Integer linear programs and heuristic solution approaches for different planning levels in underground mining

Doctoral Thesis
(Dissertation)

to be awarded the degree
Doctor of Economics (Dr. rer. pol.)

submitted by

Cinna Seifi, M. Sc.

born in Gorgan (Iran)

approved by the
Faculty of Energy and Economic Sciences
Clausthal University of Technology

Date of oral examination

April 29th, 2021

Dissertation, Clausthal University of Technology, 2021

D 104

Dean: Prof. Dr. rer. nat. habil. Bernd Lehmann

Chairperson of the Board of Examiners: Prof. Dr. sc. pol. Roland Menges

Supervising tutor: Prof. Dr. rer. pol. Jürgen Zimmermann

Reviewer: Prof. Dr. rer. nat. Stephan Westphal

Acknowledgments

I wrote the present dissertation while working as a research assistant in the Operations Research Group of the Institute of Management and Economics at the Clausthal University of Technology.

My special thanks go to my doctoral supervisor, **Prof. Dr. Jürgen Zimmermann**, for his constant promotion of the work, constant willingness to discuss, and excellent working conditions.

Writing this dissertation wouldn't have been possible without my best friend, **Kaijie Chen**. Kaijie was the best friend I made when I started to work. He stood by me during every struggle and all my successes. That is true friendship.

I also had the great pleasure of working with **Dr. Marco Schulze** and **Prof. Dr. Julia Rieck**. Thank you, Marco and Julia, for your profound belief in my abilities, for your invaluable contribution, and for never letting me down.

I want to extend my extraordinary thanks to my awesome wife **Shirin Kazemnejad** and my daughter **Benita** for their constant support. They were as important to this work getting done as I was. To my dearest wife: Shirin, my darling, my greatest happiness in life is that I met you at a party in Tehran 12 years ago. That moment changed my life forever. Thank you for being in my life. To my little daughter: thank you for letting me know that you are proud of me. I cannot express my deepest love for you in words. I am so grateful to have you in my life.

Finally, I want to thank EVERYONE who ever said anything positive to me or taught me something. I heard it all, and it meant something.

Pattensen, 20.05.2021

Cinna Seifi

Contents

1. Introduction

Humans have been extracting naturally occurring minerals from the earth for thousands of years. In southern Africa, around 43,000 years ago, iron was discovered near the earth's surface by ancient human beings. Methodically, prehistoric people used open-pit mining to extract ore and waste from the top down. Open-pit mines might be older than underground mines, but the human could also remove orebodies that lied a considerable distance below the surface in the past. From around 3,000 to 1,900 BC, an underground mine in England was exploited to retrieve flint from far below the surface of the earth. In underground mining, miners dug shafts instead of ultimately revealing the mining face and used ladders to retrieve the hard stone (cf. Gregory, 1980).

First attempts for applying operations research techniques in mining industries have presumably been taking place in the early 1960s, where Helmut Lerchs and Ingo F. Grossmann developed an algorithm to find the optimum design for an open-pit mine (cf. Musingwini, 2016). However, that algorithm for open-pit design is not published until 1965. Thus, Lerchs and Grossmann (1965) is the first published work that expresses a problem in a mining company as an optimization problem, where an exact solution approach is provided.

According to Newman et al. (2010), the mining process can be divided into five stages: 1. prospecting, 2. exploration, 3. development, 4. exploitation, and 5. reclamation. First of all, geologists use visual observation and physical measurements of the earth's properties to locate mineral deposits. Geologists assess the deposit's value by drilling holes in the exploration process to measure the mineral concentration and its variability in the orebody. Tonnage-grade curves are generated by interpolation techniques reflecting the possible benefits of exploiting the orebody for a given set of economic parameters. During the development stage, access rights to the land are acquired, and the ground is prepared to be mined by removing overlying waste. Moreover, the mining method is determined, production capacity and infrastructure capital are calculated, and a detailed engineering design is achieved. Ore is extracted from the ground at the exploitation stage through the surface and/or underground mining methods. It is transported in trucks (via haulage ramps) or in shafts to the surface. There, it can be stored (and ultimately sent

to a processing plant), sent directly to a processing plant, or taken to a dump. The last stage, reclamation, consists of restoring, to the extent possible, the region in which mining took place to its natural state. In mining, operations research (OR) has been mainly used for the stages of development and exploitation. Mine planners must make decisions about when and how both surface and underground mining will be carried out. Extraction decisions consist of deciding how the material can be retrieved and what to do with the material extracted. Since resources, e.g., machines and workers, are used to extract ore, there are also decisions on what kind of resources to use, and how to allocate them.

It is very widespread to classify decision problems according to their time horizons. For example, Newman et al. (2010) classify the existing approaches and models in mining according to the planning horizon or the hierarchy level into long-term (strategic), medium-term (tactical), and short-term (operational) problems. On the strategic level, mine layout and design models are determined. Scheduling models for mine production belong to the tactical level, and models for mine operating equipment allocation are part of the operational level. Furthermore, Bjørndal et al. (2012) and Leal Gomes Leite et al. (2020) propose a similar categorization. Bjørndal et al. (2012) classify the occurring problems in mining companies into strategic mine planning, tactical mine planning, and operational mine planning including transportation. Analogously, Leal Gomes Leite et al. (2020) consider 1. layout and design problems, 2. production and scheduling problems, and 3. operational equipment allocation problems on strategic, tactical, and operational planning levels, respectively.

In this work, we consider one of the biggest German potash mines and address three optimization problems on three different planning levels. Each optimization problem deals with a specific objective function and particular constraints that are relevant for the corresponding planning level.

First, we consider a so-called "extraction program planning" for a time horizon of one month on the tactical planning level. The related quality-oriented objective function aims at an even extraction of potash regarding the potassium content. For mathematically formulating the objective function, the amount of potassium contained in the output must be determined. Since the amount of total output is a priori unknown, the potassium amount can be determined primarily using non-linear constraints. The principal challenge is the linearization of the corresponding constraints, which has not been considered in the literature. Moreover, a solution procedure is proposed that solves realistically-sized problem instances regarding the quality-related objective function for the first time in a reasonable amount of time. The result of the extraction program planning provides

a statement about at what point in time which *block* (from a given set of blocks) is available or expected to be processed. A block is a part of the orebody that is retrieved by a detonation.

Next, we deal with a "preliminary (conceptual) planning of machines" within a time horizon of one week. That problem can be classified between the tactical and operational planning levels and investigates whether the results of the extraction program planning can be implemented for the first week. For this purpose, a machine scheduling problem to minimize the makespan is taking into account. That means we minimize the maximum completion time of the blocks that have to be processed within the first week of the following month according to the plan. In this regard, a mixed-integer linear program is formulated, where particular circumstances in an underground mine (e.g., reentry) are considered, and small-sized problem instances can be solved optimally. The main challenge is to provide a solution approach that can find near-optimal solutions for large-sized problem instances. In practice, it is very common that a job that is released will be scheduled on an idle machine without any delay, i.e., *non-delay scheduling* is applied. We propose a heuristic approach considering conscious delays in front of production stages. That means a job will not be scheduled on an idle machine in a production stage if there are some prioritized jobs in previous stages that are not completed yet; however, they can reach the current stage in a specific time interval. We show that in the problem at hand, so-called *active scheduling* provides more promising results than non-delay scheduling. Based on the achieved results, a prioritization of the excavation sequence will be passed on to the lower planning level in form of priority values.

Finally, a "detailed shift planning" considering a simultaneous assignment of machines and workers is taken into account on the operational planning level. That problem pursues an even progress in the underground mine under consideration. Within a work shift, i.e., during a relatively short time horizon, the skill levels of assigned workers cannot be neglected. Thus, we also deal with a personnel allocation problem. Moreover, to provide solutions that can be put into practice, different kinds of setup times must be considered, depending on the processing sequence of the operations on machines and workers. The major challenge is to express the specific circumstances of a work shift mathematically, e.g., considering workers' breaks for a possible delay in the processing time of a job, determining the processed percentage of a job during a work shift, observing removal and changeover times, etc. A part of real constraints is mathematically formulated in a relaxed program as part of a heuristic solution approach. The proposed heuristic procedure consists of two steps. In the first step, a relaxed program neglecting some setup times is solved, and the typically unfeasible solution achieved is repaired

in the second step by inserting the neglected times. Furthermore, we mathematically formulate all the problem specifications and introduce a compact mixed-integer linear program using TSP-variables. The introduced mixed-integer linear program can be easily customized to be applied in other industries and outperforms the existing solution approaches.

The results of a problem on each planning level are "plan data" for the next lower planning level and "actual data" for the next upper planning level. Thus, a constant comparison between plan and actual data between different planning levels supports the decision-making process. In this regard, decision-makers can react early enough by rescheduling the activities or equipping the mine with the needed resources to achieve the goals defined for each planning level.

The remainder of this thesis is organized as follows: In Chapter 2, we give a short introduction to the mining method considered in this thesis, discuss the individual steps of the extraction process, and describe a typical spatial division of an underground mine. Subsequently, the problem specifications on each planning level are given. Chapter 3 introduces the publications integrated into this cumulative dissertation thesis. In this regard, contributions of each publication to the related problem and contributions of the authors to each publication are described. In Chapter 4, the results of the work are summarized, and an outlook on further research is given.

2. Optimization problems on different planning levels in a potash underground mine

Potash ores are mainly found in deep deposits in Germany. As a result, potash mines are usually underground mines with a deposit that extends over a vast area, known as a flat-bedded deposit. According to Musingwini (2016), even though the planning logic is the same, optimization in underground mine planning is less specified and used than in open-pit mine planning. O'Sullivan et al. (2015) assert that due to the complex nature of precedence constraints, the characteristics of the operations and activities, and the irregularity of the blocks' size and shape, scheduling in underground mines is more complex than scheduling in open-pit mines. Musingwini (2016) argues that the most significant difference in difficulty between open-pit and underground mine planning stems from the fact that open-pit mines' mining direction is essentially down and outward to the pit limits. However, depending on the mining process used, there are various permutations in the direction of mining in underground mines (cf. Seifi et al., 2021a).

A drill-and-blast technique using the room-and-pillar mining method is the most widely used extraction method for flat-bedded deposits of limited thickness. By employing the room-and-pillar mining method, a grid-like structure appears. The reason is that the material is extracted across a horizontal plane, and pillars are left to protect the roof (cf. Hamrin, 2001; Schulze et al., 2016; and Seifi et al., 2021a). Figure 2.1 demonstrates a grid structure caused by the room-and-pillar mining method.

Using a conventional drill-and-blast technique, mining activities are carried out at the salt faces (cf. Fig. 2.1). More explicitly, nine discrete mining operations must be performed at a salt face in chronological order to remove the material. Those mining operations are listed below (K+S AG, 2013):

(1) scaling the mine roof and sidewalls,

(2) removing the scaled material,

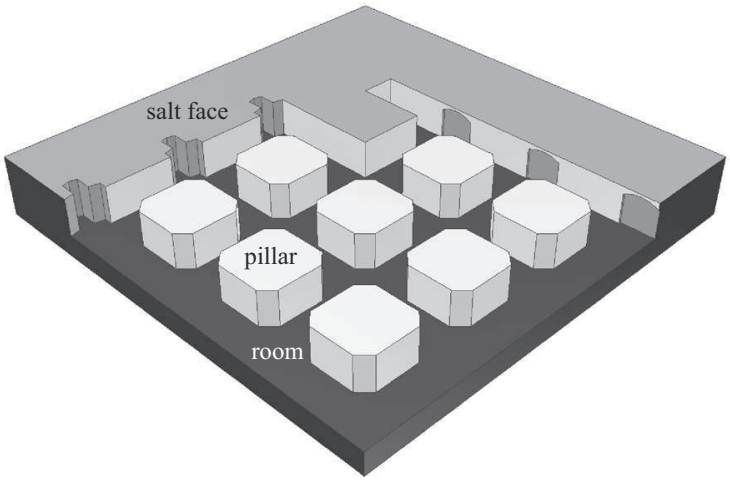

Figure 2.1.: Grid structure caused by the room-and-pillar mining method. Reprinted from Schulze et al. (2016).

(3) bolting the roof with anchors,

(4) drilling large diameter boreholes,

(5) removing drilling dust,

(6) drilling blast holes,

(7) filling blast holes with explosive substances,

(8) blasting,

(9) transporting broken material to a feeder breaker.

In the first step, layers of salt loosened by detonation are detached by *scaling machines* to provide security and prevent any possible danger to miners and mining machines. The salt accumulated in the previous process step is then removed with *small loaders* during mining operation (2). In the third step, *roof bolters* are used to install anchor bolts, which connect the individual rock layers in the roof with one another and thus give them more stability. In the fourth process step, a *drill jumbo* drills three horizontally arranged large-hole bores, each with a diameter of 280 mm and a length of seven meters. Those drill holes create the necessary cavity for the rock to collapse during blasting. The starting point of the drilling is determined on the salt face by laser aiming devices and marked in color. The inclination of the hole bores is either specified by the mine surveying or results from geological explorations. Subsequently, during mining operation (5), the dust created by the drilling is removed with a *small loader* so that the following process steps can be carried out on a level surface. In the sixth step, a *computer-controlled drilling machine* records the drilling depth of the large-hole bores. After the drill arm or the

drill carriage has been calibrated in the primary position, several blast holes are drilled with an air-water flush according to a drilling pattern. The number of blast holes to be drilled and thus the choice of the appropriate drilling scheme is primarily based on the given mining width and height. The subsequent blasting work is also mechanized. In mining operation (7), so-called *blast trucks*, which are filled with explosive substances at the beginning of each work shift, supply the blasting locations. The blast holes are then filled with the explosives via loading hoses using compressed air. The detonators are also inserted into the depth of the borehole via the charging hoses. After all the blast holes have been filled, the ignition wires are connected to one another and subsequently to mobile ignition distributors. The detonation (mining operation 8) is ignited centrally through a ripple control device near the shaft during the work shift change. The time between work shifts is long enough for the toxic, explosive clouds to escape. Following the detonation, a three-dimensional *block* of potash is extracted, and chambers, also known as *rooms*, are built in the direction of the mining activity. During mining operation (9), diesel or electrically powered *sheltered trucks* with a shovel capacity of up to 20 tonnes pick up the blasted crude material in the next work shift and transport it to the nearest feeder breaker. The position of a salt face will be shifted after completing all mining operations by the corresponding block's length in the mining activity direction. Thus, a new salt face becomes available that can be processed on the next occasion. Mining operations (1) to (9) repeat and can therefore be called a *production cycle* (cf. Schulze, 2016; Schulze et al., 2016; Seifi et al., 2021b).

The area of the underground mine under consideration is about 624 square kilometers (cf. Clausen, 2013). To achieve a proper operation, some smaller spatial areas, so-called *mining districts*, are considered for underground mines. A potash underground mine may have up to 5 mining districts. It must be noted that the machines and workers assigned to a particular mining district cannot generally be exchanged with the other mining districts. A mining district can be several square kilometers in size. Accordingly, several *tipple areas* are designed for a mining district to split the region into smaller parts and avoid long transportation routes caused by this spatial expansion. According to the size of a mining district, four to six tipple areas are usually included. The key element in a tipple area is a feeder breaker that is a machine for crushing and breaking down the extracted potash to precise measurements. The mining operations are carried out on a salt face, as previously mentioned. Following that, the explosion occurs in the mining direction, resulting in the extraction of a block of potash. After the detonation and removing the extracted potash, a new salt face appears, which is the current working place. An *underground location* is defined as a chain of consecutive blocks extracted one after the other in a specific mining direction. The material removed from each

underground location must be delivered to one particular feeder breaker. In this regard, to a feeder breaker, the underground locations in a radius of 100 meters are assigned. In general, there are five to 11 underground locations assigned to a specific feeder breaker, i.e., included in a tipple area (cf. Seifi et al., 2021b).

Figure 2.2 shows three underground locations and the assigned feeder breaker. The illustrated underground locations (UL 1 to UL 3) have different lengths in terms of the number of containing blocks. The solid-lined squares denote the room caused by the extraction of a block of potash. The dashed squares indicate the blocks in an underground location that can be removed within a given planning horizon. The number of blocks, which can be extracted in an underground location within a given time horizon, can be determined according to the geological and mining investigations. In Fig. 2.2, UL 1 and UL 2 have two and three blocks, respectively, that are available and can be excavated according to the plan for the considered planning horizon. In UL 3, on the other hand, no more blocks can be removed within the planning horizon.

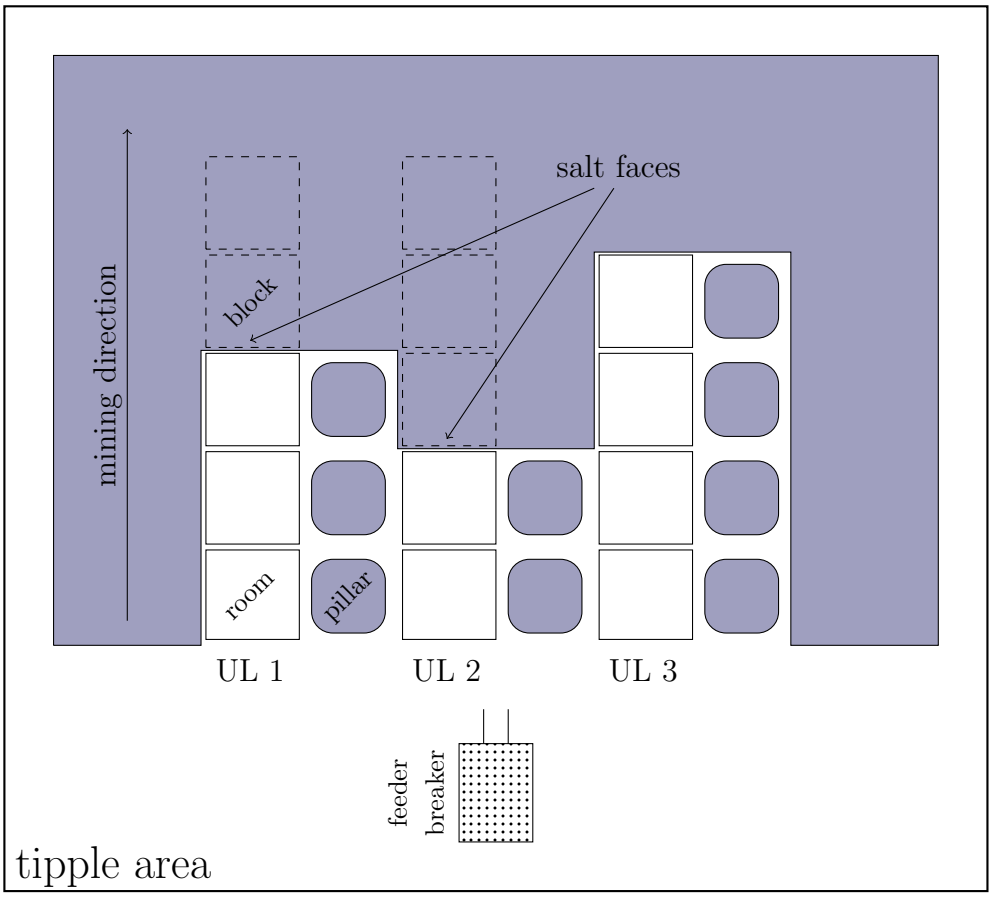

Figure 2.2.: A tipple area with associated underground locations. Adapted from Clausen (2013), p. 23.

Removing a block is a *job* that is processed in different stages. Note that blasting (mining operation 8) does not require any resources. By considering mining operations (9) and the mining operations (1) to (7) as *production stages*, our job (removing a block of potash) must be processed in eight stages. The operation on each stage is processed by exactly one machine and one worker. In the underground mine under consideration, some identical or uniform (mobile) machines are used for processing each mining operation. That means a particular operation of a job will be completed on one of the available machines in the corresponding production stage. Since a machine is coped by a skilled worker, the processing time of an individual operation depends on the assigned machine's speed and the skill level of the assigned worker. Accordingly, the sum of the processing times for the mining operations is the processing time required to remove a block (cf. Schulze et al., 2016; Seifi et al., 2021b; and Seifi et al., 2021a). For better clarity, we summarize in Table 2.1 the aforementioned mining terms.

Table 2.1.: Definitions of the mining terminology introduced. Reprinted from Seifi et al. (2021b).

Term	Definition
block	a cube of material with known dimensions removed after a detonation
feeder breaker	a crushing machine in a tipple area where the lumps extracted from the underground locations assigned to this tipple area are delivered to
mining district	the largest unit of an underground mine that comprises some smaller units (see the definition of a tipple area)
pillars	parts of underground mines that are not extracted to support the roof from collapsing
room	a space of known dimensions created after a detonation
salt face	a place at which the mining operations are conducted, i.e., it is the front side of the block that is extracted
tipple area	the largest unit of a mining district that is characterized by the assigned underground locations and a feeder breaker
underground location	a chain of consecutive blocks that can be removed in the mining direction

In Section 2.1, the problem of "extraction program planning" is described that is occurring on the tactical planning level. The results of that problem provides the right selection and sequence of the blocks to be processed during the next month. Subsequently, in Section 2.2, "preliminary (conceptual) planning of machines" is discussed that is mainly used to validate the results achieved by "extraction program planning" within a time horizon of one week. In addition, the results of preliminary (conceptual) planning of machines can be passed on to the operational planning level as block sequence prioritization. Thus, that problem connects the tactical planning level to the operational planning level and can be categorized on the tactical-operational planning level (cf. Newman et al.,

2010). Finally, in Section 2.3, the problem of "detailed shift planning" is introduced that occurs on the operational planning level and takes a simultaneous assignment of machines and workers into consideration. For each problem, the considered objective function and the corresponding restrictions are given.

2.1. Extraction program planning

In this section, we describe a block selection and sequencing problem on the tactical planning level. The problem specification is based on the description introduced in Seifi et al. (2021b).

The major valuable minerals in potash ores are potassium chloride and sodium chloride. Especially, potassium chloride is essential because it is most commonly used as a fertilizer and is an additive in the chemical and medical industry as well as human and animal food processing (cf. Chesworth, 2008; USGS, 2011; Schulze et al., 2016). The crude salt extracted in an underground mine is transported to the surface and then to the processing plants. In above-ground processing plants, potassium chloride is separated from potash by flotation, recrystallization, or electrostatic separation[1]. Like every other device, the processing plants above ground have an *optimal operating point* at which they perform best. Various variables may be used to assess this point depending on the considered device. If the potassium content of the removed potash is equal to a given value, the processing plants above ground considered in this work have the most cost-effective performance.

By removing a block of potash, the length of the large-hole bores and the drilling pattern of the blast holes determine the dimensions of the excavation. In that way, the amount of crude material expected to be excavated is known. Based on geological and mining investigations, the potassium content in a block *in percent* can be estimated. Thus, the quality of a block can be quantified by the percentage of potassium contained in the corresponding block, the so-called *quality value* of a block (in percent). If we multiply the quality value of a block by the amount of potash extracted from that block, the amount of potassium obtained is determined.

The amount of potassium extracted within a particular time interval, i.e., the quality of the removed potash, depends on the quality values of blocks excavated in that time. Moreover, the excavation sequence affects the quality value of the output within the sub-

[1] 100% potassium chloride is exactly equal to 63.17% potassium oxide. Thus, the percentage of potassium in the potassium chloride is the equivalent content of potassium oxide by weight (cf. Heinz and von der Osten, 1982, p. 147).

intervals included in a planning horizon. Note that, on the one hand, the blocks vary in terms of the volume and the quality, and on the other hand, not all the available blocks can be processed within a specific time horizon. If the quality value greatly fluctuates, we speak of a non-homogeneous output that leads to high costs for above-ground processing. Since quality fluctuations are highly probable by the excavation, quality-oriented mining of blocks plays a decisive role in reducing the processing costs. In this regard, we want to answer two questions: 1. which blocks should be excavated (i.e., block selection), and 2. if a block is extracted at which time it must be removed (i.e., block sequencing).

We defined an underground location as a chain of available blocks that can be removed during a given planning horizon. One specific block in an underground location can be processed only if the previous blocks in the same underground location have been extracted. Accordingly, each block can have one direct predecessor and one direct successor block. In other words, there are precedence relationships between the blocks in an underground location. As mentioned, the time needed to remove a block is the sum of processing times required for every single mining operation. On the tactical planning level, because of a relatively long time horizon, the processing times required for the extraction of blocks are estimated. Hence, not the assigned machines and workers for each mining operation, but the dimensions or shapes of blocks are crucial factors for determining the corresponding processing times. Additionally, the current status of a block at the beginning of the planning horizon must be considered to estimate the needed processing time. A block for which all mining operations must be processed has a longer processing time in comparison to a block for which not all the mining operations (e.g., only mining operation 5–9) must be carried out.

We mentioned that the extracted material from an underground location is first transported to the assigned feeder breaker in the corresponding tipple area via loaders. Feeder breakers are connected by a conveyor belt system to a central bunker system close to the shaft. From the shaft, the removed material finally reaches the surface. The conveyor system, the bunker, and the processing plants above ground have all a limited capacity. Accordingly, for each tipple area and each mining district, an upper limit of the output must be observed. On the other hand, a lower limit for the total output over the planning horizon must be considered, since the primary task of a mining company is the extraction of raw minerals. In this regard, in each work shift and every mining district, a minimum output must also be satisfied.

The quality of the potash regarding the potassium content must lie in a prescribed tolerance range during a predefined time interval that is typically shorter than the entire planning horizon. Therefore, the given planning horizon on the tactical planning level

(e.g., one month) is divided into some smaller sub-intervals (e.g., weeks). The length of the planning horizon and the corresponding sub-intervals can be customized depending on the current state of the mine. Let A_b in tonnes be the amount of potash extracted from block b. Moreover, let Q_b in percent be the quality value of block b. We assume that a set of blocks $\mathscr{B}$ is removed during a specific time interval. The quality value of the entire output $\overline{q}$ (in %) is the weighted average of the quality values of the extracted blocks and calculated as follows:

$$\overline{q} = \frac{\sum\limits_{b \in \mathscr{B}} Q_b \cdot A_b}{\sum\limits_{b \in \mathscr{B}} A_b}.$$

We said that the processing plants above ground perform economically best at the optimal operation point. That point is represented by a quality target value in percent. The aim of our optimization problem is that $\overline{q}$ deviates as little as possible from the given quality target value. To determine the deviations, we distinguish between a negative and a positive deviation. If $\overline{q}$ is less than the quality target value, a negative deviation is a case. Analogously, there is a positive deviation if $\overline{q}$ is greater than the quality target value. In order to formulate the objective function, the absolute value of the deviation of $\overline{q}$ from the predetermined quality target value in each sub-interval must be first determined. Note that the amount of material extracted within a particular time interval must be known so that the value of $\overline{q}$ and accordingly the values of negative and positive deviations can be calculated. However, the value of the denominator of $\overline{q}$, i.e., the entire output, is a priori unknown with the result that some non-linear constraints in the mathematical formulation are required for determining the deviations from the target value; those constraints must be then linearized. The objective is to minimize the average of the calculated deviations over the number of sub-intervals contained in the planning horizon.

In summary, a quality-oriented objective function must be minimized, where the following constraints are observed:

Precedence relationships between the blocks in an underground location;

Minimum limit of the output over the entire planning horizon, in each work shift, and for every mining district;

Maximum limit of the output for each tipple area and every mining district within every single work shift; and

Quality tolerance range over each certain sub-interval in the planning horizon.

2.2. Preliminary (conceptual) planning of machines

In this section, we describe a machine scheduling problem that can be classified between the tactical and the operational planning level with regard to the considered planning horizon. The problem specification is based on the description introduced in Schulze et al. (2016).

As part of the preliminary (conceptual) planning of machines, it is examined whether the number of blocks determined from extraction program planning (see Section 2.1) can be removed for the next week. We said that removing a block is a job that consists of some operations (mining operations described above). As mentioned before, blasting (mining operation 8) does not require any resources, and mining operations (9) and mining operations (1) to (7) are the *production stages*. According to a planning horizon of one week, the workers do not have to be considered on this planning level, and the only resources are the available machines in each production stage.

In the problem at hand, a so-called *reentry*, which plays a crucial role in the production cycle, must be given special attention. After completing the first process step, the remaining production cycle for each block should be completed within a defined period of time to ensure the safety of personnel; otherwise, there is a risk of rock layers falling due to the rock pressure. The mining operations (2) to (6) should therefore be finished within $\tau > 0$ time units after completing mining operation (1). If during or with the completion of mining operation $k, (2) \leq k \leq (6)$, however, τ time units are achieved or exceeded, a renewed scaling is necessary for safety reasons. That means process step (1) must be carried out once again (cf. Fig. 2.3) before the next process step $k + 1$, as planned, can be processed. As a result, the roof and side walls are continually monitored to ensure that loose material does not detach unexpectedly.

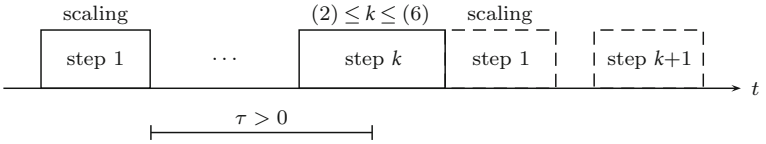

Figure 2.3.: Execution of mining operations over the time axis, where τ time units are exceeded and mining operation (1) is revisited. Reprinted from Schulze et al. (2016).

When checking whether τ time units have been reached or exceeded, mining operations (7) to (9) are not taken into account. This restriction is due to the fact that it is no longer necessary to scale again after the blast holes have been filled with the subsequent blasting.

The blasted material is then transported away by large front shovel loaders equipped with protected cabins so that the employees are well protected against possible falling layers of rock. We assume that a repetition of the first process step has to be carried out at most once for each block since a value for τ, which is valid in practice, is usually much greater than the minimum duration of an entire production cycle.

Our problem has structural similarities to the known *hybrid flow shop* (HFS) *scheduling problem*; a set of jobs must be processed on different production stages, and in every production stage, a given number of machines is available. Basic HFS scheduling problems have the following characteristics (see, e.g., the surveys of Ruiz and Vázquez-Rodríguez, 2010, as well as Ribas et al., 2010):

- jobs must be processed in $k \geq 2$ production stages (separated by unlimited buffers);
- there are $\left|M^k\right| \geq 1$ parallel machines in production stage k, where there is at least one production stage with more than one machine;
- jobs are not preemptable and have several operations that have to be processed on machines, following the same production flow; and
- Each machine in each production stage can process at most one job at a time and each job can be processed by at most one machine at a time.

Accordingly, we can classify our problem as a k-stage HFS scheduling problem with $k = 8$ if we consider removing a block as a job, and mining operations (1) to (7) as well as mining operation (9) as production stages. Figure 2.4 illustrates the resulting production environment with eight production stages. Note that production stage (8) does not have to be illustrated in the environment.

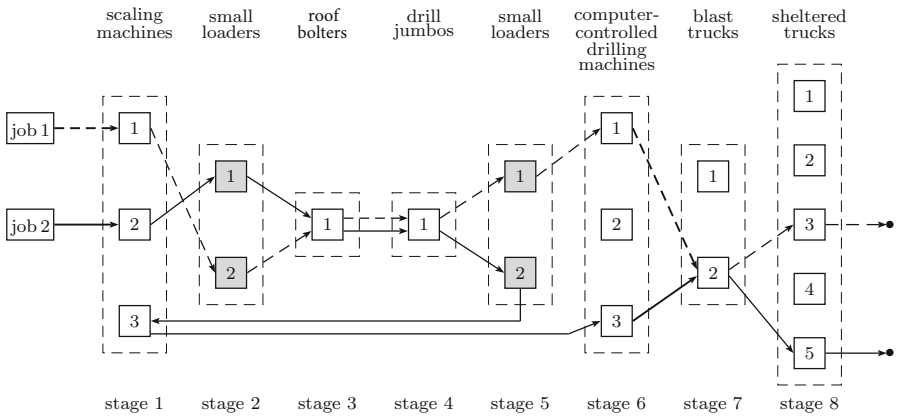

Figure 2.4.: Production environment of the proposed HFS scheduling problem. Reprinted from Schulze et al. (2016).

In the considered problem, the machines used in production stages (2) and (5) are the same small loaders. Those are represented in gray color in Fig. 2.4. That means if a machine is processing a job in stage (2), that machine cannot be used to process another job in stage (5). Furthermore, a job has to revisit the first stage if τ time units are achieved or exceeded with the completion of step $k, (2) \leq k \leq (6)$ for the corresponding job, i.e., a situation-related reentry must be taken into account. The production paths of two different jobs are shown in Fig. 2.4. We see that job 2, after completion in production stage (5), has to be processed in production stage (1), where no situation-related reentry for job 1 must be considered.

Figure 2.5 illustrates five underground locations u_1 to u_5. For example, block j_7 at location u_2 can be excavated only if the preceding production cycle for block j_6 at the same underground location is completed. Thus, a precedence relationship between the jobs of an underground location must be observed. That means at the beginning of the planning horizon, only a subset of jobs $J^0 = \{j_1, j_6, j_9, j_{15}, j_{17}\} \subset J$ can be processed. Dynamically, new jobs become available at different points in time, i.e., if a job is completed, the direct successor job in the same underground location can be operated on.

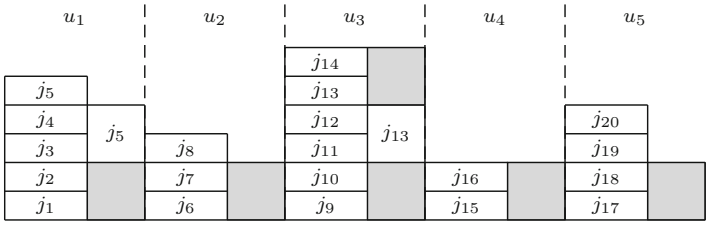

Figure 2.5.: Precedence relationship between the jobs in an underground location. Reprinted from Schulze et al. (2016).

Altogether, we have an 8-stage hybrid flow shop scheduling problem, where reentry and precedence relationship between jobs of an underground location must be taken into consideration. The aim is to minimize the maximum completion time of excavations, i.e., makespan. On the one hand, the results show whether the solution obtained from the tactical planning level can be implemented in the first week. On the other hand, with the aid of achieved results, a prioritization of the block sequence is derived, which is passed on to the lower planning level in form of priority values.

2.3. Detailed shift planning

In this section, we describe a shift scheduling problem with a simultaneous assignment of machines and workers on the operational planning level. The problem specification is based on the description introduced in Seifi et al. (2019) and Seifi et al. (2021a).

Flexible planning and control of the extraction are some of the characteristics that become possible because of a flat-bedded deposit in an underground mine. As mentioned before, the room-and-pillar mining method is the most applied extraction method in potash underground mines. That method is particularly characterized by a vast expansion of the deposit and the resulting large number of potential extraction points, the so-called *working places*. The flexibility of a plan results from the large number of working places. However, creating work orders for staff and mobile equipment in work shifts can be complex. The fact that different mining operations must be processed at different working places makes the planning more difficult. Furthermore, it must be taken into account that not all the available working places can be processed, and a production cycle started for a working place cannot necessarily be completed within a planning horizon of one work shift. Thus, providing a schedule to determine which working places in which sequence must be processed plays a decisive role. In the field of scheduling, the fact that only a subset of the given set of jobs can be processed during a prescribed planning horizon is less studied. In the considered problem, the machines are mobile, i.e., the jobs are not delivered to the machines, but a skilled worker drives a particular machine to a job. Hence, a simultaneous assignment of machines and workers must be considered, where not all workers can drive and handle all machines. Additionally, workers have different skill levels based on their experiences, leading to different handling times with a machine. Considering the different speeds of the machines, the processing time of an operation strongly depends on the assigned machine and worker. A further important point that is less investigated in the field of personnel scheduling is that a worker can change an assigned machine within a work shift (cf. Schulze and Zimmermann, 2017; Van den Bergh et al., 2013). The equipment and resources (e.g., machines and workers) assigned to a particular mining district cannot be exchanged with other mining districts during a work shift. It is therefore reasonable that we schedule a work shift for each mining district separately.

As mentioned in Section 2.2, without considering the workers, the production environment can be classified as a variant of a hybrid flow shop (HFS) scheduling problem within a time horizon of one week (cf. Schulze et al., 2016). In the problem on the operational planning level, because of many uncertainties affecting the availability of machines, we

consider a planning horizon of one work shift. As a result, a "reentry" (see Section 2.2) is not considered.

The production environment in our shift scheduling problem is another variant of a hybrid flow shop scheduling problem, where the characteristics of the basic HFS scheduling problem are as follows (cf. Section 2.2):

(i) we have eight production stages;

(ii) there are at least two machines in production stage nine;

(iii) we consider the operations as non-preemptable because the assigned machines complete the processing of the operations regardless of possible interruptions. After any possible interruption (e.g., because of the workers' breaks or at the end of the work shift), the processing continues at the next available opportunity by the same machine; and

(iv) although different working places can have different states so that the order of operations for different jobs can begin with different operation index numbers; but, a job must be processed in at least one stage.

For our machine scheduling problem, some further aspects must be taken into account:

(v) it is possible that the processing of a job or an operation is interrupted at the end of the work shift, and not all available jobs have to be processed within a work shift (job selection);

(vi) the machines must be driven to jobs, and driving times must be considered as setup times; and

(vii) mining-specific requirements imply precedence relationships between the jobs.

Considering the points (v) to (vii), we classify our problem as a variant of a HFS scheduling problem. The points (vi) and (vii) are described in the following in more detail.

Each machine in a production stage must be handled by an operator, i.e., a worker. It must be taken into account that not all workers can be assigned to all machines since workers have different skills. The workers who are eligible for a machine may have different skill levels, i.e., they can process the same job with different processing times. Thus, during a short planning horizon, the role of workers cannot be neglected. Causing different processing times is not the only reason for integrating a worker assignment to our machine scheduling problem. For a worker who goes from one machine to another assigned machine, a *changeover time* must be considered. Moreover, some setup times

must be performed on the new assigned machine by the worker. Those setup and changeover times are considerable regarding the duration of a work shift. Hence, assigning and sequencing of the workers largely affect the results of our problem.

The setup times that must be considered in our problem depend on machines, the processing sequence of operations on machines, and the processing sequence of operations by workers. In this regard, we can distinguish between 1. machine- and sequence-dependent setup times, 2. sequence-dependent changeover times, and 3. machine- and sequence-dependent removal times. Those times are described below in detail.

machine- and sequence-dependent setup times

A worker must perform preventive maintenance, the so-called first technical services, for a machine before starting an operation. The time of the first technical services only depends on the machine. Whether preventive maintenance must be done or not depends on the processing sequence of the operations; either the machine processes the first allocated operation within the considered work shift, or the assigned worker has changed his machine. Note if a worker goes to a new assigned machine, the first technical services must be performed even if the new assigned machine was already in use.

Moreover, driving times between two operations that are processed by the same machine, i.e., driving times from a working place to another working place, depend on the driving speed of the assigned machine and the locations of the corresponding jobs. That means the driving times are machine- and sequence-dependent, too.

sequence-dependent changeover times

As mentioned before, workers can change the assigned machines within the work shift under consideration. If a worker has to process two operations with two different machines, he is picked up by a transport vehicle or walks on foot to the new assigned machine. The new machine is parked near the location of the previous operation processed on that machine, and the driving speed of the transport vehicle or the walking speed of a worker is assumed as constant. Hence, the occurring times to change a machine are sequence-dependent changeover times.

machine- and sequence-dependent removal time

If a machine is not going to be used anymore within the work shift under consideration, it must be cleaned and fueled (last technical services). That would be a case if the machine processed the last operation assigned, which is known only if we have a schedule a priori or if the work shift is over. The time needed for the last technical services depends, analogous to the first technical services, only on the considered machine. In the case that the work shift is not over yet, we have to

know the processing sequence of the operations on a machine to determine the last operation processed. Thus, removal times are machine- and sequence-dependent.

As mentioned above, we schedule a work shift for one mining district. In a mining district, some specific mining requirements (MR) must be taken into account:

MR (1) According to the labor law, a miner has to make a Δ-minute break during his work shift. The workers' breaks must start within a given time interval $[\varphi^{\alpha}, \varphi^{\omega}]$. Note that taking a break causes a possible delay in processing an operation. That means the break of a worker may interrupt the processing of an operation, but the assigned worker will continue with the same operation after his break. The value of φ^{ω} is set so that if a worker starts his break at φ^{ω}, the break will be finished before the corresponding work shift is over.

MR (2) In an underground location, it is not allowed that two operations are processed at the same time. For safety reasons, not more than one machine can be utilized in an underground location at the same time. As a result, there are disjunctive constraints between the different working places of an underground location.

MR (3) It is very desirable that all the working places of an underground location have the same state, i.e., the underground location is evenly progressed. For many reasons, it is, however, not a case, i.e., different mining operations must be processed for different working places of an underground location (cf. Fig. 2.6). As a result, the working places in an underground location must be prioritized. In an underground location, the currently available jobs with mining operation (9) have the greatest priority and must be processed at first. If none of the currently available jobs requires mining operation (9), a job with the smallest operation type number has the greatest priority value.

Figure 2.6 illustrates a tipple area with four underground locations (I) to (IV). The underground locations have two to three working places. The number in parentheses is the mining operation that has to be done next for the related working place. In underground location (II), the jobs with mining operation (9) have a greater priority value in comparison to the job with mining operation (1). Besides, between those jobs, there is a disjunctive constraint according to MR (2). Taking MR (2) and MR (3) into account, in underground location (III), the job with mining operation (3) is first completed before starting with the other job. A similar situation is given in underground location (IV), where the job with mining operation (5) must be first completed before starting with the job with mining operation (6). For underground location (I), MR (2) must only be respected.

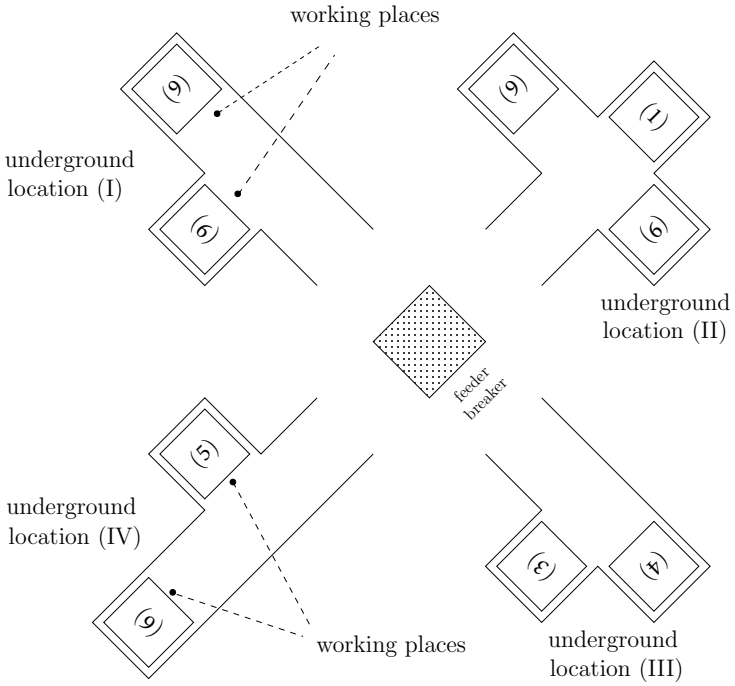

Figure 2.6.: The prioritization between working places in an underground location. Adapted from Seifi et al. (2021a).

MR (4) As mentioned, an operation can be interrupted at the end of the work shift. Suppose there is such an interrupted job from the previous work shift in an underground location. In that case, we can start the processing of an operation in that underground location only if the interrupted operation from the last work shift is processed during the current work shift. Note that the interrupted operation does not have to be the first operation processed in the corresponding underground location.

MR (5) Due to the high time required for the calibration to be prepared, the drilling of large diameter boreholes (mining operation 4) is not allowed to be interrupted. In other words, those operations can be started only if they can be completed by the end of the work shift.

MR (6) In order to reduce the number of machine changes and the involved risk, the number of machine changes is limited by two in the underground mine under consideration. That means a maximum of two different workers can be assigned to a machine, and a worker can be assigned to a maximum of two different machines during one work shift.

In the presented problem, we pursue the goal that the mine is evenly progressed. In this regard, we consider an amount of crude material for each operation of a job that

is expected to be excavated after the completion of the operation. For each mining operation, a target value for the output should be achieved within a work shift. The target values are predetermined from a superordinate planning level with the aid of constant comparison of target and actual data of the excavated raw material. At the end of the work shift, the amount of material extracted is calculated for each operation. In this regard, if an operation of a job is interrupted at the end of the work shift, the percentage of that operation that has been processed within the work shift must be determined. Accordingly, the lower deviation from the predefined target value is figured out. Lower deviation means, if the excavated material exceeds the target value, the difference will not be considered in the objective function. The aim of the optimization problem under consideration is to minimize the lower deviations accumulated over all of the mining operations so that the progress of mining stays consistent.

3. Overview of the publications

The provision of research results in the form of scientific lectures and publications is a fundamental prerequisite for actively participating in current research developments. For this cumulative dissertation thesis, the following papers are considered:

I. Seifi, C., Schulze, M., Zimmermann, J. (2021). Solution procedures for block selection and sequencing in flat-bedded potash underground mines. OR Spectrum. https://doi.org/10.1007/s00291-021-00618-z (cf. Appendix A)

II. Schulze, M., Rieck, J., Seifi, C., Zimmermann, J. (2016). Machine scheduling in underground mining: an application in the potash industry. OR Spectrum, 38(2):365–403. (cf. Appendix B)

III. Seifi, C., Schulze, M., Zimmermann, J. (2019). A two-stage solution approach for a shift scheduling problem with a simultaneous assignment of machines and workers. In Mueller et al. (Eds), Mining goes digital: proceedings of the 39th international symposium application of computers and operations research in the mineral industry (APCOM). June 2019, Wroclaw, Poland, pages 377–385. Taylor & Francis Group, London. (cf. Appendix C)

IV. Seifi, C., Schulze, M., Zimmermann, J. (2021). A new mathematical formulation for a potash-mine shift scheduling problem with a simultaneous assignment of machines and workers. European Journal of Operational Research, 292(1): 27–42. (cf. Appendix D)

The scientific quality, depth, and validity of articles can be primarily evaluated with the help of rankings. Three of the papers considered in this cumulative dissertation have been published in journals that are assessed by the "Association of University Teachers for Business Administration (VHB)." The first version of the ranking was developed in 2003. Table 3.1 shows the ranking values achieved for the relevant journals and the number of articles published in the respective journals. The ranking value "**A**" means that the corresponding journal is a leading scientific journal in the field of operations research[1].

[1] Quelle: https://vhbonline.org/fileadmin/user_upload/JQ3_OR_01.pdf

Journal	Ranking	Number of papers
OR Spectrum	**A**	2
European Journal of Operational Research	**A**	1

Table 3.1.: Ranking values of the relative journals.

One other paper is published in "Mining goes Digital," the proceedings of the international symposium application of computers and operations research in the mineral industry (APCOM). APCOM is one of the leading conferences in the field of mining that not only focuses on geostatistics and resource estimation but has broadened its horizon to information and communication technology in the mineral industry. Mining goes Digital is a collection of high-quality, peer-reviewed papers covering recent developments in, e.g., geostatistics and resource estimation, mine planning, scheduling and dispatch, internet of things, robotics, etc.

In what follows, we denote the papers with the numbers indicated above. Paper I and Paper II address the problems introduced in Sections 2.1 and 2.2, respectively. Papers III and IV both deal with the problem described in Section 2.3. The publications are given in Appendix A to D. In the following sections, the contributions of the published papers to the related problems are given. Furthermore, the authors' contributions of each paper are described.

3.1. Paper I

In this paper, we deal with a block selection and sequencing problem with a quality-oriented objective function in terms of the potassium contained in the entire output. A quality-related objective function is less studied in the literature. Since the amount of material removed within a planning horizon is a priori unknown, determining the quality values of the extracted material requires some non-linear constraints. The major contribution of this paper is to linearize those constraints to introduce a mixed-integer linear mathematical program that can be used to solve small-sized problem instances to optimality. Some precedence relationships, maximum and minimum limits of the output, and a quality tolerance range must be observed to generate a feasible solution. Another contribution of Paper I is to show that the problem under consideration is $\mathscr{NP}$-hard in the strong sense even if the quality-related constraints are neglected. That is the reason why a MILP-solver cannot find feasible solutions for the most challenging problem instances. From a practical point of view, the significant contribution of Paper I is to develop a solution approach that finds high-quality solutions for realistically-sized

problem instances. In this regard, a problem-specific heuristic solution procedure is proposed. Furthermore, we show that a subtle combination of the proposed heuristic with the mathematical program improves the results found by the heuristic considerably.

The paper is written by three authors:

Cinna Seifi, *Marco Schulze*, and *Jürgen Zimmermann*.

Paper I is published in *OR Spectrum* in 2021 and is in press. Articles in press are accepted, peer reviewed articles that are not yet assigned to volumes/issues, but are citable using DOI. The author of this dissertation thesis is the first author of Paper I. The contributions of every author to each part of this paper is given below:

Literature study

Primarily Cinna Seifi with the collaboration of Marco Schulze

Linearization of the mathematical formulation

Cinna Seifi

Showing the $\mathcal{NP}$-hardness in the strong sense

Cinna Seifi

Providing the heuristic approach

Primarily Cinna Seifi with the collaboration of Marco Schulze

Implementation of the mathematical model and the heuristic approach

Cinna Seifi

Generating the test instances

Primarily Cinna Seifi with the collaboration of Marco Schulze

Evaluating the results

Primarily Cinna Seifi with the collaboration of Marco Schulze

Manuscript

Primarily Cinna Seifi with the collaboration of Marco Schulze and Jürgen Zimmermann

Revision after review

Primarily Cinna Seifi with the collaboration of Marco Schulze and Jürgen Zimmermann

3.2. Paper II

The paper considers a machine scheduling problem that appears in potash mining, where a block excavation sequence has to be found. One of the major contributions of Paper II is to introduce a mixed-integer linear program for a hybrid flow shop scheduling problem taking "reentry" into consideration with the result that small-scale instances can be solved to optimality. Moreover, reasonable lower bounds are provided to evaluate the results that are not proven to be optimal. The most significant contribution of the paper is to suggest a heuristic solution procedure in order to solve larger problem instances. In this regard, a priority rule-based construction procedure embedded in a multi-start environment is developed. The proposed heuristic solution procedure performs very well for medium-sized instances. The basic multi-start algorithm is then extended to an advanced multi-start algorithm that considers conscious delays of jobs in front of production stages. For large-sized problem instances, the advanced multi-start algorithm results are shown to be best compared to the other procedures.

The paper is written by four authors:

Marco Schulze, *Julia Rieck*, *Cinna Seifi*, and *Jürgen Zimmermann*.

