Computers and Operations Research 115 (2020) 104590

Contents lists available at ScienceDirect

Computers and Operations Research

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

Hyper-heuristic approaches for strategic mine planning under uncertainty

Amina Lamghari*, Roussos Dimitrakopoulos

COSMO – Stochastic Mine Planning Laboratory, McGill University, FDA Building, 3450 University Street, Montreal, Quebec, H3A 2A7, Canada

ARTICLE INFO

Article history:
Received 11 September 2017
Revised 3 June 2018
Accepted 16 November 2018
Available online 17 November 2018

Keywords:
Hyper-heuristics
Strategic mine planning
Decision making under uncertainty
Large-scale optimization
Local search

ABSTRACT

A hyper-heuristic refers to *a search method or a learning mechanism for selecting or generating heuristics to solve computational search problems*. Operating at a level of abstraction above that of a metaheuristic, it can be seen as an algorithm that tries to find an appropriate solution method at a given decision point rather than a solution. This paper introduces a new hyper-heuristic that combines elements from reinforcement learning and tabu search. It is applied to solve two complex stochastic scheduling problems arising in mining, namely the stochastic open-pit mine production scheduling problem with one processing stream (SMPS) and one of its generalizations, SMPS with multiple processing streams and stockpiles (SMPS+). The performance of the new hyper-heuristic is assessed by comparing it to several solution methods from the literature: problem-specific algorithms tailored for the two problems addressed in the paper and general hyper-heuristics, which use only limited problem-specific information. The computational results indicate that not only is the proposed new hyper-heuristic approach superior to the other hyper-heuristics, but it also provides results that are comparable to or improve on the results obtained by the state-of-the-art problem-specific methods.

1. Introduction

Strategic mine planning is a critical element in the process of extracting minerals from the ground for profit, a complex operation that involves investments in the order of hundreds of millions of dollars and whose revenues are tied to scheduling and production strategies. A strategic mine plan involves devising a long-term production plan over the life-of-the-mine, typically a 10–30 year time frame, that maximizes the net present value (NPV) of the mining operation while meeting various physical and operational requirements at the extraction and processing levels. These requirements depend on the specificities and characteristics of the mining operation, not all of which are the same from one operation to another. The sources of supply might be open-pit mines, underground mines, or both. The processing facilities and processing paths might also be different, as they depend on the main minerals in the mines, on the different products produced in the various processing streams, and on the geographical location. This creates different strategic mine planning problems, and the problem to be solved depends on the specificities and characteristics of the mining operation under consideration. To date, a number of solution methods, mainly (meta)heuristics, have been developed for strategic mine planning problems, but not all methods work for every problem. Often, they are tailored and fine-tuned for a particular problem. When faced with a new problem specific to a given mining operation, the question is, how can we determine which method will work best for this problem? In other words, is there a methodology that will, given a particular problem and a number of solution methods, help determine which method or combination of methods is the best for the given problem?

The simplest and most commonly studied strategic mine planning problem involves one open-pit mine, one processing facility, and one waste dump, and is often referred to as the open-pit mine production scheduling problem (MPS). In the MPS, the mine is discretized into a set of blocks, each of which represents a volume of material that can be mined. The objective is to establish a schedule for the removal of material from the mine (which blocks to mine at each period of the life-of-the-mine) that maximizes the NPV. Every block must be mined after its predecessors (slope constraints), and the mining and processing equipment are assumed to have a certain capacity (resource constraints). MPS is modelled as a linear integer program. It generalizes the constrained max-

* Corresponding author.
E-mail addresses: amina.lamghari@mcgill.ca (A. Lamghari), roussos.dimitrakopoulos@mcgill.ca (R. Dimitrakopoulos).

https://doi.org/10.1016/j.cor.2018.11.010

imum closure problem and is therefore NP-hard (Bienstock and Zuckerberg, 2010; Hochbaum and Chen, 2000). This makes solving large instances of practical interest computationally challenging and beyond the scope of exact methods and general-purpose solvers.[1] This difficulty is exacerbated by the need to incorporate operational constraints in addition to the usual slope and resource constraints. For example, lower bounds on mining, processing, and metal production are often required to ensure that the resources are utilized evenly and that the demand is satisfied at each period. While this is useful and closer to the problem encountered in practical settings, the introduction of such constraints in the MPS formulation makes the problem harder to solve (Cullenbine et al., 2011).