Paper II is published in *OR Spectrum* in 2016. The author of this dissertation thesis is the third author of Paper II. The contributions of every author to each part of this paper is given below:

Literature study

Marco Schulze and Julia Rieck

Introducing the mixed-integer linear mathematical formulation

Marco Schulze and Julia Rieck with the collaboration of Jürgen Zimmermann

Providing the lower bounds

Marco Schulze and Julia Rieck

Providing the constructive multi-start heuristic

Marco Schulze and Julia Rieck with the collaboration of Cinna Seifi

Providing the advanced multi-start heuristic

Cinna Seifi

Modifying the Giffler-Thompson procedure

Primarily Cinna Seifi with the collaboration of Marco Schulze and Julia Rieck

Implementation of the mathematical model

Marco Schulze

Implementation of the heuristic procedures

Cinna Seifi

Generating the test instances

Marco Schulze

Evaluating the results

Primarily Cinna Seifi with the collaboration of Marco Schulze and Julia Rieck

Manuscript

Marco Schulze and Julia Rieck with the collaboration of Jürgen Zimmermann

Revision after review

Marco Schulze and Julia Rieck with the collaboration of Jürgen Zimmermann

3.3. Paper III

This paper deals with a work shift scheduling problem taking a simultaneous assignment of machines and workers into account. One of the major contributions of Paper III is to introduce a mixed-integer linear formulation for a relaxation of the problem considering the specific processing time and duration of each operation with respect to the breaks of workers within a work shift. With the aid of the proposed mathematical program, the processed part of each operation during the work shift must also be determined. On the other hand, some restrictions regarding the setup times are neglected. Another contribution is to propose a solution approach that provides high-quality solutions for realistically-sized problem instances within an adequate time limit. In this regard, a two-stage solution approach is proposed, where the relaxed mathematical program provides a schedule in the first stage. The schedule is generally unfeasible and must be modified in a second stage by integrating the necessary time intervals that are not considered in the first stage. A preliminary performance analysis using realistic problem instances shows that the solutions of our proposed two-stage approach outperform the solutions currently generated by an existing constructive heuristic procedure proposed by Schulze and Zimmermann (2017).

The paper is written by three authors:

Cinna Seifi, *Marco Schulze*, and *Jürgen Zimmermann*.

Paper III is published in the proceedings of the 39th international symposium application of computers and operations research in the mineral industry (APCOM) in 2019. The author of this dissertation thesis is the first author of Paper III. The contributions of every author to each part of this paper is given below:

Literature study
Cinna Seifi

Considering the workers' breaks in the processing times
Cinna Seifi

Determining the processed part of each operation within a work shift
Cinna Seifi

Introducing the two-stage approach
Primarily Cinna Seifi with the collaboration of Jürgen Zimmermann

Implementation of the two-stage approach
Cinna Seifi

Generating the test instances
Primarily Cinna Seifi with the collaboration of Marco Schulze

Evaluating the results
Cinna Seifi

Manuscript
Primarily Cinna Seifi with the collaboration of Marco Schulze and Jürgen Zimmermann

Revision after review
Primarily Cinna Seifi with the collaboration of Marco Schulze and Jürgen Zimmermann

3.4. Paper IV

In this paper, a mixed-integer linear program for a shift scheduling problem is introduced. The contribution of this paper is to introduce a mixed-integer linear mathematical program for a real-life shift scheduling problem in a hybrid flow shop production environment that considers 1. staff scheduling and job selection, 2. sequence- and machine-dependent setup and removal times, as well as sequence-dependent changeover times, 3. workers' breaks for a possible delay for the processing of operations, and 4. mining-specific restrictions in an underground mine. In this regard, a new mixed-integer linear program using TSP-variables is proposed that is more compact in comparison to a time-indexed or a position-based mathematical formulation. Moreover, the introduced mathematical program expresses the very complex problem in a comprehensible and understandable way. Our performance analysis shows that even though our mathematical

formulation does not prove the optimality of the solutions found, the achieved solutions are much better than the solutions provided by the existing approaches.

The paper is written by three authors:

Cinna Seifi, *Marco Schulze*, and *Jürgen Zimmermann*.

Paper IV is published in *European Journal of Operational Research (EJOR)* in 2021 (online available since October 12th, 2020). The author of this dissertation thesis is the first author of Paper IV. The contributions of every author to each part of this paper is given below:

Literature study

Cinna Seifi

Using TSP-variables to determine the setup, changeover, and removal times

Cinna Seifi

Introducing the mixed-integer linear mathematical formulation

Cinna Seifi

Implementation of the mathematical model

Cinna Seifi

Generating the test instances

Primarily Cinna Seifi with the collaboration of Marco Schulze

Evaluating the results

Cinna Seifi

Manuscript

Primarily Cinna Seifi with the collaboration of Marco Schulze and Jürgen Zimmermann

Revision after review

Primarily Cinna Seifi with the collaboration of Marco Schulze and Jürgen Zimmermann

4. Conclusion

In this cumulative thesis, we addressed optimization problems that are dealt with on different planning levels in an underground mine. On every planning level, some specific circumstances according to the planning horizon are taken into account, where an appropriate objective function is considered.

On the tactical planning level, the optimization problem aims at a steady output of potash regarding the potassium content. For this purpose, we minimize the deviations of the output quality value from a prescribed quality target value. The deviations are calculated for certain sub-intervals within a given planning horizon. In that problem, the capacities of the available facilities must be respected. Furthermore, a minimum output from several time-related and geographic-related points of view must be fulfilled. A solution to that problem provides a set of blocks that should be removed within the next month. For the problem occurring on the tactical planning level, we introduce a mixed-integer linear program and a heuristic approach to solve a block selection and sequencing problem. The performance analysis is conducted on randomly generated 100 problem instances. The problem instances consist of five test sets of 20 problem instances each. Moreover, we consider five different levels for the lower limit of the total output. The greater the considered total output, the more challenging are the problem instances. The results show that the proposed heuristic approach can find near-optimal solutions for all problem instances, where the smaller and less challenging problem instances can be optimally solved using the proposed mixed-integer linear program. For the large-sized and most challenging problem instances, an initial solution found by the proposed heuristic approach can be improved with the consecutive use of CPLEX-solver utilized for the suggested mathematical program. In that way, we can find high-quality solutions for practice-relevant problems in potash mines within a reasonable amount of time.

On the tactical-operational planning level, the blocks that must be extracted during the next week (based on the result provided on the tactical planning level) are considered. Removing a block is a job consisting of several operations that must be processed by a specific machine and a skilled worker. According to the planning horizon on the considered planning level, workers can be neglected in the production environment. The

machine scheduling problem is classified as a hybrid flow shop scheduling problem, where for safety reasons, reentry must be considered. That means some mining operations have to be finished within a specific time interval; otherwise, a security precaution is made, and the first mining operation is visited once more. The objective function is to minimize makespan to answer whether the blocks can be removed according to the plan in the next week. In that way, the plan data can be corrected according to the more detailed prediction made. On the other hand, prioritizing the blocks and the corresponding underground locations can be passed on to the operational planning level. For the machine scheduling problem on the tactical-operational planning level, we introduce a mixed-integer linear program and provide some lower bounds to facilitate the solution process and evaluate the results achieved. Additionally, we propose a priority rule-based construction heuristic that is embedded in a multi-start algorithm. The heuristic procedure is then improved to an advanced multi-start algorithm considering conscious delays of jobs. We generated six test sets considering the different numbers of jobs and underground locations. The performance analysis shows that small problem instances with 30 jobs and five underground locations can be optimally solved with CPLEX-solver. On the contrary, for larger problem instances with 60 to 240 jobs and 10 to 30 underground locations, the best performance is achieved by the suggested advanced multi-start heuristic considering a conscious delay of jobs.

On the operational planning level, the available machines and workers during one work shift are assigned to the mining operations that must be performed at a working place to remove a block. The underground locations that can be processed during one work shift are primarily determined based on the prioritization provided on the upper planning level. The aim is that the mine is evenly progressed at the end of the work shift. For this purpose, a given amount of material for each mining operation must be achieved. From a mathematical perspective, a lower deviation from the given target value accumulated over the mining operations must be minimized. We have to decide which jobs (operations) are performed in the current work shift, assigning an appropriate machine and worker to each operation and appointing the job's right processing time. To provide a realistic schedule, machine- and sequence-dependent setup and removal times, as well as sequence-dependent changeover times, must be considered. Moreover, workers' breaks that typically cause a delay in the processing of the operations must be taken into account. For that problem, we first propose a two-stage heuristic approach. In this regard, we formulate a relaxation of the problem described and introduce an algorithm to generate feasible solutions using the solution achieved by the relaxation. The results of a preliminary performance analysis considering realistic problem instances show that the two-stage approach can find for 70% of the problem instances a better solution than an existing

heuristic procedure, where the solutions found for the other problem instances are, on average, 20.3% far from the best solution found. In addition, since a linear mathematical formulation has not existed for our shift scheduling problem, we formulate a mixed-integer linear program using TSP-variables taking all the problem's aspects into consideration. In contrast to other mathematical formulations, e.g., position-based or time-indexed, the proposed program is more compact and more promising from a computational point of view. The proposed mixed-integer linear program outperforms the two-stage approach, based on the computational study conducted on the realistically-sized problem instances. In particular, according to the values of the objective function, our MILP can find a better solution for 81% of the problem instances. We can show that there is no specific correlation between the number of operations or the number of underground locations in a problem instance and the performance of the considered approaches.

Further research

on the tactical planning level will consider the uncertainties in terms of the calculated processing times for removing a block. Furthermore, metaheuristics should be proposed to improve the results or observe some other aspects in an underground potash mine. After extracting, the material does not have to be immediately processed in above-ground processing plants. The storage of the material in a bunker can change the quality value of the output. That point can be considered to generate more realistic solutions.

on the tactical-operational planning level will consider machine unavailability according to the machine breakdowns. In that case, a possible rescheduling must be taken into account. Moreover, alternative objective functions related to the objective functions on the other planning levels can be investigated. Finally, a metaheuristic can be used to seek the neighborhood of the solutions achieved by the presented procedure for better results.

on the operational planning level will develop reasonable lower bounds to evaluate the results achieved by the proposed solution approaches. In addition to an even progress, the total output is a crucial factor for a mining company. Suppose a solution is optimal regarding the described objective function. There may exist some opportunities to improve the total output after achieving the prescribed target values within a work shift. In this regard, a heuristic solution approach can be implemented to consider a possible improvement in the total output without changing the current objective value.

Bibliography

Bjørndal, T., Herrero, I., Newman, A., Romero, C., and Weintraub, A. (2012). Operations research in the natural resource industry. *International Transactions in Operational Research*, 19(1-2):39–62.

Chesworth, W. (2008). *Encyclopedia of Soil Science*. Springer, Dordrecht.

Clausen, E. (2013). *Konzept für einen integrierten Produktionssteuerungsansatz bei Anwendung eines Örterbaus*. PhD thesis, Clausthal University of Technology. Papierflieger, Clausthal-Zellerfeld.

Gregory, C. (1980). *A Concise History of Mining*. Pergamon Press, Oxford, UK.

Hamrin, H. (2001). Underground mining methods and applications. In Hustrulid, W. A. and Bullock, R. L., editors, *Underground Mining Methods: engineering Fundamentals and International Case Studies*, pages 3–14. SME, Littleton.

Heinz, A. and von der Osten, R. (1982). *ABC Kali und Steinsalz*. Deutscher Verlag für Grundstoffindustrie, Leipzig.

K+S AG (2013 (accessed 13-Feb-2021)). A Day in the Mine. http://www.k-plus-s.com/en/ks-zum-anfassen/tag-in-der-grube.html.

Leal Gomes Leite, J. M., Arruda, E. F., Bahiense, L., and Marujo, L. G. (2020). Modeling the integrated mine-to-client supply chain: a survey. *International Journal of Mining, Reclamation and Environment*, 34(4):247–293.

Lerchs, H. and Grossmann, I. (1965). Optimum design of open-pit mines. *Canadian Institute of Mining Bulletin*, 58:47–54.

Musingwini, C. (2016). Optimization in underground mine planning-developments and opportunities. *Journal of the Southern African Institute of Mining and Metallurgy*, 116(9):809–820.

Newman, A., Rubio, E., Caro, R., Weintraub, A., and Eurek, K. (2010). A review of operations research in mine planning. *Interfaces*, 40(3):222–245.

O'Sullivan, D., Brickey, A., and Newman, A. (2015). Is openpit production scheduling easier than its underground counterpart? *Mining Engineering*, 67(4):68–73.

Ribas, I., Leisten, R., and Framiñan, J. M. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research*, 37(8):1439–1454.

Ruiz, R. and Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research*, 205(1):1–18.

Schulze, M. (2016). *Ein hierarchischer Ansatz zur Lösung von Ablaufplanungsproblemen im Bergbau: Darstellung am Beispiel des Örterbaus*. PhD thesis, Clausthal University of Technology. Shaker, Aachen.

Schulze, M., Rieck, J., Seifi, C., and Zimmermann, J. (2016). Machine scheduling in underground mining: an application in the potash industry. *OR Spectrum*, 38(2):365–403.

Schulze, M. and Zimmermann, J. (2017). Staff and machine shift scheduling in a German potash mine. *Journal of Scheduling*, 20(6):635–656.

Seifi, C., Schulze, M., and Zimmermann, J. (2019). A two-stage solution approach for a shift scheduling problem with a simultaneous assignment of machines and workers. In Mueller, C., Assibey-Bonsu, W., Baafi, E., Dauber, C., Doran, C., Jerzy Jaszczuk, M., and Nagovitsyn, O., editors, *Mining goes Digital: proceedings of the 39th International Symposium Application of Computers and Operations Research in the Mineral Industry (APCOM). June, 2019, Wroclaw, Poland*, pages 377–385. Taylor & Francis Group, London.

Seifi, C., Schulze, M., and Zimmermann, J. (2021a). A new mathematical formulation for a potash-mine shift scheduling problem with a simultaneous assignment of machines and workers. *European Journal of Operational Research*, 292(1):27–42.

Seifi, C., Schulze, M., and Zimmermann, J. (2021b). Solution procedures for block selection and sequencing in flat-bedded potash underground mines. *OR Spectrum*. *DOI: https://doi.org/10.1007/s00291-021-00618-z*.

USGS (2011). *Metals and Minerals: united States Geological Survey Minerals Yearbook*. United States Government Printing Office, Washington.

Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., and De Boeck, L. (2013). Personnel scheduling: A literature review. *European Journal of Operational Research*, 226(3):367–385.

A. Paper I

OR Spectrum
https://doi.org/10.1007/s00291-021-00618-z

REGULAR ARTICLE

Solution procedures for block selection and sequencing in flat-bedded potash underground mines

Cinna Seifi[1] · **Marco Schulze**[2] · **Jürgen Zimmermann**[1]

Received: 28 June 2019 / Accepted: 25 January 2021

Abstract

Phosphates, and especially potash, play an essential role in the increase in crop yields. Potash is mined in Germany in underground mines using a conventional drill-and-blast technique. The most commercially valuable mineral contained in potash is the potassium chloride that is separated from the potash in aboveground processing plants. The processing plants perform economically best if the amount of potassium contained in the output is equal to a specific value, the so-called optimal operating point. Therefore, quality-oriented extraction plays a decisive role in reducing processing costs. In this paper, we mathematically formulate a block selection and sequencing problem with a quality-oriented objective function that aims at an even extraction of potash regarding the potassium content. We, thereby, have to observe some precedence relations, maximum and minimum limits of the output, and a quality tolerance range within a given planning horizon. We model the problem as a mixed-integer nonlinear program which is then linearized. We show that our problem is $\mathcal{NP}$-hard in the strong sense with the result that a MILP-solver cannot find feasible solutions for the most challenging problem instances at hand. Accordingly, we develop a problem-specific constructive heuristic that finds feasible solutions for each of our test instances. A comprehensive experimental performance analysis shows that a sophisticated combination of the proposed heuristic with the mathematical program improves the feasible solutions achieved by the heuristic, on average, by 92.5%.

Keywords Underground mining · Block selection and sequencing · Quality objective · Mixed-integer linear programming · Priority rule-based procedure

✉ Cinna Seifi
cinna.seifi@tu-clausthal.de

[1] Institute of Management and Economics, Operations Research Group, Clausthal University of Technology, 38678 Clausthal-Zellerfeld, Germany

[2] K+S Aktiengesellschaft, 34131 Kassel, Germany

Published online: 06 March 2021

1 Introduction

This paper considers one of the biggest German potash mines. In Germany, the potash ores are generally found in deep deposits. Hence, potash mines are typically underground mines with an area-wide expansion of the deposit, a so-called flat-bedded deposit. According to Musingwini (2016), optimization in underground mine planning is not as well developed and widely applied as in open pit mine planning, although the logic of the planning is the same. O'Sullivan et al. (2015) state by comparing the common mathematical formulations for both open-pit and underground mining that scheduling in underground mines is more complex than the scheduling in the open-pit mines based on the complex structure of precedence constraints, the characteristics of the operations and activities, and the irregularity of the size and shape of the blocks. Musingwini indicates that the main reason for the complexity difference between open-pit and underground mine planning is that the direction of mining in open-pit mines is essentially down and outward to the pit limits. However, in underground mines, there are numerous permutations of the direction of mining depending on the mining method chosen.

The mostly applied extraction method for flat-bedded deposits of limited thickness is a drill-and-blast technique using the room-and-pillar mining method. By the use of the room-and-pillar mining method, the material is extracted across a horizontal plane, and pillars, arranged in regular patterns, are left for support purposes. Thus, a grid-like structure is formed, as demonstrated in Fig. 1 (Hamrin 2001; Schulze et al. 2016).

By employing a conventional drill-and-blast technique, the mining activities are conducted at the salt faces (cf. Fig. 1). At each salt face, the following discrete steps (mining operations) must be processed in chronological order (K+S 2013):

1. scaling the mine roof and sidewalls,
2. removing the scaled material,
3. bolting the roof with anchors,
4. drilling large diameter boreholes,
5. removing drilling dust,

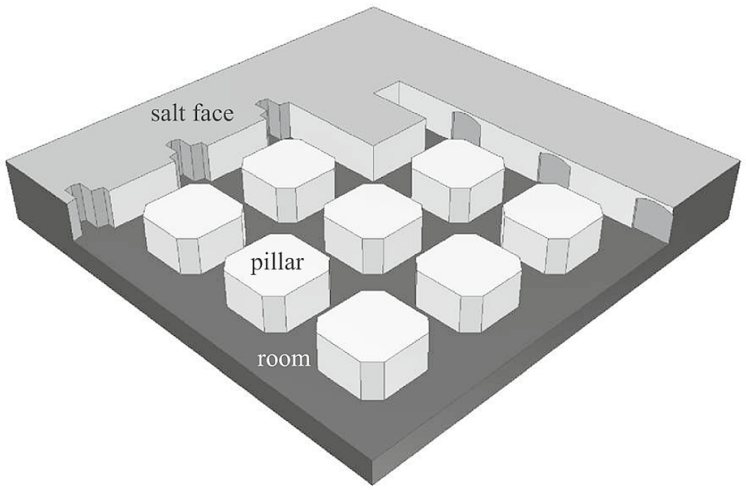

Fig. 1 Grid structure caused by the room-and-pillar mining method. Reprinted from Schulze et al. (2016)

6. drilling blast holes,
7. filling blast holes with explosive substances,
8. blasting,
9. transporting broken material to a feeder breaker.

During mining operation 4 (see the enumerated list above), large drill jumbos drill three adjacent horizontal boreholes with a diameter of 0.28 m and a length of 7 m at each salt face. The large boreholes act in particular as a direction guideline for blasting. After the detonation (mining operation 8), chambers, so-called *rooms*, are created in the direction of the mining activity (cf. Fig. 1). The height and the width of the exploded area are then scaled based on a given plan. Thus, after each detonation, a three-dimensional *block* of potash is removed, and a new room of the same size arises. During mining operation 9, the raw material is delivered using loaders from each salt face to a feeder breaker, where the lumps are broken. After completing mining operations 1 to 9, the position of the current salt face is shifted by the respective block's length, exposing a new salt face, which can be operated on directly afterward.

The onward transportation of crude materials from feeder breakers takes place using a conveyor belt system to a bunker near the shaft. The excavated crushed potash is transported from the bunker through the shaft to the surface. The potash ores are rich in potassium chloride, sodium chloride, and different associated minerals. Potassium chloride is a valuable salt that is mainly used as a fertilizer. Furthermore, it is an integral additive in the chemical, medical as well as human and animal food-processing industry (Chesworth 2008; USGS 2011; Schulze et al. 2016). After transporting the crude salt to the surface, potassium chloride is separated from the extracted potash by flotation, recrystallization, or electrostatic separation in aboveground processing plants. For each technical device, there is an *optimal operating point* at which the device has the best performance. This point can be determined based on various factors. The processing plants above ground can be most cost-effectively utilized if the amount of potassium contained in the extracted material is equal to a specific value. The equivalent content of potassium oxide is often reported to indicate the percentage of potassium by weight in the potassium chloride, where 100% potassium chloride is precisely equal to 63.17% potassium oxide (cf. Heinz and von der Osten 1982, p. 147). As mentioned, at a salt face, a block of potash can be unearthed. For each block, the amount of potash is measured according to the dimensions of the excavation. Moreover, the potassium contained in each block of potash *in percent* is determined based on geological investigations. The percentage of potassium contained in the extracted potash from a block is defined as the *quality value* of the corresponding block. Accordingly, the quality value of a block multiplied by the amount of potash obtained from that block determines the amount of potassium contained. In general, the blocks are different regarding the amount and the quality value of the potash contained. Moreover, not all the available salt faces and the corresponding blocks can be mined within a given planning horizon. Therefore, the quality of the amount of potash extracted within different time intervals

strongly depends on the selected blocks and the sequence of the extraction. Vast fluctuations in the quality value result in non-homogeneous output that leads to high costs for aboveground processing. Because quality fluctuations occur frequently owing to the way in which the potash is extracted, quality-oriented mining of blocks plays a decisive role in reducing the processing costs.

Newman et al. (2010) classify the existing approaches and models in mining companies according to the planning horizon or the hierarchy level into long-term (strategic), medium-term (tactical), and medium- and short-term (tactical-operative) problems. In this paper, we consider a medium-term planning horizon. Within the planning horizon, we want to have a homogeneous output regarding the quality value of the extracted potash. More precisely, the quality value of the extracted material should deviate as little as possible from a given quality target value so that the mineral processing above ground is conducted economically. In this regard, we want to answer two questions: 1. which block should be excavated (i.e., block selection), and 2. if a block is extracted, the time at which it is extracted (i.e., block sequencing). Taken together, we solve a block selection and sequencing problem with a quality-oriented objective function at a tactical planning level. For our problem, precedence relations, maximum and minimum limits of the output, and a quality tolerance range have to be taken into account.

The remainder of the paper is organized as follows: In Sect. 2, the characterization of the problem at hand and a literature review are given. Sect. 3 introduces a mathematical formulation for our problem. Since some constraints are not linear, more decision variables and constraints are introduced to linearize the mathematical program. Subsequently, we show that our problem is $\mathcal{NP}$-hard in the strong sense. Accordingly, in Sect. 4, we devise a constructive heuristic, which is embedded in a multi-start environment and provides good, feasible solutions for the problem through a sophisticated time-saving procedure. In Sect. 5, based on some generated realistic problem instances, we compare the introduced mathematical program and the proposed constructive heuristic. We also show if we solve our problem heuristically and use the feasible solution found as an initial solution for the mathematical program, we obtain much better results, especially for large and challenging problem instances. The paper concludes with a summary of the achieved results and an outlook on future research in Sect. 6.

2 Problem specification and related literature

For a proper operation, from a geographical point of view, underground mines are usually divided into smaller spatial areas, so-called *mining districts*. Accordingly, an underground potash mine has, on average, up to 5 mining districts. The area of a mining district may be several square kilometers. Due to this spatial expansion, several *tipple areas* are constructed for a mining district to divide the area into smaller parts avoiding long transportation routes. In each mining district, depending on its area, 4 to 6 tipple areas are involved. In a tipple area, a feeder breaker is installed, where the lumps of the extracted potash in that tipple area are delivered to and crushed. As mentioned, the mining operations are conducted at a salt face.

Subsequently, the detonation occurs in the mining direction, in which a block of potash is extracted. After mining operation 8 (blasting), a three-dimensional block of potash with the known dimensions is removed, and a room is created. A chain of consecutive blocks that are extracted one after another in a certain mining direction is defined as an *underground location*. Since a tipple area can have an extent of up to a few 100 meters, several underground locations (usually between 5 and 11) are assigned to the single tipple areas. That means the blocks of material extracted from each underground location are transported to the feeder breaker installed in the corresponding tipple area.

Figure 2 illustrates a tipple area with three underground locations (UL 1 to UL 3) that are assigned to a feeder breaker, i.e., the potash unearthed from UL 1 to UL 3 is transported to the illustrated feeder breaker. In UL 1, three blocks are already removed in the mining direction. The associated rooms are designated by squares with solid lines. After mining a block, a new salt face (a potential block) in the mining direction becomes available. Geological sampling and mining investigations determine how many of the consecutive blocks in an underground location can be removed within a considered planning horizon. In Fig. 2, the dashed squares in UL 1 and UL 2 indicate the blocks that can be mined according to the plan for the considered time horizon. On the contrary, no further block can be removed in UL 3

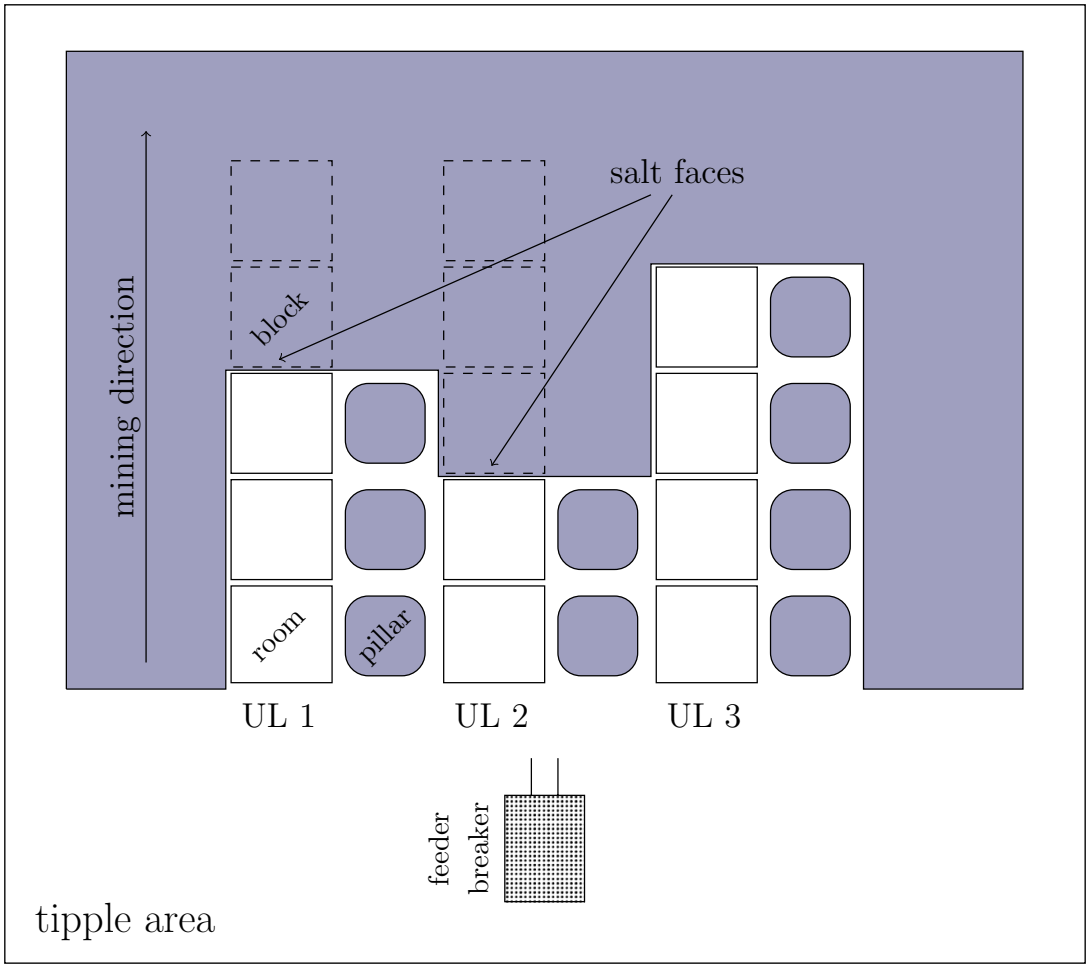

Fig. 2 A tipple area with associated underground locations. Adapted from Clausen (2013), p. 23

Table 1 Definitions of the mining terminology introduced

Term	Definition
Block	A cube of material with known dimensions removed after a detonation
Feeder breaker	A crushing machine in a tipple area where the lumps extracted from the underground locations assigned to this tipple area are delivered to
Mining district	The largest unit of an underground mine that comprises some smaller units (see the definition of a tipple area)
Pillars	Parts of underground mines that are not extracted to support the roof from collapsing
Room	A space of known dimensions created after a detonation
Salt face	A place at which the mining operations are conducted, i.e., it is the front side of the block that is extracted
Tipple area	The largest unit of a mining district that is characterized by the assigned underground locations and a feeder breaker
Underground location	A chain of consecutive blocks that can be removed in the mining direction

within the planning horizon. For better clarity, we summarize in Table 1 the aforementioned mining terms.

The excavation of a block can only then be started if the previous blocks in the same underground location have been fully excavated. A block has, at most, one direct predecessor and, at most, one direct successor block in the corresponding underground location. The time needed to remove a block is the sum of the processing times of the required mining operations 1 to 9. Each mining operation must be processed by one machine and by one skilled worker. According to the speed of the assigned machine and the skill level of the assigned worker, different processing times are needed to accomplish a mining operation. Machines and workers are scheduled at the beginning of each work shift at an operational planning level (see, e.g., Schulze and Zimmermann 2017; Seifi et al. 2019, and Seifi et al. 2020). Since we deal with a block sequencing problem at a tactical planning level, the processing times required for the extraction of blocks must be estimated. In doing so, the dimensions or shapes of blocks are crucial factors. Moreover, the current status of a block at the beginning of the planning horizon affects the needed processing time. For example, a block for which mining operations 3–9 must be carried out has a shorter processing time than a block for which all mining operations 1–9 must still be processed.

In every tipple area, the extracted material is initially carried out via the loaders from underground locations to the assigned feeder breaker. The conveyance of the extracted potash from the particular tipple areas of each mining district takes place through a conveyor belt system to a central bunker system close to the shaft, from where the material ultimately reaches the surface. The capacities of the conveyor system, the bunker, and the processing plants above ground are limited. Hence, in each work shift, an upper limit of the output for each tipple area and each mining district must be observed. On the other hand, the primary task of mining companies is the extraction of raw minerals. Accordingly, a lower

limit for the total output over the planning horizon must be considered. Moreover, a minimum output in each work shift and every mining district must be satisfied.

In our problem, we consider a planning horizon (e.g., a month) that is a union of some smaller sub-intervals (e.g., weeks). Within those sub-intervals, the quality value of the entire extracted potash must be within a given quality tolerance range. The assumptions regarding the length of the planning horizon and the corresponding sub-intervals can be customized according to the current situation of the mine at hand. Assume that a set of blocks is excavated within a specific time interval. The quality value of the entire extracted material within that time interval is the weighted average of the quality values of the removed blocks. Due to our objective, the quality value of the extracted material should deviate as little as possible from a given quality target value. The quality target value in percent typically represents the optimal operating point of the processing plant above ground. We speak of a negative deviation if the weighted average of the quality of the extracted material is less than the quality target value. Analogously, there is a positive deviation if the weighted average of the quality of the extracted material is greater than the quality target value. For formulating the objective function, we first determine the absolute deviation's value of the weighted average of the output quality from the predetermined quality target value in each considered sub-interval in the planning horizon. The value of the weighted average of the output quality and, thus, the values of negative and positive deviations are determined based on the amount of material extracted within the considered time interval. Since the amount of material removed is not known a priori, determining the deviations from the given target value requires some nonlinear constraints in the mathematical formulation that must be linearized. The aim is then to minimize the average of the calculated deviations over the number of sub-intervals considered in the planning horizon.

Altogether, we minimize a quality-oriented objective function observing the following groups of constraints:

Precedence relations between the blocks in an underground location;
Minimum limit of the output over the entire planning horizon, in each work shift, and for every mining district;
Maximum limit of the output for each tipple area and every mining district within every single work shift; and
Quality tolerance range over each certain sub-interval in the planning horizon.

Newman et al. (2010), Kozan and Liu (2011), as well as Blom et al. (2019) published survey articles on the application of operations research methods in the field of mining. Lately, Leite et al. (2020) gave a review on state-of-the-art applications of operational research techniques to mining problems taking the mentioned surveys into account. Leite et al. (2020) consider (1) layout and design problems, (2) production and scheduling problems, and (3) operational equipment allocation problems at strategic, tactical, and operational planning levels, respectively; consequently, they review the published articles in both open-pit and underground mines.

For more convenience, in Table 2, we list the most significant works published in the previous decade that introduce a mathematical formulation for a block

Table 2 Literature on block selection and sequencing problems

	Constraints			OF
	Quantity	Quality	Precedence	
Bley et al. (2010)	•	–	•	M
Nehring et al. (2010)	•	•	•	M
Martinez and Newman (2011)	•	–	•	Q
Askari-Nasab et al. (2011)	•	•	•	M
Chicoisne et al. (2012)	•	–	•	M
Nehring et al. (2012)	•	•	•	M
Ramazan and Dimitrakopoulos (2013)	•	•	•	M
Clausen (2013)	•	•	•	Q
Espinoza et al. (2013)	•	•	•	M
Smith and Wicks (2014)	•	•	•	Q
Lambert and Newman (2014)	•	–	•	M
O'Sullivan and Newman (2015)	•	•	•	Q
Montiel and Dimitrakopoulos (2015)	•	•	•	M
Lamghari and Dimitrakopoulos (2016)	•	–	•	M
Mousavi et al. (2016)	•	•	•	M
Jélvez et al. (2016)	•	–	•	M
Blom et al. (2016)	•	•	•	Other
Liu and Kozan (2016)	•	–	•	M
Vossen et al. (2016)	•	–	•	M
King et al. (2017)	•	–	•	M
Samavati et al. (2017)	•	–	•	M
Azzamouri et al. (2018)	•	•	•	Q
Reus et al. (2018)	•	–	•	M
Samavati et al. (2018)	•	–	•	M
Mousavi and Sellers (2019)	•	•	•	M
Mai et al. (2019)	•	•	•	M
Elsayed et al. (2020)	•	–	•	M
Jélvez et al.(2020)	•	•	•	M
Campeau and Gamache (2020)	–	–	•	M
Rivera Letelier et al. (2020)	•	•	•	M

•, constraints for the group are considered; –, constraints for the group are not considered

scheduling problem in the field of mining. Under columns "Constraints," we observe the three types of constraints we deal with in the problem at hand (*Quality*, *Quantity*, and *Precedence*). The "OF" column specifies whether the objective function is quantity- ("Q") or monetary-related ("M").

Regarding our quality-related objective function, we first have to calculate the weighted average of the output quality. Assume that a set of blocks $\mathcal{B}$ is available to be extracted. Let A_b and Q_b be the amount of material (in tonnes) and the

quality value (in percent) of block $b \in \mathcal{B}$, respectively. Moreover, let x_b be the binary decision variable that is 1 if block b is excavated. Then, the quality value of the entire extracted material is the weighted average of the quality values of the removed blocks, denoted by $\overline{q}$. The value of $\overline{q}$ (in %) is calculated as follows:

$$\overline{q} = \frac{\sum_{b \in \mathcal{B}} Q_b \cdot A_b \cdot x_b}{\sum_{b \in \mathcal{B}} A_b \cdot x_b}.$$

In the literature, the average proportion of the most valuable mineral contained in the ore is indicated as "ore grade." Martinez and Newman (2011) formulate a mixed-integer linear program to minimize the deviations from a given target demand for three different ore grades in LKAB's Kiruna iron ore mine. However, the target demand for a particular ore grade is given in tonnes. Accordingly, expressed in our notation, they only consider the linear term $\sum_{b \in \mathcal{B}} Q_b \cdot A_b \cdot x_b$ (the amount of extracted iron in tonnes) to measure the deviations. Thus, we categorized the proposed objective function as quantity-related. Azzamouri et al. (2018) minimize the deviations from the demand for a certain grade of the extracted material, where the objective function and the related constraints are formulated quantity-oriented, too. Analogously, Clausen (2013) determines the amount of potassium oxide contained in the extracted potash in tonnes to minimize the deviations from a given target value in an underground potash mine.

In the literature, the quality-related constraints are mostly known as "grade control constraints" or "grade blending constraints." Those constraints ensure that $\overline{q}$ is within a permitted tolerance range $[Q^-, Q^+]$. Thus, the following inequalities must apply:

$$Q^- \leq \frac{\sum_{b \in \mathcal{B}} Q_b \cdot A_b \cdot x_b}{\sum_{b \in \mathcal{B}} A_b \cdot x_b} \leq Q^+.$$

If we multiply both sides of the above inequalities with the denominator of $\overline{q}$, we obtain linear constraints (see, e.g., Rivera Letelier et al. 2020). Except for Montiel and Dimitrakopoulos (2015) and Blom et al. (2016), all the quality-related constraints observed in the papers from Table 2 are linear grade control constraints. Montiel and Dimitrakopoulos (2015) maximize discounted profits in an open-pit copper mine. They penalize the deviations regarding mining, processing, transportation, and blending targets and consider a penalty cost in the objective function. In the problem they consider, multiple material types are sent to the available processes or to stockpiles where they are blended to meet the quality requirements. The grade of the material handled in a given period corresponds to the grade of the stockpiles. Montiel and Dimitrakopoulos emphasize in their work that the corresponding blending constraint is nonlinear; consequently, the authors propose a risk-based heuristic approach to tackle the problem without suggesting any linear mathematical formulation. Blom et al. (2016) consider a multiple mine, multiple time-period, open-pit production scheduling problem. The authors define the productivity of a mine in terms of the desirable utilization of dig and trucking resources, i.e., dig and trucking resources should be fully utilized. In each period, ore produced at each mine

is transported by rail to a set of ports and blended into products for shipping. The objective function minimizes the deviation between the composition of port products and desired bounds and maximizes the productivity achieved at each mine. The observed objective function is denoted by "Other" in Table 2. The authors propose a nonlinear mathematical program to ensure blending constraints at each port while generating schedules for each mine. To tackle the problem, Blom et al. suggest a decomposition-based algorithm that can find high-quality solutions.

Our literature review suggests that no one, to the best of our knowledge, has introduced a linearization of the nonlinear quality-related constraints taking a quality-oriented objective function into account. It is straightforward to show that the scheduling problems observing upper and lower limits for the operational resources are $\mathcal{NP}$-hard. The computational time required for solving $\mathcal{NP}$-hard problems may be affected by the numerical parameters of the input data. A $\mathcal{NP}$-hard problem **in the strong sense** is still $\mathcal{NP}$-hard even when all of its numerical parameters are bounded by a polynomial in the length of the input. The contributions of this paper are:

- to introduce a **linear** mathematical program for the problem described;
- to show that the problem at hand is $\mathcal{NP}$-hard **in the strong sense**; and
- to introduce a solution approach that provides high-quality solutions for realistically sized problem instances.

In the next section, we first introduce the original mathematical program in its nonlinear structure and then linearize the corresponding nonlinear constraints. Lastly, we show that our problem is $\mathcal{NP}$-hard in the strong sense.

3 Mathematical model

In this section, we introduce a linear mathematical program for the block selection and sequencing problem described in Sect. 2. From an operational point of view, we consider $\tau_{\max}$ sub-intervals in the given planning horizon. Each sub-interval τ consists of several periods, where each period represents one work shift in the corresponding sub-interval. Each work shift is represented by time interval $(t-1, t]$. Let $T_{\max}$ denote the number of work shifts in the planning horizon under consideration. Thus, a planning horizon of $T_{\max}$ work shifts is a union of time intervals $(t-1, t]$ for $t = 1, 2, \ldots, T_{\max}$ and the point in time 0. Our mathematical formulation is based on the discrete completion times for the extraction of the blocks that are selected and mined in the considered planning horizon. Accordingly, we only focus on the discrete points in time that represent the end times of work shifts in the planning horizon. Note that a completion time at point 0 is not relevant since the point in time 0 is not the end time of any work shift in the planning horizon under consideration. Let $\mathcal{T}$ be the set of positive, discrete points in time in the entire planning horizon. Moreover, let $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_{\tau_{\max}}$ be the disjoint subsets of $\mathcal{T}$, where $\bigcup_{\tau \in \{1,\ldots,\tau_{\max}\}} \mathcal{T}_\tau = \mathcal{T}$. The elements of $\mathcal{T}_\tau$ are the positive, discrete points in time that represent the end times of the work shifts contained in sub-interval τ. In Fig. 3, a planning horizon

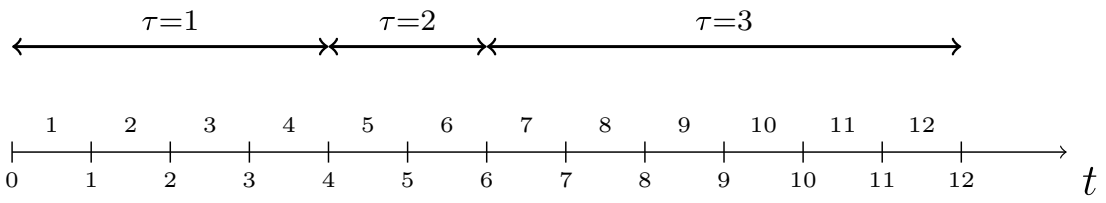

Fig. 3 The planning horizon and the associated sub-intervals

with $T_{\max} = 12$ work shifts and $\tau_{\max} = 3$ sub-intervals is depicted. The whole planning horizon is a union of disjoint time intervals [0, 4], (4, 6], and (6, 12]. Since the point in time 0 is not the end time of any work shift in $\mathcal{T}$, we have $\mathcal{T} = \{1, 2, \ldots, 12\}$, where $\mathcal{T}_1 = \{1, 2, 3, 4\}$, $\mathcal{T}_2 = \{5, 6\}$, and $\mathcal{T}_3 = \{7, 8, \ldots, 12\}$.

Table 3 demonstrates the sets, parameters, and decision variables used in the mathematical program. Note that the auxiliary decision variable $\overline{q}_\tau$ is used for better clarity in the mathematical formulation and is calculated by definition as follows:

$$\overline{q}_\tau = \frac{\sum_{t\in\mathcal{T}_\tau}\sum_{b\in\mathcal{B}} Q_b \cdot A_b \cdot x_{bt}}{\sum_{t\in\mathcal{T}_\tau}\sum_{b\in\mathcal{B}} A_b \cdot x_{bt}} \quad \forall \tau \in \{1, \ldots, \tau_{\max}\}.$$

The mathematical program for our problem is formulated as follows:

$$\text{minimize} \qquad \frac{1}{\tau_{\max}} \sum_{\tau\in\{1,\ldots,\tau_{\max}\}} (\delta_\tau^+ + \delta_\tau^-) \tag{1}$$

subject to

$$\sum_{t\in\mathcal{T}} x_{bt} \le 1 \quad \forall b \in \mathcal{B} \tag{2}$$

$$\sum_{t\in\mathcal{T}} t \cdot x_{bt} = c_b \quad \forall b \in \mathcal{B} \tag{3}$$

$$Z_b \le c_b + M \cdot (1 - \sum_{t\in\mathcal{T}} x_{bt}) \quad \forall b \in \mathcal{B} \tag{4}$$

$$\sum_{t\in\mathcal{T}} x_{lt} \le \sum_{t\in\mathcal{T}} x_{bt} \quad \forall (b, l) \in \mathcal{V} \tag{5}$$

$$c_b \le c_l - Z_l \cdot \left(\sum_{t\in\mathcal{T}} x_{lt}\right) + M \cdot (1 - \sum_{t\in\mathcal{T}} x_{lt}) \quad \forall (b, l) \in \mathcal{V} \tag{6}$$

Table 3 Sets, parameters, and decision variables used in the mathematical program

Sets	
$\mathcal{B}$	Set of blocks in an underground mine
$\mathcal{B}_k$	Set of blocks assigned to tipple area $k \in \mathcal{K}_r$
$\mathcal{B}_r$	Set of blocks assigned to mining district $r \in \mathcal{R}$
$\mathcal{K}_r$	Set of tipple areas in mining district $r \in \mathcal{R}$
$\mathcal{R}$	Set of mining districts in an underground mine
$\mathcal{T}$	Set of positive, discrete points in time in the given planning horizon
$\mathcal{T}_\tau$	Set of positive, discrete points in time contained in sub-interval τ ($\mathcal{T}_\tau \subset \mathcal{T}$)
$\mathcal{V}$	Set of precedence relations between blocks with each element representing an ordered pair of blocks; if $(b, l) \in \mathcal{V} : b, l \in \mathcal{B}$, b must be completed before l can be started
Parameters	
A_b	Amount of potash contained in block b in tonnes
M	A constant number with a sufficiently large value (big-M)
P^-	Minimum output that must be achieved over the planning horizon
P_r^-	Minimum output that must be achieved for mining district r over the planning horizon
P_t^-	Minimum output that must be achieved within time interval $(t-1, t]$
P_{kt}^+	Upper limit of the output for tipple area k within time interval $(t-1, t]$
P_{rt}^+	Upper limit of the output for mining district r within time interval $(t-1, t]$
Q_b	Percentage of potassium contained in the extracted material from block b (quality value of block b)
Q^-	Lower limit of the quality tolerance range in %
Q^+	Upper limit of the quality tolerance range in %
Q_τ	Quality target value within sub-interval τ in %
$T_{\max}$	Number of work shifts considered in the planning horizon
$\tau_{\max}$	Number of sub-intervals considered in the planning horizon
Z_b	Processing time required to extract block b measured in work shifts
Decision variables	
c_b	Positive continuous decision variable; completion time of block b
δ_τ^-	Positive continuous decision variable; negative deviations of the output quality from the quality target value over sub-interval τ
δ_τ^+	Positive continuous decision variable; positive deviations of the output quality from the quality target value over sub-interval τ
x_{bt}	Binary decision variable; 1, if the excavation of block b is completed in time interval $(t-1, t]$, and the material removed is available at point in time t; 0, otherwise
Auxiliary decision variables	
θ_{bt}^-	Positive continuous decision variable that substitutes the product of decision variables x_{bt} and δ_τ^-
θ_{bt}^+	Positive continuous decision variable that substitutes the product of decision variables x_{bt} and δ_τ^+
$\overline{q}_\tau$	Weighted average of the quality value of the extracted blocks during sub-interval τ

$$\sum_{t\in\mathcal{T}}\sum_{b\in\mathcal{B}} A_b \cdot x_{bt} \geq P^- \tag{7}$$

$$\sum_{b\in\mathcal{B}} A_b \cdot x_{bt} \geq P_t^- \quad \forall t \in \mathcal{T} \tag{8}$$

$$\sum_{t\in\mathcal{T}}\sum_{b\in\mathcal{B}_r} A_b \cdot x_{bt} \geq P_r^- \quad \forall r \in \mathcal{R} \tag{9}$$

$$\sum_{b\in\mathcal{B}_r} A_b \cdot x_{bt} \leq P_{rt}^+ \quad \forall r \in \mathcal{R}, \forall t \in \mathcal{T} \tag{10}$$

$$\sum_{b\in\mathcal{B}_k} A_b \cdot x_{bt} \leq P_{kt}^+ \quad \forall k \in \mathcal{K}_r : r \in \mathcal{R}, \forall t \in \mathcal{T} \tag{11}$$

$$\sum_{t\in\mathcal{T}_\tau}\sum_{b\in\mathcal{B}} (Q_b - Q^-) \cdot A_b \cdot x_{bt} \geq 0 \quad \forall \tau \in \{1, \ldots, \tau_{\max}\} \tag{12}$$

$$\sum_{t\in\mathcal{T}_\tau}\sum_{b\in\mathcal{B}} (Q^+ - Q_b) \cdot A_b \cdot x_{bt} \geq 0 \quad \forall \tau \in \{1, \ldots, \tau_{\max}\} \tag{13}$$

$$\overline{q}_\tau - Q_\tau \leq \delta_\tau^+ \quad \forall \tau \in \{1, \ldots, \tau_{\max}\} \tag{14}$$

$$Q_\tau - \overline{q}_\tau \leq \delta_\tau^- \quad \forall \tau \in \{1, \ldots, \tau_{\max}\} \tag{15}$$

$$x_{bt} \in \{0, 1\} \quad \forall b \in \mathcal{B}, \quad \forall t \in \mathcal{T} \tag{16}$$

$$c_b \geq 0 \quad \forall b \in \mathcal{B} \tag{17}$$

$$\delta_\tau^+, \delta_\tau^- \geq 0 \quad \forall \tau \in \{1, \ldots, \tau_{\max}\} \tag{18}$$

We minimize the average of the positive and negative deviations of the quality of the output from a given quality target value over the predetermined sub-intervals (cf. objective function (1)). A block can be excavated at most once within the entire planning horizon (constraint set (2)). By definition of decision variable x_{bt}, if $x_{bt} = 1$, the point in time t represents the completion time of block b. Constraint set (3) determines the completion times of blocks. Note that if a block is not excavated, the completion time is 0. Constraint set (4) guarantees that the completion time of a block must be greater than or equal to its processing time, i.e., the mining of a block must start during the considered planning horizon. Constraint set (4) is active only if a block is excavated. In the corresponding big-M formulation, we can choose M equal to $T_{\max}$. Constraint sets (5) and (6) observe a precedence relation

between blocks b and l. If the ordered pair (b, l) is in $\mathcal{V}$, block l cannot be mined if block b is not excavated (constraint set (5)). Moreover, constraint set (6) ensures that the completion time of block l must be at least by Z_l time units greater than the completion time of block b if block l is ever processed. Constraint sets (7) to (11) ensure the minimum and maximum limits of the output. Constraint sets (12) and (13) guarantee that the weighted average of the quality value of the excavated blocks during every sub-interval ($\overline{q}_\tau$) is within a permitted tolerance range. Those constraints are like "grade control constraints" or "grade blending constraints" explained in Sect. 2.