Another realistic and important aspect that further complicates the solution process is metal uncertainty. Metal uncertainty, also referred to as geological or reserve uncertainty, stems from the fact that the metal content of the blocks is not known at the time decisions are made, but is inferred from limited drilling data. The benefits of integrating metal uncertainty in the optimization process are well-documented in the literature. By taking into account the effects of metal uncertainty on present decision-making, not only is risk in meeting production targets reduced, but also major improvements in NPV in the order of 10 to 30% are reached (Albor and Dimitrakopoulos, 2010; Asad and Dimitrakopoulos, 2013; Dimitrakopoulos, 2011; Dimitrakopoulos et al., 2002; Dowd, 1994; Fricke et al., 2014; Lamghari and Dimitrakopoulos, 2016b; Marcotte and Caron, 2013; Menabde et al., 2007; Ravenscroft, 1992).

Over the years, several exact and approximate methods have been proposed for strategic mine planning problems. Most of the literature around such methods deals with the simplest deterministic case — one open-pit mine, one processing facility, one waste dump, and metal uncertainty is ignored (MPS). Exact methods that exploit the structure of the problem were proposed by Boland et al. (2009) and Bley et al. (2010). Heuristic and metaheuristic approaches have been introduced by, among others, Ferland et al. (2007) and Cullenbine et al. (2011), while hybrid methods were proposed by Moreno et al. (2010), Chicoisne et al. (2012), and Lamghari et al. (2015). The stochastic version of MPS that accounts for metal uncertainty has been tackled mainly using metaheuristics. Godoy (2002) and Albor and Dimitrakopoulos (2009) proposed simulated annealing algorithms. A tabu search algorithm and a variable neighborhood descent algorithm were developed by Lamghari and Dimitrakopoulos (2012) and Lamghari et al. (2014), respectively. More complex mining operations involving multiple destinations for the extracted material have been studied by Ramazan and Dimitrakopoulos (2013), Behrang et al. (2014), and Lamghari and Dimitrakopoulos (2016a). Montiel and Dimitrakopoulos (2015) and Goodfellow and Dimitrakopoulos (2016, 2017) considered stochastic mineral value chains, also known as mining complexes. The problem is to simultaneously optimize different aspects of the chain such as extraction, processing, and transportation accounting for metal uncertainty. Operations involving both open-pit and underground mines have been considered by Montiel et al. (2016). Hoerger et al. (1999), Chanda (2007), Whittle (2007), Whittle (2009), and Kawahata et al. (2015) have also considered operations consisting of multiple mines, but their studies are restricted to deterministic environments; i.e., they do not account for metal uncertainty. For a general overview of mine planning optimization problems, see Newman et al. (2010) and Dimitrakopoulos (2011). For a literature review specifically aimed at strategic mine planning problems and solution methodologies, see Lamghari (2017).

To the best of our knowledge, all the approaches proposed in the literature to optimize strategic mine planning decisions either use aggregation techniques to reduce the size of the problem so that the resulting model is of tractable size and can be solved using exact methods, or are based on (meta)heuristics, or combine (meta)heuristics and exact methods. Each of these approaches presents some weaknesses. Aggregation can severely compromise the validity and usefulness of the solution (Bienstock and Zuckerberg, 2010). It causes loss of profitability and might even lead to infeasible solutions (Boland et al., 2009). Approaches based on (meta)heuristics have been proven to be successful for solving large-scale instances without resorting to aggregation, but they have two major flaws. First, they might involve a relatively large number of parameters and/or algorithm choices, and they generally do not provide guidance on how to make such choices. Therefore, it is not always clear *a priori* which choices will perform better in a particular situation, meaning that tuning might be required if dealing with new variants of the problem or even new instances of the same variant. Second, they are problem-specific methods. Problem-specific methods can often obtain excellent results for the problem they have been designed for, but they are not readily applicable to other problems or other variants of the same problem. They have to be adapted to the new problem, and if so, they might not perform as well as on the original problem. An illustration of such a situation can be found in the study in Lamghari and Dimitrakopoulos (2016a), which indicates that the tabu search metaheuristic developed in Lamghari and Dimitrakopoulos (2012) to solve the problem with one processing stream worked well on that particular problem but exhibits a poorer performance on the problem considering multiple destinations for the extracted material, including stockpiles. Some approaches that combine (meta)heuristics and exact methods are also limited by the same weakness; that is, they are tailored to one specific case. For example, the algorithm proposed by Moreno et al. (2010) is only applicable to the variant of the MPS with a single knapsack constraint per period.