Constraint sets (14) and (15) determine the lower bounds for δ_τ^+ and δ_τ^-, respectively. Since objective function (1) must be minimized, δ_τ^+ and δ_τ^- take the value of the positive and negative deviations of the quality of the output from Q_τ. However, the constraint sets are inherently nonlinear. We write the mathematical formula of $\overline{q}_\tau$ in the inequalities and reduce the left-hand side of each inequality to a common denominator. If we multiply both sides of the inequalities by the common denominator, we obtain the following nonlinear inequalities:

$$\sum_{t\in\mathcal{T}_\tau}\sum_{b\in\mathcal{B}}(Q_b - Q_\tau)\cdot A_b\cdot x_{bt} \le \sum_{t\in\mathcal{T}_\tau}\sum_{b\in\mathcal{B}} A_b\cdot x_{bt}\cdot \delta_\tau^+ \quad \forall\tau\in\{1,\ldots,\tau_{\max}\}$$

$$\sum_{t\in\mathcal{T}_\tau}\sum_{b\in\mathcal{B}}(Q_\tau - Q_b)\cdot A_b\cdot x_{bt} \le \sum_{t\in\mathcal{T}_\tau}\sum_{b\in\mathcal{B}} A_b\cdot x_{bt}\cdot \delta_\tau^- \quad \forall\tau\in\{1,\ldots,\tau_{\max}\}$$

In order to formulate a mixed-integer linear program, we introduce positive continuous decision variables θ_{bt}^+ and θ_{bt}^- to substitute the products $x_{bt}\cdot\delta_\tau^+$ and $x_{bt}\cdot\delta_\tau^-$ on the right-hand sides of the above inequalities, respectively. We, therefore, have the following constraint sets:

$$\sum_{t\in\mathcal{T}_\tau}\sum_{b\in\mathcal{B}}(Q_b - Q_\tau)\cdot A_b\cdot x_{bt} \le \sum_{t\in\mathcal{T}_\tau}\sum_{b\in\mathcal{B}} A_b\cdot \theta_{bt}^+ \quad \forall\tau\in\{1,\ldots,\tau_{\max}\} \tag{14-1}$$

$$\sum_{t\in\mathcal{T}_\tau}\sum_{b\in\mathcal{B}}(Q_\tau - Q_b)\cdot A_b\cdot x_{bt} \le \sum_{t\in\mathcal{T}_\tau}\sum_{b\in\mathcal{B}} A_b\cdot \theta_{bt}^- \quad \forall\tau\in\{1,\ldots,\tau_{\max}\} \tag{15-1}$$

By substituting a product of a binary decision variable and a continuous decision variable, the upper bound for the substitution variable (here θ_{bt}^+ and θ_{bt}^-) must be determined. The upper bound is related to the maximum value that the continuous decision variable can take if the binary decision variable is 1. Furthermore, the substitution variable takes the value 0 if the binary decision variable is 0. Constraint sets (14-2) and (15-2) guarantee that decision variables θ_{bt}^+ and θ_{bt}^- take the value of zero for all $t\in\mathcal{T}_\tau$ if block b is not excavated within sub-interval τ. Note that each block can be mined only once within the entire planning horizon. Thus, it is sufficient if we consider the summation of decision variables θ_{bt}^+ (θ_{bt}^-) and x_{bt} over the points in time $t\in\mathcal{T}_\tau$. Otherwise, if a block is removed at any point in time $t\in\mathcal{T}_\tau$, the left-hand side of constraint set (14-2) (constraint set (15-2)) can at most have the value of $Q^+ - Q_\tau$ ($Q_\tau - Q^-$):

$$\sum_{t\in\mathcal{T}_\tau}\theta^+_{bt} \le (Q^+ - Q_\tau)\cdot\sum_{t\in\mathcal{T}_\tau}x_{bt} \quad \forall b\in\mathcal{B}, \forall\tau\in\{1,\ldots,\tau_{\max}\} \tag{14-2}$$

$$\sum_{t\in\mathcal{T}_\tau}\theta^-_{bt} \le (Q_\tau - Q^-)\cdot\sum_{t\in\mathcal{T}_\tau}x_{bt} \quad \forall b\in\mathcal{B}, \forall\tau\in\{1,\ldots,\tau_{\max}\} \tag{15-2}$$

Note that the maximum values of the positive deviation (δ^+_τ) and the negative deviation (δ^-_τ) are bounded by $Q^+ - Q_\tau$ and $Q_\tau - Q^-$, respectively. Subsequently, the following constraint sets determine the values of δ^+_τ and δ^-_τ that are used in (1):

$$\sum_{t\in\mathcal{T}_\tau}\theta^+_{bt} \le \delta^+_\tau \quad \forall b\in\mathcal{B}, \forall\tau\in\{1,\ldots,\tau_{\max}\} \tag{14-3}$$

$$\sum_{t\in\mathcal{T}_\tau}\theta^-_{bt} \le \delta^-_\tau \quad \forall b\in\mathcal{B}, \forall\tau\in\{1,\ldots,\tau_{\max}\} \tag{15-3}$$

If we replace constraint sets (14), as well as (15) by constraint sets (14-1), (14-2), and (14-3) as well as constraint sets (15-1), (15-2), and (15-3), respectively, we have a mixed-integer linear program, where constraint set (16) indicates that decision variables x_{bt} are binary, and in addition to non-negativity constraint sets (17) and (18), the following non-negativity constraint set must be considered:

$$\theta^+_{bt}, \theta^-_{bt} \ge 0 \quad \forall b\in\mathcal{B}, \forall t\in\mathcal{T} \tag{19}$$

We denote the proposed mixed-integer linear program for our **b**lock **s**election and **s**equencing **p**roblem with **BSSP**.

In what follows, we show that our block selection and sequencing problem is $\mathcal{NP}$ -hard in the strong sense. We introduce a restricted case of BSSP (RBSSP) to which a pseudo-polynomial transformation from the well-known 3-PARTITION problem can be easily constructed.

We consider an underground mine that has one mining district, over a planning horizon of $T_{\max} = T$ work shifts with $\tau_{\max} = 1$. Let the number of blocks be $n = 3T$. We assume that there is only one block in each underground location ($\mathcal{V} = \emptyset$). Let the processing time of all blocks be equal to 1 work shift. Hence, constraint sets (3) to (6) do not have to be observed. Moreover, we assume that the maximum output for each tipple area is a very large number with the result that constraint set (11) is always satisfied. Since there is only one mining district in the restricted problem, constraint sets (7) and (9) are the same. By assuming the minimum output for the entire planning horizon equal to $\sum_{t\in\mathcal{T}}P^-_t$, constraint sets (7) and (9) are redundant. Furthermore, let the quality value of all blocks be equal to Q with $Q^- < Q < Q^+$. Thus, the deviation from the quality target value is a constant number regardless of which blocks have been excavated. Consequently, all of the quality-related constraint sets are met. We assume $P^+_{1t} = P^-_t = (\sum_{b=1}^{3T}A_b)/T$. Hence, we can denote $P^+_{1t} = P^-_t$ with P. Consequently, the restricted problem, RBSSP, can be formulated as follows:

Min. Constant number

$$\text{s. t.} \quad \sum_{b=1}^{3T} x_{bt} \leq 1 \quad \forall t \in \{1, \ldots, T\} \qquad \text{(cf. constraint set (2))}$$

$$\sum_{b=1}^{3T} (A_b \cdot x_{bt}) \geq P \quad \forall t \in \{1, \ldots, T\} \qquad \text{(cf. constraint set (8))}$$

$$\sum_{b=1}^{3T} (A_b \cdot x_{bt}) \leq P \quad \forall t \in \{1, \ldots, T\} \qquad \text{(cf. constraint set (10))}$$

$$x_{bt} \in \{0, 1\} \qquad \forall b \in \{1, \ldots, 3T\},\ \forall t \in \{1, \ldots, T\}$$

RBSSP is not an optimization, but a so-called feasibility problem, i.e., a specific decision problem. In this restricted problem, we intend to determine T subsets $\mathcal{B}_1$, $\mathcal{B}_2$, ⋯, $\mathcal{B}_T$ of blocks that are removed at points in time $1, 2, \ldots, T$, respectively, subject to the constraint sets of the problem.

Garey and Johnson (1979) showed that 3-PARTITION is $\mathcal{NP}$-complete in the strong sense.

- **3-PARTITION**

  Input: a finite set $\mathcal{U} = \{u_1, u_2, \ldots, u_{3m}\}$, a bound $X \in \mathbb{N}$, and a size $s(u_b) \in \mathbb{N}$ for each $b = 1, \ldots, 3m$, such that $\frac{X}{4} < s(u_b) < \frac{X}{2}$ for each b and $\sum_{b=1}^{3m} s(u_b) = mX$.

  Question: are there m disjoint subsets $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_m$ of $\mathcal{U}$ such that:

$$\sum_{u_b \in \mathcal{U}_b} s(u_b) = X \qquad \forall b \in \{1, \ldots, m\}?$$

If we consider each element u_b of the given set $\mathcal{U}$ in 3-PARTITION as a block $b \in \mathcal{B}$ in RBSSP, a transformation from an arbitrary instance of 3-PARTITION to an instance of RBSSP is given by $T = m$, $A_b = s(u_b)$, and $P = X$. This transformation can be performed in time polynomially in the input length. The length of the constructed RBSSP-instance is polynomially related to the length of the given 3-PARTITION instance. Furthermore, the largest number in the constructed instance is

equal to the largest number of the given 3-PARTITION instance; consequently, the conditions of a pseudo-polynomial transformation are met. In a solution of RBSSP, we have T subsets of $\mathcal{B}$. Those subsets play the same role as the sets $\mathcal{U}_1, \ldots, \mathcal{U}_m$ in the desired partition of $\mathcal{U}$ in the 3-PARTITION. As a result, a solution for RBSSP exists if and only if the desired partition exists for the given 3-PARTITION instance. Thus, RBSSP is $\mathcal{NP}$-complete in the strong sense, and optimization problem BSSP is $\mathcal{NP}$-hard in the strong sense.

For $\mathcal{NP}$-hard problems in the strong sense, an optimal solution, especially for large and challenging problem instances, cannot generally be found using a MILP-solver within a reasonable amount of time. In the next section, we propose a heuristic approach that solely seeks a subset of the feasible region using some rules to provide good, feasible solutions.

4 Heuristic solution procedure

In this section, we introduce a constructive heuristic that is embedded in a so-called *multi-start algorithm*. Constructive heuristics gradually generate a complete solution based on a partial solution. In our heuristic algorithm, at each point in time t, eligible blocks are determined according to two different factors. On the one hand, a block is eligible if it can be completed at the considered point in time according to its processing time and the completion status of its predecessors. On the other hand, a block is not eligible if its extraction results in an overrun of the upper limit of the output in the associated tipple area and the related mining district. Based on a specific priority rule, the elements of the eligible set are prioritized. Then, a roulette-wheel selection procedure is applied to randomly select a block that is scheduled next in the planning horizon (its completion time is set to t). After that, the status of the mine and, accordingly, the eligible set are updated. The selection procedure continues until the eligible set at the considered point in time is empty. By the use of the roulette-wheel selection procedure, the next block to be scheduled is randomly selected from the eligible set. That means, if the method is carried out several times in a multi-start environment, it is very likely that a large number of feasible solutions are generated. Table 4 outlines the sets, parameters, and variables used in our heuristic algorithm.

Algorithm 1 describes the developed multi-start heuristic in detail.

Algorithm 1 Multi-start algorithm with priority rules

1: Input: a problem instance, optimality interval $[0, \xi]$, priority rule Ψ, and a time limit
2: **while** time limit is not achieved, and no solution has an objective value within the predefined optimality interval **do**
3: initialization;
4: **for all** $t \in \mathcal{T}$ **do**
5: determine τ;
6: **for all** $b \in \mathcal{E}$ **do**
7: **if** $t - Z_b - es_b \geq 0$ **then**
8: insert b into set $\mathcal{E}^{\mathcal{T}}$;
9: **if** $\mathcal{E}^{\mathcal{T}} = \emptyset$ **then**
10: terminate; (no feasible solution found)
11: $\mathcal{E}^{\mathcal{CT}} := \emptyset$;
12: **for all** $b \in \mathcal{E}^{\mathcal{T}}$ **do**
13: **if** $A_b \leq \rho^{\text{res}}_{R_b,t} \wedge A_b \leq \rho^{\text{res}}_{K_b,t}$ **then**
14: insert b into set $\mathcal{E}^{\mathcal{CT}}$;
15: **repeat**
16: choose $b^* \in \mathcal{E}^{\mathcal{CT}}$ with respect to priority rule Ψ;
17: update $\rho^{\text{res}}_{R_{b^*},t}$ and $\rho^{\text{res}}_{K_{b^*},t}$;
18: update $\overline{q}_\tau$;
19: **if** block l is the successor of b^* **then**
20: insert l in $\mathcal{E}$;
21: $es_l := t$;
22: remove b^* from $\mathcal{E}$;
23: remove b^* and all $b \in \mathcal{E}^{\mathcal{CT}}$ with $R_b = R_{b^*}$ and $K_b = K_{b^*}$ for which $A_b > \rho^{\text{res}}_{R_{b^*},t}$ or $A_b > \rho^{\text{res}}_{K_{b^*},t}$ from $\mathcal{E}^{\mathcal{CT}}$;
24: **until** $\mathcal{E}^{\mathcal{CT}} = \emptyset$
25: **if** constraint sets (7), (8), (9), (12), and (13) are satisfied **then**
26: store the solution;
27: **else**
28: discard the solution;
29: **return** the best solution found

In our minimization problem, all of the decision variables are non-negative. Thus, the objective function cannot take a value smaller than 0. With the prescribed value of $\xi > 0$, we denote a tiny quality tolerance value of the production process. Accordingly, a feasible solution is called an optimal solution if the associated value of the objective function lies in the narrow interval $[0, \xi]$. For a given problem instance, feasible solutions are generated using priority rule Ψ until the objective value of a feasible solution is within the predefined interval or a given time limit is exceeded (while-loop at line 2). Within an *initialization* step (line 3 in Alg. 1), we store for each block $b \in \mathcal{B}$ the associated mining district R_b and tipple area K_b. Moreover, we store the blocks with no predecessor block in set $\mathcal{E}$. For all $b \in \mathcal{E}$, the earliest start

Table 4 Sets, parameters, and variables used in the heuristic algorithm

Sets	
$\mathcal{E}$	Set of blocks that have no predecessor, or their predecessor blocks are already excavated
$\mathcal{E}^{\mathcal{CT}}$	Set of blocks that can be excavated regarding the considered point in time and the considered output capacities ($\mathcal{E}^{\mathcal{CT}} \subset \mathcal{E}^{\mathcal{T}}$)
$\mathcal{E}^{\mathcal{T}}$	Set of blocks that can be excavated regarding the considered point in time ($\mathcal{E}^{\mathcal{T}} \subset \mathcal{E}$)
Parameters	
ϵ	A tiny number that is set to 0.0001
K_b	Tipple area that contains block $b \in \mathcal{B}$
Ψ	Prescribed priority rule; $\Psi \in \{1, 2, 3, 4\}$
R_b	Mining district that contains block $b \in \mathcal{B}$
ξ	A tiny quality tolerance value of the production process
Variables	
b^*	Selected block from $\mathcal{E}^{\mathcal{CT}}$ (using a roulette-wheel-selection procedure)
es_b	Earliest start time of block b according to the completion time of its predecessor; if block b does not have any predecessor, es_b is 0
ψ_b	Probability value of block b
π_b	Priority value of block b according to prescribed priority rule Ψ
ρ_{kt}^{res}	Residual capacity of the output for tipple area k in time interval $(t-1, t]$
ρ_{rt}^{res}	Residual capacity of the output for mining district r in time interval $(t-1, t]$
$\overline{q}_\tau^b$	New value of $\overline{q}_\tau$ if block b is mined next in sub-interval τ in %

time is 0 ($es_b = 0$). Furthermore, the residual capacities ρ_{rt}^{res} and ρ_{kt}^{res} are set to the corresponding maximum limits of the output P_{rt}^{+} and P_{kt}^{+}, respectively.

After the initialization step, we pass the entire planning horizon in a for-loop (line 4). For the current point in time t, we determine the related sub-interval τ and put those blocks from $\mathcal{E}$ into set $\mathcal{E}^{\mathcal{T}}$ that can be completed until t (if-condition at line 7). At the beginning of the planning horizon ($t = 1$), the earliest start times of all blocks $b \in \mathcal{E}$ are 0. Accordingly, only the blocks $b \in \mathcal{E}$ with processing times $Z_b = 1$ can be added to $\mathcal{E}^{\mathcal{T}}$. If no block can be inserted into $\mathcal{E}^{\mathcal{T}}$, the output at the point in time t is 0. Hence, constraint set (8) is violated, and no feasible solution can be found. Thus, the algorithm terminates (line 10). Otherwise, blocks from $\mathcal{E}^{\mathcal{T}}$ are reconsidered according to the residual capacities of the corresponding mining districts and tipple areas. For block $b \in \mathcal{E}^{\mathcal{T}}$, if A_b does not exceed the associated residual capacities (if-condition at line 13), b is inserted into $\mathcal{E}^{\mathcal{CT}}$. In a realistic problem instance, we always have:

$$\max_{b \in \mathcal{B}} \{A_b\} \leq P_{R_b,t}^{+} \quad \forall t \in \mathcal{T}, \text{ and}$$

$$\max_{b \in \mathcal{B}} \{A_b\} \leq P_{K_b,t}^{+} \quad \forall t \in \mathcal{T}.$$

Therefore, at least one block from $\mathcal{E}^{\mathcal{T}}$ (if there is any) can always be inserted into $\mathcal{E}^{\mathcal{CT}}$ at each point in time. The block that must be scheduled next is selected from $\mathcal{E}^{\mathcal{CT}}$ according to the given priority rule Ψ (line 16). For our heuristic, we consider four different priority rules. Based on priority rule $\Psi \in \{1, 2, 3, 4\}$, we determine for each block b in $\mathcal{E}^{\mathcal{CT}}$ a priority value π_b. In the following, we explain the way how π_b is calculated according to a specific priority rule.

priority rule 1: $\pi_b = \dfrac{1}{|Q_b - Q_\tau| + \epsilon}$

For this rule, we put emphasis on the quality value of a block. A block with a smaller deviation from the quality target value takes a larger priority value. This priority rule gives a block that may contribute to a better objective function value a greater probability to be extracted next. Since some blocks may have no deviation from the quality target value, we add a tiny number $\epsilon > 0$ to the denominator of the fraction. In our work, we set $\epsilon = 0.0001$.

priority rule 2: $\pi_b = \dfrac{A_b}{|Q_b - Q_\tau| + \epsilon}$

For priority rule 2, we additionally consider the amount of material extracted from a block. If quality values of blocks are the same, a block with a larger amount of material is given a larger priority value, or—in other words—for the same amount of material, a block that has a smaller deviation from the quality target receives a larger priority value (like priority rule 1). Using priority rule 2, the lower limits of output are also considered to avoid generating infeasible solutions.

priority rule 3: $\pi_b = \dfrac{1}{|\overline{q}_\tau^b - Q_\tau| + \epsilon}$

Here, we calculate the value of $\overline{q}_\tau^b$ that represents the change of $\overline{q}_\tau$ if block b is excavated next. A block takes a larger priority value if its excavation results in a smaller deviation from the quality target value. We can, therefore, give a more considerable priority value to the blocks that allow the best possible improvement in the objective function value in each step.

priority rule 4: $\pi_b = \begin{cases} \epsilon, & \text{if } P^-_{R_b} \text{ is achieved;} \\ A_b, & \text{otherwise.} \end{cases}$

Priority rule 4 helps to avoid infeasible solutions in terms of constraint set (9). For block b, if the lower limit of the output for the corresponding mining district R_b is achieved, we set the priority value of block b equal to a tiny number $\epsilon > 0$. Otherwise, block b receives a priority value equal to A_b. Using priority rule 4, we focus only on finding feasible solutions regarding the lower limits of output.

According to the priority values, all blocks $b \in \mathcal{E}^{\mathcal{CT}}$ receive a probability value ψ_b as follows:

$$\psi_b = \frac{\pi_b}{\sum_{l \in \mathcal{E}^{\mathcal{CT}}} \pi_l}.$$

We apply a roulette-wheel selection procedure, where each block occupies an area on the roulette-wheel proportional to its ψ_b-value (Michalewicz and Fogel 2004, Sect. 6.1). That is equivalent to partitioning the interval [0, 1] into $|\mathcal{E}^{\mathcal{CT}}|$ parts, where the b-th sub-interval (part) has the width ψ_b representing block b. Subsequently, a random number between 0 and 1 is generated. The sub-interval that contains the random number determines block b^*. At lines 17 and 18, the associated residual capacities and the value of $\overline{q}_\tau$ are updated. If b^* has a successor l (line 19), we insert l in $\mathcal{E}$. The earliest start time of l is set equal to the completion time of b^*, i.e., $es_l = t$ (line 21). Block b^* is then removed from $\mathcal{E}^{\mathcal{CT}}$. Moreover, all blocks $b \in \mathcal{E}^{\mathcal{CT}}$ with $R_b = R_{b^*}$ and $K_b = K_{b^*}$, for which $A_b > \rho^{\text{res}}_{R_{b^*},t}$ or $A_b > \rho^{\text{res}}_{K_{b^*},t}$, have to be removed from $\mathcal{E}^{\mathcal{CT}}$ (lines 22 and 23). The repeat-until-loop in Alg. 1 will be executed until $\mathcal{E}^{\mathcal{CT}}$ is empty.

After violating the condition of the repeat-until-loop, t is incremented by one. The entire for-loop at line 4 will be executed for all $t \in \mathcal{T}$. We then check the feasibility of the solution found according to constraint sets (7), (8), (9), (12), and (13). Feasible solutions are stored, and infeasible solutions are discarded.

We observe the loops in Alg. 1 to determine the time complexity of the presented heuristic approach in every run. The for-loop at line 4 is executed $T_{\max}$ times. Moreover, the for-loop at line 6 and the repeat-until-loop are conducted, at most, $|\mathcal{B}|$ times. Line 11, the for-loop at line 12, as well as line 16, if-condition at line 19, and lines 22 and 23 within the repeat-until-loop have, at most, $|\mathcal{B}|$ steps. Accordingly, the algorithm has a time complexity of $\mathcal{O}(T_{\max} \cdot |\mathcal{B}|^2)$ that implies a pseudo-polynomial algorithm. Note that in practical applications, $T_{\max}$ is given as a constant number. Therefore, the time complexity of the algorithm is $\mathcal{O}(|\mathcal{B}|^2)$, and the heuristic proposed is polynomial.

In the next section, we compare the results achieved by a MILP-solver with the solutions provided by the proposed heuristic approach. Moreover, we introduce a sophisticated combination of the heuristic and the mathematical program to improve the results obtained by the heuristic procedure for the most challenging problem instances, for which our MILP-solver cannot find a feasible solution within a reasonable amount of time.

5 Computational study

In this section, we perform an experimental performance analysis for the proposed mixed-integer linear program and the constructive heuristic. The computational study is executed on randomly generated problem instances that are based on real-world data derived from Clausen (2013). Table 5 shows some parameters that are typical for a potash mine and used to generate realistic problem instances.

To normalize the problem instances, we set the number of blocks in each mining district r equal to 325. Thus, the problem instances with more underground locations have fewer blocks in each underground location and vice versa.

The allowed maximum output in each mining district r for time interval $(t-1, t]$ depends on the number of loaders available in the corresponding mining district

Table 5 Parameters used to generate realistic problem instances

Parameters	Symbols	Values
Number of mining districts in an underground mine		1–5
Number of tipple areas in each mining district		4–6
Number of underground locations in each tipple area		5–11
Number of blocks in each underground location		5–10
Amount of material contained in each block in tonnes	A_b	700–1200
Quality value of each block in %	Q_b	9.8–16.2
Lower limit of the quality value of the output in %	Q^-	11.1
Upper limit of the quality value of the output in %	Q^+	14.1
Quality target value for each sub-interval in %	Q_τ	12.6
Processing time of the first blocks in each underground location measured in work shifts	Z_b	1–9
Processing time of other (not the first) blocks in each underground location measured in work shifts		6–12

during the associated time interval. A loader can typically transport 750 tonnes of crude material within a work shift. In each work shift, 1 to 4 loader(s) are available so that $P^+_{rt} \in \{750, 1500, 2250, 3000\}$. The minimum output in each mining district depends on the capacity of loaders, too. The parameter P^-_r can be determined as a certain percentage γ of the capacity of all the available loaders in mining district r within the entire time horizon:

$$P^-_r = \gamma \cdot \sum_{t \in \mathcal{T}} P^+_{rt} \qquad \forall r \in \mathcal{R}.$$

The parameter γ varies between 70% and 90%, with a step size of 5%. Furthermore, the lower limit of the total output is the sum of P^-_r over the mining districts contained in the underground mine:

$$P^- = \sum_{r \in \mathcal{R}} P^-_r.$$

The upper limits of the output for tipple areas P^+_{kt} are randomly chosen from the set $\{1000, 2000, \ldots, 5000\}$. Those numbers are given due to the characteristics of typical feeder breakers. Even though the parameters P^+_{kt} have different values in terms of tipple areas k, they are the same for all time intervals $(t-1, t]$ in the planning horizon. In each time interval $(t-1, t]$, an output greater than 0 has to be achieved. We set the parameter P^-_t as follows:

$$P^-_t = \min_{b \in \mathcal{B}} \{A_b\} \qquad \forall t \in \mathcal{T}.$$

We consider a planning horizon of one month that without loss of generality, is supposed to have only four weeks. Each sub-interval τ represents a time interval of one week ($\tau_{\max} = 4$). A week contains 18 work shifts, so that the planning horizon has $T_{\max} = 72$ work shifts.

The quality values of blocks are stated in percentage and usually have decimals. We multiply the quality values of blocks and, accordingly, the quality parameters from Table 5 by 100 to avoid rounding errors. For example, a quality value of 13.7% is 1370 in a problem instance. As mentioned in Sect. 4, from a practical point of view, a solution is called an optimal solution if the associated value of the objective function lies within the small interval of $[0, \xi]$ with $\xi > 0$. Parameter ξ is the quality tolerance value of the production process and is chosen to be 0.1%. Like the quality values of blocks, we multiply ξ by 100 as well; ξ is, therefore, equal to 10.

We distinguish between 5 different test sets according to the number of mining districts. Hence, there is 1 mining district in the first test set, 2 mining districts in the second test set, and so on (5 mining districts in the fifth test set). We randomly generated 20 problem instances for each test set using the introduced parameter ranges. Additionally, we consider five different levels of γ, i.e., $\gamma \in \{70\%, 75\%, \ldots, 90\%\}$, concerning the lower limit of the output (P_r^- and accordingly P^-, see above). Thus, we have 100 problem instances for each γ-level. Test sets with 5 mining districts represent the largest problem instances. On the other hand, the problem instances with a γ-level of 90% are the most challenging problem instances to solve.

We used GAMS 24.9 to implement our mixed-integer linear program and solved the problem instances using CPLEX 12.7.1. We set the solver parameters in the way that the solution procedure terminates in the following cases:

1. if a solution found is optimal;
2. if a solution found is in the predefined interval [0, 10]; or
3. if a time limit of 1800 s is exceeded.

The corresponding solution procedure is called **CPLEX**. According to the cases mentioned above, if we say that CPLEX finds an optimal solution for a problem instance, case 1. or case 2. holds.

The heuristic algorithm is implemented in the programming language **C++** and executed with the compiler Microsoft Visual Studio 2010. Since we use four different priority rules in the heuristic, we set the upper limit of the solution time equal to 450 s for each priority rule. Thus, for each problem instance, we run the multi-start heuristic approach for 1800 (4×450) s to have a relatively fair comparison with the results of CPLEX, which has a time limit of 1800 s, too. The resulting multi-start heuristic approach with four randomized priority rules is called **MSH-4R**.

All tests are executed on an Intel(R) i7-7700K@4.20GHz machine with 64 GB RAM under Windows 10.

We can compare the results achieved by CPLEX and MSH-4R from different aspects. On the one hand, the number of problem instances for which no feasible solution can be provided is important. On the other hand, we want to know whether a feasible solution obtained is optimal, and if not, what can be said about the solution quality. We discussed that the problem instances with a γ-level of 90% are the most challenging problem instances to solve, and the ones with 5 mining districts are the largest problem instances. Accordingly, the results of each solution procedure for

each γ-level and regarding the size of the problem instances are of most interest. Our computational study suggests the following results:

- In general, the greater the problem instances, the more time CPLEX needs to find an optimal solution.
- For γ-levels of 70% to 85%, CPLEX finds an optimal solution for almost all problem instances. However, for the most challenging problem instances with $\gamma = 90\%$, CPLEX can hardly find any feasible solution.
- MSH-4R can find for all of the problem instances at least one feasible solution that is not far from the best solutions found.
- MSH-4R using priority rule 3 provides the most high-quality solutions in comparison to the other priority rules.
- If we use the solutions found by MSH-4R using priority rule 3 as initial solutions for CPLEX, the results found for the most challenging problem instances are improved by, on average, 92.5%.

A detailed explanation of the results achieved by the solution approaches is given in the following.

Table 6 compares the results achieved by CPLEX with the solutions found by MSH-4R. Each row in Table 6 shows the information for a test set that consists of 20 problem instances. Columns # *Optimal* depict the number of test instances for which CPLEX and MSH-4R could find an optimal solution (see the explanation for an optimal solution above). The number of test instances for which a feasible solution could be found but the optimality of the solution could not be proven or shown is given under columns # *Feasible* for both solution procedures. Columns # *Unknown* present the number of test instances for which no feasible solution is found within the considered time limit (1800 s). Note that the numbers appearing under each column cannot be greater than 20. Columns *Solution time* report the average solution time in seconds for the considered 20 problem instances in each test set. According to the setting, the average solution time cannot be greater than 1800 s. Analogous to CPLEX, MSH-4R terminates if a solution found by a given priority rule is in the predefined interval [0, 10]. In that case, we say that MSH-4R finds an optimal solution for a problem instance. Note that MSH-4R is not able to state whether a feasible solution with an objective function value greater than 10 is optimal or not. For a problem instance solved by MSH-4R, an optimal solution may be found by using any priority rule. In those cases, the procedure terminates before the time limit (450 s) is exceeded for the respective priority rule. The time in seconds under *Solution time* MSH-4R is the sum of solution times taken by every priority rule and cannot be greater than 1800 s (4×450).

We see in Table 6 that CPLEX could find for the whole 100 problem instances with $\gamma = 70\%$ an optimal solution. The greater the number of mining districts in a test set, the more time CPLEX needs to find an optimal solution. For $\gamma = 70\%$, the solution time is, on average, 14 s for the 20 problem instances with 1 mining district and 132 s for problem instances with 5 mining districts. For the whole 100 problem instances with $\gamma = 70\%$, CPLEX needed, on average, 53 s to prove the optimality of the solutions found. For $\gamma = 75\%$, there is the same trend. All

Table 6 Comparison of the results achieved by CPLEX with the solutions found by MSH-4R (each row represents a test set including 20 problem instances)

γ[in %]	$\lvert\mathcal{R}\rvert$	# Optimal		# Feasible		# Unknown		Solution time [s]		Ave. deviation from best. [$\frac{\%}{100}$]	
		CPLEX	MSH-4R	CPLEX	MSH-4R	CPLEX	MSH-4R	CPLEX	MSH-4R	CPLEX	MSH-4R
70	1	20	7	0	13	0	0	14	1392	0.00	7.42
	2	20	3	0	17	0	0	22	1615	0.00	8.40
	3	20	0	0	20	0	0	44	1800	0.00	14.60
	4	20	2	0	18	0	0	52	1740	0.00	12.74
	5	20	0	0	20	0	0	132	1800	0.00	13.35
75	1	20	7	0	13	0	0	26	1392	0.00	7.42
	2	20	3	0	17	0	0	62	1621	0.00	8.40
	3	20	0	0	20	0	0	241	1800	0.00	14.60
	4	20	2	0	18	0	0	279	1740	0.00	12.74
	5	20	0	0	20	0	0	330	1800	0.00	13.35
80	1	18	7	2	13	0	0	202	1392	0.00	6.48
	2	20	3	0	17	0	0	230	1615	0.00	8.40
	3	20	0	0	20	0	0	404	1800	0.00	14.60
	4	20	2	0	18	0	0	473	1740	0.00	12.75
	5	20	0	0	20	0	0	676	1800	0.00	13.35
85	1	18	7	2	13	0	0	324	1392	0.02	5.74
	2	17	3	3	17	0	0	568	1615	0.00	7.13
	3	18	0	2	20	0	0	788	1800	0.00	14.04
	4	19	2	1	18	0	0	904	1740	0.00	12.51
	5	18	0	2	20	0	0	1020	1800	0.00	12.86
90	1	7	6	1	14	12	0	1241	1460	0.00	1.07
	2	0	2	1	18	19	0	1800	1668	0.00	0.36
	3	0	0	0	20	20	0	1800	1800	–	0.00
	4	0	1	0	19	20	0	1800	1762	–	0.00
	5	0	0	0	20	20	0	1800	1800	–	0.00

of the test instances are solved to optimality by CPLEX; the solution time is, on average, 26 s for problem instances with 1 mining district and 330 s for problem instances with 5 mining districts. Notably, the computational times are higher (for the 100 problem instances with $\gamma = 75\%$, the computational time is, on average, 187.6 s) since the problem instances are more challenging. For γ-levels of 80% and 85%, the same trend can be seen. For $\gamma = 80\%$ ($\gamma = 85\%$), CPLEX could optimally solve 98 (90) problem instances. At the γ-level of 80% (85%), the solution time is, on average, 202 (324) s for the 20 problem instances with 1 mining district and 676 (1020) s for the 20 problem instances with 5 mining districts. For the other problem instances with $\gamma = 80\%$ and $\gamma = 85\%$ that could not be solved to optimality, CPLEX could find a feasible solution (cf. columns # *Feasible*). For

the most challenging problem instances with $\gamma = 90\%$, CPLEX could find for only seven problem instances with 1 mining district an optimal solution within, on average, 1241 s. Furthermore, for only two other problem instances, a feasible solution could be found by CPLEX. In other words, for 91 problem instances at the γ-level of 90%, CPLEX cannot find any feasible solutions within a reasonable amount of time.

On the contrary, MSH-4R was able to find an optimal solution for 12 problem instances each at the γ-levels of 70%, 75%, 80%, and 85% as well as for nine problem instances with $\gamma = 90\%$ (altogether 57 of the whole 500 problem instances). Hence, it cannot be said at which level MSH-4R works best. MSH-4R could find for every problem instance at every level and with every size at least one feasible solution (the numbers under # *Unknown* MSH-4R are all 0), even for the γ-level of 90%. To evaluate the solution quality of the results achieved by MSH-4R, we calculate the average of the absolute deviations from the best solutions found (in $\frac{\%}{100}$), which is stated in the last columns *Ave. deviation from best* of Table 6. The following example illustrates how the deviation from the best solution found is calculated.

Example 1 Let 6.64 be the objective function value of the best solution found by CPLEX (S_C), and 24.34 the objective function value of the best solution found by MSH-4R (S_H). Since $S_C \leq \xi = 10$ is true, we consider S_C as an optimal solution. Accordingly, the deviation of S_C from the best solution found is 0, and the deviation of S_H from an optimal solution is $S_H - \xi = 14.34\ \frac{\%}{100}$.

Now, let $S_C = 16.34$ and $S_H = 30.71$. The best solution found is S_C and, consequently, the deviation of S_C from the best solution found is 0. Independent of whether we can prove the optimality of S_C or not, the deviation of S_H from the best solution found is $S_H - S_C = 14.37\ \frac{\%}{100}$.

For the problem instances with $\gamma = 70\%$, the solutions found by MSH-4R deviate, on average, $11.30\ \frac{\%}{100}$ (0.113%) from the optimal solution found by CPLEX. The minimum value is for the test set with 1 mining district ($7.42\ \frac{\%}{100}$), and the maximum value belongs to the test set with 3 mining districts ($14.60\ \frac{\%}{100}$). For $\gamma = 70\%$, MSH-4R could find for only 12 problem instances an optimal solution. Hereby, seven problem instances that are solved to optimality by MSH-4R have 1 mining district. On the other hand, for the test set with 3 mining districts, no optimal solution is found by MSH-4R. Since the average is calculated over the whole 20 problem instances for each test set, the differences regarding the average deviations from the best solutions found can be explained by the different number of optimal solutions found for each test set. The same trend exists at γ-levels of 75%, 80%, and 85%, where the solutions found by MSH-4R deviate, on average, 11.3, 11.12, and $10.45\ \frac{\%}{100}$ from the best solution found, respectively. We can conclude that the quality of the solutions found by MSH-4R is quite promising for the γ-levels of 70%, 75%, 80%, and 85%, where optimal solutions for 388 of 400 problem instances are known, which is a strong measure to evaluate the results. In addition, we see that for $\gamma = 90\%$, where CPLEX could find for only nine problem instances a feasible solution, MSH-4R found for all of the

100 problem instances at least one feasible solution. Therefore, it is reasonable to use MSH-4R to solve the problem instances at the γ-level of 90%.

In the next step, we compare the applied priority rules in MSH-4R (cf. Sect. 4) in Table 7. The first columns on the left-hand side of Table 7 show the number of problem instances for which a specific priority rule **exclusively** found the best solution. For example, in the first row of Table 7, the sum of the numbers under *Number of best-known solutions exclusively found* for rules 1 to 4 is 14. That means there are six problem instances in the test set for which the best solution could be found by means of at least two different priority rules. At the same line, we see that, e.g., priority rule 2 could exclusively find the best solutions for two problem instances, which means that the other priority rules could not find the best solution for those problem instances. We see in Table 7 that for 407 of the whole 500 problem instances, the best solutions were exclusively obtained using priority rule 3,

Table 7 Comparison among 4 priority rules (each row represents a test set including 20 problem instances)

γ [*in%*]	$\lvert\mathcal{R}\rvert$	Number of best-known solutions exclusively found				Average t^{bs} (t^{fs}) for the best (first) feasible solution [s]			
		Rule 1	Rule 2	Rule 3	Rule 4	Rule 1	Rule 2	Rule 3	Rule 4
70	1	2	0	12	0	186 (0.00)	196 (0.00)	153 (0.00)	186 (0.00)
	2	0	0	18	0	249 (0.00)	182 (0.00)	209 (0.00)	260 (0.00)
	3	2	0	18	0	229 (0.00)	151 (0.00)	210 (0.00)	223 (0.00)
	4	1	0	18	0	256 (0.00)	213 (0.00)	250 (0.00)	258 (0.00)
	5	1	0	19	0	231 (0.00)	260 (0.00)	240 (0.00)	267 (0.00)
75	1	2	0	12	0	185 (0.00)	196 (0.00)	153 (0.00)	207 (0.00)
	2	0	0	18	0	249 (0.00)	181 (0.00)	209 (0.00)	148 (0.00)
	3	2	0	18	0	229 (0.00)	151 (0.00)	210 (0.00)	224 (0.00)
	4	1	0	18	0	256 (0.00)	213 (0.00)	250 (0.00)	200 (0.00)
	5	1	0	19	0	231 (0.00)	260 (0.00)	240(0.00)	223 (0.00)
80	1	2	0	12	0	186 (0.00)	196 (0.00)	153 (0.00)	228 (0.00)
	2	0	0	18	0	249 (0.00)	181 (0.00)	209 (0.00)	174 (0.00)
	3	2	0	18	0	229 (0.00)	151 (0.00)	209 (0.00)	187 (0.00)
	4	1	0	18	0	256 (0.00)	213 (0.00)	236 (0.00)	224 (0.00)
	5	1	0	19	0	231 (0.00)	259 (0.00)	239 (0.00)	194 (0.00)
85	1	2	0	12	0	186 (0.00)	195 (0.00)	153 (0.00)	238 (0.00)
	2	0	0	18	0	249 (0.00)	182 (0.00)	210 (0.00)	194 (0.00)
	3	2	0	18	0	229 (0.00)	151 (0.00)	210 (0.00)	232 (0.00)
	4	1	0	18	0	256 (0.00)	213 (0.00)	236 (0.00)	204 (0.00)
	5	1	0	19	0	231 (0.00)	260 (0.00)	240 (0.00)	254 (0.00)
90	1	3	1	11	0	144 (0.00)	180 (0.00)	190 (0.00)	234 (0.00)
	2	0	3	15	0	216 (0.00)	217 (0.00)	227 (0.00)	197 (0.00)
	3	0	4	16	0	212 (0.05)	245 (0.00)	261 (0.00)	257 (0.00)
	4	1	3	13	0	250 (0.00)	196 (0.00)	214 (3.00)	246 (0.05)
	5	0	7	12	1	201 (0.25)	220 (0.05)	214 (0.10)	230 (0.00)

which suggests priority rule 3 as the most potent rule. Remember that priority rule 3 enables the best possible improvement regarding the objective function in each step.

Let t^{bs} and t^{fs} be the times in seconds that are taken to find the best (*bs*) and the first feasible solution (*fs*), respectively. Those times are given on the right-hand side of Table 7. MSH-4R using priority rules 1, 2, 3, and 4 needed, on average, 225, 202, 213, and 220 s, respectively, to find the best feasible solutions. The first feasible solution could be quickly found by MSH-4R regardless of the applied priority rules (e.g., priority rules 2 and 4 find the first feasible solution by no longer than, on average, 0.05 s).

We saw that CPLEX performs for $\gamma = 90\%$ worse, in particular, with respect to the number of feasible solutions found (cf. Table 6). At the γ-level of 90%, where we have the most challenging problem instances, and where it is tough to find a feasible solution by a MILP-solver, MSH-4R can find at least one feasible solution for each problem instance. In the following, we introduce another solution approach to tackle the problem instances with $\gamma = 90\%$. The idea is to support CPLEX by starting with an initial feasible solution that is found by MSH-4R. We distinguish between the following kinds of initial solutions:

first solution (*fs*): MSH-4R using priority rules 2 and 4 is the fastest approach to find a feasible solution (cf. Table 7). That may be the case because both priority rules consider the amount of material of a block for determining the priority values to observe the lower limits of output. If we observe the quality of the solutions found, MSH-4R using priority rule 2 (priority rule 4) could exclusively find the best solution for 18 problem instances (one problem instance) with $\gamma = 90\%$. Remember that priority rule 2 considers both block's amount of material and the block's quality deviation from the quality target. By contrast, priority rule 4 focuses only on the amount of material, which justifies the higher quality of the solutions found using priority rule 2 than using priority rule 4. Thus, we solve the problem instances at the γ-level of 90% by MSH-4R using priority rule 2 and store the time t^{fs}. Then, we give the first feasible solution found as an initial solution to CPLEX and set the upper limit of time equal to $1800 - t^{fs}$ seconds. The corresponding solution approach is called **CPLEX with *fs***.

best solution (*bs*): Table 7 shows that MSH-4R using priority rule 3 performs best among all four priority rules. MSH-4R using priority rule 3 could exclusively find for 67 problem instances with $\gamma = 90\%$ the best solution within, on average, 221.2 s. Hence, we solve the problem instances at the γ-level of 90% by MSH-4R using priority rule 3, set the upper limit of the solution time equal to 250 s, and store the best feasible solution found. Then, we solve the problem instances by CPLEX using the best feasible solutions found by MSH-4R as an initial solution with a time limit of 1550 s. The corresponding solution approach is called **CPLEX with *bs***.

Table 8 depicts that using an initial solution leads to a much better performance of CPLEX at the γ-level of 90%. Without using an initial solution, CPLEX could find for only nine of 100 problem instances a feasible solution (thereof seven optimal solutions) within, on average, 1688 s (cf. Table 6, last 5 rows). If we apply CPLEX

Table 8 Comparison of the solutions found by CPLEX with and without initial solutions (each row represents a test set including 20 problem instances)

γ [*in%*]	$\lvert\mathcal{R}\rvert$	CPLEX with *fs*			CPLEX with *bs*		
		# Opt.	# Feas.	Solution time [s]	# Opt.	#Feas.	Solution time [s]
90	1	12	8	922	14	6	789
	2	12	8	1115	11	9	1298
	3	5	15	1626	7	13	1567
	4	8	12	1651	8	12	1575
	5	5	15	1757	5	15	1786

with *fs*, 42 problem instances can be solved to optimality within, on average, 1414.2 s. Typically, the larger the problem instances, the harder it is to solve. For the problem instances with 1 mining district, CPLEX with *fs* could find an optimal solution for 12 of 20 problem instances. For the problem instances with 5 mining districts, only five of 20 problem instances can be solved to optimality by CPLEX with *fs*. CPLEX with *bs* performs even better, where 45 problem instances can be optimally solved within, on average, 1403 s (some more problem instances within some less time in comparison to CPLEX with *fs*). The same trend regarding the size of the instances is recognizable. We can see that 14 of 20 problem instances with 1 mining district and five of 20 problem instances with 5 mining districts can be solved to optimality by CPLEX with *bs*.

Finally, we compare the results achieved by MSH-4R with the combination of MSH-4R and CPLEX. For this purpose, the average of the absolute deviations from the best solution known is depicted in Table 9 for each test set with $\gamma = 90\%$. Let S_{H_i} be the objective function value of the best solution found by MSH-4R for problem instance i. Moreover, let $S^{fs}_{C_i}$ and $S^{bs}_{C_i}$ denote the objective function values of the solutions found by CPLEX with *fs* and *bs* for problem instance i, respectively. The numbers in parentheses are the average of $(S_{H_i} - S^{fs}_{C_i})/(S_{H_i})$ as well as $(S_{H_i} - S^{bs}_{C_i})/(S_{H_i})$ over the problem instances in every corresponding test set. We see that the solutions

Table 9 Comparison of the solutions found by MSH-4R with the results achieved by CPLEX using initial solutions (each row represents a test set including 20 problem instances)

γ [*in%*]	$\lvert\mathcal{R}\rvert$	Average deviation from the best solution found $[\frac{\%}{100}]$		
		MSH-4R	CPLEX with *fs* (improvement)	CPLEX with *bs* (improvement)
90	1	3.11	0.98 (68.5%)	0.03 (99.0%)
	2	5.11	1.96 (61.6%)	0.27 (94.7%)
	3	8.88	2.43 (72.6%)	0.72 (91.9%)
	4	8.11	1.93 (76.2%)	0.57 (93.0%)
	5	7.70	1.22 (76.2%)	1.21 (84.3%)

found by CPLEX with an initial solution are much better than the solutions found by MSH-4R. In particular, if we first solve a problem instance heuristically using priority rule 3 and then give the best feasible solution found as an initial solution to CPLEX (CPLEX with *bs*), the solutions are improved by, on average, 92.5%.

6 Conclusion

In this paper, we introduced a mixed-integer linear program and a heuristic approach to solve a block selection and sequencing problem occurring in underground potash mines. In the problem under consideration, we minimized the deviations of the output quality value from a prescribed quality target value. The deviations were calculated for certain sub-intervals within a given planning horizon.

To evaluate the solution approaches, we randomly generated 100 problem instances (5 test sets of 20 problem instances each) based on realistic data. We solved the problem instances on 5 different levels in terms of the lower limit of the output using the proposed mixed-integer linear program and the suggested constructive heuristic. We can conclude that if the MILP-solver starts with an initial solution, an optimal solution for a problem instance can be found more quickly. We can also say, if we solve a problem instance within a reasonable amount of time heuristically using introduced priority rule 3 and then give the best feasible solution found as an input to the MILP-solver, we obtain quite promising results. Hence, practice-relevant problems in potash mines can be solved with an acceptable quality within an adequate time frame. Consequently, the results achieved could support the decisions of underground mining operators to provide the aboveground processing plants with homogenous output by a systematic comparison between the actual and target performance.

Further research will apply metaheuristics to improve the results achieved by the proposed constructive heuristic. Uncertainty regarding the processing times needed to excavate the blocks must also be investigated. On the other hand, the material excavated in a work shift can be stored in a bunker during a specific time interval. That can lead to another value of the output quality in comparison to having all the material conveyed directly to the surface. Ongoing research can distinguish those cases to make the generated solutions more realistic.