As evidenced from the above discussion, there is a need for solution approaches that are able to tackle large-scale instances without resorting to aggregation, that are self-managed, and that are more general than currently existing methodologies. The latter feature is particularly important since, as mentioned earlier, in strategic mine planning, the mining and processing requirements are often different as they depend on the specificities and characteristics of the mining operation under consideration, and thus lead to different problems. We believe that a general algorithmic framework that is re-usable without major structural modifications is more appropriate than a problem-specific approach that must be re-adapted (if possible) to each new problem tackled.

In response to this need, we propose using hyper-heuristic approaches. Operating at a level of abstraction above that of a metaheuristic, a hyper-heuristic is an emergent search methodology that seeks to automate the process of selecting and combining simpler heuristics or the process of generating new heuristics from components of existing heuristics in order to solve hard computational search problems (Burke et al., 2013; 2003a; Ross, 2005). A hyper-heuristic can be seen as an algorithm that tries to find an appropriate solution method at a given decision point rather than a solution. The ideas behind hyper-heuristics date back to the 1960s (Fisher and Thompson, 1963), but the term hyper-heuristic was first used only in the late 1990ÃIt was coined in the context of automated theorem proving to describe a protocol that combines different Artificial Intelligence methods (Denzinger et al., 1997), then independently in the context of combinatorial optimization to describe a high-level heuristic to choose lower-level heuristics using only limited problem-specific information (Cowling et al.,

[1] In practical settings, a medium-size open-pit mine consists of tens thousands of blocks, and the resulting optimization problem has hundreds of thousands binary variables. This is far more than any of the existing exact methods can handle in a reasonable amount of time.

2001). Over the last decade, many papers presenting successful applications of hyper-heuristics to various difficult combinatorial problems have appeared in the literature, including hyper-heuristics for personal scheduling, sports scheduling, educational timetabling, space allocation, cutting and packing, and vehicle routing problems. For a recent and comprehensive survey of the literature, see Burke et al. (2013).

Despite the work that has been done in the above areas, hyper-heuristics, to the best of our knowledge, have not been applied to solve mine planning optimization problems, and the objective of this paper is to propose such a work. There are two main categories of hyper-heuristics: heuristic selection, which are methodologies for choosing existing heuristics, and heuristic generation, which are methodologies for creating new heuristics from a set of components of other existing heuristics (Burke et al., 2010). The three hyper-heuristic approaches proposed in this paper fall under the first category. Specifically, they use a set of simple perturbative low-level heuristics to improve a candidate solution. The decision of which low-level heuristic should be applied at a given step of the search process relies on a score-based learning mechanism, whereby a score is associated with each heuristic reflecting its past performance, and the heuristics are selected based on these scores. The scores are updated periodically, and the process terminates when a pre-specified stopping criterion is met. While this general framework is similar in the three hyper-heuristic approaches considered in this paper, the score update rules and the heuristic selection strategy are different.

Two of the hyper-heuristic approaches come from the literature (Burke et al., 2003b; Drake et al., 2012), while the third one is a novel approach that uses some of the ideas of the first two but also includes new features aimed to overcome their weaknesses. The generality of the three proposed hyper-heuristics is demonstrated by applying them to various instances of two strategic mine planning problems having different characteristics. The performance of the hyper-heuristics is assessed by comparing them to each other and to other search methodologies from the literature. This comparison indicates that the new hyper-heuristic that we propose in this paper outperforms the two others, providing results that are comparable to or improve on the results obtained by the state-of-the-art problem-specific methods.