References

Askari-Nasab H, Pourrahimian Y, Ben-Awuah E, Kalantari S (2011) Mixed integer linear programming formulations for open pit production scheduling. J Min Sci 47(3):338–359

Azzamouri A, Fénies P, Fontane F, Giard V (2018) Scheduling of open-pit phosphate mine extraction. Int J Prod Res 56(23):7122–7141

Bley A, Boland N, Fricke C, Froyland G (2010) A strengthened formulation and cutting planes for the open pit mine production scheduling problem. Comput Oper Res 37(9):1641–1647

Blom M, Pearce AR, Stuckey PJ (2019) Short-term planning for open pit mines: a review. Int J Min Reclam Environ 33(5):318–339

Blom ML, Pearce AR, Stuckey PJ (2016) A decomposition-based algorithm for the scheduling of open-pit networks over multiple time periods. Manag Sci 62(10):3059–3084

Campeau L-P, Gamache M (2020) Short-term planning optimization model for underground mines. Comput Oper Res. https://doi.org/10.1016/j.cor.2019.02.005

Chesworth W (2008) Encyclopedia of soil science. Springer, Dordrecht

Chicoisne R, Espinoza D, Goycoolea M, Moreno E, Rubio E (2012) A new algorithm for the open-pit mine production scheduling problem. Oper Res 60(3):517–528

Clausen E (2013) Konzept für einen integrierten Produktionssteuerungsansatz bei Anwendung eines Örterbaus. PhD diss., Clausthal University of Technology

Elsayed S, Sarker R, Essam D, Coello CC (2020) Evolutionary approach for large-scale mine scheduling. Inf Sci 523:77–90

Espinoza D, Goycoolea M, Moreno E, Newman A (2013) Minelib: a library of open pit mining problems. Ann Oper Res 206(1):93–114

Garey MR, Johnson MD (1979) Computers and intractability: a guide to the theory of NP-completeness. Freeman, New York

Hamrin H (2001) Underground mining methods and applications. In: Hustrulid WA, Bullock RL (eds) Underground mining methods: engineering fundamentals and international case studies. SME, Littleton, pp 3–14

Heinz A, von der Osten R (1982) ABC Kali und Steinsalz. Deutscher Verlag für Grundstoffindustrie, Leipzig

Jélvez E, Morales N, Nancel-Penard P, Peypouquet J, Reyes P (2016) Aggregation heuristic for the open-pit block scheduling problem. Eur J Oper Res 249(3):1169–1177

Jélvez E, Morales N, Nancel-Penard P, Cornillier F (2020) A new hybrid heuristic algorithm for the precedence constrained production scheduling problem: a mining application. Omega 94:102046

King B, Goycoolea M, Newman A (2017) Optimizing the open pit-to-underground mining transition. Eur J Oper Res 257(1):297–309

Kozan E, Liu SQ (2011) Operations research for mining: a classification and literature review. ASOR Bull 30(1):2–23

K+S AG (2013) A day in the mine. http://www.k-plus-s.com/en/ks-zum-anfassen/tag-in-der-grube.html. Accessed 10 Sep 2020

Lambert WB, Newman AM (2014) Tailored Lagrangian relaxation for the open pit block sequencing problem. Ann Oper Res 222(1):419–438

Lamghari A, Dimitrakopoulos R (2016) Network-flow based algorithms for scheduling production in multi-processor open-pit mines accounting for metal uncertainty. Eur J Oper Res 250(1):273–290

Leite LG, Marcelo J, Arruda EF, Bahiense L, Marujo LG (2020) Modeling the integrated mine-to-client supply chain: a survey. Int J Min Reclam Environ 34(4):247–293

Liu SQ, Kozan E (2016) New graph-based algorithms to efficiently solve large scale open pit mining optimisation problems. Expert Syst Appl 43:59–65

Mai NL, Topal E, Erten O, Sommerville B (2019) A new risk-based optimisation method for the iron ore production scheduling using stochastic integer programming. Resour Policy 62:571–579

Martinez MA, Newman AM (2011) A solution approach for optimizing long-and short-term production scheduling at LKAB's Kiruna mine. Eur J Oper Res 211(1):184–197

Michalewicz Z, Fogel D (2004) How to solve it: modern heuristics. Springer, Berlin

Montiel L, Dimitrakopoulos R (2015) Optimizing mining complexes with multiple processing and transportation alternatives: an uncertainty-based approach. Eur J Oper Res 247(1):166–178

Mousavi A, Sellers E (2019) Optimisation of production planning for an innovative hybrid underground mining method. Resour Policy 62:184–192

Mousavi A, Kozan E, Liu SQ (2016) Comparative analysis of three metaheuristics for short-term open pit block sequencing. J Heuristics 22(3):301–329
Musingwini C (2016) Optimization in underground mine planning-developments and opportunities. J South Afr Inst Min Metall 116(9):809–820
Nehring M, Topal E, Little J (2010) A new mathematical programming model for production schedule optimization in underground mining operations. J South Afr Inst Min Metall 110(8):437–446
Nehring M, Topal E, Kizil M, Knights P (2012) Integrated short-and medium-term underground mine production scheduling. J South Afr Inst Min Metall 112(5):365–378
Newman AM, Rubio E, Caro R, Weintraub A, Eurek K (2010) A review of operations research in mine planning. Interfaces 40(3):222–245
O'Sullivan D, Newman A (2015) Optimization-based heuristics for underground mine scheduling. Eur J Oper Res 241(1):248–259
O'Sullivan D, Brickey A, Newman A (2015) Is openpit production scheduling easier than its underground counterpart? Min Eng 67(4):68–73
Ramazan S, Dimitrakopoulos R (2013) Production scheduling with uncertain supply: a new solution to the open pit mining problem. Optim Eng 14(2):361–380
Reus L, Belbèze M, Feddersen H, Rubio E (2018) Extraction planning under capacity uncertainty at the Chuquicamata underground mine. Interfaces 48(6):543–555
Rivera Letelier O, Espinoza D, Goycoolea M, Moreno E, Muñoz G (2020) Production scheduling for strategic open pit mine planning: a mixed-integer programming approach. Oper Res. 4:5. https://doi.org/10.1287/opre.2019.1965
Samavati M, Essam D, Nehring M, Sarker R (2017) A local branching heuristic for the open pit mine production scheduling problem. Eur J Oper Res 257(1):261–271
Samavati M, Essam D, Nehring M, Sarker R (2018) A new methodology for the open-pit mine production scheduling problem. Omega 81:169–182
Schulze M, Zimmermann J (2017) Staff and machine shift scheduling in a German potash mine. J Sched 20(6):635–656
Schulze M, Rieck J, Seifi C, Zimmermann J (2016) Machine scheduling in underground mining: an application in the potash industry. OR Spectr 38(2):365–403
Seifi C, Schulze M, Zimmermann J (2019) A two-stage solution approach for a shift scheduling problem with a simultaneous assignment of machines and workers. In: Mining goes digital: proceedings of the 39th international symposium application of computers and operations research in the mineral industry (APCOM (2019) June 2019, Wroclaw, Poland. CRC Press, pp 377–385
Seifi C, Schulze M, Zimmermann J (2020) A new mathematical formulation for a potash-mine shift scheduling problem with a simultaneous assignment of machines and workers. Eur J Oper Res. https://doi.org/10.1016/j.ejor.2020.10.007
Smith ML, Wicks SJ (2014) Medium-term production scheduling of the Lumwana Mining Complex. Interfaces 44(2):176–194
USGS (2011) Metals and minerals: United States geological survey minerals yearbook. United States Government Printing Office, Washington
Vossen TWM, Kevin Wood R, Newman AM (2016) Hierarchical Benders decomposition for open-pit mine block sequencing. Oper Res 64(4):771–793

B. Paper II

OR Spectrum (2016) 38:365–403
DOI 10.1007/s00291-015-0414-y

REGULAR ARTICLE

Machine scheduling in underground mining: an application in the potash industry

Marco Schulze[1] · **Julia Rieck**[1] · **Cinna Seifi**[1] · **Jürgen Zimmermann**[1]

Received: 21 November 2014 / Accepted: 24 July 2015 / Published online: 1 September 2015

Abstract In this paper, a scheduling problem that occurs in potash mining is introduced, where a block excavation sequence has to be found taking into account a limited number of underground machines as well as safety-related restrictions. The aim is to minimize the maximum completion time of excavations, i.e., the makespan. The resulting problem can be transformed into a hybrid flow shop scheduling problem with reentry, unrelated machines, and job-precedences. A mixed-integer linear model is presented and small-scale instances are solved with CPLEX. In order to tackle medium- and large-scale instances heuristically, a basic and an advanced multi-start algorithm are developed, based on a specific priority rule-based construction procedure. In addition, a modified version of the Giffler and Thompson procedure is applied. Computational experiments are conducted on problem instances derived from real-world data in order to evaluate the performances of the proposed solution procedures.

Keywords Underground mining · Hybrid flow shop · Reentrant flows · Mixed-integer linear program · Priority rule-based procedure · Multi-start

✉ Marco Schulze
marco.schulze@tu-clausthal.de

Julia Rieck
julia.rieck@tu-clausthal.de

Cinna Seifi
cinna.seifi@tu-clausthal.de

Jürgen Zimmermann
juergen.zimmermann@tu-clausthal.de

[1] Institute of Management and Economics, Operations Research Group, Clausthal University of Technology, 38678 Clausthal-Zellerfeld, Germany

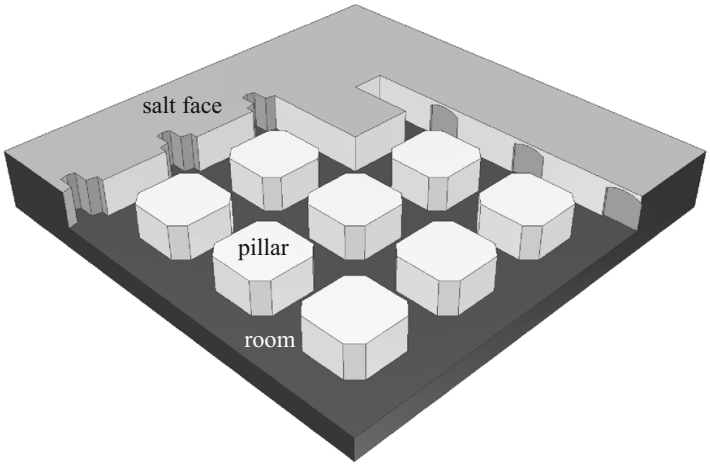

Fig. 1 Grid structure caused by the room-and-pillar mining method

1 Introduction to potash mining

Potash represents mined and manufactured salts that contain the element potassium (K) in water-soluble form. Main locations where potash is currently being mined are Canada, Russia, Belarus, China, and Germany. Typically, potash mines are underground mines (in contrast to open pits), since this kind of ore is generally found in deep deposits. The deposits are a naturally occurring mixture of potassium chloride (KCl) and sodium chloride (NaCl). In above-ground processing plants, the KCl is separated by flotation, recrystallization, or electrostatic separation. The majority of KCl produced is used for agricultural fertilizers. Furthermore, it is an input in the chemical, medical as well as human and animal food-processing industry (Chesworth 2008; USGS 2011).

For flat-bedded deposits of limited thickness (typical for potash, coal, or limestone), the *room-and-pillar mining method* is generally applied. Using the method, material is extracted across a horizontal plane and pillars, arranged in regular patterns, are left for support purposes. Thus, a grid-like structure is formed, as demonstrated in Fig. 1. The distance in which pillars are arranged depends on the depth of excavation and the thermal structure. The paths in *mining direction*, i.e., the direction in which the excavation is executed, are often broader than the paths parallel to the salt face. In order to expose the rooms, material is blasted in *blocks* (each of which consists of a volume of material and the corresponding mineral properties) and then removed via trackless loaders. Potash contained in pillars is unrecoverable, even if pillars contain valuable material (Hamrin 2001).

During excavation of potash, several miners work at the salt face, where the *drilling and blasting technique* is typically used. The technique provides that the miners drill holes and fill them with explosive substances. After a subsequent detonation, the crude material is delivered to a tipple, where the lumps are broken by a crushing machine. Then, the material is carried to the shaft on a conveyor belt or to an interim storage facility (a so-called bunker). Figure 2 depicts a ground plan with four *underground locations*, in which work can proceed simultaneously. Each underground location is characterized by a path in mining direction (illustrated by an arc) and a neighboring pillar row. The sequence of block excavation in location 4 is demonstrated on the right

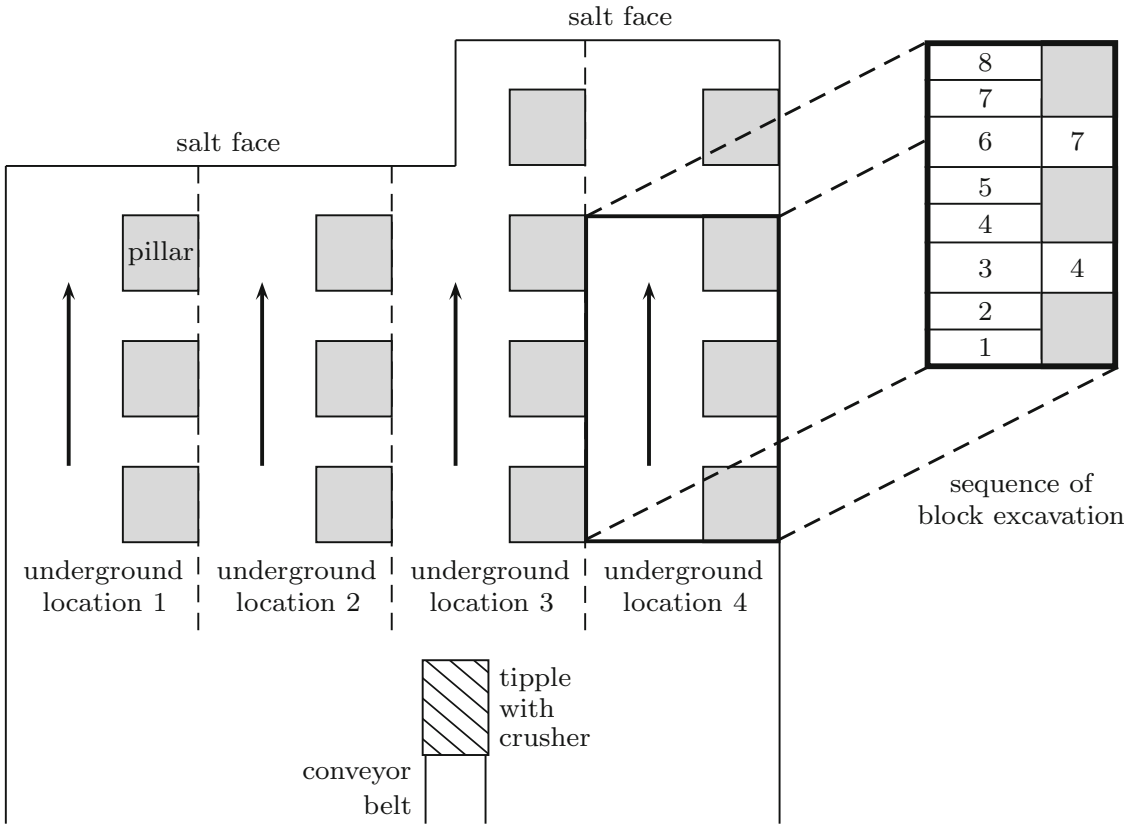

Fig. 2 Ground plan and schematic demonstration of block excavation

hand side of Fig. 2, where block 1 is processed first, then block 2, block 3 etc., one after another. Only if all relevant processing steps of the current block are executed, the work on the next block can be started. In addition to the "linear" (or chain-like) processing of blocks, a "lateral" processing must further be performed at regular intervals in order to insert transits between pillars and to create the preferred grid structure. If possible, blocks in both directions (i.e., linear and lateral) will be processed simultaneously. For example, the two blocks with number 4 are excavated at the same time.

The excavation of one block in a potash mine, where the drilling and blasting technique is applied, typically involves the following nine consecutive steps (K+S AG 2015):

(1) scaling of mine roof and side walls,
(2) removing the scaled material,
(3) bolting of roof with anchors,
(4) drilling large diameter boreholes,
(5) removing the drilling dust,
(6) drilling blast holes,
(7) filling blast holes with explosive substances,
(8) blasting,
(9) transporting broken material to the tipple.

In the first step, loose rock fragments on the mine roof or side walls are carefully detached using *scaling machines*. Afterwards, the scaled material is removed with *small loaders* (step 2). In the third step, *drilling trucks* drill holes for roof anchors

(threaded rods up to 1.2 m in length). The roof anchors are installed onto the roof structure in order to bind the salt layers together and give them a greater degree of stability. Then, large *drill jumbos* are used to drill three adjacent horizontal boreholes with a diameter of 0.28 m and a length of 7 m (step 4). The large boreholes act on the one hand as a direction guideline for the drilling of blast holes and on the other hand as a collecting area for material generated during blasting. The resultant dust as well as the loose material are removed with *small loaders* (step 5) to improve work conditions and to increase productivity in subsequent steps. In the sixth step, a *computer-controlled drilling machine* is used to drill about 60 holes with a diameter of 0.03 m 7 m deep into the rock according to a set plan. The blast holes are then filled with explosive substances carried by *blast trucks* (step 7). The blasting always takes place between work shifts, when one shift has left the salt face and the next is waiting for the elevator to go down (step 8). After the detonation, *sheltered trucks* with shovels take the crude material away from the salt face and deliver it to the tipple. Steps (1)–(9) repeat and can therefore be treated as a *production cycle*. Each step of the production cycle (except the blasting) requires a specific machine type that only exists in a limited number. Thus, the availability of machines has to be considered as an important restriction in the block excavation planning process of an underground mine.

In order to guarantee safeness for all miners, the production cycle should, once started, be performed within a certain time interval. Hence, after completing the first step, steps (2)–(6) have to be finished within $\tau > 0$ time units. If τ is achieved or exceeded with the completion of step i, $(2) \leq i \leq (6)$, a security precaution is made after i in which the roof is scaled once more, i.e., the first step is revisited (cf. Fig. 3). Thus, the roof and side walls are controlled continuously, so that loose material does not become detached unexpectedly. When the scaling process is finished, step $i + 1$ and the succeeding steps of the production cycle are executed.

The time period, in which we check the achieving or exceeding of τ, lies between the end of the first step and the end of the sixth step, i.e., steps (7)–(9) are not considered in the calculation. This decision is based on the following reasons: a return to step (1) after filling the blast holes is unnecessary, since the blasting is performed in step (8). Subsequently, the crude material is delivered to the tipple with "sheltered" trucks (diesel or electric) and so, the miners are adequately protected against the possibility of rockfall.

The described scheduling problem in potash mining, where a block excavation sequence has to be found taking into account a limited number of machines as well as a possible return to step (1) for safety purposes, is part of an industry project funded by a German potash provider. The aim is to minimize the maximum completion time of excavations (i.e., the makespan or the schedule length). In what follows, we take advantage of the fact that the problem can be transformed into a hybrid flow shop

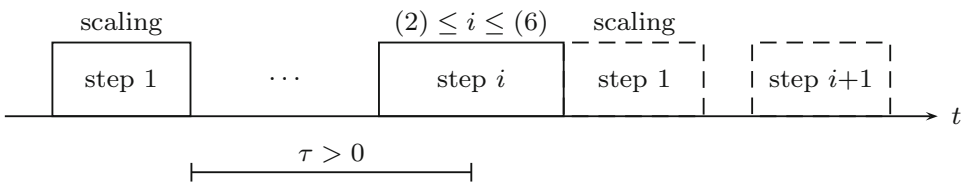

Fig. 3 Execution of steps over the time axis, where τ time units are exceeded and step (1) is revisited

scheduling problem (cf. Sect. 2). However, approaches known from the literature cannot be used directly in order to solve the problem, as they do not consider all described constraints, in particular the situation-related revisit of step (1). Since the consideration of a time-based threshold τ is also relevant for applications in coal, zinc, and limestone mining as well as for scheduling problems in the chemical industry, where products are included with a limited shelf life, the problem is of fundamental importance from a theoretical as well as practical point of view. Preliminary tests have shown that a targeted planning process can lead to a significant makespan reduction compared to the current manual planning process (the makespan could be halved in specific problem situations). As a consequence, there is a need for suitable exact and heuristic algorithms that can be applied to solve the problem under consideration.

The remainder of the paper is organized as follows: Sect. 2 describes the proposed scheduling problem in more detail. In order to show the originality and the characteristics of the problem, an overview of the literature on corresponding hybrid flow shop scheduling problems and on scheduling problems in underground mining is presented. In Sect. 3, a mathematical model is formulated, which can be given to a standard solver (e.g., CPLEX). Based on the model, we proceed to describe methods for improving the quality of the model in terms of computation time and solution gap (cf. Sect. 4). Section 5 is devoted to a basic and an advanced multi-start algorithm that efficiently construct near-optimal solutions. Furthermore, a modified Giffler–Thompson procedure for constructing active schedules is described. The results of computational experiments, where realistic instances are considered, are given in Sect. 6. Finally, conclusions are presented in Sect. 7.

2 Problem specification and related literature

The problem of finding a sequence in which a predefined set of blocks should be removed from the mine, where a limited number of vehicles or mobile machines is available, can be treated as a special variant of the *hybrid flow shop* (HFS) *scheduling problem*. Basic HFS scheduling problems are characterized by (see, e.g., the surveys of Ruiz and Vázquez-Rodríguez 2010, as well as Ribas et al. 2010):

- $k \geq 2$ production stages separated by unlimited intermediate buffers (problems are usually classified into two-stage, three-stage, and k-stage HFS scheduling problems with $k > 3$),
- $|M^k| \geq 1$ parallel machines at stage k, where at least one stage includes more than one machine,
- non-preemptable jobs consisting of several operations that have to be processed on machines, with positive processing times and no setup times, following the same production flow, where all stages have to be visited (problems in which jobs can skip stages are named flexible HFS scheduling problems), and
- a production environment in which each machine at each stage can process at most one job at a time and each job can be processed by at most one machine at a time (problems in which a job at a certain stage requires more than one machine are named HFS scheduling problems with multiprocessor tasks).

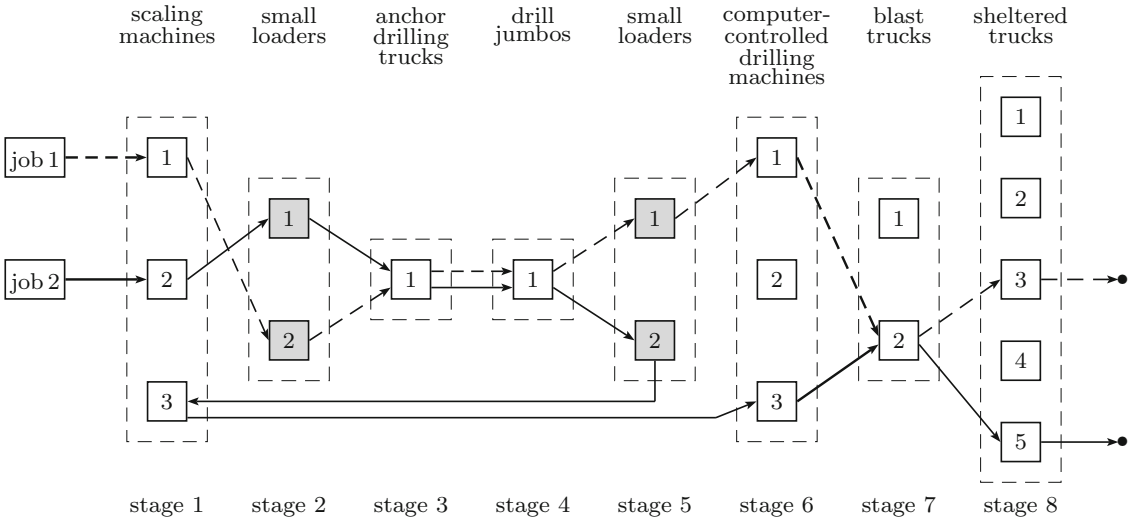

Fig. 4 Production environment of the proposed HFS scheduling problem

The proposed scheduling problem in underground mining is a k-stage HFS scheduling problem if we identify

- the production stages with steps (1)–(7) and step (9) of our production cycle,
- the machines at each stage with the underground machines used in the different steps (e.g., step 1 requires scaling machines, step 2 small loaders), and
- a job with a block excavation in an underground location.

The resulting production environment in which block excavations are executed consists of eight stages and is depicted in Fig. 4. Step (8) must not be considered in the environment, since the blasting does not need any vehicle or machine. Note that the depicted numbers of machines per stage (1–5 parallel machines) reflect realistic sizes for a mining district (or region) with up to 30 underground locations.

Almost all steps of our production cycle require a different set of special mobile machines. Only steps (2) and (5) make use of the same small loaders (given in gray color in Fig. 4) in order to remove the scaled material or the drilling dust. Hence, a job may revisit the same machines in the HFS and a *reentrant* problem has to be considered. Repeated use of the same machines by the same job means that there may be resource conflicts among jobs at different levels in the process. Moreover, if τ time units are achieved or exceeded with the completion of step i, $(2) \leq i \leq (6)$, the respective job has to revisit the first stage (scaling machines) for safety reasons, i.e., a situation-related reentry appears. We assume that a situation-related reentry may happen only once, since a practical relevant value of τ is significantly higher than the minimum processing time of the whole production cycle. Figure 4 shows the paths of two jobs through the system of machines, where job 2 performs a situation-related reentry after leaving stage 5. At the revisit of stage (1), the same (as in the previous path) or a different machine can be used.

Usually, the machine fleet consists of underground machines with different features (e.g., shovel volume, exhaust gas emission, feed rate). Therefore, the processing time $p_{jmk} \in \mathbb{N}$ of job j at stage k depends on the specific machine m within the stage, i.e., *unrelated* parallel machines must be taken into account.

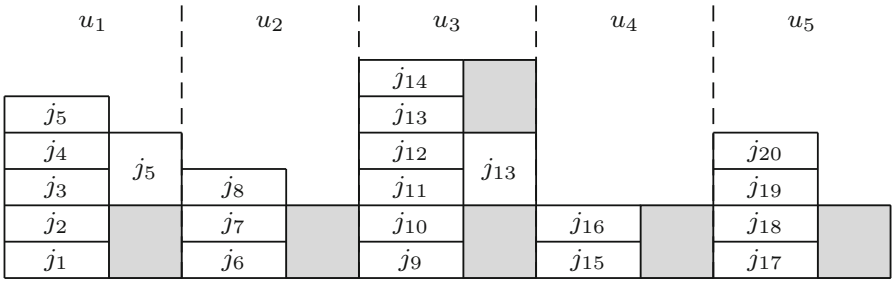

Fig. 5 Mining district with five underground locations $u_1, \ldots, u_5$ and 2–6 blocks per underground location

In order to excavate one block of a certain underground location, e.g., block j_2 at location u_1 (cf. Fig. 5), it is necessary that all steps of the preceding production cycle have been finished, i.e., the preceding block j_1 is completely removed. Hence, *precedence constraints* between jobs of an underground location exist and only a subset of jobs (here jobs j_1, j_6, j_9, j_{15}, and j_{17}) is available for excavation at time zero. New jobs appear dynamically, i.e., at each point in time, where a block excavation is completed and the block just removed is not the last block in an underground location.

In summary, we obtain a k-stage HFS scheduling problem with reentry, unrelated parallel machines, and precedence constraints between jobs. HFS are usually found in common manufacturing environments for products like concrete blocks (Grabowski and Pempera 2000), circuit boards (Jin et al. 2002), label stickers (Lin and Liao 2003), steel (Voß and Witt 2007; Kreutz et al. 2000), or solar cells (Chen et al. 2013). Furthermore, examples are found in non-manufacturing areas like container handling systems (Chen et al. 2006, 2007; Fereidoonian and Mirzazadeh 2011). Gupta (1988) showed that HFS scheduling problems restricted to two processing stages, even for the case in which one stage contains one machine and the other stage two machines, are $\mathcal{NP}$-hard in the strong sense.

In what follows, we confine our literature review on the one hand to k-stage HFS scheduling problems containing reentry, unrelated parallel machines, and job-precedences and on the other hand to scheduling problems in underground mining.

Choi et al. (2005) proposed a 4-stage HFS scheduling problem with identical parallel machines. Two types of jobs are considered: jobs that can be completed after being processed once at the stages and jobs with *reentrant* flows, i.e., jobs that should visit processing stages twice. For each job, a due date is given and the objective is to minimize the total tardiness of jobs. The problem is solved using list-scheduling algorithms (also known as dispatching rule-based or priority rule-based algorithms), where the job with highest priority among all jobs at a stage is assigned to the idle machine with highest priority at the stage. Choi et al. (2011) considered a reentrant HFS scheduling problem with five stages and identical parallel machines per stage. In reentrant flows, each job may visit serial stages two or more times. Multiple performance measures are considered: system throughput, mean flow time, mean tardiness, and the number of tardy jobs. A real-time scheduling mechanism is suggested in which a decision tree is used to select an appropriate dispatching rule. Cho et al. (2011) developed a Pareto genetic algorithm with local search strategies in order to solve a reentrant HFS scheduling problem with $k \geq 5$ stages, where each stage consists of identical parallel machines. The objectives are to minimize the makespan and to minimize the total tardiness. A solution is represented by a permutation of n jobs which indicates the

sequence of jobs to be considered for scheduling at the first stage. Afterwards, jobs are ordered according to non-decreasing completion times at the stages.

Ruiz and Maroto (2006) proposed a HFS scheduling problem with $k \in \{5, 10, 20\}$ stages and *unrelated* parallel machines, sequence-dependent setup times, and machine eligibility. A genetic algorithm is presented, where each solution is encoded as a permutation of n jobs and evaluated using the makespan criterion. In production environments with unrelated parallel machines, the calculation of the makespan must be executed cautiously, since the assignment of a job to the first available machine, which is, e.g., a very slow machine, can result in a later completion time compared to other machines. Therefore, jobs are assigned to the machine that can finish the job as soon as possible at a given stage, taking into account, among other things, different processing speeds. A similar genetic algorithm is proposed by Yaurima et al. (2009) for the same problem with the additional consideration of limited buffers for storing work in progress. Xu and Wang (2011) considered 3- and 4-stage hybrid flow shops with two to four unrelated parallel machines per stage. In order to solve the problem, while trying to minimize the makespan, an evolutionary algorithm combining differential evolution and local search is applied. A real-valued $(k \times n)$-matrix is used to encode a solution, where element m_{kj} defines that job j is processed on machine number $\lfloor m_{kj} \rfloor$ at stage k.

Botta-Genoulaz (2000) studied the scheduling of jobs in a k-stage HFS ($k = 4, 5$) with identical parallel machines, when jobs are subject to *job-precedence* constraints, minimum time lags, setup and removal times, and due dates. The objective is to minimize the maximum lateness. The problem is solved in two steps: Sequencing jobs on k machines, applying pure flow shop heuristics, and assigning jobs to the machines at each stage, using a list-scheduling algorithm. Voß and Witt (2007) dealt with a production environment of a German steel manufacturer that consists of 16 stages with parallel machines. The machines are uniform, i.e., they run with different speeds. At all stages, breakdowns may occur due to maintenance activities or holidays. The problem contains due dates, precedence constraints among jobs, and a weighted tardiness objective function. A mathematical model based on the resource-constrained project scheduling problem is presented and a heuristic solution procedure using dispatching rules is applied.

The literature on underground mining (regardless of whether an unsupported, a supported, or a caving method is used) can be classified according to the planning horizon into strategic, tactical, and operational problems (see, e.g., the surveys proposed by Weintraub et al. 2007; Newman et al. 2010). Strategic problems may consider how to geographically position facilities such as mills or haulage systems. Tactical problems are needed to determine whether or not to mine a block in an underground location in a particular time period. Operational problems consider machine allocations and/or machine capacities and entail targeted dispatching rules. The focus of the paper on hand is to generate a so-called "master" plan for block excavations in which activities or jobs have to be assigned to machines. The problem under consideration can be categorized between the tactical and operational planning level. Therefore, the following literature review covers newer approaches to determine block excavation or activity sequences and to dispatch vehicles or machines.

Carlyle and Eaves (2001) proposed a mixed-integer programming model in order to analyze development and production scenarios for an underground platinum and palladium mine. The model takes as input the planned mine layout, the ore quality as well as the costs for mining activities (e.g., drilling and preparation) and produces as output a near-optimal *schedule of activities* that maximizes the discounted ore revenue over a given planning horizon (i.e., 10 and 20 quarters). Rahal et al. (2003) considered a kimberlite diamond mine with several mine sections consisting of blocks. Blocks are characterized by a specific ore quality and waste rate. The goal of the study is to find a production schedule (i.e., the tonnage to be excavated from blocks in planning periods) such that the deviation of the ideal depletion and the deviation from a target production rate is minimized. Further, minimum and maximum production and waste rates for blocks as well as precedence constraints between blocks are taken into account. A mixed-integer linear model is introduced and embedded in a rolling planning horizon approach. Sarin and West-Hansen (2005) investigated an underground coal mine consisting of sections processed with the three mining methods: longwall, room-and-pillar, and retreat mining. Given a mine layout, the problem is to schedule the mining of sections so as to maximize the net present value. A solution methodology based on Benders' decomposition is used to solve problems with a time horizon of 100 weeks. Newman and Kuchta (2007) formulated a multi-period mixed-integer linear program for iron ore production. The model determines whether or not to start mining an aggregated block in a given time period. The aim is to minimize deviations from planned production quantities. A decomposition method is applied, where a tractable model with aggregated periods is built. Then, using information gained from the aggregated model, the original model is solved. Nehring et al. (2010b) considered a production scheduling problem focusing on sublevel stoping operations, where a decision is made on which stope is excavated in which period. A mixed-integer linear programming model is presented in order to maximize the net present value. Martinez and Newman (2011) extended the model proposed by Newman and Kuchta (2007) by incorporating a finer level, where individual production blocks (instead of aggregated blocks) are applied. Instances with a time horizon of 42-months are solved using an optimization-based heuristic algorithm. Epstein et al. (2012) proposed a problem to optimize production plans for copper mines considering underground and open-pit ore deposits, multiple products, and multiple downstream processing plants. One major decision is how to allocate mineral from the different mines to the downstream processes. A general multi-commodity network flow problem is developed in order to maximize the net present value. Approximate solutions are generated with rounding heuristics. Finally, O'Sullivan and Newman (2015) pursued the objective of determining a schedule for activities (e.g., block excavation, pillar extraction and backfilling) in a lead and zinc mine in order to maximize the quantity of metal produced. A mathematical formulation is provided and exact as well as heuristic methods are used to reduce the problem size. Instances with a time horizon of up to 156 weeks are solved.

Gamache et al. (2005) considered the problem of *dispatching*, routing and scheduling load-haul-dump *vehicles*, whenever they need to be assigned to a new task. Each dispatching decision is based on a predetermined criterion, e.g., minimizing cycle time or waiting time, and takes into account the current excavation progress of the mine, the current traffic on all bi-directional road segments, and operational equip-

ment constraints. To solve the problem, a shortest-path algorithm is presented. Later on, Beaulieu and Gamache (2006) proposed an enumeration algorithm based on dynamic programming for tackling the aforementioned fleet management problem. Saayman et al. (2006) introduced the problem of optimizing an autonomous vehicle dispatch system. The simulated environment is based on the layout of a diamond mine implementing a block cave mining technique. Five different dispatching strategies are used over one week in order to maximize the total tons produced, to keep the ore level between adjacent draw points as even as possible, and to have only few crusher shutdowns. Simsir and Ozfirat (2008) presented a simulation model in order to assess the efficiency of shearer-loaders, stage-loaders, crushers, and conveyor belts in a coal mine. Nehring et al. (2010a) considered a short-term scheduling and machine allocation problem, where the focus is on sublevel stoping operations. A mixed-integer linear programming model is proposed in order to minimize the deviation from targeted metal production. The model optimizes the shift based schedule and allows rapid equipment reassignment to take place as underground operating conditions change. In Nehring et al. (2012), the task of integrating short- and medium-term production plans is addressed by combining the short-term objective of minimizing deviations from production targets with the medium-term objective of maximizing net present value. Apart from dealing with the extraction of each stope and its subsequent ore movement, all production drilling and backfilling activities are incorporated into a mathematical model. Instances considering 18-month periods are solved using CPLEX.

The review shows that hybrid flow shop scheduling problems are studied extensively in the literature. However, there is still a shortcoming between previously published problems and the proposed scheduling problem in underground mining, since a new and complex combination of constraints is imposed. Furthermore, articles focusing on underground mining do not consider both block excavation sequences and machine allocations. Additionally, machine allocation problems include only one or a few machine type(s), neglect safety-related restrictions, and reposition machines by using enhanced shortest-path algorithms. To the best of our knowledge, the problem presented in this paper has not been studied in the literature and thus no exact or heuristic solution algorithm exists.

3 Model formulation

In order to describe the proposed problem of finding a block excavation sequence in underground mining more precisely, a mixed-integer linear model formulation is presented. The notation used in the optimization model is summarized in Table 1.

Objective function (1) denotes the makespan which is to be minimized. Constraints (2) specify the makespan as the latest completion time of a job at stage 8, i.e., the last stage in the production environment.

$$\text{Minimize} \quad C^{\max} \tag{1}$$

$$\text{subject to} \quad C^{\max} \geq C_{j8} \quad j \in J \tag{2}$$

Table 1 Sets, parameters, and variables (alphabetically ordered in each subsection)

Sets	
J	Set of jobs
J^{α}	Set of jobs that are the first jobs in an underground location, i.e., that are available at time zero, $J^{\alpha} \subseteq J$
J^{ω}	Set of jobs that are the last jobs in an underground location, $J^{\omega} \subseteq J$
J^{u}	Set of jobs that belong to underground location $u \in U$
K	Set of production stages, $\lvert K \rvert = 8$
K'	Set of production stages, where a situation-related reentry can occur, $K' = \{2, \ldots, 6\}$
M^{k}	Set of parallel machines at stage $k \in K$
U	Set of underground locations, where each underground location $u \in U$ contains at least one job
Parameters	
$\hat{C}_j$	Completion time of job $j \in J^{\alpha}$ with $\kappa_j > 1$ at the first production stage; $\hat{C}_j \leq 0$, since $\hat{C}_j$ lies before or at the beginning of the planning period
κ_j	Production stage at which job $j \in J$ will be processed at the beginning of the planning period, i.e., $1 \leq \kappa_j \leq 8$ for all $j \in J^{\alpha}$ and $\kappa_j = 1$ for all $j \in J \backslash J^{\alpha}$
p_{jmk}	Processing time of job $j \in J$ on machine $m \in M^{k}$ at stage $k \in K$
τ	Time units that lie between the completion times at stages one and $k \in K'$, whose achieving or exceeding will result in a situation-related reentry
Θ	Suitably large number
z_{j,κ_j-1}	Reference point for all jobs $j \in J$ in order to identify a situation-related re-entry (if there is one) at stage $k \in \{\max\{\kappa_j, 2\}, \ldots, 6\}$; we set $z_{j,\kappa_j-1} := 0$
Decision variables	
$C^{\max}$	Makespan
C_{jk}	Completion time of job $j \in J$ at stage $k \in \{\kappa_j, \ldots, 8\}$
C_{j,κ_j-1}	Earliest start time of job $j \in J$ at production stage κ_j
$C_j^{\circlearrowleft}$	Completion time of job $j \in J$ at the first stage if j is processed for the second time
x_{ijk}	1, if job $i \in J$ precedes job $j \in J$ at stage $k \in \{3, 4, 6, 7, 8\}$: $i < j, k \geq \max\{\kappa_i, \kappa_j\}$; 0, otherwise
$\hat{x}_{ijkk'}$	1, if job $i \in J$ (currently positioned at production stage k) is processed before job $j \in J$ (currently positioned at stage k'): $i < j, k, k' \in \{2, 5\}$; 0, otherwise
$x_{ij}^{\theta,\eta}$	1, if job $i \in J$ precedes job $j \in J$ at the first stage, where job i is processed for the first ($\theta = 1$) or the second ($\theta = 2$) time and job j is processed for the first ($\eta = 1$) or the second ($\eta = 2$) time: $i < j$; 0, otherwise
$\hat{y}_{jm}$	1, if job $j \in J$ is assigned to machine $m \in M^{1}$ at the first stage, but for the second time; 0, otherwise
y_{jmk}	1, if job $j \in J$ is assigned to machine $m \in M^{k}$ at stage $k \in \{\kappa_j, \ldots, 8\}$; 0, otherwise
z_{jk}	1, if τ time units are achieved or exceeded with the completion of job $j \in J$ at stage $k \in \{\max\{\kappa_j, 2\}, \ldots, 6\}$; 0, otherwise

Constraints (3) ensure that the completion time of job j at stage $k > \kappa_j$ ($k = \kappa_j$) is larger than or equal to the completion time (earliest start time) of j at the previous stage (at stage k) plus the processing time of j on the chosen machine at stage k.

$$C_{jk} \geq C_{j,k-1} + \sum_{m \in M^k} p_{jmk}\, y_{jmk} \quad j \in J;\ k \in \{\kappa_j, \ldots, 8\} \tag{3}$$

We assume that the jobs in a specific underground location are numbered consecutively. Figure 5 shows, for example, that jobs $j_1, \ldots, j_5$ belong to underground location u_1 and jobs j_6, j_7, j_8 belong to underground location u_2. Moreover, sets J^α and J^ω may be specified by $J^\alpha = \{j_1, j_6, j_9, j_{15}, j_{17}\}$ and $J^\omega = \{j_5, j_8, j_{14}, j_{16}, j_{20}\}$. Consequently, equalities (4) guarantee that the completion time of job j at stage 8 is equal to the earliest start time of the succeeding job $j+1$ at the first stage, where both jobs are part of the same underground location.

$$C_{j+1,0} = C_{j8} \quad j \in J \backslash J^\omega \tag{4}$$

Equalities (5) ensure that each job j is assigned to exactly one machine at each stage $k \geq \kappa_j$. Constraints (6)–(9) guarantee that decision variables $z_{jk}, z_{j,k+1}, \ldots, z_{j6}$ are equal to one if the difference between the completion time of job j at the first stage and the completion time of j at stage $k \in \{\max\{\kappa_j, 2\}, \ldots, 6\}$ achieves or exceeds τ time units. Hence, a situation-related reentry occurs between stages 2 and 6 if decision variable z_{j6} is equal to one. In that case, job j returns to the first stage and is there assigned to a machine $m \in M^1$ (cf. constraints (10)).

$$\sum_{m \in M^k} y_{jmk} = 1 \quad j \in J;\ k \in \{\kappa_j, \ldots, 8\} \tag{5}$$

$$\tau \leq C_{jk} - C_{j1} + \Theta(1 - z_{jk}) \quad j \in J : \kappa_j = 1;\ k \in K' \tag{6}$$

$$\tau > C_{jk} - C_{j1} - \Theta z_{jk} \quad j \in J : \kappa_j = 1;\ k \in K' \tag{7}$$

$$\tau \leq C_{jk} - \hat{C}_j + \Theta(1 - z_{jk}) \quad j \in J^\alpha : \kappa_j > 1;\ k \in \{\max\{\kappa_j, 2\}, \ldots, 6\} \tag{8}$$

$$\tau > C_{jk} - \hat{C}_j - \Theta z_{jk} \quad j \in J^\alpha : \kappa_j > 1;\ k \in \{\max\{\kappa_j, 2\}, \ldots, 6\} \tag{9}$$

$$z_{j6} = \sum_{m \in M^1} \hat{y}_{jm} \quad j \in J : \kappa_j \leq 6 \tag{10}$$

The difference $(z_{j,k-1} - z_{jk})$ of decision variables is equal to -1 if a situation-related reentry appears for job j at stage k. Then, disjunctive constraints (11) and (12) determine the completion time of the respective job j at the first stage, where j is processed for the second time, and the completion time of job j at stage $k+1$, where j continues its production cycle afterwards.

$$C_j^{\circlearrowleft} \geq C_{jk} + \sum_{m \in M^1} p_{jm1}\, \hat{y}_{jm} - \Theta(1 + z_{j,k-1} - z_{jk}) \quad j \in J;\ k \in \{\max\{\kappa_j, 2\}, \ldots, 6\} \tag{11}$$

$$C_{jk} \geq C_j^{\circlearrowleft} + \sum_{m \in M^k} p_{jmk}\, y_{jmk} - \Theta(1 + z_{j,k-2} - z_{j,k-1}) \quad j \in J;\ k \in \{\max\{\kappa_j + 1, 3\}, \ldots, 7\} \tag{12}$$

Resource precedence relations identify the sequence in which jobs are processed on the same machines. In order to include the characteristics of different stages (i.e., at stages 3, 4, 6, 7, 8 jobs are processed once, at stages 2, 5 the same machines may

be revisited, and at stage 1 jobs may be processed twice due to a situation-related reentry), we distinguish between three categories of constraints.

The excavation of blocks is performed in such a way that at most one job $j \in J^u$ that belongs to underground location u is processed at a specific point in time. As a result, only jobs that belong to different underground locations, say, e.g., u and $\hat{u}$, compete for machines at stages $k \in \{3, 4, 6, 7, 8\}$. Hence, constraints (13) and (14) specify the completion time of job j for the case in which job i precedes job j.

$$C_{ik} \geq C_{jk} + p_{imk} - \Theta(2 - y_{imk} - y_{jmk} + x_{ijk}) \quad i \in J^u, j \in J^{\hat{u}} : i < j; u, \hat{u} \in U : u \neq \hat{u}; m \in M^k; k \in \{3, 4, 6, 7, 8\} : k \geq \max\{\kappa_i, \kappa_j\} \tag{13}$$

$$C_{jk} \geq C_{ik} + p_{jmk} - \Theta(3 - y_{imk} - y_{jmk} - x_{ijk}) \quad i \in J^u, j \in J^{\hat{u}} : i < j; u, \hat{u} \in U : u \neq \hat{u}; m \in M^k; k \in \{3, 4, 6, 7, 8\} : k \geq \max\{\kappa_i, \kappa_j\} \tag{14}$$

With inequalities (15) and (16), the completion time of job j at stage $k \in \{2, 5\}$, where the same mobile machines are used, is determined analogously for the situation in which job i precedes job j.

$$C_{ik} \geq C_{j\tilde{k}} + p_{imk} - \Theta(2 - y_{imk} - y_{jm\tilde{k}} + \hat{x}_{ijk\tilde{k}}) \quad i \in J^u, j \in J^{\hat{u}} : i < j; u, \hat{u} \in U : u \neq \hat{u}; m \in M^k \cup M^{\tilde{k}}; k, \tilde{k} \in \{2, 5\} : k \geq \kappa_i, \tilde{k} \geq \kappa_j \tag{15}$$

$$C_{j\tilde{k}} \geq C_{ik} + p_{jm\tilde{k}} - \Theta(3 - y_{imk} - y_{jm\tilde{k}} - \hat{x}_{ijk\tilde{k}}) \quad i \in J^u, j \in J^{\hat{u}} : i < j; u, \hat{u} \in U : u \neq \hat{u}; m \in M^k \cup M^{\tilde{k}}; k, \tilde{k} \in \{2, 5\} : k \geq \kappa_i, \tilde{k} \geq \kappa_j \tag{16}$$

In order to determine the completion times of jobs at the first stage, four different cases have to be considered. Inequalities (17) and (18) (inequalities (23) and (24)) state the case in which job i precedes job j and both jobs are processed for the first (second) time. Constraints (19) and (20) (constraints (21) and (22)) specify the event that job i precedes job j, where j (i) is processed for the first and i (j) is processed for the second time.

$$C_{i1} \geq C_{j1} + p_{im1} - \Theta(2 - y_{im1} - y_{jm1} + x_{ij}^{1,1}) \quad m \in M^1; i, j \in J : i < j, \kappa_i = \kappa_j = 1 \tag{17}$$

$$C_{j1} \geq C_{i1} + p_{jm1} - \Theta(3 - y_{im1} - y_{jm1} - x_{ij}^{1,1}) \quad m \in M^1; i, j \in J : i < j, \kappa_i = \kappa_j = 1 \tag{18}$$

$$C_i^{\circlearrowleft} \geq C_{j1} + p_{im1} - \Theta(2 - \hat{y}_{im} - y_{jm1} + x_{ij}^{2,1}) \quad m \in M^1; i, j \in J : i < j, \kappa_i \leq 6, \kappa_j = 1 \tag{19}$$

$$C_{j1} \geq C_i^{\circlearrowleft} + p_{jm1} - \Theta(3 - \hat{y}_{im} - y_{jm1} - x_{ij}^{2,1}) \quad m \in M^1; i, j \in J : i < j, \kappa_i \leq 6, \kappa_j = 1 \tag{20}$$

$$C_{i1} \geq C_j^{\circlearrowleft} + p_{im1} - \Theta(2 - y_{im1} - \hat{y}_{jm} + x_{ij}^{1,2}) \quad m \in M^1; i, j \in J : i < j, \kappa_i = 1, \kappa_j \leq 6 \tag{21}$$

$$C_j^{\circlearrowleft} \geq C_{i1} + p_{jm1} - \Theta(3 - y_{im1} - \hat{y}_{jm} - x_{ij}^{1,2})$$
$$m \in M^1;\ i, j \in J : i < j, \kappa_i = 1, \kappa_j \leq 6 \tag{22}$$
$$C_i^{\circlearrowleft} \geq C_j^{\circlearrowleft} + p_{im1} - \Theta(2 - \hat{y}_{im} - \hat{y}_{jm} + x_{ij}^{2,2})$$
$$m \in M^1;\ i, j \in J : i < j, \kappa_i, \kappa_j \leq 6 \tag{23}$$
$$C_j^{\circlearrowleft} \geq C_i^{\circlearrowleft} + p_{jm1} - \Theta(3 - \hat{y}_{im} - \hat{y}_{jm} - x_{ij}^{2,2})$$
$$m \in M^1;\ i, j \in J : i < j, \kappa_i, \kappa_j \leq 6 \tag{24}$$

Finally, constraints (25) to (32) impose the domains of decision variables.

$$C_{jk} \geq 0 \quad j \in J;\ k \in \{\kappa_j - 1, \ldots, 8\} \tag{25}$$
$$C_j^{\circlearrowleft} \geq 0 \quad j \in J \tag{26}$$
$$x_{ijk} \in \{0, 1\} \quad i, j \in J : i < j;\ k \in \{3, 4, 6, 7, 8\} : k \geq \max\{\kappa_i, \kappa_j\} \tag{27}$$
$$\hat{x}_{ijk\tilde{k}} \in \{0, 1\} \quad i, j \in J : i < j;\ k, \tilde{k} \in \{2, 5\} : k \geq \kappa_i, \tilde{k} \geq \kappa_j \tag{28}$$
$$x_{ij}^{\theta,\eta} \in \{0, 1\} \quad i, j \in J : i < j;\ \theta, \eta \in \{1, 2\} \tag{29}$$
$$\hat{y}_{jm} \in \{0, 1\} \quad j \in J;\ m \in M^1 \tag{30}$$
$$y_{jmk} \in \{0, 1\} \quad j \in J;\ m \in M^k;\ k \in \{\kappa_j, \ldots, 8\} \tag{31}$$
$$z_{jk} \in \{0, 1\} \quad j \in J;\ k \in \{\max\{\kappa_j, 2\}, \ldots, 6\} \tag{32}$$

A solution of the problem can be visualized in a Gantt-chart, where $C^{\max} := \max_{j \in J^\omega} C_{j8}$ indicates the makespan. Figure 6 depicts a Gantt-chart for a problem instance with three underground locations, u_1, u_2, u_3, seven jobs, $j_1, \ldots, j_7$, and $(|M^k|)_{k \in K} = (1, 2, 1, 1, 2, 1, 2, 1)$ machine(s) at stages 1–8. Each job is placed over the time axis and the processing times of jobs $j_1, j_4, \ldots, j_7$ at production stages are individually given. Jobs j_1 and j_6 perform a situation-related reentry, i.e., stage 1 is revisited.

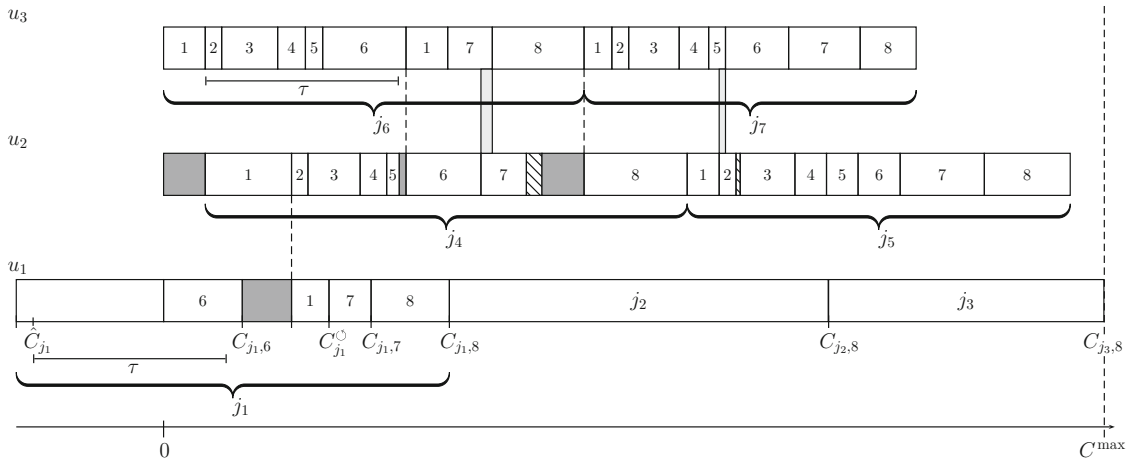

Fig. 6 Feasible solution for an instance with three underground locations

The set of first jobs, that are available at time zero, is $J^{\alpha} = \{j_1, j_4, j_6\}$ with $\kappa_{j_1} = 6$ and $\kappa_{j_4} = \kappa_{j_6} = 1$. The first stage involves only one machine and job j_6 is processed first. Hence, job j_4 has to wait in front of the machine until j_6 is completed and j_1 has to wait until j_4 is completed (the waiting times are depicted as gray boxes). Additionally, job j_4 has to wait in front of stages 6 and 8 with $|M^6| = |M^8| = 1$, until job j_6 is finished. Stage 7 involves two machines and therefore jobs j_4 and j_6 can be processed simultaneously for some time units (the time units are depicted as light-gray box). Since j_6 arrives first, it is assigned to the fastest machine. Thus, j_4 has to use a slower machine and has to accept a longer duration (the additional duration is demonstrated as hatched box). The same holds true for jobs j_5 and j_7; since j_7 is processed on the fastest machine at stage 5 (which is equal to stage 2), job j_5 has to utilize a slower one.

4 Improvements in modeling

In order to find exact solutions to the problem under consideration, the model presented in Sect. 3 can be tackled with standard solvers (e.g., CPLEX). Since already for small-scale instances running times are quite long, we provide the solver with supplementary knowledge such as lower and upper bounds for the makespan as well as additional constraints (cf. Sect. 4.1–4.3). Preliminary tests have shown that the following modeling techniques result in significant improvements in terms of computation time and solution gap.

4.1 Initial solution and makespan upper bound

The solution process of a standard solver, which uses a branch-and-bound or branch-and-cut algorithm, is usually faster if a feasible initial solution (i.e., a leaf of the enumeration tree) is provided. Using the multi-start procedure described in Sect. 5.2, we are able to determine a good initial solution $(\widetilde{C}, \widetilde{x}, \widetilde{y}, \widetilde{z})$ and a corresponding upper bound $\widetilde{C}^{\max}$ on the objective function value. Since a strong linear programming (LP) relaxation will benefit from choosing a small "large number" in disjunctive constraints, we set $\Theta := \widetilde{C}^{\max}$.

4.2 Makespan lower bounds

Good makespan lower bounds inform about the quality of upper bounds and can serve to prune parts of the enumeration tree. In what follows, we introduce four different lower bounds. One specifies the time it takes at least to process blocks of an underground location (lower bound LB^0) and the other three determine in a stage-based perspective a waiting time, a stagnation time as well as the time it takes at least to process all jobs (lower bounds LB, $\overline{\mathrm{LB}}$, and $\widehat{\mathrm{LB}}$).

Precedence constraints exist between jobs that belong to one underground location (cf. Fig. 5). As a precedence relation for a job $j \in J^u$, with $j \notin J^{\alpha}$, is given by an immediate predecessor and for a job $j \in J^u$, with $j \notin J^{\omega}$, by an immediate successor, the precedence constraints can be interpreted as "chain constraints" for underground locations. Considering the chain constraints, a *chain-based lower bound* LB^0 may be

calculated by the longest time it takes to handle the chain of an underground location, assuming that all jobs are processed on the fastest machines. Hence, we obtain

$$\mathrm{LB}^0 := \max_{u \in U} \left\{ \sum_{j \in J^u} \sum_{k=\kappa_j}^{8} \min_{m \in M^k} \{p_{jmk}\} \right\}.$$

An "unavoidable" situation-related reentry can be integrated in LB^0 by the addition of the term $\Upsilon_j \min_{m \in M^1}\{p_{jm1}\}$, where indicator Υ_j is equal to 1, if condition $\sum_{k \in \{\max\{\kappa_j,2\},\ldots,6\}} \min_{m \in M^k}\{p_{jmk}\} \geq \tau$ is satisfied for job j, and 0, otherwise. Since the benchmark instances provided by our cooperation partner and described in Sect. 6.1 do not involve unavoidable situation-related reentries, we neglect hereafter the consideration of Υ_j.

Santos et al. (1995) considered a common HFS with identical parallel machines and computed a *stage-based lower bound*. In what follows, the determination of the described stage-based lower bound is enhanced in such a way that it is suitable for the problem under consideration with unrelated parallel machines and precedence constraints among jobs. Further, we assume that each stage must be passed by at least one job, i.e., at least one job j with $\kappa_j \leq k$ exists for all $k \in K$. Initially, a "head" H_{jk} is calculated in order to determine the minimum time that must elapse before job j reaches stage k. The first jobs that will pass stage k are either jobs $j \in J^\alpha$ with $k \geq \kappa_j$ or jobs $j+1$ (if there are any) with $j \in J^\alpha$ and $k < \kappa_j$. Therefore, heads can be calculated as follows:

$$H_{jk} := \sum_{h=\kappa_j}^{k-1} \min_{m \in M^h} \{p_{jmh}\} \quad j \in J^\alpha; k \in \{\kappa_j, \ldots, 8\} \tag{33}$$

$$H_{j+1,k} := \sum_{h=\kappa_j}^{8} \min_{m \in M^h} \{p_{jmh}\} + \sum_{h=1}^{k-1} \min_{m \in M^h} \{p_{j+1,m,h}\} \quad \begin{array}{l} j \in J^\alpha : |J^u| \geq 2; \\ k \in \{1, \ldots, \kappa_j - 1\}. \end{array} \tag{34}$$

Jobs $j \in J^\alpha$ with $k < \kappa_j$ that belong to an underground location consisting of only one element (i.e., $|J^u| = 1$) must not be considered at stage k; as a result, we set $H_{jk} := \Theta$ (i.e., the head is equal to a suitably large number).

Afterwards, a "tail" T_{jk} is computed that contains information about the time that must at least elapse before a last job $j \in J^\omega$, departing stage $k \in \{\kappa_j, \ldots, 8\}$, leaves the system. Thus, the tails are computed by

$$T_{jk} := \sum_{h=k+1}^{8} \min_{m \in M^h} \{p_{jmh}\}.$$

Again, we set $T_{jk} := \Theta$ for jobs $j \in J^\omega \cap J^u$ with $k < \kappa_j$ and $|J^u| = 1$.

The head- and tail-values are ordered according to non-decreasing values; let $H_{k(i)}$ and $T_{k(i)}$ be the i-th smallest values. Assuming that $|M^k|$ machines are available at stage k, a lower bound may be specified by

$$\mathrm{LB} := \max_{k \in K} \left\{ \left\lceil \frac{1}{|M^k|} \left(A_k + \sum_{\substack{j \in J \\ \kappa_j \leq k}} \min_{m \in M^k} \{p_{jmk}\} + B_k \right) \right\rceil \right\}$$

with a waiting time of $A_k := \sum_{i=1}^{|M^k|} H_{k(i)}$ and a setting or stagnation time of $B_k := \sum_{i=1}^{|M^k|} T_{k(i)}$. The lower bound considers the $|M^k|$ jobs with the smallest processing times in the head and in the tail section, and the minimal processing times of all jobs at stage k. Note that if less than $|M^k|$ jobs exist, $|M^k|$ is reduced to the number of allocatable jobs. When dividing by $|M^k|$, it is assumed that there are at least $|M^k|$ machines at previous and subsequent stages (if fewer machines are available, times A_k and B_k would increase).

Hidri and Haouari (2011) extended the lower bound presented in Santos et al. (1995) by introducing different calculations of waiting and stagnation times. In order to compute the waiting time, an interstage h is investigated, with $1 \leq h < k$. Then, the $|M^h|$ earliest availability times $H_{h(1)}, \ldots, H_{h(|M^h|)}$ of machines at stage h are determined. These availability times are taken into account when the $|M^k|$ fastest jobs $\mu_1, \ldots, \mu_{|M^k|}$ are assigned to the machines at stage h. The assignment is performed according to the shortest processing time (SPT) first rule. The resulting completion times $\overline{C}_{\mu_1,h}, \ldots, \overline{C}_{\mu_{|M^k|},h}$ are the flow times that must at least elapse between time zero and the exit of job $\mu_i, i = 1, \ldots, |M^k|$, at stage h. For the problem at hand and jobs $j \in J$ with $k > \kappa_j$, the time lag between the exit at interstage h and the start of processing at stage k may be computed by

$$L_{jhk} := \sum_{l=\max\{\kappa_j, h+1\}}^{k-1} \min_{m \in M^l} \{p_{jml}\}.$$

We denote the i-th smallest value by $L_{hk(i)}$. Jobs $j \in J^\alpha$ with $k \leq \kappa_j$ receive a large value, i.e., $L_{jhk} := \Theta$. Consequently, we are able to determine the waiting time as $A_{hk} := \sum_{i=1}^{|M^k|} (\overline{C}_{\mu_i,h} + L_{hk(i)})$. Let I be a problem instance of the HFS scheduling problem. A corresponding symmetric instance I^{-1} is obtained by reversing the ordering of stages. Both instances I and I^{-1} have the same optimal makespan values and A^{-1}_{lk} is a valid value for the stagnation time $B_{kl}, k, l \in K: k < l \leq 8$. Hence, a lower bound can now be computed by

$$\overline{\mathrm{LB}} := \max_{\substack{h,k,l \in K \\ 1 \leq h < k < l \leq 8}} \left\{ \left\lceil \frac{1}{|M^k|} \left(A_{hk} + \sum_{\substack{j \in J \\ \kappa_j \leq k}} \min_{m \in M^k} \{p_{jmk}\} + B_{kl} \right) \right\rceil \right\},$$

Additionally, Hidri and Haouari (2011) described another possibility to determine the waiting and stagnation times. Thereby, a release date $r_j := H_{jh}$ is investigated for each job j at stage h (instead of availability times for machines). The jobs are

ordered according to non-decreasing r_j-values and separated in one-time-unit blocks. Then, these one-time-unit blocks are assigned to the existing $|M^h|$ machines at stage h using the shortest remaining processing time (SRPT) first rule. With the SRPT-rule, it may happen that the one-time-unit blocks of a job are not scheduled consecutively. For a problem instance with two jobs, one stage, two identical parallel machines, and the data $r_{j_1} = 0, r_{j_2} = 1, p_{j_1,m_i,k_1} = 5, p_{j_2,m_i,k_1} = 2, i = 1, 2$, the assignment of one-time-unit blocks is performed in such a way that a one-time-unit block of j_1 is scheduled at machine $m_i, i = 1, 2$, at time $t = 0$. At $t = 1$, both jobs are available and the remaining processing time of j_1 is equal to 3 while the remaining processing time of j_2 is equal to 2. Therefore, the one-time-unit blocks of j_2 are included. Job j_1 is completed at time $\check{C}_{j_1,k_1} = 4$ and j_2 at time $\check{C}_{j_2,k_1} = 2$. The procedure terminates iff $|M^k|$ jobs $\mu_1, \ldots, \mu_{|M^k|}$ are scheduled completely. Then, the waiting time is determined by $A'_{hk} := \sum_{i=1}^{|M^k|} (\check{C}_{\mu_i,h} + L_{hk(i)})$. The same bounding procedure can be applied on the symmetric instance for computing B'_{kl} and we obtain the lower bound

$$\widehat{\mathrm{LB}} := \max_{\substack{h,k,l \in K \\ 1 \le h < k < l \le 8}} \left\{ \left\lceil \frac{1}{|M^k|} \left(A'_{hk} + \sum_{\substack{j \in J \\ \kappa_j \le k}} \min_{m \in M^k} \{p_{jmk}\} + B'_{kl} \right) \right\rceil \right\}.$$

To summarize, a *stage-based lower bound* LB^1 for the proposed scheduling problem in underground mining may be determined by

$$\mathrm{LB}^1 := \max_{\substack{h,k,l \in K \\ 1 \le h < k < l \le 8}} \left\{ \left\lceil \frac{1}{|M^k|} \left(\mathcal{A}_{hk} + \sum_{\substack{j \in J \\ \kappa_j \le k}} \min_{m \in M^k} \{p_{jmk}\} + \mathcal{B}_{hk} \right) \right\rceil \right\},$$

with $\mathcal{A}_{hk} := \max\{A_k, A_{hk}, A'_{hk}\}$ and $\mathcal{B}_{hk} := \max\{B_k, B_{kl}, B'_{kl}\}$. Further, a good overall lower bound $\mathrm{LB}^{\max}$ can be computed as follows:

$$\mathrm{LB}^{\max} := \max\{\mathrm{LB}^0, \mathrm{LB}^1\}. \tag{35}$$

In order to illustrate the calculation of lower bounds, in particular of lower bound LB at stage 4 (LB_4 for short), we consider a mining district with 25 underground locations $u_1, \ldots, u_{25}$. Figure 7 depicts that locations $u_1, \ldots, u_6$ involve 1–5 blocks and locations $u_7, \ldots, u_{25}$ involve 2 blocks (they are identical in length, processing times, and mining progress). The production stage at which job $j \in J^\alpha$ will be processed at the beginning of the planning period is given by $\kappa_j \ge 1$. We assume that each stage contains exactly three identical parallel machines (i.e., $|M^k| = 3$ for all $k \in K$ and index m can be eliminated). Furthermore, the processing times of job j at stages $k \in K$ are identical, i.e., $p_{j1} = \cdots = p_{j8}$. Hence, index k can be neglected in the processing times. The table in the upper part of Fig. 7 shows the job processing times as well as the heads and tails for all relevant jobs at the fourth stage.

jobs	$j_1, \ldots, j_5$	j_6, j_7	j_8	$j_9, \ldots, j_{11}$	j_{12}	j_{13}	j_{14}	$j_{15}, \ldots, j_{52}$
p_j	5	10	19	20	26	10	15	30

jobs	j_1	j_6	j_8	j_9	j_{13}	j_{14}	$j_{2\xi+1}$, $\xi \in \{7, \ldots, 25\}$
H_{j4}	15	30	57	60	56	Θ	90

jobs	j_5	j_7	j_8	j_{11}	j_{13}	j_{14}	$j_{2\xi}$, $\xi \in \{8, \ldots, 26\}$
T_{j4}	20	40	76	80	40	Θ	120

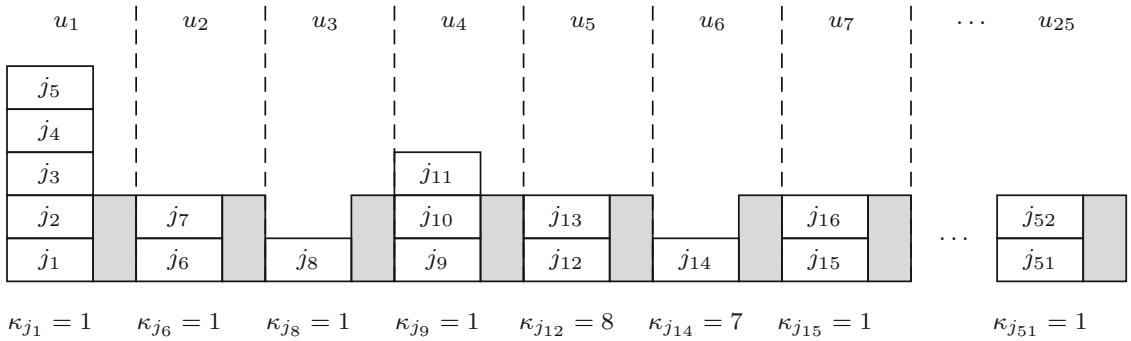

Fig. 7 Mining district with 25 underground locations $u_1, \ldots, u_{25}$

In order to calculate the heads for jobs j_1, j_6, j_8, j_9 and $j_{2\xi+1}$ with $\xi \in \{7, \ldots, 25\}$ condition (34) must be used, the head for j_{13} is calculated by condition (33), i.e., $H_{j_{13},4} = p_{j_{12}} + 3 \cdot p_{j_{13}} = 56$, and the head for j_{14} is set to Θ.

Using the data, lower bound LB_4 is computed by

$$\mathrm{LB}_4 = \left\lceil \frac{1}{3} \left(\underbrace{H_{j_1,4} + H_{j_6,4} + H_{j_{13},4}}_{A_4} + \sum_{\substack{j=1 \\ j \neq 12,14}}^{52} p_j + \underbrace{T_{j_5,4} + T_{j_7,4} + T_{j_{13},4}}_{B_4} \right) \right\rceil$$
$$= \left\lceil \frac{1}{3} (101 + 1274 + 100) \right\rceil = 492.$$

By determining lower bounds LB_k for all stages $k \in K$, we obtain the stage-based lower bound $\mathrm{LB} = \max_{k \in K} \mathrm{LB}_k = 492$. Furthermore, the chain-based lower bound can be calculated by $\mathrm{LB}^0 = p_{j_{15}} \cdot 8 + p_{j_{16}} \cdot 8 = 480$. Just for the sake of more information: $\overline{\mathrm{LB}} = 483$, $\widehat{\mathrm{LB}} = 484$, and thus $\mathrm{LB}^{\max} = 492$.

4.3 Domain reductions and additional constraints

A proper definition of domains of continuous decision variables helps the solver to early identify an empty solution space at an enumeration node. We consider decision variables C_{jk}, $j \in J, k \in \{\kappa_j, \ldots, 8\}$, and $C_j^{\circlearrowleft}$, $j \in J$, (cf. constraints (25) with $k \geq \kappa_j$ instead of $k \geq \kappa_j - 1$ and (26)) and compute tighter domains. The earliest completion time EC_{jk} of job $j \in J^u$ at stage $k \in \{\kappa_j, \ldots, 8\}$, can be obtained by

$$\mathrm{EC}_{jk} := \sum_{\substack{i \in J^u \\ i < j}} \sum_{h=\kappa_j}^{8} \min_{m \in M^h} \{p_{imh}\} + \sum_{h=\kappa_j}^{k} \min_{m \in M^h} \{p_{jmh}\}.$$

With a predefined upper bound $\widetilde{C}^{\max}$, the latest completion time LC_{jk} of job j at stage $k \in \{\kappa_j, \ldots, 8\}$, may be computed as follows:

$$\mathrm{LC}_{jk} := \widetilde{C}^{\max} - \sum_{\substack{i \in J^u \\ j<i}} \sum_{h \in K} \min_{m \in M^h} \{p_{imh}\} - \sum_{h=k+1}^{8} \min_{m \in M^h} \{p_{jmh}\}.$$

Thus, we obtain the following stronger constraints:

$$\mathrm{EC}_{jk} \leq C_{jk} \leq \mathrm{LC}_{jk} \quad j \in J; k \in \{\kappa_j, \ldots, 8\} \tag{36}$$

$$\mathrm{EC}_{j2} + \min_{m \in M^1} \{p_{jm1}\} \leq C_j^{\circlearrowleft} \leq LC_{j6} + \min_{m \in M^1} \{p_{jm1}\} \quad j \in J : \kappa_j \in \{1, 2\} \tag{37}$$

$$\mathrm{EC}_{j,\kappa_j} + \min_{m \in M^1} \{p_{jm1}\} \leq C_j^{\circlearrowleft} \leq LC_{j6} + \min_{m \in M^1} \{p_{jm1}\} \quad j \in J : 3 \leq \kappa_j \leq 6. \tag{38}$$

Additional constraints in the model can influence the behavior, scope, and interaction of the solver solution process. In particular, achieving a better result in terms of computation time is possible by adding the constraints

$$\sum_{k=\kappa_j}^{8} \sum_{m \in M^k} y_{jmk} = (8 - \kappa_j) + 1 \quad j \in J \tag{39}$$

$$\sum_{\substack{k \in \{3,4,6,7,8\} \\ k \geq \max\{\kappa_i, \kappa_j\}}} x_{ijk} \leq \min\{\gamma_i, \gamma_j\} \quad \begin{array}{c} i, j \in J : i < j, \text{ where } \gamma_i = 1(2 \mid 3 \mid 4 \mid 5) \\ \text{if } \kappa_i = 8(7 \mid 6 \text{ or } 5 \mid 4 \mid 3, 2 \text{ or } 1). \end{array} \tag{40}$$

Equalities (39) count the number of assignments to machines at stages. Since each job must be assigned to exactly one machine at a stage, the number of assignments must be equal to $(8 - \kappa_j) + 1$. If two jobs are processed on the same machine at stage $k \in \{3, 4, 6, 7, 8\}$, they must have the order "i before j" or "j before i". Hence, the number of those resource precedence relations between jobs i and j with $\kappa_i = \kappa_j = 1$ is at most 5, and the number decreases if one job, i.e., i or j, starts at a later stage at the beginning of the planning period (cf. inequalities (40)).

5 Solution procedure

In order to solve medium- and large-scale instances of the proposed underground mine scheduling problem, a construction procedure, which is based on priority rules, has been developed (cf. Sect. 5.1). The choice of a construction procedure or priority rule-based method, respectively, has been inspired by the promising results provided by Choi et al. (2005), Ruiz and Maroto (2006), Ruiz et al. (2008), Lee (2009), and Li et al. (2013). The construction procedure is embedded in a multi-start algorithm in order to generate different near-optimal solutions (cf. Sect. 5.2). Since the basic version of the construction procedure does not integrate conscious delays or waiting times in front of production stages, a more sophisticated method is developed in a second step (cf. Sect. 5.3). In addition, motivated by Kreutz et al.

(2000), Baykasoğlu et al. (2014), and Rossi et al. (2015), a modified version of the Giffler–Thompson algorithm is considered as a third heuristic solution procedure (cf. Sect. 5.4).

5.1 Construction procedure

The *construction procedure* schedules, at specific discrete points in time, all jobs that are eligible with respect to precedence and resource constraints. Hence, the approach has some similarity to the classical parallel schedule-generation scheme provided by Kelley (1963) as well as Bedworth and Bailey (1982), or to the corresponding priority rule-based method described by Kolisch (1996). The structure of the construction procedure is based on an *initialization*, an *outer loop*, and an *inner loop*.

In the *initialization*, a sequence of machines at stages $k \in K$ is determined for each job $j \in J$. Correspondingly, we order the machines $1, \ldots, |M^k|$ at stage k according to non-decreasing processing times; machine $m_{jk(i)}$ denotes the i-th fastest machine for job j at stage k. Afterwards, jobs $j \in J^\alpha$ that are first jobs in underground locations are considered, since only these jobs are available at time zero. Depending on the mining progress, each job $j \in J^\alpha$ will be processed at stage κ_j at the beginning of the planning period. Thus, j is inserted in an artificial buffer B_{κ_j} in front of stage κ_j.

In each iteration of the *outer loop*, the *schedule time* $t^* \geq 0$ is specified, i.e., the earliest point in time at which a job can be feasibly scheduled on an idle machine at some stage k^*. Note that schedule times t^* are monotonically non-decreasing. In the event that several jobs can be scheduled at t^* at different stages, stage k^* with the smallest index number is chosen. The buffer B_{k^*} in front of stage k^* includes all unscheduled jobs that are candidates for scheduling.

The *inner loop* mainly consists of two phases: *job selection* and *machine allocation*. In the **job selection**, a priority-rule is used in order to determine job $j^* \in B_{k^*}$ to be scheduled next. A large number of different priority rules have been examined in the literature for scheduling problems (see, e.g., Panwalkar and Iskander 1977; Haupt 1989). Preliminary tests have shown that the following priority rules are efficient for the problem under consideration:

FIFO-rule (first in first out), i.e., choice of job j^* with $\max_{j \in B_{k^*}} \{\frac{1}{t_j}\}$, where t_j is the arrival time of job j in buffer B_{k^*}; if job j arrives at time zero, we set $t_j := 1$.

MWR-rule (most work remaining first), i.e., choice of job j^* with

$$\max_{j \in B_{k^*}} \left\{ \sum_{k=k^*}^{8} \min_{m \in M^k} \{p_{jmk}\} + \sum_{\substack{i \in J^u \\ j<i \wedge j \in J^u}} \sum_{k \in K} \min_{m \in M^k} \{p_{imk}\} \right\}.$$

Let $\Omega \subseteq M^{k^*}$ be the set of idle machines at current stage k^*. In the **machine allocation** phase, job j^* is assigned to an idle machine from set Ω. Two alternative

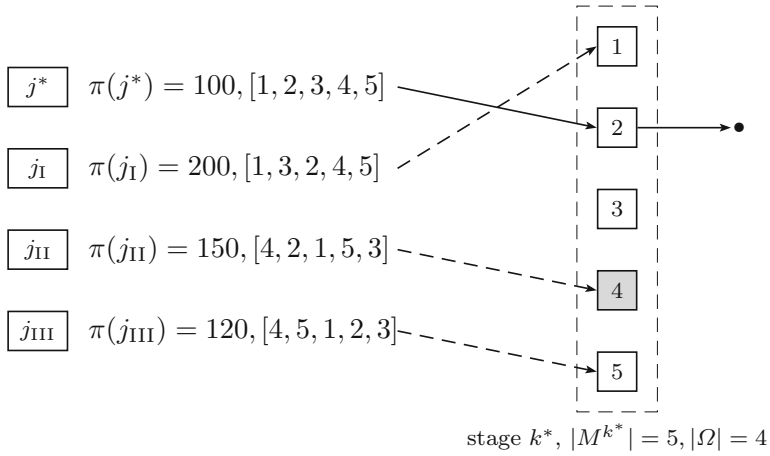

Fig. 8 Assignment Method [b] for a problem instance with five machines at stage k^*

assignment methods are implemented. In the first (Method [a]), job j^* is assigned to the fastest idle machine; i.e., j^* is assigned to idle machine $m_{j^*k^*(i)}$, where i is as small as possible. In the second method (Method [b]), j^* is assigned to an idle machine (that is not necessarily the fastest) with respect to arriving jobs with a higher priority value than j^*. Thereby, all jobs, which will arrive at stage k^* in time interval $[t^*, t^* + p_{j^*,m_{j^*k^*(i)},k^*})$, where $m_{j^*k^*(i)}$ is idle and i as small as possible, are considered in order to anticipate a better assignment for these jobs at a subsequent point in time. Note that if more than $|\Omega| - 1$ such jobs exist, the $|\Omega| - 1$ best, according to the priority values, are selected. We assume that the respective jobs are ordered in such a way that the first job, j_{I}, has the highest, the second job, j_{II}, the second highest etc. priority value. In this order, the jobs are temporarily assigned to machines, which are then marked as "blocked". In order to incorporate that a machine, which is in use at schedule time t^*, could possibly be idle at time $t^* + 1$, all machines at stage k^* (idle or not idle) are taken into account. For jobs j_{I}, j_{II} etc., the fastest machine, which is not blocked, is chosen. In the last step, job j^* is assigned to machine $m_{j^*k^*(i)}$, where i is as small as possible, idle and not blocked.

Figure 8 exemplifies assignment Method [b] for a problem instance with $|M^{k^*}| = 5$ machines at stage k^*, four of them are idle and machine 4 (in gray color) is currently in use. The priority values $\pi(j)$ as well as the machine orders $[m_{jk^*(1)}, \ldots, m_{jk^*(5)}]$ are depicted next to the respective jobs $j \in \{j^*, j_{\mathrm{I}}, j_{\mathrm{II}}, j_{\mathrm{III}}\}$. Job j^* is selected in the job selection phase and jobs j_{I}, j_{II}, and j_{III} with a higher priority value will arrive until the processing time of job j^* on machine $m_{j^*k^*(1)} = 1$ at stage k^* has elapsed. Job j_{I} is considered first in the machine allocation phase (the priority value is the highest) and is temporarily assigned to machine 1 (illustrated with a dashed arc). Consequently, machine 1 is marked as blocked. Then, job j_{II} is temporarily assigned to machine 4 and job j_{III} to machine 5 (because 4 is blocked). Finally, job j^* is actually assigned to machine 2 (illustrated with a continuous arc) that is idle and not blocked.

After the machine allocation phase, the completion time $\widetilde{C}_{j^*,k^*} \geq 0$ of job j^* at stage k^* can be computed. Furthermore, job j^* must be included in the "next" buffer in front of a stage. Thereby, we distinguish between the following if-statements:

(1) **if** job j^* performs the regular production cycle without a situation-related reentry and $k^* \in K \backslash \{8\}$ **then** insert j^* in buffer B_{k^*+1} in front of stage $k^* + 1$.

(2) **if** a situation-related reentry is identified for job j^* at stage $k^* \in K'$ **then** insert j^* in buffer B_1 and store pair (j^*, k^*).
(3) **if** job j^* is completed at the first stage ($k^* = 1$) for the second time, i.e., j^* is part of a pair (j^*, k'), $k' \in K'$, as a result of if-statement (2) **then** insert j^* in buffer $B_{k'+1}$.
(4) **if** $k^* = 8$ **then** set j^* as "finished," initialize succeeding job $j^* + 1$ (if existing) in the respective underground location, and insert $j^* + 1$ in buffer B_1.

The job selection and the machine allocation phases are solved at stage k^* as long as buffer B_{k^*} is not empty and further idle machines exist (*inner loop* of the procedure). The algorithm terminates if all jobs are set as finished. The corresponding algorithm is depicted in Algorithm 1.

Algorithm 1 Construction procedure

Input: problem instance I, priority rule
Specification of machines $m_{jk(i)}$ for all $j \in J, k \in K, i = 1, \ldots, |M^k|$ {Initialization}
Insertion of jobs $j \in J^\alpha$ in corresponding buffers B_{κ_j}
while a job exists that is not set as "finished" **do** {Outer loop}
  Specification of **schedule time** $t^* \geq 0$
  Determination of stage k^* with idle machine(s) at time t^* and with $B_{k^*} \neq \emptyset$
  while buffer $B_{k^*} \neq \emptyset$ and an idle machine exists at stage k^* **do** {Inner loop}
    Job selection: choice of job j^* with highest priority value to be scheduled
    Machine allocation: assignment of job j^* to
      [a] idle machine $m_{j^*k^*(i)}$, where i is as small as possible or
      [b] idle machine $m_{j^*k^*(i)}$, where i is as small as possible and not marked as "blocked"
    Determination of $\widetilde{C}_{j^*,k^*}$
    Insertion of j^* in the "next" buffer according to **if**-statements (1)–(4)
return solution $(\widetilde{C}, \widetilde{x}, \widetilde{y}, \widetilde{z})$ and upper bound $\widetilde{C}^{\max}$.

5.2 Multi-start algorithm

The construction procedure is embedded in a *multi-start algorithm*, where the priority values are used in order to determine selection probabilities for jobs. Particularly, within the **job selection** phase, the priority values of jobs are employed *directly* or in a *position-based* fashion. In the first case, we define $\pi(j)$ as the priority value of job j. In the second case, we order all jobs in buffer B_{k^*} according to non-decreasing priority values and set $\pi(j) := \sigma$ if job j is on position σ in the respective list. The selection probability ψ_j of job $j \in B_{k^*}$ is then computed by

$$\psi_j := \frac{\pi(j)}{\sum_{i \in B_{k^*}} \pi(i)}. \tag{41}$$

Afterwards, job j^* is chosen by the roulette-wheel selection, where each job occupies an area on the roulette wheel proportional to its ψ-value (Michalewicz and Fogel 2004, Sect. 6.1). The algorithm terminates once a stopping criterion is satisfied, e.g., after a prescribed computation time or a prescribed number of generated solutions.

5.3 Advanced multi-start algorithm

The multi-start algorithm (based on the construction procedure presented in Sect. 5.1) generates a set of so-called *non-delay* schedules. A schedule is termed non-delay if no job, which can be scheduled with respect to precedence and resource constraints, waits in front of a stage. (Baker 1974, Sect. 7.2) proved that the set of non-delay schedules must not contain an optimal solution for flow and job shop scheduling problems with regular objective functions and more than three machines. Moreover, Baker (1974) showed that the set of *active* schedules contains at least one optimal solution. A schedule is termed active if it is not possible to construct another schedule, through changes in the order of processing on the machines, with at least one job at a stage finishing earlier and no job finishing later. A non-delay schedule is always active but the reverse is not necessarily true (Pinedo 2008, Sect. 2.3). For the problem under consideration, where unrelated parallel machines exist, a non-delay schedule must not be active. This is due to the fact that an available job can be finished earlier on a machine that is currently occupied instead of being scheduled on an idle machine.

In order to illustrate the difference between non-delay as well as active (and not non-delay) schedules, we consider a system with two jobs, j_1, j_2 belonging to different underground locations, u_1, u_2, that have to be scheduled at stages 7 and 8, where $|M^7| = |M^8| = 1$. Job j_1 is in buffer B_7 at schedule time t^* and job j_2 will arrive in B_7 at time $t^* + 1$. The processing times are $p_{j_1,7} = 5$, $p_{j_1,8} = 1$, $p_{j_2,7} = 2$, and $p_{j_2,8} = 3$ (index m can be eliminated).

A non-delay schedule is depicted on the left hand side of Fig. 9, where j_1 starts its processing immediately at time t^* at stage 7. Thus, j_2 has to wait four time units (waiting time is given as a gray box) until the machine at stage 7 becomes idle. Both jobs can be assigned to the machine at stage 8 without waiting times and the resulting makespan is equal to $t^* + 10$. An active, but not non-delay, schedule is given on the right hand side of Fig. 9, where the order of processing at stages is reversed. Thus, job j_1 must accept a conscious waiting time of one time unit (demonstrated as a hatched box), i.e., the machine is kept idle for one time unit. In spite of the consideration of a conscious delay, the makespan is decreased to $t^* + 9$ (obviously an optimal solution).

The example demonstrates the usefulness of *conscious delays* or waiting times of jobs in front of stages when regarding the makespan. Therefore, the algorithm presented in Sects. 5.1 and 5.2 is enhanced so that the resulting *advanced multi-start algorithm* determines active schedules that must not be necessarily non-delay.

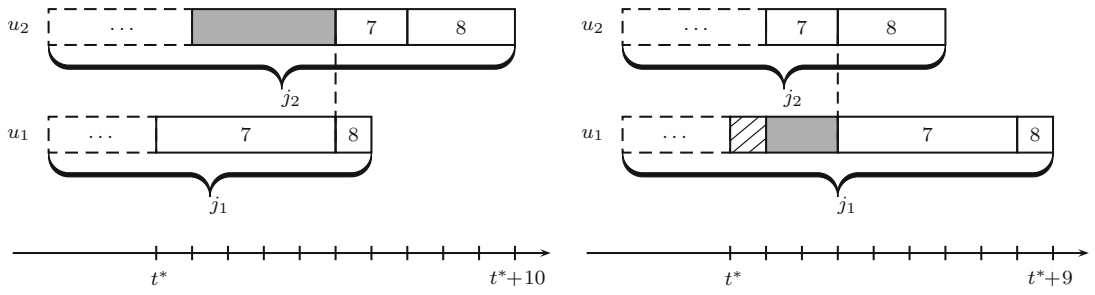

Fig. 9 Non-delay schedule with $C^{\max} = t^*+10$ and active, but not non-delay, schedule with $C^{\max} = t^*+9$

In order to integrate conscious delays of jobs in front of stages, the **job selection** phase is adapted. Let P_{k^*} be the set of jobs that are in process at schedule time t^* and that will be included in buffer B_{k^*}, after processing at their current stage, according to **if**-statements (1)–(4). In a first step, set B_{k^*} is extended to set $\mathcal{E}_{k^*} := B_{k^*} \cup P_{k^*}$ of eligible jobs. The minimum completion time of a job in $\mathcal{E}_{k^*}$ can be calculated by $\delta(t^*) := \min_{j \in \mathcal{E}_{k^*}} \min_{m \in M^{k^*}} \{C_{jmk^*}\}$, where C_{jmk^*} is the earliest possible completion time of job j on machine $m \in M^{k^*}$ at stage k^*. Note that the $\delta(t^*)$-value is determined by job $\tilde{j} \in \mathcal{E}_{k^*}$ on machine $\tilde{m}$. We set

$$\delta'(t^*) := \delta(t^*) - (1 - \varrho)\, p_{\tilde{j}\tilde{m}k^*},$$

with $\varrho \in (0, 1]$. In order to ensure that no job in $\mathcal{E}_{k^*}$ will be started at time $\delta'(t^*)$ or later, the set of eligible jobs is updated in a second step. Let $\mathcal{S}_{jk^*}$ be the earliest possible start time of job j at stage k^*. Then, we receive

$$\mathcal{E}'_{k^*} := \mathcal{E}_{k^*} \backslash \{j \in \mathcal{E}_{k^*} \mid \mathcal{S}_{jk^*} \geq \delta'(t^*)\}. \tag{42}$$

In the **machine allocation** phase, again two alternative assignment methods are implemented. In the first (Method [c]), job $j^* \in \mathcal{E}'_{k^*}$ is assigned to machine m^* that can finish j^* as early as possible. In the second method (Method [d]), a priority value for machine $m \in M^{k^*}$ is defined as the completion time of j^* on machine m at stage k^*, i.e., $\pi(m) := C_{j^*mk^*}$. The selection probability ψ_m of machine $m \in M^{k^*}$ can then be computed by

$$\psi_m := \frac{1/\pi(m)}{\sum_{m \in M^{k^*}} 1/\pi(m)}. \tag{43}$$

The resulting inner loop of the multi-start algorithm is described in Algorithm 2.

Algorithm 2 Inner loop of the advanced multi-start algorithm

while buffer $B_{k^*} \neq \emptyset$ and an idle machine exists at stage k^* **do** {Inner loop}
  Job selection: choice of job j^* from set $\mathcal{E}'_{k^*}$ (cf. condition (42)) using the roulette-wheel selection (cf. condition (41))
  Machine allocation: assignment of job j^* to
    [c] machine m^* that can finish j^* as early as possible, or
    [d] machine m^* specified by the roulette-wheel selection (cf. condition (43))
  Determination of $\widetilde{C}_{j^*,k^*}$
  Insertion of j^* in the "next" buffer according to **if**-statements (1)–(4)

Note that it is not ensured that all active schedules can be generated in the course of the procedure, since it is possible that a job at stage $k^* - 2$ (which is not an element of $\mathcal{E}_{k^*}$) causes the minimum completion time at stage k^*. Moreover, assignment Method [d] can lead on the one hand to schedules that are not active, but on the other hand to active schedules that could not be generated by assignment Method [c].

In order to show different schedules that can be generated by the advanced multi-start algorithm, the following example is considered. We assume that only two jobs j_1 and j_2 are not set as "finished" at schedule time t^*. Job j_2 is currently in process

at stage 6 and will be completed at $\widetilde{C}_{j_2,6} = t^* + 5$. Job j_1 is in buffer B_8 and both machines at stage 8 (i.e., $|M^8| = 2$) are idle. Hence, $\mathcal{E}'_8 = \{j_1\}$ holds and due to $p_{j_1,m_1,8} = 180$ as well as $p_{j_1,m_2,8} = 190$, j_1 is assigned to the first machine (cf. assignment Method [c]). This leads to $\widetilde{C}_{j_1,8} = t^* + 180$. Afterwards, j_2 is scheduled at stage 7 ($|M^7| = 1$) with $\widetilde{C}_{j_2,7} = t^* + 20$. Given the processing times $p_{j_2,m_1,8} = 140$ and $p_{j_2,m_2,8} = 180$, j_2 is finally processed on the second machine at stage 8 (note that the first machine is occupied by j_1). The makespan can then be calculated by $C^{\max} = \widetilde{C}_{j_2,8} = (t^* + 20) + 180 = t^* + 200$. For the case in which Method [d] is applied, j_1 could also be assigned to the second machine at stage 8 which results in $\widetilde{C}_{j_1,8} = t^* + 190$. Assuming that j_2 is then assigned to the first machine at stage 8, we get $\widetilde{C}_{j_2,8} = (t^* + 20) + 140 = t^* + 160$ and a reduced makespan of $C^{\max} = t^* + 190$.

5.4 Modified Giffler–Thompson procedure

A systematic approach for generating active schedules for production scheduling problems is presented by Giffler and Thompson (1960), the Giffler–Thompson (GT) procedure. In what follows, a modified version of the GT-procedure is applied, where the concept of a schedule time t^* is not used. In each iteration, the completion times of all eligible jobs $j \in \mathcal{E}_k$ on machines $m \in M^k$ at each stage $k \in K$ are determined. The minimum completion time can then be calculated by

$$\delta'' := \min_{k \in K} \min_{j \in \mathcal{E}_k} \min_{m \in M^k} \{C_{jmk}\}. \tag{44}$$

Note that the δ''-value is determined by stage k^* and job $\tilde{j} \in \mathcal{E}_{k^*}$ on machine $\tilde{m}$. In order to generate active schedules, the set of eligible jobs $\mathcal{E}_{k^*}$ at stage k^* must be reduced as follows:

$$\mathcal{E}'_{k^*} := \mathcal{E}_{k^*} \backslash \{j \in \mathcal{E}_{k^*} \mid \mathcal{S}_{jk^*} \geq \delta'' - (1 - \varrho)\, p_{\tilde{j}\tilde{m}k^*}\}, \tag{45}$$

with $\varrho \in (0, 1]$. Afterwards, a job $j^* \in \mathcal{E}'_{k^*}$ will be chosen by the roulette-wheel selection (cf. condition (41)) and will be assigned to machine m^* that can finish j^* as early as possible. The algorithm terminates if all jobs are set as finished. The corresponding algorithm is depicted in Algorithm 3.

Let us consider the example described in Sect. 5.3 again. The minimum completion time is determined by job $j_2 \in \mathcal{E}_7$ and we get $\delta'' := t^* + 20$. Consequently, j_2 is assigned to the machine at stage 7. In the following iteration, both jobs are eligible, i.e., $\mathcal{E}_8 = \{j_1, j_2\}$. Within the job selection phase, both jobs have a positive selection probability and, thus, both makespans $C^{\max} = t^* + 190$ and $C^{\max} = t^* + 200$ can be generated.

6 Computational study

This section covers the results of an extensive computational study that was conducted in order to investigate the performance of the presented solution methods. We start

Algorithm 3 Modified GT-procedure

Input: problem instance I, priority rule
Insertion of jobs $j \in J^{\alpha}$ in corresponding buffers B_{κ_j}
while a job exists that is not set as "finished" **do**
    Specification of δ'' with k^*, $\tilde{j}$ and $\tilde{m}$ (cf. condition (44))
    Job selection: choice of job $j^* \in \mathcal{E}'_{k^*}$ (cf. condition (45)) using the roulette-wheel selection (cf. condition (41))
    Machine allocation: assignment of job j^* to machine m^* that can finish j^* as early as possible
    Determination of $\widetilde{C}_{j^*k^*}$
    Insertion of j^* in the "next" buffer according to **if**-statements (1)–(4)
return solution $(\widetilde{C}, \widetilde{x}, \widetilde{y}, \widetilde{z})$ and upper bound $\widetilde{C}^{\max}$.

by describing the composition and generation of problem instances (cf. Sect. 6.1). In our performance analysis, we used GAMS 24.0 to model and CPLEX 12.6 to solve small-scale instances to optimality. Moreover, medium- and large-scale instances are tackled by the basic and advanced multi-start as well as the modified Giffler–Thompson algorithm (cf. Sect. 6.2). All tests are executed on an Intel i7X990@3.47GHz machine with 24 GB RAM. The algorithms are implemented in C++ and compiled under MS Visual Studio 2010.

6.1 Benchmark instances

The computational tests have been performed on 600 randomly generated instances that are derived from real-world data provided by our cooperation partner. The underground mine under consideration consists of several mining districts, where every district is equipped with a predefined machine fleet (Fig. 4 shows realistic numbers of machine types for a typical potash mine). Each mining district is composed of "tipple areas," where a tipple area involves 2–10 underground locations, which provide the associated tipple with crude material. The length of underground locations (i.e., the number of jobs in a chain) that has to be extracted within a given planning period (e.g., month) depends, among other factors, on the depth of excavation as well as on the salt face characteristics (i.e., composition of crude salt, quality of potash). The instances comprise 30–240 jobs and consist of 5–30 underground locations. Hence, our test set covers a wide range of possible ground plans for mining districts. We distinguish between instances with an *identical* number of jobs per underground location and instances with a *diverse* number, where the number varies within a predefined range. Although a steady and consistent mining progress is preferred that results in identical chain lengths, both situations may occur in underground mining, since excavations may be perturbed due to machine breakdowns or other technical failures and diverse chain lengths appear. Furthermore, we consider five types of machine arrangements: exactly one (two, or three) machine(s) per stage, or 1–2 (1–3) machines per stage, where the number is randomly determined. Note that instances with only one machine per stage can be regarded as baseline scenarios that typically do not exist in practice. Table 2 shows the different test sets with their specific characteristics. For each

Table 2 Detailed information on test sets used in the performance analysis

Test set	$\|J\|$	$\|U\|$	Identical	Diverse	$\|M^k\|$				
1	30	5	6	1–8	1	2	3	1–2	1–3
2	60	10	6	3–9	1	2	3	1–2	1–3
3	120	20	6	3–9	1	2	3	1–2	1–3
4	120	15	8	6–10	1	2	3	1–2	1–3
5	240	30	8	6–10	1	2	3	1–2	1–3
6	240	20	12	8–16	1	2	3	1–2	1–3

combination of identical and diverse chain length as well as machine arrangement, 10 instances were generated.

The processing time p_{jmk} of job j on machine m is defined randomly within the range 85–105 (min) for stage $k = 1$ and within the ranges 30–45, 120–135, 60–70, 40–55, 165–200, 100–130, 130–250 for stages $k = 2, 3, 4, 5, 6, 7, 8$. Thus, a block excavation needs approximately 1000 min non-stop working time at the salt face, i.e., it can be executed within three 8-h work shifts. By contrast, our cooperation partner currently needs 13 work shifts for an excavation; 9 work shifts are the target value of our partner, derived by the simple assumption that each step of the production cycle lasts exactly one work shift plus one backup shift. Considering a smooth workflow, test sets 1 to 3 refer to a planning period of one month and test sets 4 to 6 to a planning period of two months.

6.2 Performance analysis

We begin our analysis by considering small- and medium-scale instances with up to 120 jobs. Each instance is solved using model (1)–(32), plus stronger constraints (36)–(38), as well as valid inequalities (39) and (40). In order to improve the bounding accuracy and to speed up the solver, a constraint $C^{\max} \geq \mathrm{LB}^{\max}$ as well as mixed-integer rounding cuts provided by CPLEX are added. Furthermore, an initial solution $(\widetilde{C}, \widetilde{x}, \widetilde{y}, \widetilde{z})$ is given to the solver, which is generated by the basic multi-start algorithm with priority rule "Random", where the job to be scheduled next is selected randomly, and a stopping criterion of 1000 generated solutions. Since the problem under consideration is $\mathcal{NP}$-hard and therefore difficult to solve to optimality, we used a maximum computation time of $|J|$ minutes after which the best solution found is returned.

Table 3 shows the computational results, where the different test sets are given by the names $|J|\_|U|\_|M^k|$ in order to demonstrate the impact of numbers of jobs, underground locations, and machines per stage. In column "#Opt" ("#Feas"), the number of instances solved to optimality (feasibility, but not to optimality) within the time limit is given. Column t_{cpu} displays the average computation times (s) for all optimally solved instances. Finally, column "Gap[1]" shows the average gap (%) between the best integer solution $\mathrm{UB}^{\mathrm{MIP}}$ and the best lower bound $\mathrm{LB}^{\mathrm{MIP}}$ found by the solver, i.e., we calculate $\frac{\mathrm{UB}^{\mathrm{MIP}} - \mathrm{LB}^{\mathrm{MIP}}}{\mathrm{LB}^{\mathrm{MIP}}} \cdot 100\,\%$ for each instance and obtain Gap^1 by arithmetic averaging. Analogously, "Gap[2]" describes the average gap (%) between

Table 3 Computational results for the MIP-model

	Identical					Diverse				
	#Opt	#Feas	t_{cpu}	Gap^1	Gap^2	#Opt	#Feas	t_{cpu}	Gap^1	Gap^2
30_5_1	10	0	91.36	0.00	4.10	10	0	18.46	0.00	1.80
30_5_2	1	9	10.30	1.59	4.43	10	0	7.02	0.00	0.30
30_5_3	2	8	19.99	1.71	2.67	8	2	69.46	0.42	0.84
30_5_1–2	10	0	438.93	0.00	3.06	10	0	102.30	0.00	1.46
30_5_1–3	8	2	215.14	0.45	3.51	10	0	38.46	0.00	1.50
60_10_1	6	4	5.03	0.69	0.69	4	6	750.21	0.88	0.88
60_10_2	0	10	–	7.98	7.98	0	10	–	5.70	5.80
60_10_3	0	10	–	9.34	10.47	6	4	1369.04	0.59	0.65
60_10_1–2	6	4	170.72	0.41	0.41	3	7	3.54	2.57	2.68
60_10_1–3	4	6	3.85	0.72	0.72	3	7	3.77	4.07	4.07
120_20_1	6	4	1148.55	0.31	0.31	6	4	7.37	0.13	0.13
120_20_2	0	10	–	9.39	9.39	0	10	–	12.32	12.32
120_20_3	0	10	–	16.71	16.71	0	10	–	26.87	26.90
120_20_1–2	4	6	8.21	0.15	0.15	5	5	7.94	2.04	2.04
120_20_1–3	3	7	8.93	1.28	1.28	3	7	9.31	1.35	1.35
120_15_1	4	6	6.45	0.62	0.62	5	5	6.15	0.47	0.47
120_15_2	0	10	–	9.38	9.38	0	10	–	12.27	12.27
120_15_3	0	10	–	15.50	15.50	0	10	–	16.77	17.03
120_15_1–2	3	7	8.85	0.34	0.34	2	8	8.68	1.56	1.56
120_15_1–3	3	7	9.46	2.29	2.29	3	7	7.97	1.40	1.40

UB^{MIP} and the best lower bound $\text{LB}^{\max}$ found by condition (35). Note that 0 % is included in the average gap-calculation if an instance is solved to proven optimality.

The model formulation performs well for small-scale instances with 30 jobs. In particular, most instances with 1, 1–2, or 1–3 machines per stage are solved to proven optimality. For the remaining instances with $|J| = 30$ and two or three machines per stage, the average Gap^1-values given by the solver are relatively small (lower than 2 %). The results for instances with 60 or 120 jobs are not as good as expected. Less than a third of these instances are solved to optimality within the time limit and the average Gap^1-values achieve up to 26.87 %. Particularly, instances with two or three machines per stage perform poorly. Here, the relatively large number of machines allows jobs to quickly pass through the system, where several job sequences result in good makespans, i.e., in (near-)optimal solutions. Therefore, the effort to prove an optimal solution is significantly higher than for instances with, e.g., only one machine per stage. Furthermore, the results show that instances with $|J| = 120$ and $|U| = 15$ are harder to solve than instances with $|J| = 120$ and $|U| = 20$. The same holds to be true for instances with an identical number of jobs per underground location and a diverse number. The solver finds optimal solutions for 35 % of all "identical"-instances and for 44 % of all "diverse"-instances. These results emerge from a different number

Table 4 Number of instances for which the best-known solution is jointly or exclusively found

	Number of best-known solutions found						Number of best-known solutions exclusively found					
	(1)	(2)	(3)	(4)	(5)	(6)	(1)	(2)	(3)	(4)	(5)	(6)
30_5	66	63	65	66	66	65	0	0	0	0	0	34
60_10	56	51	46	60	64	64	6	2	1	4	11	22
120_20	42	41	47	61	52	67	1	0	7	12	5	25
120_15	31	30	33	53	54	61	3	2	3	14	18	25
240_30	40	39	41	62	61	53	1	0	2	20	19	15
240_20	36	35	38	62	66	56	1	1	2	14	22	14

of constraints (13)–(16), which are generated for all jobs $i \in J^u$ and $j \in J^{\hat{u}}$ with $i < j, u, \hat{u} \in U$, and $u \neq \hat{u}$. Obviously, a large number of constraints (in instances with $|U| = 20$ and in "diverse"-instances) supports the solver during the branch-and-bound algorithm. Finally, a comparison of Gap^1- and Gap^2-values demonstrate that the solver is often not in the position to increase the given lower bound $\text{LB}^{\max}$.

In order to analyze the effects of modeling techniques described in Sect. 4, each instance with 30 jobs is solved using only model (1)–(32). In comparison to the results presented in Table 3, the outcomes are significantly worse; 68 instances (instead of 79 instances) are solved to proven optimality. Furthermore, the average computation times increased by 41 % and the average solution gaps increased from 0.42 to 1.08 %.

We continue our analysis by considering all instances with up to 240 jobs. Each instance is solved with the *basic multi-start algorithm* presented in Sects. 5.1 and 5.2 by using the following combinations of priority-rules and assignment methods:

(1) Random-rule, where job $j^* \in B_{k^*}$ is selected randomly,
(2) FIFO-rule and assignment Method [a],
(3) FIFO-rule (position-based) and assignment Method [a],
(4) MWR-rule and assignment Method [a],
(5) MWR-rule (position-based) and assignment Method [a],
(6) MWR-rule and assignment Method [b].

For each setting (1)–(6), 10,000 solutions are generated. Hence, the algorithm terminates after 60,000 solutions and returns the best (non-delay) solution found so far. Note that settings (4) and (6) are equal for instances with exactly one machine per stage, and the execution of assignment Method [b] is only suitable for the MWR-rule. In order to illustrate that all chosen settings generate best-known solutions, we consider each setting and count the number of instances for which the best-known solution by multi-start is found or exclusively found.

Table 4 shows the results. Since the highest value in each cell could be equal to 100, the numbers of best-known solutions found are quite large (in the interval [30,67]). For the case in which a best-known solution is exclusively found, settings (4)–(6) perform best (the maximum value per column lies in the interval [20,34]). As the number of jobs in underground locations do not influence that much the results of

Table 5 Comparison of MIP-model and basic multi-start algorithm

	t_{cpu}	Gap^3	MS = MIP	MS < MIP	MS > MIP	$\text{dev}_{<0}$	$\text{dev}_{>0}$
30_5_1	63.35	6.04	3	0	17	–	3.61
30_5_2	59.70	3.63	5	0	15	–	1.63
30_5_3	58.90	2.86	5	0	15	–	1.43
30_5_1–2	62.10	4.51	5	0	15	–	2.95
30_5_1–3	61.90	4.26	6	0	14	–	2.45
60_10_1	149.70	0.75	14	5	1	0.19	0.31
60_10_2	126.55	6.91	0	11	9	0.94	1.26
60_10_3	114.80	5.83	6	5	9	0.74	1.00
60_10_1–2	137.00	1.52	11	5	4	0.57	0.62
60_10_1–3	137.00	1.92	11	7	2	1.42	0.58
120_20_1	393.25	0.13	12	8	0	0.22	–
120_20_2	341.60	7.42	0	20	0	3.08	–
120_20_3	309.50	16.20	0	20	0	4.49	–
120_20_1–2	336.90	0.82	11	8	1	0.74	0.81
120_20_1–3	316.35	0.93	6	11	3	0.73	0.31
120_15_1	375.45	0.38	8	10	2	0.34	0.05
120_15_2	316.90	7.70	0	20	0	2.80	–
120_15_3	288.60	12.99	0	20	0	2.81	–
120_15_1–2	336.25	0.64	7	12	1	0.51	0.36
120_15_1–3	317.05	1.31	6	11	3	1.01	0.45

multi-start, a further distinction between instances of types "identical" and "diverse" is not considered.

Table 5 summarizes the results obtained by comparing the MIP- and the multi-start solutions. In column t_{cpu} the average computation times (s) of the multi-start (MS) algorithm are given. Column "Gap^3" shows the average gap (%) between the best integer solution UB^{MS} found by multi-start and the lower bound $\text{LB}^{\max}$. Furthermore, column "MS = MIP" ("MS<MIP", "MS>MIP") indicates the number of solutions for which the heuristic value is equal to (smaller, larger than) the MIP-solution. Columns "$\text{dev}_{<0}$" and "$\text{dev}_{>0}$" depict the corresponding average negative and positive deviations (%), i.e., we calculate $\frac{\text{UB}^{\text{MIP}}-\text{UB}^{\text{MS}}}{\text{UB}^{\text{MIP}}} \cdot 100\,\%$ as well as $\frac{\text{UB}^{\text{MS}}-\text{UB}^{\text{MIP}}}{\text{UB}^{\text{MIP}}} \cdot 100\,\%$ and obtain $\text{dev}_{<0}$ as well as $\text{dev}_{>0}$ by arithmetic averaging.

The multi-start algorithm offers the advantage of fast run times (lower than 7 min for instances with 120 jobs) along with good results. For small-scale instances with 30 jobs, where the MIP-model usually finds the optimal solution within the time limit, the results of multi-start are slightly poorer than the results of CPLEX. However, both methods are generally able to produce the same objective function values for medium-scale instances with 60 jobs. For instances with 120 jobs, the multi-start algorithm performs best. By comparing the Gap^3-values to the Gap^2-values in Table 3, it becomes apparent that the values decrease for large-scale instances. This fact also

Table 6 Comparison of MIP-model and advanced multi-start algorithm

	t_{cpu}	Gap[3]	$\widetilde{\text{MS}} = \text{MIP}$	$\widetilde{\text{MS}} < \text{MIP}$	$\widetilde{\text{MS}} > \text{MIP}$	$\text{dev}_{<0}$	$\text{dev}_{>0}$
30_5_1	25.40	3.54	9	0	11	–	1.07
30_5_2	25.00	2.87	2	0	18	–	0.54
30_5_3	25.10	2.24	4	0	16	–	0.60
30_5_1–2	25.25	2.71	9	0	11	–	0.80
30_5_1–3	25.15	3.00	6	0	14	–	0.68
60_10_1	75.10	0.71	15	5	0	0.28	–
60_10_2	61.70	5.62	0	17	3	1.51	0.80
60_10_3	53.30	5.48	0	12	8	0.64	0.84
60_10_1–2	70.30	1.34	12	7	1	0.60	0.29
60_10_1–3	70.05	1.70	12	8	0	1.61	–
120_20_1	282.05	0.11	12	8	0	0.28	–
120_20_2	233.45	5.32	0	20	0	4.97	–
120_20_3	216.70	11.58	0	20	0	8.31	–
120_20_1–2	228.70	0.49	10	10	0	1.12	–
120_20_1–3	215.55	0.74	7	13	0	0.82	–
120_15_1	254.40	0.38	9	11	0	0.31	–
120_15_2	208.35	5.44	0	20	0	4.84	–
120_15_3	199.80	9.91	0	20	0	5.46	–
120_15_1–2	222.85	0.45	7	13	0	0.70	–
120_15_1–3	210.65	0.78	6	14	0	1.40	–

emphasizes the efficiency of our multi-start algorithm for realistic instances. Moreover, the average negative as well as the positive deviations are approximately 1.9 %.

In order to integrate conscious delays and to generate active and not necessarily non-delay schedules, each instance is solved by the *advanced multi-start* and the *modified Giffler–Thompson algorithm* (cf. Sect. 5.3–5.4). Thereby, the following priority-rules are used:

(i) Random-rule, where job $j^* \in \mathcal{E}_{k^*}$ is selected randomly,
(ii) FIFO-rule,
(iii) FIFO-rule (position-based),
(iv) MWR-rule,
(v) MWR-rule (position-based).

Within the job selection phase, we used $\varrho = \{\epsilon, 0.1, 0.3, 0.6, 1\}$ in order to determine the set of eligible jobs. Preliminary tests have shown that the described ϱ-values on average lead to good objective function values.

When applying the advanced multi-start algorithm ($\widetilde{\text{MS}}$), each setting (i)–(v) was combined with assignment Method [c] as well as assignment Method [d]. The stopping criterion was set to 60,000 solutions, i.e., 2000 solutions for each combination of setting (i)–(v) and each ϱ-value with Method [c] as well as 400 solutions for each combination of setting (i)–(v) and each ϱ-value with Method [d], respectively. In order to obtain

Table 7 Comparison of MIP-model and modified Giffler–Thompson algorithm

	t_{cpu}	Gap^3	GT = MIP	GT < MIP	GT > MIP	$dev_{<0}$	$dev_{>0}$
30_5_1	37.30	3.54	7	0	13	–	0.90
30_5_2	38.25	2.68	7	0	13	–	0.45
30_5_3	40.95	2.08	6	0	14	–	0.46
30_5_1–2	39.65	2.47	8	0	12	–	0.33
30_5_1–3	42.85	2.77	7	0	13	–	0.39
60_10_1	76.30	0.72	14	5	1	0.30	0.17
60_10_2	75.50	5.39	0	18	2	1.58	0.41
60_10_3	77.80	5.30	0	12	8	0.74	0.56
60_10_1–2	75.60	1.24	12	8	0	0.72	–
60_10_1–3	77.25	1.57	12	8	0	1.89	–
120_20_1	241.05	0.13	12	7	1	0.27	0.16
120_20_2	240.55	5.24	0	20	0	5.04	–
120_20_3	240.30	11.80	0	20	0	8.14	–
120_20_1–2	233.55	0.47	10	10	0	1.15	–
120_20_1–3	229.45	0.73	7	13	0	0.84	–
120_15_1	231.65	0.42	9	11	0	0.23	–
120_15_2	226.30	5.45	0	20	0	4.82	–
120_15_3	230.00	9.75	0	20	0	5.60	–
120_15_1–2	229.40	0.50	7	12	1	0.69	0.04
120_15_1–3	226.75	0.86	6	13	1	1.40	0.03

a fair comparison of results, the modified Giffler–Thompson algorithm (GT) is also terminated after 60,000 solutions, i.e., 2400 solutions for each combination of setting (i)–(v) and each ϱ-value.

At first, we compared the MIP-solutions to the advanced multi-start (modified Giffler–Thompson) solutions (cf. Tables 6, 7). The results show that both algorithms outperform the basic multi-start algorithm. The total number of instances with the same or a smaller objective function value than the MIP-model is higher, i.e., 318 (321) instead of 289 in Table 5. Moreover, the average Gap^3-values as well as the average positive deviations decrease, e.g., the Gap^3-values decrease from 4.3 to 3.2 % (3.2 %). Further, the average negative deviations increase from 1.37 to 2.19 % (2.23 %). The average run time for all instances decreases from 472 to 346 (351) s. Note that an analysis of solutions for instances with 60 jobs and a stopping criterion of 150 s (instead of 60,000 solutions) showed similar results as described in Tables 5, 6, and 7.

Considering all aforementioned numbers in brackets, which refer to the modified Giffler–Thompson procedure, we can say that this procedure performs slightly better than the advanced multi-start algorithm. However, both algorithms are obviously better than the basic multi-start. A detailed comparison of the modified Giffler–Thompson procedure and basic as well as advanced multi-start algorithms is given in Table 8. Column "GT < MS" includes the number of solutions for which GT performs better

Table 8 Comparison of modified Giffler–Thompson algorithm with basic and advanced multi-start algorithm

	GT = MS	GT < MS ($dev_{<0}$)	GT > MS ($dev_{>0}$)	GT = $\widetilde{MS}$	GT < $\widetilde{MS}$ ($dev_{<0}$)	GT > $\widetilde{MS}$ ($dev_{>0}$)	dev_{Best} (GT)	dev_{Best} ($\widetilde{MS}$)	dev_{Best} (MS)
30_5_1	3	17 (2.75)	0 (–)	8	5 (0.50)	7 (0.34)	0.58	0.59	3.07
30_5_2	5	15 (1.21)	0 (–)	3	12 (0.37)	5 (0.11)	0.30	0.49	1.22
30_5_3	5	14 (1.07)	1 (0.25)	5	14 (0.23)	1 (0.09)	0.32	0.48	1.07
30_5_1–2	5	15 (2.56)	0 (–)	11	6 (0.91)	3 (0.19)	0.20	0.44	2.22
30_5_1–3	5	14 (2.00)	1 (0.05)	11	6 (0.91)	3 (0.32)	0.26	0.48	1.72
60_10_1	16	2 (0.46)	2 (0.10)	17	1 (0.13)	2 (0.09)	0.01	0.01	0.05
60_10_2	0	20 (1.42)	0 (–)	0	13 (0.44)	7 (0.20)	0.11	0.33	1.56
60_10_3	0	13 (1.03)	7 (0.55)	0	12 (0.33)	8 (0.06)	0.29	0.46	0.78
60_10_1–2	14	5 (1.09)	1 (0.04)	16	3 (0.64)	1 (0.01)	0.00	0.10	0.28
60_10_1–3	12	5 (1.40)	3 (0.19)	14	4 (0.63)	2 (0.08)	0.03	0.15	0.36
120_20_1	15	1 (0.39)	4 (0.12)	15	1 (0.02)	4 (0.13)	0.03	0.00	0.02
120_20_2	0	20 (2.03)	0 (–)	0	13 (0.21)	7 (0.16)	0.05	0.14	2.13
120_20_3	0	20 (3.80)	0 (–)	0	6 (0.26)	14 (0.38)	0.27	0.08	4.23
120_20_1–2	13	5 (1.30)	2 (0.03)	17	1 (0.47)	2 (0.06)	0.01	0.02	0.34
120_20_1–3	14	6 (0.65)	0 (–)	18	2 (0.12)	0 (–)	0.00	0.01	0.20
120_15_1	9	4 (0.16)	7 (0.20)	13	1 (0.01)	6 (0.15)	0.07	0.03	0.03
120_15_2	0	20 (2.08)	0 (–)	1	10 (0.24)	9 (0.30)	0.13	0.12	2.27
120_15_3	0	20 (2.86)	0 (–)	1	13 (0.38)	6 (0.33)	0.10	0.25	3.06
120_15_1–2	12	4 (0.79)	4 (0.14)	13	0 (–)	7 (0.14)	0.06	0.01	0.19

Table 8 continued

	GT = MS	GT < MS ($dev_{<0}$)	GT > MS ($dev_{>0}$)	GT = $\widetilde{MS}$	GT < $\widetilde{MS}$ ($dev_{<0}$)	GT > $\widetilde{MS}$ ($dev_{>0}$)	dev_{Best} (GT)	dev_{Best} ($\widetilde{MS}$)	dev_{Best} (MS)
120_15_1–3	11	7 (1.25)	2 (0.05)	13	2 (0.02)	5 (0.31)	0.08	0.00	0.52
240_30_1	14	0 (–)	6 (0.15)	14	0 (–)	6 (0.08)	0.05	0.02	0.00
240_30_2	0	20 (2.75)	0 (–)	1	13 (0.29)	6 (0.27)	0.08	0.19	2.92
240_30_3	0	20 (5.44)	0 (–)	0	9 (0.27)	11 (0.18)	0.10	0.12	5.86
240_30_1–2	13	4 (0.98)	3 (0.08)	13	4 (0.17)	3 (0.04)	0.01	0.04	0.20
240_30_1–3	14	5 (1.42)	1 (0.03)	16	1 (0.01)	3 (0.27)	0.04	0.00	0.41
240_20_1	14	1 (0.08)	5 (0.07)	15	0 (–)	5 (0.05)	0.02	0.01	0.00
240_20_2	0	20 (2.57)	0 (–)	0	9 (0.13)	11 (0.20)	0.11	0.06	2.75
240_20_3	0	20 (4.90)	0 (–)	1	7 (0.47)	12 (0.32)	0.20	0.16	5.37
240_20_1–2	16	3 (0.30)	1 (0.00)	19	0 (–)	1 (0.00)	0.00	0.00	0.04
240_20_1–3	11	6 (1.13)	3 (0.09)	12	7 (0.09)	1 (0.18)	0.01	0.04	0.35

than MS; the corresponding average negative deviation is given in brackets behind. Column "$\text{dev}_{\text{Best}}(\text{GT})$" depicts the average deviation (%) between the best integer solution UB^{GT} found by GT and the best known solution UB^{Best}, i.e., we calculate $\frac{\text{UB}^{\text{GT}}-\text{UB}^{\text{Best}}}{\text{UB}^{\text{Best}}} \cdot 100\,\%$ and obtain $\text{dev}_{\text{Best}}(\text{GT})$ by arithmetic averaging.

The strength of the modified Giffler–Thompson procedure compared to the basic multi-start algorithm appears when regarding the results in columns "GT $<$ MS ($\text{dev}_{<0}$)". For 326 out of 600 instances, GT produces lower objective function values. Moreover, the average negative deviations between GT and MS, which are equal to 1.72 %, are larger than the average positive deviations, which are equal to 0.13 %.

As expected, the modified Giffler–Thompson procedure performs very well for instances with 30 or 60 jobs compared to the advanced multi-start algorithm. In particular, this is emphasized by the results in columns "$\text{dev}_{\text{Best}}(\text{GT})$" and "$\text{dev}_{\text{Best}}(\widetilde{\text{MS}})$". When regarding larger instances with 120 or 240 jobs, the results show that the GT and $\widetilde{\text{MS}}$ generate the same objective function values for 46 % of the instances, while for 30 % of the instances $\widetilde{\text{MS}}$ generates better results. The behavior can be explained as follows: For instances with 30 or 60 jobs, algorithm GT outperforms $\widetilde{\text{MS}}$ since the set of active schedules is relatively small and the probability of finding a (near-) optimal solution is higher for GT. For larger instances with 120 or 240 jobs, the set of active schedules is quite large. Therefore, it is highly probable that GT and $\widetilde{\text{MS}}$ deliver similar solutions.

7 Conclusion

The paper considers an underground mine scheduling problem that appears in potash mining, where a block excavation sequence has to be found. The problem is modeled as a mixed-integer linear program and small-scale instances are solved with standard optimization software. In order to facilitate the solution process, additional constraints, lower bounds, and an initial solution are given to the solver. Furthermore, a priority rule-based construction procedure is developed in order to solve large-scale problem instances. The procedure is embedded in a basic multi-start algorithm, which is then extended to an advanced multi-start algorithm that considers conscious delays of jobs in front of stages. In addition, a modified version of the Giffler–Thompson procedure is developed. Computational experiments are conducted on randomly generated problem instances that resemble the structure of realistic applications.

The results show that the MIP-model is efficient for instances with 30 jobs and 1, 1–2, or 1–3 machine(s) per stage. The corresponding branch-and-cut algorithm is able to solve 58 instances to proven optimality and for the remaining two instances the average gap is lower than 0.5 % after a 30 min time limit. For the cases in which the number of jobs and the number of machines per stage are increased, the proposed heuristic solution procedures should be used. In particular, the modified Giffler–Thompson procedure performs very well for instances with up to 60 jobs. For larger instances with up to 240 jobs, the results obtained by the advanced multi-start algorithm are slightly better than by the modified Giffler–Thompson procedure. In practice, it is common that an available machine is not kept idle when a job could be assigned to it. This approach is considered in the basic multi-start algorithm. However, the results show that this variant seems not to be an appropriate choice for solving the presented problem.

Future research will include the consideration of other real-life conditions, e.g., machine failures, stochastic job processing times, and repositioning of vehicles. Further, the minimization of the flow time per job is an alternative objective function. Additionally, the improvement of lower bounds by considering a combined stage for the current stages 2 and 5 would be an interesting topic. In order to explore the search space more efficiently, a metaheuristic (e.g., a genetic algorithm) could finally be implemented, where the presented procedures are used as a basis.

Acknowledgments The benchmarks presented herein may be downloaded from http://www.wiwi.tu-clausthal.de/abteilungen/unternehmensforschung/forschung/benchmark-instances/.

References

Baker KR (1974) Introduction to sequencing and scheduling. Wiley, New York

Baykasoğlu A, Hamzadayi A, Yelkenci Köse S (2014) Testing the performance of teaching–learning based optimization (TLBO) algorithm on combinatorial problems: Flow shop and job shop scheduling cases. Inf Sci 276:204–218

Beaulieu M, Gamache M (2006) An enumeration algorithm for solving the fleet management problem in underground mines. Comput Oper Res 33:1606–1624

Bedworth DD, Bailey JE (1982) Integrated production and control systems: management, analysis, and design. Wiley, New York

Botta-Genoulaz V (2000) Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness. Int J Prod Econ 64:101–111

Carlyle W, Eaves B (2001) Underground planning at stillwater mining company. Interfaces 31:50–60

Chen L, Bostel N, Dejax P, Cai J, Xi L (2007) A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal. Eur J Oper Res 181:40–58

Chen L, Xi L-F, Cai J-G, Bostel N, Dejax P (2006) An integrated approach for modeling and solving the scheduling problem of container handling systems. J Zhejiang Univ Sci A 7:234–239

Chen Y-Y, Cheng C-Y, Wang L-C, Chen T-L (2013) A hybrid approach based on the variable neighborhood search and particle swarm optimization for parallel machine scheduling problems - a case study for solar cell industry. Int J Prod Econ 141:66–78

Chesworth W (2008) Encyclopedia of soil science. Springer, Dordrecht

Cho H-M, Bae S-J, Kim J, Jeong I-J (2011) Bi-objective scheduling for reentrant hybrid flow shop using pareto genetic algorithm. Comput Ind Eng 61:529–541

Choi H-S, Kim J-S, Lee D-H (2011) Real-time scheduling for reentrant hybrid flow shops: a decision tree based mechanism and its application to a TFT-LCD line. Exp Syst Appl 38:3514–3521

Choi S-W, Kim Y-D, Lee G-C (2005) Minimizing total tardiness of orders with reentrant lots in a hybrid flowshop. Int J Prod Res 43:2149–2167

Epstein R, Goic M, Weintraub A, Catalan J, Santibanez P, Urrutia R, Cancino R, Gaete S, Aguayo A, Caro F (2012) Optimizing long-term production plans in underground and open-pit copper mines. Oper Res 60:4–17

Fereidoonian F, Mirzazadeh A (2011) A genetic algorithm for the integrated scheduling model of a container-handling system in a maritime terminal. J Eng Marit Environ 226:62–77

Gamache M, Grimard R, Cohen P (2005) A shortest-path algorithm for solving the fleet management problem in underground mines. Eur J Oper Res 166:497–506

Giffler B, Thompson GL (1960) Algorithms for solving production-scheduling problems. Oper Res 8:487–503

Grabowski J, Pempera J (2000) Sequencing of jobs in some production system. Eur J Oper Res 125:535–550

Gupta JND (1988) Two-stage, hybrid flow shop scheduling problem. J Oper Res Soc 39:359–364

Hamrin H (2001) Underground mining methods and applications. In: Hustrulid WA, Bullock RL (eds) Underground mining methods: engineering fundamentals and international case studies. SME, Littleton, pp 3–14

Haupt R (1989) A survey of priority rule-based scheduling. OR Spektrum 11:3–16

Hidri L, Haouari M (2011) Bounding strategies for the hybrid flow shop scheduling problem. Appl Math Comput 217:8248–8263

Jin ZH, Ohno K, Ito T, Elmaghraby SE (2002) Scheduling hybrid flowshops in printed circuit board assembly lines. Prod Oper Manag 11:216–230
Kelley JE (1963) The critical-path method: resources planning and scheduling. In: Muth JF, Thompson GL (eds) Industrial scheduling. Prentice-Hall, New Jersey, pp 347–365
Kolisch R (1996) Serial and parallel resource-constrained project scheduling methods revisited: theory and computation. Eur J Oper Res 90:320–333
Kreutz M, Hanke D, Gehlen S (2000) Solving extended hybrid-flow-shop problems using active schedule generation and genetic algorithms. In: Schoenauer M, Deb K, Rudolph G, Yao X, Lutton E, Merelo J, Schwefel H-P (eds) Parallel problem solving from nature PPSN VI, Lecture notes in computer science, vol 1917. Springer, Berlin, pp 293–302
K+S AG (2015) A day in the mine. http://www.k-plus-s.com
Lee G-C (2009) Estimating order lead times in hybrid flowshops with different scheduling rules. Comput Ind Eng 56:1668–1674
Li Z-T, Chen Q-X, Mao N, Wang X, Liu J (2013) Scheduling rules for two-stage flexible flow shop scheduling problem subject to tail group constraint. Int J Prod Econ 146:667–678
Lin H-T, Liao C-J (2003) A case study in a two-stage hybrid flow shop with setup time and dedicated machines. Int J Prod Econ 86:133–143
Martinez MA, Newman AM (2011) A solution approach for optimizing long- and short-term production scheduling at LKAB's Kiruna mine. Eur J Oper Res 211:184–197
Michalewicz Z, Fogel D (2004) How to solve it: modern heuristics. Springer, Berlin
Nehring M, Topal E, Kizil M, Knights P (2012) Integrated short- and medium-term underground mine production scheduling. J S Afr Inst Min Metall 112:365–378
Nehring M, Topal E, Knights P (2010a) Dynamic short term production scheduling and machine allocation in underground mining using mathematical programming. Trans Inst Min Metall Sect A Min Technol 119:212–220
Nehring M, Topal E, Little J (2010b) A new mathematical programming model for production schedule optimization in underground mining operations. J S Afr Inst Min Metall 110:437–446
Newman AM, Kuchta M (2007) Using aggregation to optimize long-term planning at an underground mine. Eur J Oper Res 176:1205–1218
Newman AM, Rubio E, Caro R, Weintraub A, Eurek K (2010) A review of operations research in mine planning. Interfaces 40:222–245
O'Sullivan D, Newman A (2015) Optimization-based heuristics for underground mine scheduling. Eur J Oper Res 241:248–259
Panwalkar SS, Iskander W (1977) A survey of scheduling rules. Oper Res 25:45–61
Pinedo ML (2008) Scheduling: theory, algorithms, and systems, 3rd edn. Springer, New York
Rahal DC, Smith ML, van Hout G, von Johannides A (2003) The use of mixed integer linear programming for long-term scheduling in block caving mines. In: Camisani-Calzolari FA (ed) Application of computers and operations research in the minerals industries. South African Institute of Mining and Metallurgy, Cape Town, pp 123–131
Ribas I, Leisten R, Framiñan JM (2010) Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. Comput Oper Res 37:1439–1454
Rossi A, Soldani S, Lanzetta M (2015) Hybrid stage shop scheduling. Exp Syst Appl 42:4105–4119
Ruiz R, Şerifoğlu FS, Urlings T (2008) Modeling realistic hybrid flexible flowshop scheduling problems. Comput Oper Res 35:1151–1175
Ruiz R, Maroto C (2006) A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. Eur J Oper Res 169:781–800
Ruiz R, Vázquez-Rodríguez JA (2010) The hybrid flow shop scheduling problem. Eur J Oper Res 205(1):1–18
Saayman P, Craig IK, Camisani-Calzolari FR (2006) Optimization of an autonomous vehicle dispatch system in an underground mine. J S Afr Inst Min Metall 106:77–86
Santos DL, Hunsucker JL, Deal DE (1995) Global lower bounds for flow shops with multiple processors. Eur J Oper Res 80:112–120
Sarin S, West-Hansen J (2005) The long-term mine production scheduling problem. IIE Trans 37:109–121
Simsir F, Ozfirat MK (2008) Determination of the most effective longwall equipment combination in longwall top coal caving (LTCC) method by simulation modelling. Int J Rock Mech Min Sci 45:1015–1023

USGS (2011) Metals and minerals: united states geological survey minerals yearbook. United States Government Printing Office, Washington

Voß S, Witt A (2007) Hybrid flow shop scheduling as a multi-mode multi-project scheduling problem with batching requirements: A real-world application. Int J Prod Econ 105:445–458

Weintraub A, Romero C, Bjørndal T, Epstein R (2007) Handbook of operations research in natural resources. In: International series in operations research & management science, vol. 99. Springer, New York

Xu Y, Wang L (2011) Differential evolution algorithm for hybrid flow-shop scheduling problems. J Syst Eng Electron 22:794–798

Yaurima V, Burtseva L, Tchernykh A (2009) Hybrid flowshop with unrelated machines, sequence-dependent setup time, availability constraints and limited buffers. Comput Ind Eng 56:1452–1463

C. Paper III

Mining Goes Digital – Mueller et al. (Eds)
© 2019 Taylor & Francis Group, London, ISBN 978-0-367-33604-2

A two-stage solution approach for a shift scheduling problem with a simultaneous assignment of machines and workers

Cinna Seifi
Clausthal University of Technology, Clausthal-Zellerfeld, Germany

Marco Schulze
K+S Aktiengesellschaft, Kassel, Germany

Jürgen Zimmermann
Clausthal University of Technology, Clausthal-Zellerfeld, Germany

ABSTRACT: We consider a short-term production scheduling problem in a German potash underground mine where drill-and-blast mining operations have to be assigned to machines and workers and scheduled simultaneously. In addition, several mining-specific requirements have to be taken into account. In order to solve the problem at hand, we propose a two-stage solution approach. In the first stage, we apply a mixed-integer linear program where some time-consuming restrictions are neglected. Afterward, we modify the obtained schedule by integrating the necessary time intervals that were dismissed within the mathematical model. Since an existing heuristic solution procedure for the same problem is currently in use in a German potash mine, we will present results for computational experiments conducted on problem instances derived from real-world data in order to evaluate the performance of the two solution approaches.

1 PROBLEM DESCRIPTION AND LITERATURE REVIEW

This paper addresses a short-term production scheduling problem in a German potash underground mine that was already studied by Schulze & Zimmermann (2017) as well as Schulze et al. (2017) who proposed a rule-based constructive procedure. The extraction of the examined potash mine is done by room-and pillar mining method and the excavation of potash is based on drilling and blasting technique. This kind of underground mining is characterized by eight consecutive sub-steps, i.e., operations, that can be seen as a production cycle: scaling the roof, bolting the roof with expansion-shell bolts, drilling large diameter bore holes, removing the drilled material, drilling blast holes, filling the blast holes with an explosive substance, blasting, and transportation of the broken material to a crusher. For each operation, except blasting, one special mobile machine out of a set of identical or uniform machines is required that is handled by a worker with the corresponding qualification, i.e., skill. Hence, the underlying problem consists of the determination of a shift schedule where (i) a set of jobs[1] has to be selected and determined for execution, (ii) start times of the selected jobs have to be specified, and (iii) machines and workers have to be assigned to the jobs simultaneously while the individual skills of the workers as well as the technological-based precedence relations for the jobs have to be taken into account. The objective is the minimization of the average positive deviation between a predetermined quantity and the amount of extracted crude salt, cumulated over all operations.

Taking the aforementioned characteristics into consideration, we deal with two different problem types that have to be solved simultaneously, that is a machine scheduling problem on the one hand, and an employee timetabling problem on the other hand. The machine

1. Note that a job is characterized by the corresponding operation type from the production cycle and the place within the mine where the operation has to be executed.

scheduling problem can be classified as a variant of a so-called hybrid flow shop (HFS) scheduling problem, if we identify each mining operation from the production cycle as a stage, see Schulze et al. (2016). Note that we assume the mining operations as non-preemptable, because an assigned machine stays at the corresponding working place after an interruption (due to workers' breaks or end of a shift) and processing will be resumed at the next possible occasion. In comparison to the classical HFS scheduling problem, due to the short planning horizon of one single working shift, it is not possible to perform all mining operations at all working places. Thus, on the one hand, it has to be decided whether a job at a working place will be processed within the shift under consideration or not, and on the other hand, some possible interruption of the processing of a job at the end of the working shift must be taken into account.

The machines in the problem at hand are mobile and travel from job to job, therefore, no buffers are needed between the stages. Instead, we have to consider driving times between jobs that are processed by the same machine what results in sequence-dependent setup times. In addition, there are so-called technical services that have to be performed for the machines. Before processing the first operation assigned to a machine, a preventive maintenance (first technical services) must be done. After processing the last job assigned to a machine as well as before the end of the working shift if a job has to be interrupted, each machine must be cleaned and fueled (last technical services). Furthermore, for the case in which a worker changes his machine, he has to perform the first technical services for the new machine.

The second problem type, an employee timetabling problem, comprises the assignment of suitable workers to machines within a working shift. We assume that a worker can handle at most one machine at the same time and a machine can be operated by at most one worker at the same time. Since the workers have particular skills on different levels, not all workers can handle all machines. The different skill levels result in different handling times for each machine. Furthermore, the machines on each stage can have different speeds, which means that the processing time of a job depends on both the assigned worker and machine.

In order to generate a schedule that is accepted from the shift supervisor and that meets legal obligations, we furthermore have to satisfy the following mining-specific requirements.

R1: Due to legal regulations, a Δ-minute break for the workers has to be incorporated in the schedule within a predetermined time window. It has to be noted that the break can lead to a delay in the processing of a job.

R2: We have to consider disjunctive constraints for subsets of jobs that are physically close to each other (these jobs "belong" to so-called underground locations). Due to security reasons, it is not allowed that more than one machine is processing there at the same time, i.e., only one job in an underground location can be processed at any point in time.

R3: Although a consistent progress at all working places of the underground locations would be desirable, different excavation states appear, i.e., not the same operation can be performed there. In order to achieve a harmonized state, we prioritize the jobs in an underground location in a way that the progress strives for consistency.

R4: Jobs that are interrupted at the end of the pre-shift have a higher priority than the other jobs within the same underground location.

R5: The operations that symbolize the step drilling large diameter bore holes are non-interruptible. If those operations cannot be finished until the end of the corresponding working shift, they must not be started.

R6: The number of assigned worker to a machine is limited by two. That means no more than two workers can be assigned to a machine and a worker is not allowed to work on more than two machines during a working shift.

In the literature, most works concerning machine scheduling neglect the assignment of workers, while in the field of employee timetabling (i.e., staff scheduling and personnel assignment) the machine scheduling problem is rarely considered. Therefore, we confined our literature review to the integrated employee timetabling and machine scheduling problems. An overview of the studied literature is given in Table 1, where the abbreviations in column *production environment* characterize, whether the authors analyzed a flow shop (FSP), job

Table 1. Literature review.

	Production environment	Objective function	Formulation
Daniels & Mazzola (1994)	FSP	makespan	time-indexed
Daniels et al. (2004)	FSP	makespan	time-indexed
Huq et al. (2004)	FSP	multi-obj.	seq.-based
Artigues et al. (2006)	JSP	empl. cost	time-indexed
Artigues et al. (2009)	JSP	multi-obj.	seq.-based
Puttkammer et al. (2011)	FSP	multi-obj.	time-indexed
Mencia et al. (2013)	JSP	flow time	
Ramya & Chandrasekaran (2014)	JSP	empl. cost	time-indexed
Frihat et al. (2014)	HJSP	empl. cost	seq.-based
Benavides et al. (2014)	FSP	makespan	seq.-based
Guyon et al. (2014)	JSP	empl. cost	
Agnetis et al. (2014)	JSP	makespan	seq.-based
Campos Ciro et al. (2016)	OSP	flow time	seq.-based
Ahmadi-Javid & Hooshangi-Tabrizi (2017)	JSP	makespan	seq.-based
Santos et al. (2018)	JSP	throughput	time-indexed

shop (JSP), open shop (OSP), or hybrid job shop (HJSP) scheduling problem. Moreover, we indicate what kind of *objective function* as well as *formulation* is considered within the corresponding study.

In brief, our literature review shows that the approaches discussed in the studies are not suitable for the problem at hand what is mainly due to the aspects that we consider (i) a selection of jobs, (ii) setup times for machines and workers, (iii) possible interruption of jobs at the end of the shift, (iv) breaks that could delay the processing of jobs, and (v) that the workers can change their machine within a working shift. In the next section, we introduce a two-stage approach to tackle the problem.

2 TWO-STAGE APPROACH

In our two-stage approach, we first solve a relaxation of the problem described in the previous section using a MIP solver and then, we repair the solution found and generate a feasible one. First, we describe the relaxation (**R-Model**) of our short-term scheduling problem, where some restrictions concerning the breaks or technical services are omitted.

Let J be the set of jobs in the underground mine under consideration. Binary decision variables b_j are 1 if $j \in J$ is processed. Moreover, we introduce binary decision variables x_{jw} and y_{jm} that are 1 if j is processed by worker $w \in W_j$ and machine $m \in G_j$, respectively. G_j and W_j are subsets of the set of available machines G and workers W that can process job j. We also define binary decision variables z_{jwm} that are 1 if worker w and machine m are assigned to job j.

$$\sum_{m \in G_j} y_{jm} = b_j \quad \forall j \in J \tag{1}$$

$$\sum_{w \in W_j} x_{jw} = b_j \quad \forall j \in J \tag{2}$$

$$x_{jw} + y_{jm} \leq 1 + z_{jwm} \quad \forall j \in J, \forall w \in W_j, \forall m \in G_j \tag{3}$$

$$z_{jwm} \leq x_{jw} \quad \forall j \in J, \forall w \in W_j, \forall m \in G_j \tag{4}$$

$$z_{jwm} \leq y_{jm} \quad \forall j \in J, \forall w \in W_j, \forall m \in G_j \tag{5}$$

Let PT_{jwm} be the given parameter that denotes the processing time of job j by worker w and machine m. The actual processing time of job j is then $p_j = \sum_{w \in W_j} \sum_{m \in G_j} z_{jwm} \cdot PT_{jwm}$. In

this paper, we use a sequence-based formulation for our **R-Model**. So, we introduce binary decision variables v_{jr} that are 1 if job j is completed before job r is started.

$$v_{jr} + v_{rj} \leq 1 \quad \forall j,r \in J : j \neq r \tag{6}$$

We have to consider a break for each worker (see **R1**) that leads to the absence of this worker in a specific time. Each worker w has to make a Δ-minute break so that the start time of the break ρ_w lies in a predefined interval $[varphi^{\alpha}, \varphi^{\omega}]$. In our relaxation, we do not allow that the processing of a job overlaps the break of the worker who processes this job. However, the break may overlap the drive between two jobs or the technical services that may be executed for a machine.

$$\varphi^{\alpha} \leq \rho_w \quad \forall w \in W \tag{7}$$

$$\rho_w \leq \varphi^{\omega} \quad \forall w \in W \tag{8}$$

$$S_j \leq \rho_w + \left(1 - \omega_j^s\right) M + \left(1 - x_{jw}\right) M \quad \forall j \in J, \forall w \in W_j \tag{9}$$

$$\rho_w + \boxtimes \leq S_j + \omega_j^s M + \left(1 - x_{jw}\right) M \quad \forall j \in J, \forall w \in W_j \tag{10}$$

$$\rho_w \leq S_j + p_j + \left(1 - \omega_j^e\right) M + \left(1 - x_{jw}\right) M \quad \forall j \in J, \forall w \in W_j \tag{11}$$

$$S_j + p_j \leq \rho_w + \omega_j^e M + \left(1 - x_{jw}\right) M \quad \forall j \in J, \forall w \in W_j \tag{12}$$

$$\omega_j^s + \omega_j^e \leq 1 + \delta_j \quad \forall j \in J \tag{13}$$

$$\delta_j \leq \omega_j^s \quad \forall j \in J \tag{14}$$

$$\delta_j \leq \omega_j^e \quad \forall j \in J \tag{15}$$

If the start time of the break of a worker, who processes job j, is during the processing of j ($\omega_j^s = 1$ and $\omega_j^e = 1$), binary decision variable δ_j takes the value of 1 and a Δ-minute break must be considered for the duration of j, additionally.

At the beginning of the working shift, the first technical services td_m^{α} must be performed on machine m and the assigned worker drives d_{0jm} time units to the first job.

$$\sum_{m \in G_j} (td_m^{\alpha} + d_{0jm}) \cdot y_{jm} \leq S_j + \left(1 - b_j\right) M \quad \forall j \in J \tag{16}$$

At each working place in an underground location, several operation types must be executed in a specific order related to the prescribed production cycle. Let ul_j be the underground location of j, ml_j be the working place of j in ul_j, and $order_j$ be the position of j in the given order for ml_j. In a working place, always job j with the minimum value of $order_j$ must be completed before any job r with a greater value of $order_r$ can be started.

$$b_r \leq b_j \quad \forall j,r \in J : j \neq r, ul_j = ul_r, ml_j = ml_r, order_j < order_r \tag{17}$$

$$\begin{aligned} &S_j + p_j + \delta_j \cdot \Delta \leq S_r + \left(2 - b_j - b_r\right) M \\ &\forall j, r \in J : j \neq r, ul_j = ul_r, ml_j = ml_r, order_j < order_r \end{aligned} \tag{18}$$

For jobs that are processed by the same worker, a precedence relation must be considered. As mentioned in Sect. 1, if a worker changes his machine, he has to go to the new machine (transfer time), has to do the first technical services, and he can drive the machine to the location of the new job. In our relaxed model, we neglect the time for the case, where a worker changes his machine.

$$S_j + p_j + \delta_j \cdot \Delta \leq S_r + \left(2 - x_{jw} - x_{rw}\right) M + \left(1 - v_{jr}\right) M \quad \forall j,r \in J : j \neq r, \forall w \in W_j \bigcap W_r \tag{19}$$

$$S_r + p_j + \delta_r \cdot \Delta \leq S_j + \left(2 - x_{jw} - x_{rw}\right) M + v_{jr} M \quad \forall j,r \in J : j \neq r, \forall w \in W_j \bigcap W_r \tag{20}$$

Moreover, if two jobs are processed by the same machine, a driving time between the jobs must be taken into account.

$$S_j + p_j + d_{jrm} + \delta_j \cdot \Delta \leq S_r + \left(2 - y_{jm} - y_{rm}\right) M + \left(1 - v_{jr}\right) M \quad \forall j,r \in J : j \neq r, \forall m \in G_j \bigcap G_r \tag{21}$$

$$S_r + p_j + d_{rjm} + \delta_r \cdot \Delta \leq S_j + \left(2 - y_{jm} - y_{rm}\right) M + v_{jr} M \quad \forall j,r \in J : j \neq r, \forall m \in G_j \bigcap G_r \tag{22}$$

After processing the last job on a machine, last technical services must be performed for the machine. Let *Shift* be the duration of the working shift. The following constraints guarantee that if the processing of a job exceeds *Shift* ($id_j = 1$), continuous decision variable $grad_{jwm}$ specifies, which percentage of j is achieved during the shift.

$$S_j + p_j + \delta_j \cdot \Delta + \sum_{m \in G_j} td_m^{\omega} \cdot y_{jm} \leq Shift + id_j M \quad \forall j \in J \tag{23}$$

$$Shift \leq S_j + p_j + \delta_j \cdot \Delta + \sum_{m \in G_j} td_m^{\omega} \cdot y_{jm} + \left(1 - id_j\right) M \quad \forall j \in J \tag{24}$$

$$S_j + \delta_j \cdot \Delta + \sum_{w \in W_j} \sum_{m \in G_j} grad_{jwm} \cdot PT_{jwm} + \sum_{m \in G_j} td_m^{\omega} \cdot y_{jm} \leq Shift + \left(1 - id_j\right) M \, \forall j \in J \tag{25}$$

$$S_j + \delta_j \cdot \Delta + \sum_{w \in W_j} \sum_{m \in G_j} grad_{jwm} \cdot PT_{jwm} + \sum_{m \in G_j} td_m^{\omega} \cdot y_{jm} \geq Shift \cdot id_j \, \forall j \in J \tag{26}$$

Consequently, if a job is processed ($b_j = 1$) and its duration does not exceed the working shift ($id_j = 0$), $grad_{jwm}$ must take the value of 1.

$$\sum_{w \in W_j} \sum_{m \in G_j} grad_{jwm} \leq \left(id_j + b_j\right) M \quad \forall j \in J \tag{27}$$

$$1 - \left(id_j - b_j + 1\right) M \leq \sum_{w \in W_j} \sum_{m \in G_j} grad_{jwm} \quad \forall j \in J \tag{28}$$

For the other mining-specific requirements **R2**–**R5**, we formulate the following constraints. Note that we write $j \prec r$ if job j must be completed before job r can be started.

R2:

$$S_j + p_j + \delta_j \cdot \Delta \leq S_r + \left(2 - b_j - b_r\right) M + \left(1 - v_{jr}\right) M \quad \forall j,r \in J : j \neq r, ul_j = ul_r \tag{29}$$

$$S_r + p_j + \delta_r \cdot \Delta \leq S_j + \left(2 - b_j - b_r\right) M + v_{jr} M \quad \forall j,r \in J : j \neq r, ul_j = ul_r \tag{30}$$

R3:

$$b_r \le b_j \quad \forall j,r \in J : j \ne r, ul_j = ul_r, j \prec r \tag{31}$$

$$S_j + p_j + \delta_j \cdot \Delta \le S_r + \left(2 - b_j - b_r\right) M \quad \forall j,r \in J : j \ne r, ul_j = ul_r, j \prec r \tag{32}$$

R4:

$$b_r \le b_j \quad \forall j,r \in J : j \ne r, ul_j = ul_r, started_j = 1, started_r = 0 \tag{33}$$

R5:

$$id_j = 0 \quad \forall j \in J : type_j = 4 \tag{34}$$

To realize **R6**, we introduce binary decision variables ma_{wm} that are 1 if w is assigned to m.

$$y_{jm} + x_{jw} \le 1 + ma_{wm} \quad \forall j \in J, \forall w \in W_j, \forall m \in G_j \tag{35}$$

$$\sum_{m \in G} ma_{wm} \le 2 \quad \forall w \in W \tag{36}$$

$$\sum_{w \in W} ma_{wm} \le 2 \quad \forall m \in G \tag{37}$$

Let ton_j be the expected amount of material after processing of job j and ton_k^{pre} be the predetermined quantity (target value) for production step k. We can determine the lower deviation from ton_k^{pre} for each production step by the following constraints.

$$ton_k^{pre} - \sum_{j \in J : type_j = k} \sum_{w \in W_j} \sum_{m \in G_j} grad_{jwm} \cdot ton_j \le dev_k \quad \forall k \in K \tag{38}$$

Our goal is to have a consistent progress so that the following function must be minimized.

$$\sum\nolimits_{k \in K} dev_k^2$$

If we determine the maximum lower deviation as follows:

$$dev_k \le dev^{max} \quad \forall k \in K, \tag{39}$$

we can then approximate the quadratic objective function by the following linear one.

$$\sum_{k \in K} dev_k + dev^{max} \tag{40}$$

After finding a solution for **R-Model** (Min. (40) s.t. (1)–(39)), the solution is used as an input for Algorithm 1 to generate a feasible solution for the problem instance at hand. In Algorithm 1, we first determine the sequence of the processing of jobs for each machine and each worker. After that, for each machine m, between two consecutive jobs j and r that are processed by different workers, a first technical service must be inserted before starting r. Subsequently, start times of all of the jobs that have to be started after the completion of r must be updated. For each worker w, if w changes his machine and goes from machine m to m', we eventually have to consider last technical services if w processed the last task on m. The worker then goes to the parking location of m', performs first technical services, and drives m' to the location of the next job. Consequently, start times of all of the related jobs must be updated. Then, we check if there are overlaps between breaks of workers and the activities that workers have to perform. In this case, we consider the effect of workers' breaks on start times or durations of jobs and update the start times of all of the affected jobs. The steps above are repeated until there are no more changes in start times of jobs.

Algorithm 1. Repair solution.

Input: problem instance D, solution of **R-Model**
repeat
  For each machine, determine the sequence of processed jobs by this machine;
  For each worker, determine the sequence of processed jobs by this worker;
  for all processed jobs j **do**
    $S_j^1 = S_j$
  for all machines m **do**
    if two consecutive jobs j and r on m are performed by different workers **then**
      Insert the time for first technical services for m before S_r
      for all jobs j' with $v_{rj'} = 1$ **do**
        update $S_{j'}$
  for all workers w **do**
    if two consecutive jobs j and r on w are performed by different machines **then**
      Insert the potential last technical services for the machine assigned to j, the transfer time, first technical services for the machine assigned to r, and the driving time from the direct predecessor of r on the assigned machine to r
      for all jobs j' with $v_{rj'} = 1$ **do**
        update $S_{j'}$
  for all workers w **do**
    if the break of w, who processes job j, overlaps any of the first technical services for the machine assigned to j, driving times to j, the processing of j, or the last technical services for the machine assigned to j **then**
      Consider the break of w for S_j or the duration of j
      for all jobs j' with $v_{jj'} = 1$ **do**
        update $S_{j'}$
  for all jobs j with $type_j = 4$ **do**
    Eliminate j if the processing of j exceeds the duration of the working shift
until $S_j^1 = S_j \forall_j$
Determine the objective value according to the new schedule
return The feasible schedule

3 COMPUTATIONAL STUDY

To show the suitability of our proposed solution approach, we compare the results of our two-stage approach with the heuristic procedure introduced by Schulze & Zimmermann (2017). For this purpose, we generated 100 test instances based on the case study presented in Schulze & Zimmermann (2017), which depict realistic problems in a German potash underground mine. In Algorithm 2, an overview of the constructive heuristic approach is given (for more details see Schulze & Zimmermann (2017) and Schulze (2016)).

Algorithm 2. Constructive heuristic introduced by Schulze & Zimmermann (2017).

Initialization (* *construction* *)
Priority-based scheduling
Staff changes (* *improvement* *)
repeat
  Scheduling downstream operations
  Staff changes
  Replenishment
until total amount of potash cannot be increased
Job reassignment
Insertion of **technical services** (* *post-processing* *)
Insertion of **breaks** for workers
return solution

Table 2. Comparison of the approaches.

	Number of best solutions found	Gap to the best solution found
Two-Stage approach	70	6.1% (20.3%)
Constructive heuristic	30	45.2% (64.6%)

The heuristic procedure is embedded in a multi-start algorithm, where jobs, machines, and workers are chosen based on selection probabilities that are determined by priority values. For the heuristic approach in this paper, we use the setting that is currently used in the underground mine under consideration. In the corresponding assigning method, jobs and workers are randomly chosen, and machines are selected regarding the shortest driving times to the selected job or regarding the shortest processing time based on the selected job and the selected worker.

All tests are executed on an Intel i7–7700 K@4.20 GHz machine with 64 GB RAM under Windows 10. The heuristic algorithm is implemented in Xpress IVE 8.4. For the two-stage approach, we used GAMS 25.1 and GUROBI solver 8.1.0 to solve **R-Model** and C++ to generate a feasible solution with the aid of Algorithm 1. Since we schedule only one working shift, we set an upper time limit of 900 seconds for both approaches that symbolizes a typical duration of a shift handover.

To compare the results achieved by the procedures, we use the value of $\sum_{k \in K} dev_k^2$ that shows how consistent the desired progress could be implemented at the end of the working shift compared to the given state at the beginning of the working shift.

Table 2 presents the number of best solutions found and an average gap to the best solution found. Let S_i^* be the solution found for instance i by procedure * and S^{best} be the best solution found. We calculate $\frac{S_i^* - S^{best}}{S_i^*}$ to determine the gap for instance i (gap_i). The numbers presented under "Gap to the best solution found" are obtained by arithmetic averaging over all instances. Note that the number in parentheses is the obtained gap by arithmetic averaging over the number of the instances for which the solution found is not equal to the best solution found (i.e., $\text{gap}_i \neq 0$).

We see that the two-stage approach can find for 70 instances the best solution, where the solutions found for the other 30 instances are 20.3% far from the best solution found. On the other hand, the solutions found by the constructive heuristic are on average 45.2% worse than the best solutions found. This number gets significantly worse (64.6%) if we make an average over the 70 instances for which the heuristic could not find the best solution. So, we can conclude that our two-stage approach performs quite promising.

4 CONCLUSION

In this paper, we consider a shift scheduling problem where machines and workers are simultaneously assigned to a selection of the available jobs. We formulate a relaxation of the problem described in Sect. 1 and introduced an algorithm to generate feasible solutions using the solution achieved by the relaxation. The results of a preliminary performance analysis using realistic instances show that the solutions of our proposed two-stage approach clearly outperform the solutions which are currently generated by a constructive heuristic procedure.

Future work concerns the development of good lower bounds for our problem. Moreover, the total output can additionally be taken into account. Considering the trade-off between the presented objective function and the total excavated amount of material could provide new insights.

REFERENCES

Agnetis, A., Murgia, G., & Sbrilli, S., 2014. A job shop scheduling problem with human operators in handicraft production. *International Journal of Production Research* 52(13): 3820–3831.

Ahmadi-Javid, A. & Hooshangi-Tabrizi, P., 2017. Integrating employee timetabling with scheduling of machines and transporters in a job-shop environment: A mathematical formulation and an anarchic society optimization algorithm. *Computers & Operations Research* 84: 73–91.

Artigues, C., Gendreau, M. & Rousseau, L.M., 2006. A flexible model and a hybrid exact method for integrated employee timetabling and production scheduling. In *International Conference on the Practice and Theory of Automated Timetabling* (pp. 67–84). Berlin, Heidelberg: Springer.

Artigues, C., Gendreau, M., Rousseau, L.M. & Vergnaud, A., 2009. Solving an integrated employee timetabling and job-shop scheduling problem via hybrid branch-and-bound. *Computers & Operations Research* 36(8): 2330–2340.

Benavides, A.J., Ritt, M. & Miralles, C., 2014. Flow shop scheduling with heterogeneous workers. *European Journal of Operational Research* 237(2): 713–720.

Campos Ciro, G., Dugardin, F., Yalaoui, F. & Kelly, R., 2016. Open shop scheduling problem with a multi-skills resource constraint: A genetic algorithm and an ant colony optimisation approach. *International Journal of Production Research* 54(16): 4854–4881.

Daniels, R.L. & Mazzola, J.B., 1994. Flow shop scheduling with resource flexibility. *Operations Research* 42(3): 504–522.

Daniels, R.L., Mazzola, J.B. & Shi, D., 2004. Flow shop scheduling with partial resource flexibility. *Management Science* 50(5): 658–669.

Frihat, M., Sadfi, C. & Hadj-Alouane, A.B., 2014. Optimization of integrated employee timetabling and hybrid job shop scheduling under time lag constraints. In *International Conference on Control, Decision and Information Technologies (CoDIT)*, (pp. 282–287). IEEE.

Guyon, O., Lemaire, P., Pinson, E. & Rivreau, D., 2014. Solving an integrated job-shop problem with human resource constraints. *Annals of Operations Research* 213(1): 147–171.

Huq, F., Cutright, K. & Martin, C., 2004. Employee scheduling and makespan minimization in a flow shop with multi-processor work stations: a case study. *Omega* 32(2): 121–129.

Mencia, C., Sierra, M.R. & Varela, R., 2013. An efficient hybrid search algorithm for job shop scheduling with operators. *International Journal of Production Research* 51(17): 5221–5237.

Puttkammer, K., Kleber, R., Schulz, T. & Inderfurth, K., 2011. Simultane Maschinenbelegungs- und Personaleinsatzplanung in KMUs anhand eines Fallbeispiels aus der Druckereibranche. In Sucky, E., Asdecker, B., Dobhan, A., Haas, S. & Wiese, J. (eds), *Logistikmanagement: Herausforderungen, Chancen und LÃ¶sungen*: 229–247. University of Bamberg Press.

Ramya, G. & Chandrasekaran, M., 2014. Shuffled frog leaping algorithm approach to employee timetabling and job shop scheduling. *International Journal of Internet Manufacturing and Services 4 & 25* 3(3): 178–203.

Santos, F., Fukasawa, R. & Ricardez-Sandoval, L., 2018. An integrated personnel allocation and machine scheduling problem for industrial size multipurpose plants. *IFAC-PapersOnLine* 51(18): 156–161.

Schulze, M., 2016. Ein hierarchischer Ansatz zur LÃ¶sung von Ablaufplanungsproblemen im Bergbau: Darstellung am Beispiel des Ã–rterbaus. Aachen: Shaker.

Schulze, M., Mathiak, T. & Haney, C., 2017. Using operations research techniques to increase efficiency in German potash mining. In *Application of Computers and Operations Research in the Mineral Industry; Proc. intern. symp., Golden, Colorado, 2017* 5: 9–15.

Schulze, M., Rieck, J., Seifi & C., Zimmermann, J., 2016. Machine scheduling in underground mining: An application in the potash industry. *OR Spectrum* 38(2): 365–403.

Schulze, M., Zimmermann, J., 2017. Staff and machine shift scheduling in a German potash mine. *Journal of Scheduling* 20(6): 635–656.

D. Paper IV

European Journal of Operational Research 292 (2021) 27–42

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

A new mathematical formulation for a potash-mine shift scheduling problem with a simultaneous assignment of machines and workers

Cinna Seifi*, Marco Schulze, Jürgen Zimmermann

Julius-Albert-Straße 2, Clausthal-Zellerfeld 38678, Germany

ARTICLE INFO

Article history:
Received 13 February 2020
Accepted 7 October 2020
Available online 12 October 2020

Keywords:
Scheduling
Shift scheduling
Sequence-dependent setup times
Mixed-integer linear programming
Underground mining

ABSTRACT

In this paper, we introduce a mixed-integer linear program for a shift scheduling problem in a German potash mine. In particular, we consider a short-term (work shift) production scheduling problem, where drill-and-blast mining operations have to be assigned to machines and workers simultaneously. Since we deal with several sequence-dependent setup, changeover, and removal times, TSP-variables are used in the mathematical program to determine the processing-sequence of the operations on each worker and each machine, respectively. In addition, several mining-specific requirements are taken into account to obtain a solution that can be put into practice. Computational experiments are conducted on problem instances of realistic size derived from real-world data. The results show that our new mixed-integer linear formulation outperforms both existing solution procedures for the problem at hand.

1. Introduction

The dynamics in the commodities industry has put mining operators under constant pressure to stabilize operating margins. This fact provides a good challenge and opportunity for using digital tools and operations research techniques to increase the efficiency and effectiveness of the mine's production and engineering processes (Schulze, Mathiak, & Haney, 2017).

In this paper, we address a short-term production scheduling problem in a potash mine, where a discontinuous production cycle using drilling and blasting takes place. For the same problem, Schulze and Zimmermann (2017) and Schulze et al. (2017) proposed a rule-based constructive procedure, and Seifi, Schulze, and Zimmermann (2019) devised a two-stage approach. We show that the problem can be mathematically formulated as a mixed-integer linear program using TSP-variables. The results, achieved in a reasonable amount of time using the proposed mathematical formulation, clearly surpass the results obtained by the existing solution approaches.

Due to the flat-bedded deposit, the room-and-pillar mining method is applied in the underground potash mine under consideration. The main characteristics of this mining method are an area-wide expansion of the deposit and a resulting high number of possible extraction points, the so-called working places. The excavation of potash at the working places is based on the drilling and blasting technique. This means that the underground production process follows discrete process steps which need to be conducted in a specific order. On the one hand, the high number of working places allows a very flexible planning and control of the extraction, but on the other hand, it is especially complex to generate work orders in shifts for the personnel and mobile equipment. The complexity is reinforced by the fact that not all working places are in the same state (i.e., different process steps are required at different working places) and not all currently available process steps can be operated within a single work shift. As a consequence, it is imperative to decide, for the work shift under consideration, which working places are processed and in which order. Hence, we have to consider a selection and sequencing problem which is less studied in the field of scheduling. Moreover, in contrast to classical machine scheduling problems (e.g., flow or job shop) where the jobs that have to be processed are "delivered" to the necessary production stages (machines), we consider mobile machines that have to drive to the working places. For this reason, we have to consider planning-dependent driving times of machines, i.e., the driving times depend on the job-sequence on the respective machine. Also, we have to consider a simultaneous assignment of machines and workers, where it must be taken into account that the mobile machines have different speeds and that the workers have different skills with different experiences, i.e., skill levels (Schulze & Zimmermann, 2017). Hence, the processing times are strongly dependent on the assigned machine and worker. Another important aspect that is less investigated in the field of personnel scheduling is that a worker can change an assigned machine within a work

* Corresponding author.
E-mail address: cinna.seifi@tu-clausthal.de (C. Seifi).

https://doi.org/10.1016/j.ejor.2020.10.007
0377-2217/

shift (cf. Van den Bergh, Beliën, De Bruecker, Demeulemeester, and De Boeck (2013)).

The remainder of the paper is organized as follows: in Section 2 an explicit description of the problem at hand, as well as a comprehensive literature review are given. In Section 3, we introduce a mixed-integer linear program (MILP) formulation for our shift scheduling problem using TSP-variables, where a suitable set of operations has to be selected for execution. The introduced MILP-model is verified and validated based on a small example in Appendix A. In Section 4, we evaluate the performance of the developed MILP-model in comparison to two existing approaches using problem instances derived from real-world data. Moreover, a detailed comparison of the solution approaches for problem instances with similar characteristics is given in Appendix B. The paper concludes with a summary of the achieved results and an outlook on further research.

2. Problem specification and related literature

In this section, we describe our shift scheduling problem and give a literature review on existing approaches for solving shop scheduling problems with a simultaneous assignment of machines and multi-skilled workers.

Potash mines are usually underground mines, which are divided into several *mining districts*. Each mining district may comprise an area of several square kilometers. Due to this spatial extent, several *tipple areas* (usually between three and six) are considered for a mining district. The key element in a tipple area is a feeder breaker, where the lumps are broken and the material is carried to a shaft or an interim storage facility on a conveyor belt. To each feeder breaker, a number of *underground locations* (usually between three and eleven) can be assigned. An underground location is the smallest section in an underground mine and involves *working places* that have a short geographical distance to each other. In order to extract the crude salt in an underground potash mine, the room and pillar mining method using the drilling and blasting technique is generally applied. In this method, to expose the rooms, the material is blasted in *blocks*, and pillars are left for support purposes (see Schulze, Rieck, Seifi, and Zimmermann (2016) for more details about the room and pillar mining method). At each working place to remove a block, the following nine mining operations (process steps) must be processed in a chronological order: (1) a front-end loader transports the crude salt from the working place to the assigned feeder breaker; (2) a scaling machine detaches loose rock fragments from the roof or side walls; (3) a small loader removes the detached material from the working place; (4) a roof bolter installs anchor bolts onto the roof structure; (5) a drilling jumbo drills three adjacent horizontal boreholes; (6) a small loader removes the resultant dust[1]; (7) a blasthole drilling machine drills several blastholes into the salt face; (8) a charging vehicle fills the blastholes with explosive substances; and (9) the blasting that occurs between the work shifts. After completing the whole nine mining operations at a working place, the salt face moves on in the length of the extracted block, and a new working place becomes available K+S AG (2013).

Fig. 1 illustrates a tipple area that consists of four underground locations. The underground locations in this figure have two to three working places. The number given in parentheses is the specific mining operation that has to be done next for the associated working place. For many reasons, working places usually have different states, which means mining operations that have to be executed next at different working places are not the same.

[1] Note that mining operations (3) and (6) make use of the same small loaders.

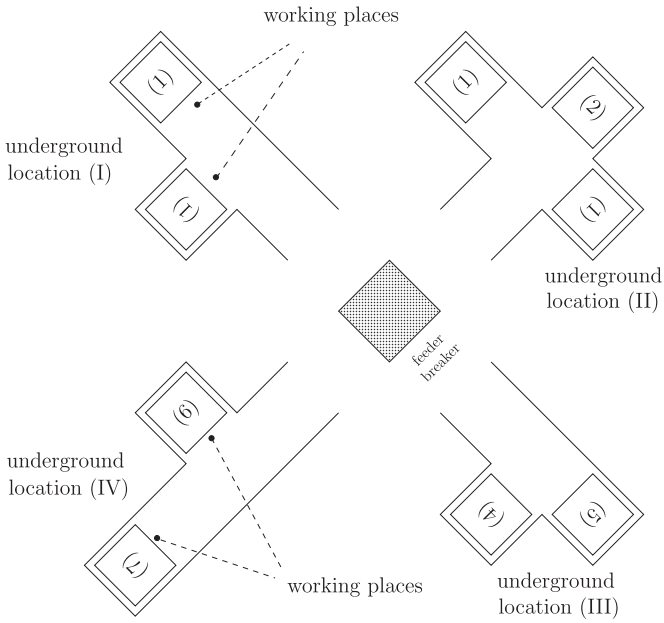

Fig. 1. The relation between a working place and an underground location in a tipple area.

The work that has to be done at each working place can be considered as a *job* that consists of several operations; each of those represents one of the nine mining operations explained above. Since blasting (mining operation (9)) does not require any machines or workers, we only consider the mining operations (1) to (8) as *production stages*. Processing an operation of a job in a production stage requires exactly one machine and one worker. In the present mine, there are some identical or uniform (mobile) machines available for each mining operation (production stage) so that operation k of a job will be done on one of the available machines in the $k-$th production stage. Note that the machines must be handled (and moved) by a worker with appropriate skill. Without considering the workers, the production environment can be classified as a variant of a hybrid flow shop (HFS) scheduling problem (see Schulze et al. (2016)). According to Ruiz and Vázquez-Rodríguez (2010), variants of HFS scheduling problems have the following attributes in common (which is the case for the problem at hand, too):

(i) a set of jobs has to be processed in at least two stages
–we have eight stages in our problem that need resources (machines and workers);
(ii) in at least one stage there is more than one machine available
–in the present mine, there are at least two machines in stage one;
(iii) each machine can operate at most one operation and an operation must be processed without preemption by at most one machine at the same time
–that is always the case in our problem. We consider the operations as non-preemptable because the assigned machines complete the processing of the operations regardless of possible interruptions. After any possible interruption (e.g., because of the workers' breaks or at the end of the work shift), the processing continues at the next available opportunity by the same machine; and
(iv) a job might skip any number of stages; however, it is processed in at least one of them
–although, different working places can have different states so that the order of operations for different jobs can begin

with different operation index numbers; but, a job must be processed in at least one stage.

From an operational point of view (e.g., because of many uncertainties affecting the availability of machines), we consider a planning horizon of one work shift. As a result, the processing of an operation or a job may typically be interrupted at the end of the work shift, since processing the whole eight operations for a job is not generally possible within this short time horizon.[2] Furthermore, because of the large number of available working places, we cannot process all of the pending jobs within a work shift. Thus, it must be decided whether an operation of a job is processed within the current shift or not.

As mentioned, each machine in each production stage must be handled by a worker with an appropriate skill. For assigning a worker to a machine, it must be noted that not all workers can cope with all machines, i.e., workers have particular skills. Moreover, workers who can deal with a machine have different skill levels, which lead to different handling times with a specific machine. Besides the fact that the available machines in each production stage can have different speeds, the effect of the different skill levels of workers on the *processing time* of an operation cannot be neglected. Furthermore, we have to consider some *setup times* and *changeover times* if a worker changes his machine. These times depend on the processing-sequence of the operations assigned to a worker on different machines, and their durations in comparison to the duration of a work shift are not negligible. Hence, assigning the proper workers to machines for processing the operations in the right sequence plays a decisive role in the solution to the problem at hand.

In addition to the setup and changeover times mentioned above, there are some setup times and removal times that are machine- and sequence-dependent. The occurrence of the setup, changeover, and removal times and their dependence on the sequence of the processing of the operations are explained below:

setup time Before processing the first operation allocated to a machine, a preventive maintenance (first technical services) must be done. Also, if a worker changes his machine, he has to perform the first technical services for the new machine for safety reasons regardless of whether the new machine was already in use or not. The duration of the first technical services is constant for each machine and does not depend on the operation assigned to a machine. However, the sequence of the processing of operations on a machine or by a worker must be determined. Therefore, we consider these times as machine- and sequence-dependent setup times. In addition, since the machines are mobile, there are driving times between operations that are processed by the same machine. Driving times between two operations depend on their locations and the driving speed of the machine that processes these operations. Hence, we can consider the driving times as machine- and sequence-dependent setup times, too.

changeover time If a worker processes two consecutive operations on different machines, he has to change his machine and go to the current position of the second machine. Whether the worker walks to the new machine or is picked up by a transport vehicle depends on the geographical distance to the new machine. The walking speed of all workers, as well as the driving speed of the transport vehicle, can be considered as constant. Moreover, the number of transport vehicles is large enough such that they are always available when they are needed. Consequently, the time that a worker needs to change the machine depends only on the location of the first operation and the location of the new machine. The parking position of a machine is usually near the location of the previous operation done by this machine so that changeover time has to be considered as sequence-dependent.

removal time After processing the last operation allocated to a machine as well as before the end of the work shift if an operation has to be interrupted, each machine must be cleaned and fueled (last technical services). Like the first technical services, performing the last technical services depends on the machine and the sequence of the processing of operations on the machine (the last operation assigned to a machine within the work shift must be determined). As a result, the last technical services can be considered as machine- and sequence-dependent removal times.

Until now, we described the production environment and pointed out the importance of considering the workers in our problem. Since the workers and machines are principally assigned to one specific mining district and cannot be exchanged between different mining districts during a work shift, we take only one mining district in our shift scheduling problem into consideration. In a mining district, besides the circumstances mentioned above, there are some mining-specific requirements (MR) that must be fulfilled:

- MR (1) In accordance with labor law, workers have to take a δ-minute break within a predetermined time interval in each work shift. It may be the case that a worker has to interrupt the processing of an assigned operation to take his break. After the break, the worker will continue with the interrupted operation. Therefore, the workers' break can lead to a delay in the processing of an operation.
- MR (2) There are disjunctive constraints between the operations of different jobs in an underground location. In an underground location, for security reasons, it is not allowed that more than one machine is applied at the same time. Therefore, only one operation of a job (working place) in an underground location can be processed at any point in time.
- MR (3) In an ideal situation, a consistent progress is given at all working places of an underground location, i.e., the same operation type can be performed for all jobs in an underground location. However, excavations may be perturbed due to machine breakdowns or other technical failures, and a different excavation state may appear for different jobs. In this situation, the jobs (working places) in an underground location are prioritized according to their next operation. In an underground location, a job with the smallest operation index number has the highest priority.
- MR (4) In addition to MR (3), in an underground location, operations that are interrupted at the end of the last shift have a higher priority. That means, if there is a job (working place) with an operation that is started but not

[2] Note that mining operation (9) always takes place between the work shifts and does not require any resources.

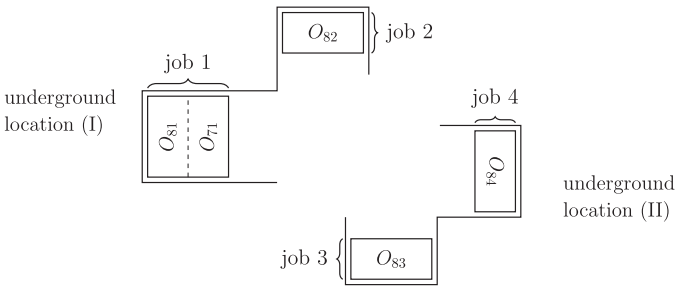

Fig. 2. An example of different jobs in two underground locations.

completed in the last shift, the operations of the other jobs in this underground location can be processed if and only if the interrupted operation is processed, too.

MR (5) Operations on stage (5) are non-interruptible. That means, if those operations cannot be completed until the end of the work shift, they must not be started.

MR (6) In the present mine, not more than two different workers can be assigned to a machine to process different operations. Also, a worker cannot be assigned to more than two different machines during the work shift. That is a decision of the superordinate planning level to reduce the number of machine changes and the risk involved.

Each operation of a job is characterized by an amount of crude material which is expected to be excavated after the completion of the operation. For each mining operation, a target value for the output is given that should be achieved within a work shift. The target values are predetermined from a superordinate planning level with the aid of a constant comparison of target and actual data of the excavated raw material. At the end of the work shift, the amount of material that is extracted is calculated for each operation. In this regard, if an operation of a job is interrupted at the end of the work shift, the percentage of this operation that has been processed within the work shift must be determined. Accordingly, the lower deviation from the predefined target value is figured. Lower deviation means, if the excavated material exceeds the target value, the difference will not be considered in the objective function. The aim of the optimization problem under consideration is to minimize the lower deviations, cumulated over all of the mining operations so that the progress of mining stays consistent.

Fig. 2 illustrates two underground locations, each of which has two working places. Jobs (working places) are numbered from job 1 to job 4, and the associated mining operations are given. Let O_{kj} denote operation k of job j. As mentioned, jobs can have different states. In underground location (I), for job 1, the operations seven and eight, and for job 2, only operation eight must be completed. In underground location (II), for both jobs 3 and 4, operation eight has to be processed. For the problem illustrated in Fig. 2, a feasible schedule for machines and workers is given in Fig. 3.

In underground location (I), O_{71} has a smaller index number and, therefore, a higher priority than O_{82} (see MR (3)). For the processing of O_{71}, worker w_1 is assigned to a blasthole drilling machine. At the beginning of the work shift, w_1 performs the first technical services on the blasthole drilling machine and moves (drives) the machine from its parking position at the beginning of the shift to the location of O_{71}. Because of MR (2), no operation of job 2 can be processed before the completion of O_{71}. Moreover, MR (2) implies that in underground location (II), there is a disjunctive constraint between O_{83} and O_{84}. According to the schedule, O_{83} is processed first in underground location (II) by worker w_2 and by charging vehicle (i). Analogously to w_1 for the blasthole drilling machine, w_2 does the first technical services for charging vehicle (i), drives the machine to the location of O_{83}, and begins with the processing. Regarding the schedule, w_1 has to process O_{84} on charging vehicle (i) after completion of O_{71}, where after completing O_{83}, w_2 is assigned to charging vehicle (ii) to process O_{82}. Since O_{71} is the last task allocated to the blasthole drilling machine, w_1 performs the last technical services and then, goes to the current position of charging vehicle (i). w_1 has to wait till O_{83} is completed, do the first technical services[3], and drive the machine to the assigned operation O_{84}. On the other hand, w_2 goes to the current position of charging vehicle (ii), takes his break, does the first technical services, and moves the machine to the location of O_{82}. w_1 takes a break during the processing of the assigned task. The processing duration of operation O_{84} without considering the worker's break is given in a dashed-box. O_{84} is the last operation assigned to charging vehicle (i), and w_1 performs the last technical services for the machine. O_{81} cannot be processed according to the schedule (not all operation can be processed during the work shift). O_{82} cannot be completed within the work shift and must be interrupted at the end of the work shift. Therefore, the duration of the last technical services, that must be done for the machine within the work shift, must be considered. As a result, only a certain part of operation O_{82} is processed during the work shift. Let ton_{kj} be the tonnage that is excavated if O_{kj} is completed. Moreover, let P_{kj} be the needed processing time for completing the operation O_{kj} by a particular worker and a specific machine, and $\tilde{P}_{kj}$ be the part of the processing of operation O_{kj} that is performed during the work shift. If we denote the excavated tonnage for operation k with T_k, then T_7 is equal to ton_{71}, and T_8 is calculated as follows:

$$T_8 = \frac{\tilde{P}_{82}}{P_{82}} \cdot ton_{82} + ton_{83} + ton_{84}.$$

Let the target value (the tonnage that is expected to be excavated) for mining operation k be ton^e_k. Our problem aims to minimize the accumulated lower deviations from the target values over the eight mining operations. That value is determined as follows:

$$\sum_{k=1}^{8} \max\{ton^e_k - T_k, 0\}.$$

In the literature, there are several works studying scheduling problems in underground mines or HFS scheduling problems. For instance, Newman, Rubio, Caro, Weintraub, and Eurek (2010) give an overview of operations research methods used for scheduling problems in underground mines, and Allahverdi (2015) presents a comprehensive survey on scheduling problems with setup times.

Considering human operators in a machine scheduling problem is known as Dual Resourced Constrained (DRC) system. Xu, Xu, and Xie (2011) and Ammar, Pierreval, and Elkosentini (2013) present overviews on contributions that considered DRC systems. The authors categorize the challenges regarding the workers based on (i) worker flexibility; (ii) worker assignment; and (iii) transfer costs of workers. According to worker flexibility, a flexibility level is introduced. For instance, a flexibility level of 2 means that a worker is capable of operating two different machines. In our work, the workers can operate a subset (or the set) of available machines with different skills; however, we limit the number of machine changes. Regarding the worker assignment, Xu et al. introduce a *where*/*when*-rule which dictates when a worker can be transferred to another production stage and where that worker is to be assigned when he is eligible for transfer. In the problem at hand, we have no additional constraints according to that classification

[3] Note that a worker, who changes his machine, has to do the first technical services for the new machine, regardless of whether the machine was already in use.

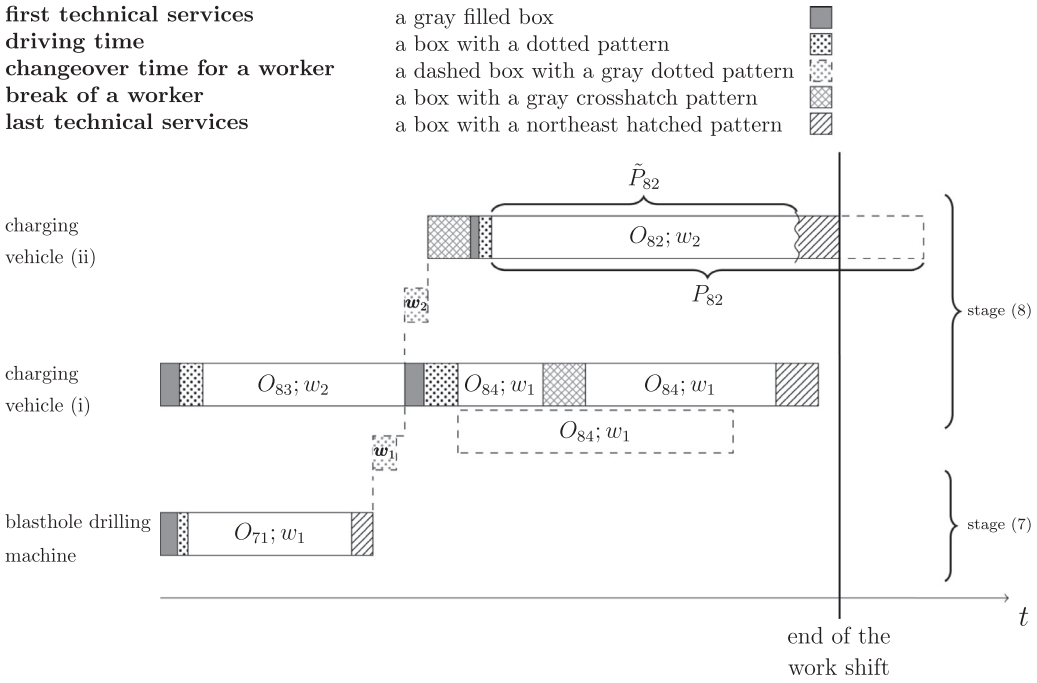

Fig. 3. A feasible solution to the problem illustrated in Fig. 2

Table 1
Literature review.

	Production environment	Objective function	Mathematical formulation	Setup times
Daniels et al. (2004)	FSP	makespan	time-indexed	–
Huq et al. (2004)	FSP	makespan	seq.-based	machine-dependent
Artigues et al. (2006)	JSP	multi-obj.	time-indexed	job-dependent
Artigues et al. (2009)	JSP	multi-obj.	seq.-based	job-dependent
Puttkammer et al. (2011)	JSP	multi-obj.	time-indexed	–
Mencía et al. (2013)	JSP	flow time	–	–
Ramya and Chandrasekaran (2014)	JSP	empl. cost	time-indexed	–
Frihat et al. (2014)	FJSP	empl. cost	seq.-based	–
Benavides, Ritt, and Miralles (2014)	FSP	makespan	seq.-based	–
Guyon et al. (2014)	JSP	empl. cost	time-indexed	–
Agnetis, Murgia, and Sbrilli (2014)	JSP	makespan	seq.-based	–
Behnamian (2014)	HFSP	diff. obj.	–	seq.-dependent
Yazdani, Zandieh, Tavakkoli-Moghaddam, and Jolai (2015)	JSP	makespan	seq.-based	–
Campos Ciro et al. (2016)	OSP	flow time	non-linear/seq.-based	machine-/job-dependent
Mencía, Sierra, Mencía, and Varela (2016)	JSP	makespan	–	–
Paksi and Ma'ruf (2016)	FJSP	tardiness	–	–
Ahmadi-Javid and Hooshangi-Tabrizi (2017)	JSP	makespan	seq.-based	seq.-dependent
Zhang, Wang, and Xu (2017)	JSP	makespan	seq.-based	–
Santos et al. (2018)	FJSP	throughput	time-indexed	–
Kress et al. (2019)	FJSP	diff. obj.	TSP-var./seq.-based	seq.-dependent
Yazdani et al. (2019)	FJSP	multi-obj.	–	–
Meng et al. (2019)	FJSP	problem-specific	pos.-based	–
Gong et al. (2020)	FJSP	diff. obj.	pos.-based	–

FSP: Flow shop scheduling problem; FJSP: Flexible job shop problem; HFSP: Hybrid flow shop scheduling problem; JSP: Job shop scheduling problem; OSP: Open shop scheduling problem; pos.-based: position-based MILP; seq.-based: sequence-based MILP; seq.-dependent: sequence-dependent; time-indexed: time-indexed MILP

where and when a worker can be transferred to another production stage. Also, Xu et al. and Ammar et al. mention that most papers did not give any consideration to costs and/or delays caused by worker transfer across different work stations. In our problem, we consider the delays caused by worker transfer.

To the few works that consider a transfer cost for human operators belong, e.g., Araz (2005) and Uzun Araz and Salum (2010), that propose a heuristic procedure and simulation-based scheduling approaches for a parallel machine scheduling problem. In what follows, we confine our literature review to the integrated worker assignment and *shop* scheduling problems in the last two decades. In Table 1, we summarized the papers regarding the production environments, the objective functions, the proposed mathematical formulations (if there is any), and whether some setup times are considered or not. Note that "multi-obj." means that the papers use multiple objective functions, where "diff. obj." means that the same problem is solved using different objective functions.

In terms of the production environment, most of the studies deal with JSPs (e.g., Artigues, Gendreau, and Rousseau (2006)) or FJSPs (e.g., Frihat, Sadfi, and Hadj-Alouane (2014)). In comparison, FSPs are considered less in the literature (e.g., Daniels, Mazzola, and Shi (2004)). The work of Behnamian (2014) is the only one that considers a hybrid flow shop scheduling problem but without suggesting any mathematical formulation.

In the view of objective functions to be considered, the most common one is makespan (e.g., Daniels et al. (2004)), while flow time (e.g., Mencía, Sierra, and Varela (2013)), employee cost (e.g., Ramya and Chandrasekaran (2014)), tardiness (e.g., Paksi and Ma'ruf (2016)), and throughput (e.g., Santos, Fukasawa, and Ricardez-Sandoval (2018)) are also used as objective functions in the reviewed papers. Artigues et al. (2006) and Artigues, Gendreau, Rousseau, and Vergnaud (2009) consider a linear combination of a production cost and an employee satisfaction cost as an objective function to be minimized. Puttkammer, Kleber, Schulz, and Inderfurth (2011) minimize a multi-objective function considering the lateness and the employee cost. In the work of Yazdani, Zandieh, and Tavakkoli-Moghaddam (2019), a multi-objective function concerning makespan as well as critical and total machine workload is taken into account. Behnamian (2014) heuristically solves one problem with different objectives, i.e., makespan, earliness, tardiness, and employee cost. Kress, Müller, and Nossack (2019) consider two objective functions, makespan and total tardiness. In the work of Gong et al. (2020), makespan, total worker cost, and a problem specific objective function, are taken into consideration.

The reviewed studies generally use time-indexed (e.g., Daniels et al. (2004)) or sequence-based (e.g., Huq, Cutright, and Martin (2004)) mathematical formulations. Meng, Zhang, Zhang, and Ren (2019) and Gong et al. (2020) propose a position-based mathematical formulation. Only Kress et al. (2019) use TSP-variables to formulate a vehicle routing problem in their work, which is showed to be a potent formulation for considering the driving times for the mobile machines. For workers' constraints, Kress et al. use a sequence-based formulation. Daniels et al. (2004), Huq et al. (2004), Artigues et al. (2006), Artigues et al. (2009), Puttkammer et al. (2011), Kress et al. (2019), Meng et al. (2019), and Gong et al. (2020) use MILP solver or suggest some exact procedures to solve small problem instances. Frihat et al. (2014) and Guyon, Lemaire, Pinson, and Rivreau (2014) apply decomposition and cut generation for solving larger problem instances up to 100 operations. They show that the proposed exact approaches outperform the solutions achieved by a MILP solver. Santos et al. (2018) solve larger instances using a MILP solver to optimality, which is probably possible because of the structure of the problem instances. In the test instances, 200 jobs must be processed on 25 processing units, where each processing unit has up to 10 identical machines, and the number of available workers is large enough. In other works mentioned in Table 1, the authors use some heuristic procedures to tackle large problem instances.

Finally, in terms of setup times, Artigues et al. (2006), Artigues et al. (2009), and Campos Ciro, Dugardin, Yalaoui, and Kelly (2016) consider some job-dependent removal times. In the work of Campos Ciro et al. (2016), some machine-dependent setup times are also considered. Behnamian (2014), Ahmadi-Javid and Hooshangi-Tabrizi (2017), and Kress et al. (2019) observe some sequence-dependent setup times, while Huq et al. (2004) take machine-dependent setup times into account. None of the papers mentioned in Table 1 deals with the setup times caused by the workers.

To the best of our knowledge, the published works on DRC systems did not consider the following aspects: (i) operating a subset of available jobs and operations; (ii) possible delays for the processing of the operations because of the workers' breaks; (iii) changeover times for workers; and (iv) possible interruptions of jobs at the end of the work shift. In summary, our literature review suggests that the solution procedures discussed in the literature are not suitable for our shift scheduling problem.

As mentioned before, Schulze and Zimmermann (2017) and Seifi et al. (2019) deal with our problem, but they do not introduce a linear program for the problem. Hence, solving the problem using a MILP solver is not possible. In Section 3, we formulate a mixed-integer linear program (MILP) using TSP-variables to model the problem mathematically.

3. Mathematical model

In this section, we introduce a MILP-formulation for the problem explained in Section 2. Since we consider sequence-dependent setup, changeover, and removal times, it is necessary to determine the exact processing-sequence of the operations on the assigned machines and the assigned workers. For this purpose, a position-based formulation is theoretically appropriate. Accordingly, we linearized the non-linear position-based program suggested by Schulze and Zimmermann (2017). Preliminary tests showed that the corresponding mixed-integer linear program, because of a large number of decision variables, cannot be generated for even small problem instances using a MILP solver within a reasonable amount of time. Seifi et al. (2019) propose a sequence-based formulation for a relaxation of the problem at hand. By complementing the formulation with additional decision variables and constraints, the exact processing-sequence of the operations on the machines and the workers can be determined. Although this extended formulation is better than a position-based one, it is still not good enough to be used for real problem instances.

The straightforward way to determine the sequencing of the processing of the operations is to use the traveling salesman problem (TSP) variables. Queyranne and Schulz (1994) investigate five different formulations for a single machine scheduling problem from a polyhedral point of view and suggest the use of the TSP-variables, especially for the problems with changeover or setup times. Our preliminary tests showed that a formulation based on TSP-variables outperforms the other kinds of formulations.

For better clarity, we divide our mathematical formulation into seven parts. Note that the parts 1 to 5 of the proposed formulation can be used to formulate any job or flow shop scheduling problem with worker constraints.

1. assigning one worker and one machine to an operation, and determining the processing-sequence of the operations on each machine and each worker;
2. determining the setup, changeover, and removal times;
3. considering the specific processing time and duration of each operation with respect to MR (1);
4. observing the precedence relations between the operations;
5. determining the processed part of each operation during the work shift;
6. formulating the mining-specific requirements MR (2) to MR (6); and
7. providing the objective function.

Notably, in part 1, it can be customized if a subset or the set of available jobs must be processed. The problem-specific setup times can be adjusted in part 2 of the formulation. In part 3, the processing times of operations can be adapted regarding the fact if the breaks of workers are considered or not. Moreover, in part 4, problem-specific precedence relations between the operations can be considered. Finally, the planning horizon (if it is a work shift or not) and the interruptibility of the operations at the end of a work shift (if we consider a work shift) can be easily tailored in part 5.

In Table 2, we first introduce the sets and the parameters which are used in our program and are known in advance.

1. Assigning one worker and one machine to an operation, and determining the processing-sequence of the operations on each machine and each worker

For all $O_{kj} \in \Omega$, we introduce binary decision variables B_{kj} and X_{kjwm}. B_{kj} is 1 if the processing of O_{kj} is started within the work shift, and X_{kjwm} is 1 if machine $m \in M_k$ and worker $w \in W_k$ process operation O_{kj}. Since any operation is processed by exactly one

Table 2
The sets and parameters used in MILP.

Sets	
J	Set of jobs
K	Set of production stages (mining operations)
M	Set of machines
M_k	Set of machines in stage $k \in K$
Ω	Set of operations O_{kj}, $k \in K$, $j \in J$
W	Set of workers
W_k	Set of workers who can be assigned to stage $k \in K$
Parameters	
χ	Duration of a work shift in minutes
d_{0jm}	Driving time of machine $m \in M$ from the parking location at the beginning of the work shift to job $j \in J$ in minutes
d_{jrm}	Driving time of machine $m \in M$ from job $j \in J$ to job $r \in J$ $(j \neq r)$ in minutes
δ	Duration of the break of workers in minutes
φ^{α}	Earliest start time of workers' break
φ^{ω}	Latest start time of workers' break
pt_{kjwm}	Processing time of operation $O_{kj} \in \Omega$, if it is processed by worker $w \in W_k$ and machine $m \in M_k$ in minutes
σ_{kj}	Binary information, 1 if operation $O_{kj} \in \Omega$ was interrupted at the end of the last shift; 0 otherwise
t^{co}_{jr}	Changeover time when a worker changes his machine and goes from job $j \in J$ to job $r \in J$ $(j \neq r)$ in minutes
td^{α}_{m}	Time for the first technical services that must be done for machine $m \in M$ in minutes
td^{ω}_{m}	Time for the last technical services that must be done for machine $m \in M$ in minutes
ton_{kj}	Estimated output of operation $O_{kj} \in \Omega$ in tonnes
ton^{e}_{k}	Given target value of output for stage (mining operation) $k \in K$ in tonnes
u_j	Underground location of Job $j \in J$

worker and one machine but has not to be necessarily processed within a work shift, the following constraint set must be considered:

$$\sum_{w \in W_k} \sum_{m \in M_k} X_{kjwm} = B_{kj} \quad j \in J; k \in K : O_{kj} \in \Omega \tag{1}$$

To determine the processing-sequence of operations on a worker and a machine respectively, we use, as mentioned, so-called TSP-variables. In this regard, we introduce a dummy operation $O_{00} \in \Omega$ (and accordingly $J^0 := J \cup \{0\}$, $K^0 := K \cup \{0\}$) that can be considered as the start city in TSP. Like the cities in TSP, each operation, that is processed within the work shift, has exactly one predecessor operation and exactly one successor operation on the assigned worker and the assigned machine. O_{00} is the first and the last operation processed by each worker and each machine. Consequently, there is a route for each worker (each machine), beginning with O_{00}, that visits all the operations processed by this worker (this machine) and returns to operation O_{00}. Let binary decision variable $Y^{W}_{kj,k'r,w}$ be 1 if operation O_{kj} is processed directly before operation $O_{k'r}$ by worker w. Constraint sets (2) and (3) appoint the sequencing of the operations processed by worker w.

$$\sum_{r \in J^0} \sum_{\substack{k' \in K^0: \\ O_{k'r} \in \Omega \wedge \\ O_{k'r} \neq O_{kj} \wedge w \in W_{k'}}} Y^{W}_{k'r,kj,w} = \sum_{m \in M_k} X_{kjwm} \quad j \in J; k \in K : O_{kj} \in \Omega; w \in W_k \tag{2}$$

$$\sum_{r \in J^0} \sum_{\substack{k' \in K^0: \\ O_{k'r} \in \Omega \wedge \\ O_{k'r} \neq O_{kj} \wedge w \in W_{k'}}} Y^{W}_{kj,k'r,w} = \sum_{m \in M_k} X_{kjwm} \quad j \in J; k \in K : O_{kj} \in \Omega; w \in W_k \tag{3}$$

Constraint sets (4) and (5) guarantee that O_{00} can be the predecessor and the successor of maximum one operation for each worker, respectively.

$$\sum_{j \in J} \sum_{\substack{k \in K: \\ O_{kj} \in \Omega \wedge \\ w \in W_k}} Y^{W}_{00,kj,w} \leq 1 \quad w \in W \tag{4}$$

$$\sum_{j \in J} \sum_{\substack{k \in K: \\ O_{kj} \in \Omega \wedge \\ w \in W_k}} Y^{W}_{kj,00,w} \leq 1 \quad w \in W \tag{5}$$

Since we do not process all eligible operations within the work shift, it is necessary to make sure that $Y^{W}_{kj,k'r,w}$ for $k, k' \in K$ and $j, r \in J$ $(O_{kj} \neq O_{k'r})$ can only take the value of 1 if both operations O_{kj} and $O_{k'r}$ are processed.

$$2Y^{W}_{kj,k'r,w} \leq \sum_{m \in M_k} X_{kjwm} + \sum_{m \in M_{k'}} X_{k'rwm}$$

$$j, r \in J; k, k' \in K : O_{kj}, O_{k'r} \in \Omega : O_{kj} \neq O_{k'r}; w \in W_k \cap W_{k'} \tag{6}$$

Analogously, we introduce the binary decision variables $Y^{M}_{kj,k'r,m}$. With the aid of the following sets of constraints, the sequencing of the operations processed by a specific machine is determined:

$$\sum_{r \in J^0} \sum_{\substack{k' \in K^0: \\ O_{k'r} \in \Omega \wedge \\ O_{k'r} \neq O_{kj} \wedge m \in M_{k'}}} Y^{M}_{k'r,kj,m} = \sum_{w \in W_k} X_{kjwm} \quad j \in J; k \in K : O_{kj} \in \Omega; m \in M_k \tag{7}$$

$$\sum_{r \in J^0} \sum_{\substack{k' \in K^0: \\ O_{k'r} \in \Omega \wedge \\ O_{k'r} \neq O_{kj} \wedge m \in M_{k'}}} Y^{M}_{kj,k'r,m} = \sum_{w \in W_k} X_{kjwm} \quad j \in J; k \in K : O_{kj} \in \Omega; m \in M_k \tag{8}$$

$$\sum_{j \in J} \sum_{\substack{k \in K: \\ O_{kj} \in \Omega \wedge \\ m \in M_k}} Y^{M}_{00,kj,m} \leq 1 \quad m \in M \tag{9}$$

$$\sum_{j \in J} \sum_{\substack{k \in K: \\ O_{kj} \in \Omega \wedge \\ m \in M_k}} Y^{M}_{kj,00,m} \leq 1 \quad m \in M \tag{10}$$

$$2Y^{M}_{kj,k'r,m} \leq \sum_{w \in W_k} X_{kjwm} + \sum_{w \in W_{k'}} X_{k'rwm}$$

$$j, r \in J; k, k' \in K : O_{kj}, O_{k'r} \in \Omega : O_{kj} \neq O_{k'r}; m \in M_k \cap M_{k'} \tag{11}$$

Note that the subtour elimination is realized if we consider a precedence relation between the operations that are processed by the same worker or by the same machine (see constraint sets (26) and (27)).

2. Determining the setup, changeover, and removal times

We classify two groups of activities that must be done for any operation: (1) *preceding activities* that are done before the start of the processing of an operation; and (2) *succeeding activities* that are performed after the completion of the processing of an operation. According to that classification, we can associate the first technical

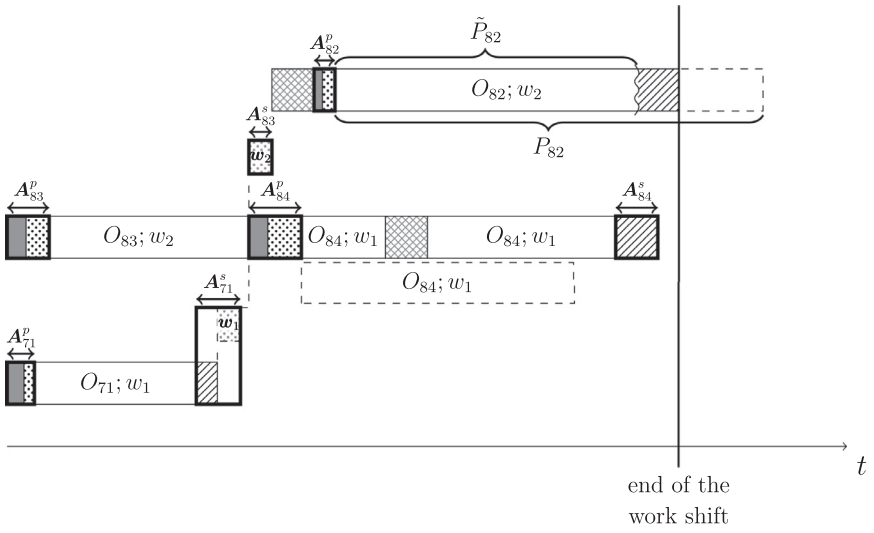

Fig. 4. Preceding and succeeding activities of an operation (cf. Fig. 3).

services and the driving times with the preceding activities. On the other hand, possible changeover and removal times belong to the succeeding activities. Let A^p_{kj} and A^s_{kj} be the duration of the preceding and succeeding activities that must be performed for operation O_{kj}, respectively. Fig. 4 illustrates these activities for the feasible schedule illustrated in Fig. 3.

Both preceding and succeeding activities for operation O_{kj} are performed by worker w, who processes O_{kj}. Utilizing that classification, we can easily formulate the precedence relations between the operations that use the same resource if we know the values of A^p_{kj} and A^s_{kj}.

The value of the non-negative decision variable A^p_{kj} for operation O_{kj} that is processed by worker w and machine m depends on the predecessor operation on w, and m. Let Θ be a constant number with a sufficiently large positive value. For A^p_{kj}, we formulate the following constraints:

$$\sum_{m\in M_k}(d_{0jm}+td^{\alpha}_{m})\cdot Y^M_{00,kj,m}\le A^p_{kj}+\Theta(1-B_{kj})\quad j\in J;k\in K:O_{kj}\in\Omega \tag{12}$$

$$\sum_{r\in J}\sum_{\substack{k'\in K:\\ O_{k'r}\in\Omega:\\ O_{k'r}\neq O_{kj}}}\sum_{m\in M_k\cap M_{k'}} d_{rjm}\cdot Y^M_{k'r,kj,m}\le A^p_{kj}+\Theta(1-B_{kj})\quad j\in J;k\in K:O_{kj}\in\Omega \tag{13}$$

$$\sum_{m\in M_k\cap M_{k'}} d_{rjm}\cdot Y^M_{k'r,kj,m}+\sum_{m\in M_k} td^{\alpha}_{m}\cdot X_{kjwm}\le A^p_{kj}+\Theta\left(1-\sum_{m\in M_k\cap M_{k'}} Y^M_{k'r,kj,m}\right)$$
$$+\Theta\left(\sum_{m\in M_{k'}} X_{k'rwm}-\sum_{m\in M_k} X_{kjwm}+1\right)+\Theta(2-B_{kj}-B_{k'r})$$
$$j,r\in J;k,k'\in K:O_{kj},O_{k'r}\in\Omega:O_{kj}\neq O_{k'r};w\in W_k \tag{14}$$

Constraint set (12) makes sure that at the beginning of the work shift, the first technical services and the driving time from the parking location of machine m to job j are considered. In this case, O_{kj} is the first operation assigned to machine m (see A^p_{71}, A^p_{83}, and A^p_{82} in Fig. 4). If $O_{k'r}$ is processed immediately before O_{kj} by machine m, constraint set (13) guarantees the consideration of d_{rjm}. Constraint set (14) ensures that the first technical services for O_{kj} must be additionally taken into account if $O_{k'r}$ is directly scheduled before O_{kj} on the same machine; still O_{kj} is processed by worker w, and $O_{k'r}$ by a different worker (see A^p_{84} in Fig. 4).

For A^s_{kj}, constraint set (15) considers the last technical services for O_{kj} if it is the last operation assigned to machine m (see A^s_{84} in Fig. 4), and constraint set (16) guarantees the consideration of the changeover times if worker w, who processed O_{kj}, performs directly another operation on a different machine (see A^s_{71} and A^s_{83} in Fig. 4).

$$\sum_{m\in M_k} td^{\omega}_{m}\cdot Y^M_{kj,00,m}\le A^s_{kj}+\Theta(1-B_{kj})\quad j\in J;k\in K:O_{kj}\in\Omega \tag{15}$$

$$\sum_{m\in M_k} td^{\omega}_{m}\cdot Y^M_{kj,00,m}+\sum_{w\in W_k} t^{co}_{jr}\cdot Y^W_{kj,k'r,w}\le A^s_{kj}+\Theta\left(\sum_{m\in M_k\cap M_{k'}} Y^M_{kj,k'r,m}\right)$$
$$+\Theta(2-B_{kj}-B_{k'r})\,j,r\in J;k,k'\in K:O_{kj},O_{k'r}\in\Omega:O_{kj}\neq O_{k'r} \tag{16}$$

3. Considering the specific processing time and duration of each operation with respect to MR (1)

Let S_{kj} be the non-negative decision variable to denote the start time of operation O_{kj}. We indicate the actual processing time of O_{kj} after assigning a particular worker and a specific machine with P_{kj} that can be calculated as follows[4]:

$$P_{kj}:=\sum_{w\in W_k}\sum_{m\in M_k} pt_{kjwm}\cdot X_{kjwm}$$

It must be noted that the duration of the processing of an operation, and therefore its completion time may be extended because of the break of the worker who processes that operation (see O_{84} in Fig. 4). Hence, to determine the right completion time of any operation, mining requirement MR (1) must be taken into account (cf. Section 2).

MR (1) Each worker has to take a $\delta-$minute break. The start time of the break must lie in a given interval $[\varphi^{\alpha},\varphi^{\omega}]$. We define positive decision variable Φ_w to denote the start time of the break of worker w.

$$\varphi^{\alpha}\le\Phi_w\le\varphi^{\omega}\quad w\in W \tag{17}$$

Additionally, we introduce binary decision variables Γ^s_{kj} and Γ^e_{kj} to determine if the break of a worker starts during the processing of O_{kj}.

$$\Gamma^s_{kj}=\begin{cases}1, & \text{if the break of worker } w\in W_k, \text{ who processes } O_{kj},\\ & \text{is not taken before } S_{kj}-A^p_{kj};\\ 0, & \text{otherwise.}\end{cases}$$

[4] Note that P_{kj} are decision variables that have not to be additionally introduced in the mathematical model. We use P_{kj} instead of the explicit summation for better clarity.

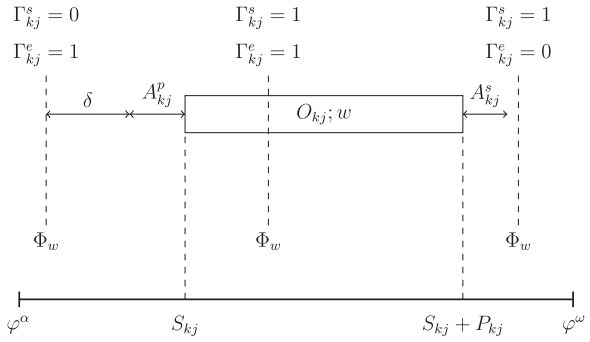

Fig. 5. Values of Γ^s_{kj} and Γ^e_{kj} depending on Φ_w.

$$\Gamma^e_{kj}=\begin{cases}1, & \text{if the start time of the break of worker } w\in W_k,\\ & \text{who processes } O_{kj}, \text{ is before } S_{kj}+P_{kj};\\ 0, & \text{otherwise.}\end{cases}$$

Fig. 5 illustrates the dependence of the values taken by Γ^s_{kj} and Γ^e_{kj} on the start time of the break of worker w, who processes O_{kj}. Both variables Γ^s_{kj} and Γ^e_{kj} must take the value of 1 if and only if Φ_w lies during the processing of O_{kj}. For this purpose, we formulate the following constraint sets:

$$\Phi_w+\delta\le S_{kj}-A^p_{kj}+\Theta\cdot\Gamma^s_{kj}+\Theta\left(1-\sum_{m\in M_k}X_{kjwm}\right)$$
$$j\in J;k\in K:O_{kj}\in\Omega;w\in W_k \tag{18}$$

$$S_{kj}\le\Phi_w+\Theta(1-\Gamma^s_{kj})+\Theta\left(1-\sum_{m\in M_k}X_{kjwm}\right)$$
$$j\in J;k\in K:O_{kj}\in\Omega;w\in W_k \tag{19}$$

$$\Phi_w\le S_{kj}+P_{kj}+\Theta(1-\Gamma^e_{kj})+\Theta\left(1-\sum_{m\in M_k}X_{kjwm}\right)$$
$$j\in J;k\in K:O_{kj}\in\Omega;w\in W_k \tag{20}$$

$$S_{kj}+P_{kj}+A^s_{kj}\le\Phi_w+\Theta\cdot\Gamma^e_{kj}+\Theta\left(1-\sum_{m\in M_k}X_{kjwm}\right)$$
$$j\in J;k\in K:O_{kj}\in\Omega;w\in W_k \tag{21}$$

Note that constraint sets (19) to (21) additionally guarantee that the break of a worker cannot be taken during the preceding and succeeding activities of an operation. The following constraint sets ensure that binary decision variables Δ_{kj} take the value of 1 if and only if $\Gamma^s_{kj}=1$ and $\Gamma^e_{kj}=1$:

$$\Gamma^s_{kj}+\Gamma^e_{kj}\le 1+\Delta_{kj}\quad j\in J;k\in K:O_{kj}\in\Omega \tag{22}$$

$$2\Delta_{kj}\le\Gamma^s_{kj}+\Gamma^e_{kj}\quad j\in J;k\in K:O_{kj}\in\Omega \tag{23}$$

Thus, the right duration of an operation is equal to $P_{kj}+\delta\cdot\Delta_{kj}$, and accordingly, the completion time of O_{kj} can be determined as follows[5]:

$$C_{kj}:=S_{kj}+P_{kj}+\delta\cdot\Delta_{kj}$$

[5] Note that C_{kj} are decision variables that have not to be introduced additionally and are used for better clarity.

4. Observing the precedence relations between the operations

As mentioned, the production environment can be classified as an HFS scheduling problem. Constraint sets (24) and (25) guarantee that a job is processed in the chronological order in the stages.

$$B_{k'j}\le B_{kj}\quad j\in J;k,k'\in K:k<k'\wedge O_{kj},O_{k'j}\in\Omega \tag{24}$$

$$C_{kj}\le S_{k'j}+\Theta(2-B_{kj}-B_{k'j})\quad j\in J;k,k'\in K:k<k'\wedge O_{kj},O_{k'j}\in\Omega \tag{25}$$

Constraint sets (26) and (27) ensure a precedence relation between the operations that are processed by the same worker and by the same machine, respectively.

$$C_{kj}+A^s_{kj}\le S_{k'r}-A^p_{k'r}+\Theta\left(1-\sum_{w\in W_k\cap W_{k'}}Y^W_{kj,k'r,w}\right)$$
$$j,r\in J;k,k'\in K:O_{kj},O_{k'r}\in\Omega:O_{kj}\ne O_{k'r} \tag{26}$$

$$C_{kj}\le S_{k'r}-A^p_{k'r}+\Theta\left(1-\sum_{m\in M_k\cap M_{k'}}Y^M_{kj,k'r,m}\right)$$
$$j,r\in J;k,k'\in K:O_{kj},O_{k'r}\in\Omega:O_{kj}\ne O_{k'r} \tag{27}$$

Note that if $Y^M_{kj,k'r,m}=1$, we must not consider A^s_{kj} in the precedence relation between O_{kj} and $O_{k'r}$ on machine m. A^s_{kj} takes a value greater than zero if: (I) O_{kj} is the last operation assigned to the machine; or (II) worker w, who processed O_{kj}, has changed his machine. In case (I), $Y^M_{kj,k'r,m}$ is zero (constraint set (27) is satisfied), and in case (II), A^s_{kj} leads to a delay for the operation that will be consequently processed by worker w (not for the successor operation on machine m) (see A^s_{83} in Fig. 4, where O_{83} and O_{84} are processed by the same machine).

5. Determining the processed part of each operation during the work shift

As explained, the operations can be interrupted at the end of the work shift. In this regard, we define binary decision variables I_{kj} as follows:

$$I_{kj}=\begin{cases}1, & \text{if } C_{kj} \text{ plus the removal time needed for the assigned}\\ & \text{machine is after the end of the work shift;}\\ 0, & \text{otherwise.}\end{cases}$$

Obviously, I_{kj} is 0 if operation O_{kj} is not processed during the work shift:

$$I_{kj}\le B_{kj}\quad j\in J;k\in K:O_{kj}\in\Omega \tag{28}$$

With the aid of the following constraint sets, decision variables I_{kj} take the right values:

$$C_{kj}+\sum_{m\in M_k}td^\omega_m\cdot Y^M_{kj,00,m}\le\chi+\Theta\cdot I_{kj}\quad j\in J;k\in K:O_{kj}\in\Omega \tag{29}$$

$$\chi\le C_{kj}+\sum_{m\in M_k}td^\omega_m\cdot Y^M_{kj,00,m}+\Theta(1-I_{kj})\quad j\in J;k\in K:O_{kj}\in\Omega \tag{30}$$

Assume machine m processes O_{kj}. If I_{kj} is 1, $C_{kj}(=S_{kj}+P_{kj}+\delta\cdot\Delta_{kj})+td^\omega_m$ lies after the end of the work shift. In this case, we have to determine the percentage of O_{kj} that is processed during the work shift. Let $\widetilde{P}_{kj}$ be the executed part of O_{kj} (see O_{82} in Fig. 4). Then, $\widetilde{P}_{kj}=\rho\cdot P_{kj}$ with $\rho\in[0,1]$, where ρ can be calculated as follows:

$$\rho=\frac{\chi-(S_{kj}+\delta\cdot\Delta_{kj}+td^\omega_m)}{P_{kj}}$$

Remember that P_{kj} has the value of pt_{kjwm} if worker w and machine m are assigned to O_{kj}. If we write $\sum_{w\in W_k}\sum_{m\in M_k} pt_{kjwm}\cdot X_{kjwm}$ instead of P_{kj} and reformulate the above fraction, we have the following equation:

$$\sum_{w\in W_k}\sum_{m\in M_k} pt_{kjwm}\cdot\rho\cdot X_{kjwm}=\chi-(S_{kj}+\delta\cdot\Delta_{kj}+td_m^{\omega})$$

We substitute the product $\rho\cdot X_{kjwm}$ by the non-negative continuous decision variable G_{kjwm} that takes a value greater than 0 if and only if X_{kjwm} is 1. The percentage of O_{kj} that is processed during the work shift is $G_{kjwm}\cdot 100\%$. Through the following constraint sets, the value of G_{kjwm} will be correctly appointed:

$$G_{kjwm}\le X_{kjwm}\quad j\in J;k\in K:O_{kj}\in\Omega;w\in W_k;m\in M_k \tag{31}$$

$$S_{kj}+\sum_{w\in W_k}\sum_{m\in M_k} pt_{kjwm}\cdot G_{kjwm}+\delta\cdot\Delta_{kj}+\sum_{m\in M_k} td_m^{\omega}\cdot Y^M_{kj,00,m}\le\chi+\Theta(1-I_{kj})\quad j\in J;k\in K:O_{kj}\in\Omega \tag{32}$$

Clearly, we can maximally operate 100 percent of any operation ($G_{kjwm}\le 1$). Moreover, if $B_{kj}=1$ and $I_{kj}=0$, the excavated percentage of O_{kj} must be 100 ($G_{kjwm}=1$).

$$\sum_{w\in W_k}\sum_{m\in M_k} G_{kjwm}\le B_{kj}\quad j\in J;k\in K:O_{kj}\in\Omega \tag{33}$$

$$\sum_{w\in W_k}\sum_{m\in M_k} G_{kjwm}\le B_{kj}+I_{kj}\quad j\in J;k\in K:O_{kj}\in\Omega \tag{34}$$

$$B_{kj}-I_{kj}\le\sum_{w\in W_k}\sum_{m\in M_k} G_{kjwm}\quad j\in J;k\in K:O_{kj}\in\Omega \tag{35}$$

6. Formulating the mining-specific requirements MR (2) to MR (6)

Let $V_{kj,k'r}$ be a binary decision variable that is 1 if O_{kj} is completed before $O_{k'r}$ is started. Moreover, Z_{wm} is a binary decision variable that is 1 if worker w is assigned at least one time to machine m to process any operation. For mining-specific requirements MR (2)–(6) (cf. Section 2), we formulate the following self-explanatory constraint sets:

MR (2)

$$C_{kj}\le S_{k'r}+\Theta(1-V_{kj,k'r})+\Theta(2-B_{kj}-B_{k'r})\quad j,r\in J:j\ne r\wedge u_j=u_r;k,k'\in K:O_{kj},O_{k'r}\in\Omega \tag{36}$$

$$C_{k'r}\le S_{kj}+\Theta\cdot V_{kj,k'r}+\Theta(2-B_{kj}-B_{k'r})\quad j,r\in J:j\ne r\wedge u_j=u_r;k,k'\in K:O_{kj},O_{k'r}\in\Omega \tag{37}$$

MR (3)

$$B_{k'r}\le B_{kj}\quad j,r\in J:j\ne r\wedge u_j=u_r;k,k'\in K:O_{kj},O_{k'r}\in\Omega\wedge k<k' \tag{38}$$

$$C_{kj}\le S_{k'r}+\Theta(2-B_{kj}-B_{k'r})\quad j,r\in J:j\ne r\wedge u_j=u_r;k,k'\in K:O_{kj},O_{k'r}\in\Omega\wedge k<k' \tag{39}$$

MR (4)

$$k'r\le B_{kj}\quad j,r\in J:j\ne r\wedge u_j=u_r;k,k'\in K:O_{kj},O_{k'r}\in\Omega\wedge\sigma_{kj}=1\wedge\sigma_{k'r}=0 \tag{40}$$

MR (5)

$$I_{kj}=0\quad j\in J;k=5:O_{5j}\in\Omega \tag{41}$$

MR (6)

$$\sum_{m'\in M_k} X_{kjwm'}+\sum_{w'\in W_k} X_{kjw'm}\le 1+Z_{wm}\quad j\in J;k\in K:O_{kj}\in\Omega;w\in W_k;m\in M_k \tag{42}$$

$$\sum_{m\in M} Z_{wm}\le 2\quad w\in W \tag{43}$$

$$\sum_{w\in W} Z_{wm}\le 2\quad m\in M \tag{44}$$

7. Providing the objective function

Our optimization problem aims to minimize the accumulated lower deviations of the output from the predetermined target values (cf. Section 2). We introduce positive continuous decision variables D_k, which take the value of the lower deviation for stage k from a given target value ton_k^e (an amount of material in tonnage that is expected to be completed in stage k within the work shift).

$$ton_k^e-\sum_{\substack{j\in J:\\ O_{kj}\in\Omega}}\sum_{w\in W_k}\sum_{m\in M_k} ton_{kj}\cdot G_{kjwm}\le D_k\quad k\in K \tag{45}$$

We only take the lower deviations into account. If the output for a stage exceeds the target value, we do not consider the difference ($D_k=0$).

From an operational point of view, we want to have consistent progress, i.e., if there are some lower deviations, they should be distributed as evenly as possible between the processing stages. For this purpose, a quadratic objective function would be appropriate.

$$\sum_{k\in K} D_k^2 \tag{QOF}$$

Preliminary tests on small instances showed that the optimal solution is the same, regardless of having a quadratic or a linear objective function. That is maybe the case, because the number of available machines, in comparison to the available operations, is quite small. Hence, we use a linear objective function to approximate the quadratic one that allows finding better solutions within a given time limit. We introduce positive continuous decision variable $D^{\max}$ that takes the value of the maximum lower deviation from a target value.

$$D_k\le D^{\max}\quad k\in K \tag{46}$$

Our modified linear objective function can then be formulated as follows:

$$\sum_{k\in K} D_k+D^{\max} \tag{MLOF}$$

We denote the minimization problem using (MLOF) as the objective function and subject to constraints (1)–(46) as **MILP**. In our MILP, the number of integer decision variables is of the order of $|J^0|^2\cdot|K^0|^2\cdot\max\left\{\max_{k\in K}\{|W_k|\},\max_{k\in K}\{|M_k|\}\right\}$ (cf., e.g., $Y^W_{kj,k'r,w}$). On the other hand, the number of constraints is of the order of $|J|^2\cdot|K|^2\cdot\max\left\{\max_{k\in K}\{|W_k|\},\max_{k\in K}\{|M_k|\}\right\}$ (cf., e.g., constraint sets (6) and (11)). Finally, the number of continuous decision variables is of the order of $|J|\cdot|K|\cdot\max_{k\in K}\{|W_k|\}\cdot\max_{k\in K}\{|M_k|\}$ (cf. G_{kjwm}). In Appendix A, we indicate the values of the decision variables of our MILP model for the feasible solution illustrated in Fig. 3 (given in Section 2).

In Section 4, we use some realistic problem instances to compare the results achieved by the presented MILP to the results provided using the heuristic approach introduced by Schulze and Zimmermann (2017) and the two-stage procedure developed by Seifi et al. (2019).

4. Computational study

In Section 4.1, we first present the essential data to generate the problem instances and give a brief overview of the constructive heuristic introduced by Schulze and Zimmermann and of the two-stage procedure suggested by Seifi et al. After that, we describe the performance analysis and compare the solution approaches based on the values of $\sum_{k\in K} D_k^2$. In Subsect. 4.2, we divide the generated test instances according to the number of operations and the number of underground locations into 4 groups. Then, we compare the results achieved by our MILP and by the two-stage procedure considering the values of $\sum_{k\in K} D_k + D^{\max}$ (MLOF).

4.1. Generation of the test instances and the primary comparison of the results

As mentioned before, machines and workers cannot be exchanged between different mining districts during a work shift. Thus, we consider only one mining district in our shift scheduling problem. To generate realistic problem instances, we used data from practice that characterize a typical mining district that generally has 4–6 tipple areas. Typically, a mining district has a prescribed minimum and maximum number of underground locations and working places. In our setting, there are 14–35 underground locations as well as 28–46 working places in the mining district. Consequently, we can assign 3–6 underground locations to each tipple area and 1–3 working places to each underground location. The allocation of underground locations (and their corresponding working places) to different tipple areas contributes only to determining the distances between the jobs. However, the associated underground location for each working place (job), u_j, is used in the mathematical program to guarantee some mining-specific requirements for different jobs in the same underground location (cf. Section 2).

At the beginning of the work shift, the processing of a job can begin on any production stage $k \in K$. Accordingly, each job must be processed in stages k to $|K|$. Note that the jobs (working places) that start with operations (1) and (2) appear more often.[6] Moreover, not all of the jobs must be processed in production stages (4), (5), and (6). The installation of anchor bolts (stage (4)) is not necessarily needed for each working place (job). Also, drilling of large diameter boreholes (stage (5)), and consequently, removing the resultant dust (stage (6)) must not be performed if the salt face has a certain shape. Thus, operations (4), (5), and (6) are neglected for jobs with a certain probability in our problem instances. Note that each underground location includes only the operations of jobs, which can be theoretically processed during the work shift if the preceding operations are processed by the most skillful worker and by the speediest machine.

The number of available identical/uniform machines for each production stage is given in Table 3.

As mentioned, it is expected that the processing of a job in a production stage provides a specific amount of crude material. This amount of material is a random number from 1 to 20, which is multiplied by 50 in tonnes. It must be taken into account that the expected amount of material for all operations of a job is the same.

In the mining district under consideration, there are 6–15 workers available during one work shift. Each worker can operate at least one and at most all of the available machines. The processing times of the operations of jobs are first determined based on machine speeds. The results are then divided by the skill levels of workers who can deal with those machines. If a worker can handle a machine, his skill level on that machine is an element of the set {0.7, 0.8, 0.9, 1}, which is determined randomly.

Table 3
Number of available machines in each production stage.

Production stage	Number of machines
(1)	2–6
(2)	1–3
(3) and (6)	1–2
(4)	1–2
(5)	1–2
(7)	1–2
(8)	1–2

All other parameters from Table 2 in Section 3, e.g., the target values of output for production stages, driving times between the jobs, etc., are chosen based on the case study presented in Schulze and Zimmermann (2017). We made the generated problem instances available online.[7]

The constructive heuristic procedure proposed by Schulze and Zimmermann (2017) is embedded in a multi-start algorithm. Using the multi-start algorithm, priority values are used to determine selection probabilities for jobs, machines, and workers. Schulze and Zimmermann studied several assigning methods regarding the way that jobs, machines, and workers are chosen. In this paper, we apply the combination that is currently used in the underground mine under consideration. In the corresponding assigning method, jobs and workers are randomly chosen, and machines are selected either 1. regarding the shortest driving times to the selected job; or 2. regarding the shortest processing time based on the chosen job and worker (for more detail, see Schulze and Zimmermann (2017) and Schulze (2016)). In what follows, we denote the applied constructive heuristic with **CH**.

Seifi et al. (2019) propose a two-stage approach to tackle the problem. They suggest a sequence-based formulation for a relaxation of the problem. In the relaxed program, if a worker processes two consecutive operations on different machines, the succeeding activities for the first operation and the preceding activities for the second operation are not considered. Moreover, the breaks of workers may overlap the preceding or the succeeding activities of the operations processed by the associated workers. In the first stage, the relaxed program is solved by a MILP solver, and in the second stage, the solution achieved is fixed by inserting the neglected time intervals (for more detail, see Seifi et al. (2019)). In the following, we denote the two-stage approach with **2SA**.

We generate feasible solutions for 100 realistic problem instances using MILP, 2SA, and CH. All tests are executed on an Intel i7-7700K@4.20GHz machine with 64 GB RAM under Windows 10. CH is implemented in Xpress IVE 8.8. For the first stage of 2SA (relaxed program) and MILP, we used GAMS 31.2 and GUROBI solver 9.0.2. The second stage of 2SA is implemented in Visual Studio 2017 by programming language C++. For all three solution approaches, we set an upper time limit of 3600 seconds. CH is embedded in a multi-start algorithm, where a roulette-wheel selection is applied to choose the jobs, workers, and machines according to their probability values. Hence, different solutions can be generated within 3600 seconds for each single problem instance. CH finds the best solution within, on average, 343.4 seconds. For 2SA, we solve the relaxation of the problem using the GUROBI solver (first stage) with an upper time limit of 3600 seconds. The time for generating a feasible solution using the solution found in the first stage is negligible. In the first stage of 2SA, the GUROBI solver

[6] Because the detonation (mining operation (9)) takes place between the work shifts, and mining operation (1) can last one work shift.

[7] https://www.wiwi.tu-clausthal.de/abteilungen/unternehmensforschung/forschung/benchmark-instances/

Table 4
Comparison of MILP to 2SA and CH.

	MILP	2SA	MILP	CH
#best solutions found	69	31	84	16
Gap^1	7.5%	24.4%	3.4%	97.8%
Gap^2	24.3%	35.3%	21.4%	116.4%

can find the best feasible solution within, on average, 1893.16 seconds. Our MILP cannot prove the optimality of any of the test instances within 3600 seconds, and the gap of the results achieved by MILP to the lower bound is, on average, 71.15%. That is probably the case since lower bounds of 23 test instances are 0, and the corresponding gap is 100%. Moreover, the GUROBI solver in our MILP needs, on average, 2557.22 seconds to find the best feasible solution, which probably means that the lower bounds cannot be improved to prove the optimality of the feasible solutions found. However, the contribution of this paper is to introduce a mixed-integer linear program for the problem described in Section 2 that can find good feasible solutions for realistic problem instances. Clearly, CH is the quickest approach to find a feasible solution.

As explained in Section 3, the value of $\sum_{k\in K} D_k^2$ shows how consistent the desired progress could be implemented at the end of the work shift compared to the given state at the beginning of the work shift. We argued that using a modified linear objective function (MLOF) for our MILP is usually sufficient to provide a good solution even for the quadratic objective function. To gauge the efficiency of our claim, we take the value of $\sum_{k\in K} D_k^2$, in this subsection, as a basis to compare the results achieved by MILP, 2SA, and CH. Since CH is directly tailored for $\sum_{k\in K} D_k^2$, a comparison of MILP and 2SA with CH concerning the provided values of (MLOF) is not reasonable.

Seifi et al. showed that 2SA outperforms CH. In this paper, we compare the results achieved by MILP to 2SA and CH, respectively. For problem instance i, we denote the solution found by procedure * with ξ_i^*. Moreover, the best solution found is indicated by ξ_i^{best}. Let $f(\xi_i)$ be the objective value for solution ξ_i. We calculate $\frac{f(\xi_i^*)-f(\xi_i^{\text{best}})}{f(\xi_i^{\text{best}})} \cdot 100\%$ to determine the gap (gap_i^*) of the solution found by procedure * with respect to the best solution found for problem instance i. To evaluate the results achieved by the procedures, we define the parameters Gap^1 and Gap^2. Gap^1 is obtained by arithmetic averaging of the gap values over all of the problem instances. Gap^2 is calculated by arithmetic averaging of the gap values over the set of the problem instances for which the solution found by the considered procedure * is not equal to the best solution found (i.e., $\text{gap}_i^* \neq 0$). Table 4 presents the number of best solutions found using each approach, as well as Gap^1 and Gap^2.

In comparison to 2SA, MILP finds for 69% of the problem instances the best solution. For the case, that both procedures do not find the best solution, the values of the sum of quadratic lower deviations of the solutions found by MILP are, on average, better than those values achieved by 2SA (24.3% vs. 35.3%).

In comparison to CH, MILP finds for 84% of the problem instances the best solution. Moreover, for the other 16 problem instances, MILP finds a solution whose objective value is, on average, 21.4% far from the objective value of the best solution found by CH. This value is 116.4% for CH, which shows that the solutions found by MILP are much better than the solutions achieved by CH.

4.2. Comparison of MILP and 2SA in more detail

MILP and 2SA can be investigated concerning the values of (MLOF) since 2SA uses (MLOF) as the objective function in the first stage. According to the values of (MLOF), MILP can find for 81% of the instances a better solution. The value of Gap^1 for the solution achieved by MILP is, on average, 10.2%, where that value for 2SA is equal to 21.9%. To find out whether there is a correlation between the number of operations or the number of underground locations in a problem instance and the quality of the solutions achieved by MILP and 2SA, we analyze the problem instances according to those numbers. The box plots in Figs. 6 and 7 illustrate the distribution of the number of available operations of all jobs and the number of underground locations in our 100 generated problem instances, respectively.

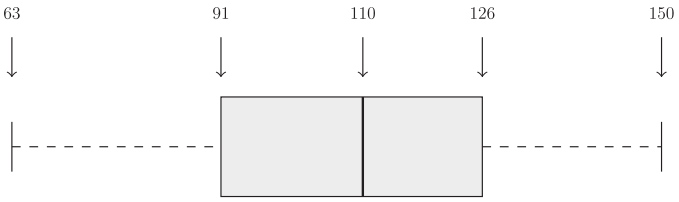

Fig. 6. Depicting of available operations of all jobs through their quartiles.

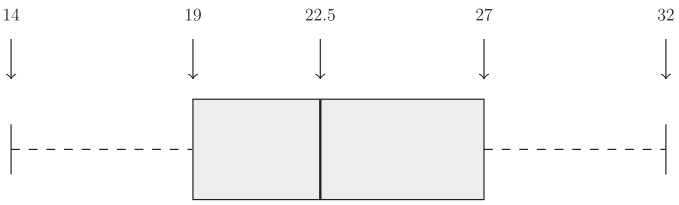

Fig. 7. Depicting of underground locations through their quartiles.

There are 63 to 150 operations of jobs with a median of 110 as well as 16 to 32 underground locations with a median of 22.5. Let Q_1, Q_2, and Q_3 be the first, second, and third quartile, respectively. By definition, Q_1 is the middle number between the smallest number (min) and the median (for operations $Q_1 = 91$ and for underground locations $Q_1 = 19$). Q_2 is the median, and Q_3 (for operations 126 and for underground locations 27) is the middle number between the median and the highest value (max). Thus, each problem instance can only belong to one of the four intervals [min, Q_1), [Q_1, Q_2), [Q_2, Q_3), and [Q_3, max] according to the number of operations or the number of underground locations. In Table 5, the number of problem instances in each interval, the percentage of the problem instances for which MILP finds a better solution than 2SA (according to the values of (MLOF)), and Gap^1 values for 2SA and MILP are given.

We see in Table 5 that in all four intervals, the number of best solutions found by MILP is much higher than the number of best solutions obtained by 2SA. The values of Gap^1 also show that the quality of the solutions found by MILP in all of the four intervals is much better than the quality of the solutions found by 2SA. Overall, the results show that a pattern cannot be recognized, and there is no particular correlation between the number of operations or the number of underground locations in a problem instance and the performance of the considered approaches.

Finally, we calculate $\frac{f(\xi_i^{2SA})-f(\xi_i^{MILP})}{f(\xi_i^{MILP})} \cdot 100\%$ for all of the problem instances. The arithmetic average of the fraction above over 100 problem instances is equal to 16.1%. It shows that the objective values of the results found by MILP are, on average, distinctly better than the objective values of the results obtained by 2SA. Hence, we can conclude that MILP significantly outperforms both existing procedures if we solve problem instances that are typical for potash underground mines.

Table 5
Analysis of the results achieved by MILP in comparison to 2SA based on the number of operations and the number of underground locations in problem instances.

According to the number of operations	[min, Q_1)	[Q_1, Q_2)	[Q_2, Q_3)	[Q_3, max]
#problem instances	24	25	23	28
MILP < 2SA	95.8%	72%	73.9%	82.1%
Gap[1] for 2SA	14.8%	21.9%	29.5%	23.6%
Gap[1] for MILP	5.7%	9.9%	11.3%	10.0%
According to the number of underground locations	**[min, Q_1)**	**[Q_1, Q_2)**	**[Q_2, Q_3)**	**[Q_3, max]**
#problem instances	22	24	27	27
MILP < 2SA	81.8%	75%	85.2%	81.4%
Gap[1] for 2SA	14.1%	22.1%	24.6%	25.5%
Gap[1] for MILP	7.1%	13.7%	11.5%	7.1%

5. Conclusions

In this paper, we consider a shift scheduling problem with a simultaneous assignment of machines and workers in a German potash mine. In the first step, we classify our shift scheduling problem as a variant of a hybrid flow shop scheduling problem where we have to

- decide which jobs (operations) are performed in the current work shift;
- consider sequence-dependent setup, changeover, and removal times;
- assign an appropriate machine and worker to each operation and thus appoint the processing time of the job;
- observe workers' breaks.

Next, we formulate a mixed-integer linear program using TSP-variables for our shift scheduling problem. In contrast to position- and sequence-based formulations, the proposed program is more compact and more promising from a computational point of view. It should be emphasized that the proposed formulation can be easily adapted for other flow and job shop scheduling problems with worker constraints. The results of a performance analysis based on a set of realistic problem instances show that the solutions of our proposed mixed-integer linear program outperform the solutions which are currently constructed by means of the existing procedures.

Future work concerns the development of good lower bounds for our shift scheduling problem and the investigation of general hybrid flow shop problems with the presented extensions. Moreover, the total output can be considered as an important objective. Assume that target values for the production stages are achieved. Then in an optimal solution, there is no need to schedule the other available operations that can lead to more crude salt. For this reason, a local search procedure would be of interest, where the achieved solutions can be enhanced concerning the total output.

Appendix A. Verification and validation of the MILP

In this section, we indicate the values of the decision variables of our MILP for the feasible solution in Fig. 3. In this way, we show how the constraints of MILP guarantee the feasibility of a solution. We denote the blasthole drilling machine with m_1, the charging vehicle (i) with m_2, and charging vehicle (ii) with m_3. Hence, we have $W_7 = \{w_1, w_2\}$, $W_8 = \{w_1, w_2\}$, $M_7 = \{m_1\}$, and $M_8 = \{m_2, m_3\}$. Constraint set (1) implies that the following decision variables have the value 1:

$$B_{71} = B_{83} = B_{84} = B_{82} = 1,$$
$$X_{71,w_1,m_1} = X_{83,w_2,m_2} = X_{84,w_1,m_2} = X_{82,w_2,m_3} = 1.$$

Constraint sets (2) to (6) guarantee in collaboration with constraint set (26) the following tours for w_1 and w_2:

$$Y^W_{00,71,w_1} = Y^W_{71,84,w_1} = Y^W_{84,00,w_1} = 1,$$
$$Y^W_{00,83,w_2} = Y^W_{83,82,w_2} = Y^W_{82,00,w_2} = 1.$$

Analogously, constraint sets (7) to (11) with constraint set (27) determine the tours for machines m_1, m_2, and m_3:

$$Y^M_{00,71,m_1} = Y^M_{71,00,m_1} = 1,$$
$$Y^M_{00,83,m_2} = Y^M_{83,84,m_2} = Y^M_{84,00,m_2} = 1,$$
$$Y^M_{00,82,m_3} = Y^M_{82,00,m_3} = 1.$$

Constraint set (12) determines the preceding activities of operation O_{kj} according to the fact if O_{kj} is the first operation assigned to a machine. In this example, we have:

$$d_{0,1,m_1} + td^{\alpha}_{m_1} \leq A^p_{71},$$
$$d_{0,3,m_2} + td^{\alpha}_{m_2} \leq A^p_{83},$$
$$d_{0,2,m_3} + td^{\alpha}_{m_3} \leq A^p_{82}.$$

Constraint set (13) considers the preceding activities regarding the driving times if operation O_{kj} has a real predecessor on the assigned machine. For our example, it is the case only for operation O_{84}:

$$d_{3,4,m_2} \leq A^p_{84}.$$

For activating constraint set (14), two operations must be processed successively on the same machine and by different workers. Since operation O_{84} is operated directly after operation O_{83} on machine m_2 ($Y^M_{83,84,m_2} = 1$), constraint set (14) is active if $\sum_{m\in M_{k'}} X_{k'rwm} - \sum_{m\in M_k} X_{kjwm} + 1$ is 0. That is the case for O_{83}, O_{84}, and w_1, since $\sum_{m\in M_8} X_{83,w_1,m_2} = 0$ and $\sum_{m\in M_8} X_{84,w_1,m_2} = 1$:

$$d_{3,4,m_2} + td^{\alpha}_{m_2} \leq A^p_{84}.$$

Note that the value of A^p_{84} must satisfy both of the above inequalities. Hence, A^p_{84} has to take the maximum value of the left-hand sides of the inequalities.

Constraint set (15) considers the succeeding activities for the last jobs assigned to a machine. For our example, we have:

$$td^{\omega}_{m_1} \leq A^s_{71},$$
$$td^{\omega}_{m_2} \leq A^s_{84},$$
$$td^{\omega}_{m_3} \leq A^s_{82}.$$

Constraint set (16) is only active if two real operations O_{kj} and $O_{k'r}$ are not processed successively on the same machine ($\sum_{m\in M_k\cap M_{k'}} Y^M_{kj,k'r,m} = 0$). Although constraint set (16) is active for many operations, the left-hand side of the inequality has the value

0, unless O_{kj} is the last task of the assigned machine or if the same worker operates O_{kj} and $O_{k'r}$. In the following, we only list the constraints that have a number greater than 0 on the left-hand side:

$td^{\omega}_{m_1} + t^{co}_{14} \le A^s_{71}$,
$t^{co}_{32} \le A^s_{83}$,
$td^{\omega}_{m_2} \le A^s_{84}$,
$td^{\omega}_{m_3} \le A^s_{82}$.

Analogously, A^s_{71} takes the maximum value of the left-hand sides of the inequalities, i.e., $td^{\omega}_{m_1} + t^{co}_{14}$.

Constraint sets (17) to (23) are explicitly described in Section 3. Note that $\Gamma^s_{84} = 1$ and $\Gamma^e_{84} = 1$, and subsequently, $\Delta_{84} = 1$.

Constraint sets (24) and (25) are not active in our example. For operations O_{71} and O_{84} according to constraint set (26), we have:

$$C_{71} + A^s_{71} \le S_{84} - A^p_{84}.$$

According to constraint set (27) for operations O_{83} and O_{84}, we write:

$$C_{83} \le S_{84} - A^p_{84}.$$

Bear in mind that S_{84} has to take the maximum value of the left-hand sides of the corresponding inequalities above. Besides, A^s_{83} cannot affect the start time of operation O_{84}. For operations O_{83} and O_{82}, according to constraint set (26), the following inequality must apply:

$$C_{83} + A^s_{83} \le S_{82} - A^p_{82}.$$

The reason why O_{82} cannot be started at point $C_{83} + A^s_{83} + A^p_{82}$ in time is that according to constraint set (18), we have:

$$\Phi_{w_2} + \delta \le S_{82} - A^p_{82}.$$

In our example, $\Phi_{w_2} + \delta$ is after $C_{83} + A^s_{83}$. Note that Φ_{w_2} cannot be between $S_{83} + P_{83}$ and $S_{83} + P_{83} + A^s_{83}$ according to constraint set (21).

According to constraint sets (29) and (30), the following values of I_{kj} are correct:

$$I_{71} = 0,\ I_{83} = 0,\ I_{84} = 0,\ I_{82} = 1.$$

Consequently, constraint set (32) is only active for O_{82}. Since $\Delta_{82} = 0$, $Y^M_{82,00,m_3} = 1$, and according to constraint set (31), only G_{82,w_2,m_3} can take a value greater than zero. Therefore, we have:

$$G_{82,w_2,m_3} \le \frac{\chi - S_{82} - td^{\omega}_{m_3}}{pt_{82,w_2,m_3}}.$$

For operations O_{71}, O_{83}, and O_{84}, B_{kj} are 1, and I_{kj} are zero. Constraint sets (34) and (35) imply that $\sum_{w\in W_k}\sum_{m\in M_k} G_{kjwm}$ for those operations are 1.

For MR (2), $V_{71,82}$ and $V_{83,84}$ take the value 1. Regarding MR (3), O_{71} is processed before O_{82}. MR (4) and MR (5) are not considered in our example. Considering MR (6) (constraint set (42)), the following values of Z_{wm} are correct:

$$Z_{w_1,m_1} = Z_{w_1,m_2} = Z_{w_2,m_2} = Z_{w_2,m_3} = 1.$$

Accordingly, constraint sets (43) and (44) are satisfied.

Constraint set (45) determines the value of positive continuous decision variables D_k. Since D_k must be minimized in the objective function, the term

$$\sum_{\substack{j\in J:\\ O_{kj}\in\Omega}} \sum_{w\in W_k} \sum_{m\in M_k} ton_{kj} \cdot G_{kjwm}$$

has to take its maximum value. G_{82,w_2,m_3}, therefore, takes the value of $\frac{\chi - S_{82} - td^{\omega}_{m_3}}{pt_{82,w_2,m_3}}$ and G_{71,w_1,m_1}, G_{83,w_2,m_2}, and G_{84,w_1,m_2} are 1. For $k = 1$, we have:

$$ton^e_1 - ton_{71} \cdot G_{71,w_1,m_1} \le D_1,$$

and for $k = 8$, we have:

$$ton^e_8 - ton_{83} \cdot G_{83,w_2,m_2} - ton_{84} \cdot G_{84,w_1,m_2} - ton_{82} \cdot G_{82,w_2,m_3} \le D_8.$$

Note that except G_{82,w_2,m_3}, G_{71,w_1,m_1}, G_{83,w_2,m_2}, and G_{84,w_1,m_2}, all other G_{kjwm} are 0 (cf. constraint set (31)). D_k takes the value of 0 if the value of the left-hand side of the inequality is negative since D_k is a positive decision variable.

Appendix B. A detailed comparison between similar test instances

In what follows, we consider 8 test instances from the 100 generated realistic instances that have the following characteristics:

1) the number of operations is between 91 (Q_1) and 126 (Q_3);
2) the number of available workers is between 9 and 12;
3) the number of available machines at processing stage 1 is between 3 and 5; and
4) the number of available machines at processing stage 2 is between 2 and 3.

The number of machines at processing stages 1 and 2 are more important than the number of machines at other processing stages. On the one hand, the jobs that start with operations (1) and (2) appear more often, and on the other hand, the number of machines at these processing stages can be more than 2. In Table B.6, the numbers of underground locations, jobs, operations, workers, and total machines (the information regarding the size of the instances) are given for every single instance. Under the column "#Machines", the numbers in parentheses are the number of machines at each processing stage as follows: ($|M_1|$, $|M_2|$, $|M_3|$ or $|M_6|$, $|M_4|$, $|M_5|$, $|M_7|$, $|M_8|$).

For comparing the results achieved by MILP, 2SA, and CH, we consider the value of (MLOF) ($= \sum_{k\in K} D_k + D^{\max}$). We set an upper time limit of 3600 seconds for each approach. In Table B.7, the values of (MLOF) achieved by each approach are given. The numbers in columns "time" are the times in seconds that MILP, 2SA, and CH needed to find the best feasible solution. The best value of MLOF is underlined in each row. Note that MILP cannot prove the optimality for any of the problem instances.

We see in Table B.7 that CH could find for 2 instances and MILP for 6 instances the solutions with the best value of (MLOF).

Table B.6
The key information of the chosen instances.

instance	#UL	#Jobs	#Operations	#Workers	#Machines
1	19	37	99	11	13 (3, 3, 2, 1, 1, 2, 1)
2	22	42	114	10	13 (4, 2, 1, 1, 1, 2, 2)
3	23	46	110	10	17 (5, 3, 2, 2, 2, 2, 1)
4	28	45	108	12	13 (5, 3, 1, 1, 1, 1, 1)
5	19	38	105	11	14 (5, 2, 1, 2, 2, 1, 1)
6	24	45	120	11	14 (4, 3, 2, 1, 2, 1, 1)
7	18	40	91	12	13 (3, 3, 1, 1, 2, 2, 1)
8	24	46	118	11	13 (4, 2, 2, 2, 1, 1, 1)

Table B.7
Values of (MLOF) and the times needed to find the corresponding solutions for the results achieved by MILP, 2SA, and CH.

instance	MILP	time (sec)	2SA	time (sec)	CH	time (sec)
1	3714.5	3469.7	5054.5	1680.5	4199.0	2,168.4
2	820.6	3365.5	1802.4	1965.5	4665.0	2.8
3	4051.8	3310.7	6011.6	2693.8	4008.0	0.6
	5439.7	3524.1	6615.0	221.9	6622.0	5.1
5	5766.3	3240.9	6347.3	3590.3	7949.0	4.9
6	5468.6	1660.5	5423.7	756.8	4870.0	6.9
7	6535.4	1823.2	6844.8	314.3	7018.0	171.1
8	3175.8	3069.9	5010.6	264.9	3559.0	5.5

Table B.8
The Gap to the best solution found considering the values of MLOF.

instance	MILP	2SA	CH
1	0.0%	36.1%	13.0%
2	0.0%	119.6%	468.5%
3	1.1%	50.0%	0.0%
4	0.0%	21.6%	21.7%
5	0.0%	10.1%	37.9%
6	12.3%	11.4%	0.0%
7	0.0%	4.7%	7.4%
8	0.0%	57.8%	12.1%

Table B.9
Some information derived from the GUROBI solver while using MILP.

instance	#nodes	#integer variables	#continuous variables	#constraints	gap to LB
1	11,145	100,703	2227	248,468	50.2%
2	7561	123,783	2028	312,503	100.0%
3	7793	101,701	2350	261,361	64.9%
4	2456	122,024	2567	299,652	57.4%
5	14,751	106,575	2002	267,661	97.7%
6	4361	145,476	2846	359,921	80.9%
7	3095	96,545	2312	232,384	69.5%
8	10,934	136,370	2390	340,905	22.4%

MILP, 2SA, and CH find the best feasible solution, on average, after 2,933.1, 1,436.0, and 295.7 seconds, respectively. Let ξ^* be the solution found by approach * and $f(\xi^*)$ be the value of (MLOF) for ξ^*. The best solution found, ξ^{best}, is the solution for which:

$$f(\xi^{\text{best}}) = \min\{f(\xi^{\text{MILP}}), f(\xi^{\text{2SA}}), f(\xi^{\text{CH}})\}.$$

A gap to the best solution found is calculated as follows:

$$\text{Gap} = \frac{f(\xi^*) - f(\xi^{\text{best}})}{f(\xi^{\text{best}})}.$$

Table B.8 presents the values of Gap to the best solution found. Regarding the values of (MLOF), the Gap to the best solution found is, on average, 1.7% for MILP, 38.9% for 2SA, and 70.1% for CH. Although 2SA could not find the best solution for any of the problem instances, it can find a much better solution than CH considering the Gap-values. CH is clearly the quickest way to find a feasible solution (cf. Table B.7) and finds the best solution for 2 (over 8) problem instances. However, the average quality of the solution found by CH is comparably the worst.

In Table B.9, some information derived from the GUROBI solver while solving an instance using MILP is summarized. For every instance, the number of nodes that a search tree required to find the best solution, the number of integer and continuous decision variables, the number of constraints, and the gap to the lower bound found by the GUROBI solver are given. The default setting of the GUROBI solver uses the formula "$\frac{f(\xi^{\text{MILP}}) - \textbf{LB}}{f(\xi^{\text{MILP}})}$" to determine the gap to lower bounds (**LB**). Therefore, the gap to the lower bound is 100% if the lower bound is equal to 0. Considering Table B.6, we cannot identify any correlation between the size of an instance and the number of nodes that a search tree required to find the best feasible solution. Moreover, there is no noticeable trend that links the size of an instance to the gap to the lower bound found by the GUROBI solver after 3600 seconds.

The gap to the lower bound found by the GUROBI solver after 3600 seconds is, on average, 67.9%. The fact that on the one hand, our MILP provides better solutions than CH and 2SA, and on the other hand, the gap to the corresponding lower bound is still big, shows that it is tough to find reasonable lower bounds for such a hard problem. Note that for test instance 2, the gap to the lower bound is 100%. However, our MILP could find the best solution among MILP, 2SA, and CH, where the gap to the best solution found by 2SA is 119.6% and by CH is 468.5% (cf. Table B.8).

References

Agnetis, A., Murgia, G., & Sbrilli, S. (2014). A job shop scheduling problem with human operators in handicraft production. *International Journal of Production Research, 52*(13), 3820–3831.

Ahmadi-Javid, A., & Hooshangi-Tabrizi, P. (2017). Integrating employee timetabling with scheduling of machines and transporters in a job-shop environment: A mathematical formulation and an anarchic society optimization algorithm. *Computers & Operations Research, 84*, 73–91.

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research, 246*(2), 345–378.

Ammar, A., Pierreval, H., & Elkosentini, S. (2013). Workers assignment problems in manufacturing systems: A literature analysis. In *Proceedings of 2013 international conference on industrial engineering and systems management (IESM)* (pp. 1–7). IEEE.

Araz, O. U. (2005). A simulation based multi-criteria scheduling approach of dual-resource constrained manufacturing systems with neural networks. In *Australasian joint conference on artificial intelligence* (pp. 1047–1052). LNAI.

Artigues, C., Gendreau, M., & Rousseau, L. M. (2006). A flexible model and a hybrid exact method for integrated employee timetabling and production scheduling. In *International conference on the practice and theory of automated timetabling* (pp. 67–84). Springer, Berlin, Heidelberg.

Artigues, C., Gendreau, M., Rousseau, L. M., & Vergnaud, A. (2009). Solving an integrated employee timetabling and job-shop scheduling problem via hybrid branch-and-bound. *Computers & Operations Research, 36*(8), 2330–2340.

Behnamian, J. (2014). Scheduling and worker assignment problems on hybrid flow-shop with cost-related objective function. *The International Journal of Advanced Manufacturing Technology, 74*(1–4), 267–283.

Benavides, A. J., Ritt, M., & Miralles, C. (2014). Flow shop scheduling with heterogeneous workers. *European Journal of Operational Research, 237*(2), 713–720.

Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., & De Boeck, L. (2013). Personnel scheduling: A literature review. *European Journal of Operational Research, 226*(3), 367–385.

Campos Ciro, G., Dugardin, F., Yalaoui, F., & Kelly, R. (2016). Open shop scheduling problem with a multi-skills resource constraint: A genetic algorithm and an ant colony optimisation approach. *International Journal of Production Research, 54*(16), 4854–4881.

Daniels, R. L., Mazzola, J. B., & Shi, D. (2004). Flow shop scheduling with partial resource flexibility. *Management Science, 50*(5), 658–669.

Frihat, M., Sadfi, C., & Hadj-Alouane, A. B. (2014). Optimization of integrated employee timetabling and hybrid job shop scheduling under time lag constraints. In *International conference on control, decision and information technologies (CODIT)* (pp. 282–287). IEEE.

Gong, G., Chiong, R., Deng, Q., Han, W., Zhang, L., Lin, W., & Li, K. (2020). Energy-efficient flexible flow shop scheduling with worker flexibility. *Expert Systems with Applications, 141*, 112902.

Guyon, O., Lemaire, P., Pinson, E., & Rivreau, D. (2014). Solving an integrated job-shop problem with human resource constraints. *Annals of Operations Research, 213*(1), 147–171.

Huq, F., Cutright, K., & Martin, C. (2004). Employee scheduling and makespan minimization in a flow shop with multi-processor work stations: A case study. *Omega, 32*(2), 121–129.

Kress, D., Müller, D., & Nossack, J. (2019). A worker constrained flexible job shop scheduling problem with sequence-dependent setup times. *OR Spectrum, 41*(1), 179–217.

K+S AG (2013(Accessed 30-Oct-2018)). A Day in the Mine. http://www.k-plus-s.com/en/ks-zum-anfassen/tag-in-der-grube.html.

Mencía, C., Sierra, M. R., & Varela, R. (2013). An efficient hybrid search algorithm for job shop scheduling with operators. *International Journal of Production Research, 51*(17), 5221–5237.

Mencía, R., Sierra, M. R., Mencía, C., & Varela, R. (2016). Genetic algorithms for the scheduling problem with arbitrary precedence relations and skilled operators. *Integrated Computer-Aided Engineering, 23*(3), 269–285.

Meng, L., Zhang, C., Zhang, B., & Ren, Y. (2019). Mathematical modeling and optimization of energy-conscious flexible job shop scheduling problem with worker flexibility. *IEEE Access, 7*, 68043–68059.

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.

Paksi, A., & Ma'ruf, A. (2016). Flexible job-shop scheduling with dual-resource constraints to minimize tardiness using genetic algorithm. In *IOP conference series: Materials science and engineering: 114* (pp. 1–9). IOP Publishing.

Puttkammer, K., Kleber, R., Schulz, T., & Inderfurth, K. (2011). Simultane Maschinenbelegungs- und Personaleinsatzplanung in KMUs anhand eines Fallbeispiels aus der Druckereibranche. In E. Sucky, B. Asdecker, A. Dobhan, S. Haas, & J. Wiese (Eds.), *Logistikmanagement: Herausforderungen, Chancen und Lösungen* (pp. 229–247). University of Bamberg Press.

Queyranne, M., & Schulz, A. S. (1994). *Polyhedral approaches to machine scheduling*. Technische Universitt Berlin, Berlin

Ramya, G., & Chandrasekaran, M. (2014). Shuffled frog leaping algorithm approach to employee timetabling and job shop scheduling. *International Journal of Internet Manufacturing and Services, 3*(3), 178–203.

Ruiz, R., & Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research, 205*(1), 1–18.
Santos, F., Fukasawa, R., & Ricardez-Sandoval, L. (2018). An integrated personnel allocation and machine scheduling problem for industrial size multipurpose plants. *IFAC-PapersOnLine, 51*(18), 156–161.
Schulze, M. (2016). *Ein hierarchischer Ansatz zur Lösung von Ablaufplanungsproblemen im Bergbau: Darstellung am Beispiel des Örterbaus*. Aachen: Shaker.
Schulze, M., Mathiak, T., & Haney, C. (2017). Using operations research techniques to increase efficiency in german potash mining. In *Proceedings of the 38th international symposium on the application of computers and operations research in the mineral industry, Golden, Colorado*.
Schulze, M., Rieck, J., Seifi, C., & Zimmermann, J. (2016). Machine scheduling in underground mining: An application in the potash industry. *OR Spectrum, 38*(2), 365–403.
Schulze, M., & Zimmermann, J. (2017). Staff and machine shift scheduling in a German potash mine. *Journal of Scheduling, 20*(6), 635–656.
Seifi, C., Schulze, M., & Zimmermann, J. (2019). A two-stage solution approach for a shift scheduling problem with a simultaneous assignment of machines and workers. In *Mining goes digital: Proceedings of the 39th international symposium application of computers and operations research in the mineral industry (APCOM 2019), June, 2019, Wroclaw, Poland* (pp. 377–385). CRC Press.
Uzun Araz, Ö., & Salum, L. (2010). A multi-criteria adaptive control scheme based on neural networks and fuzzy inference for DRC manufacturing systems. *International Journal of Production Research, 48*(1), 251–270.
Xu, J., Xu, X., & Xie, S. (2011). Recent developments in dual resource constrained (DRC) system research. *European Journal of Operational Research, 215*(2), 309–318.
Yazdani, M., Zandieh, M., & Tavakkoli-Moghaddam, R. (2019). Evolutionary algorithms for multi-objective dual-resource constrained flexible job-shop scheduling problem. *OPSEARCH, 56*(3), 983–1006.
Yazdani, M., Zandieh, M., Tavakkoli-Moghaddam, R., & Jolai, F. (2015). Two meta-heuristic algorithms for the dual-resource constrained flexible job-shop scheduling problem. *Scientia Iranica Transaction E, 22*(3), 1242–1257.
Zhang, J., Wang, W., & Xu, X. (2017). A hybrid discrete particle swarm optimization for dual-resource constrained job shop scheduling with resource flexibility. *Journal of Intelligent Manufacturing, 28*(8), 1961–1972.