In the next section, a description of the two strategic mine planning problems considered in this paper is given. The proposed hyper-heuristics are described in Section 3. Numerical results are reported in Section 4. Section 5 provides conclusions and directions for future research.

2. Description of the problems

As stated in Section 1, there are different strategic mine planning problems depending on the specificities and characteristics of the mining operation under consideration. The two that are considered in this paper are described in the following sections.

2.1. Stochastic open-pit mine production scheduling problem with one processing stream (SMPS)

This mining operation consists of one open-pit mine from which blocks are extracted,[2] one processing facility where extracted ore blocks are treated to recover the metal they contain,[3] and one waste dump where extracted blocks that cannot be processed profitably are disposed.[4] Although not profitable, low-grade blocks have to be extracted to either have access to higher-grade blocks or ensure safe wall slopes for the pit. The metal content of any given block, which determines whether the block is an ore block to be processed or a waste block to be discarded, is not known prior to the extraction. What is available is a number of equiprobable scenarios, each of which provides possible values of the blocks' metal content given the geological data and the information obtained from drilling. To generate the scenarios, geostatistical techniques of conditional simulation are used. Those can be seen as complex Monte Carlo simulation frameworks able to reproduce all available data and information, as well as spatial statistics of the data (Boucher and Dimitrakopoulos, 2009; Chiles and Delfiner, 2012; Goovaerts, 1997; Horta and Soares, 2010; Maleki and Emery, 2015; Rossi and Deutsch, 2014).

The optimization problem associated with the mining operation described above, referred to as SMPS in the rest of the paper, concerns the design of a mining sequence over a discrete finite planning horizon (the life-of-the-mine); that is, deciding which blocks should be extracted at each period of the life-of-the-mine. In doing so, various constraints must be satisfied. Logical and physical restrictions impose that each block can be extracted at most once, after all its predecessors have been extracted. Operational restrictions require that, at each period of the life-of-the-mine, the total amount of material extracted (ore and waste), the total amount of ore processed, and the total amount of metal produced should lie between specific lower and upper bounds. The extraction and processing operations incur costs, while the metal recovered from processing is sold and generates revenue. All cost and revenue components are future cash-flows and thus must be discounted to the present, so feasible solutions are evaluated by their net present value (NPV) to select the one that provides the highest value.

As mentioned earlier, extraction decisions are to be made prior to knowing the metal content of the blocks, and the latter affects: i) the amount of ore available for processing at each period; ii) the amount of metal produced from processing at each period; and iii) the NPV. Thus, according to the particular metal scenario realized, not only might the NPV shift upward or downward, but also ore and metal production targets might fail to be satisfied in some or all periods (exceed the upper bound or fall under the lower bound). Some recourse actions are available to adapt to the situation at hand, but they are subject to extra costs. For example, if an excess in ore production occurs, extra processing capacity is required and an additional cost is incurred. Clearly, for better-informed decision-making, the aforementioned effects of uncertainty must be taken into account. Because initially one has to decide on which blocks to extract, but only later, when the metal uncertainty is disclosed, does one have to decide how best to deal with the excess and shortage in ore and metal (recourse decisions), the problem is formulated as a two-stage stochastic programming model (Birge and Louveaux, 2011), where the overall objective is to maximize the expected net present value of the mining operation and to minimize the future expected recourse costs over the uncertain metal scenarios. This model is described in detail in Lamghari and Dimitrakopoulos (2012) and is briefly recalled in Appendix A.

2.2. Stochastic open-pit mine production scheduling problem with multiple processors and stockpiles (SMPS+)

In the SMPS+, several destinations for the extracted material are considered instead of only two (a processor and a waste dump),

[2] Recall that the mine is discretized into a set of blocks, each of which represents a volume of material that can be mined.

[3] This process incurs a unit cost, so a block is processed only if the revenue from the metal recovered pays for the processing and selling costs. Such a block is referred to as an ore block.

[4] Low-grade blocks, also referred to as waste blocks.

and these destinations include stockpiles. The stockpiles are used to absorb the excess of ore such that when such a situation occurs, some ore is not immediately processed in the period it is mined in but rather sent to the stockpiles from which it is reclaimed in periods where there is spare capacity. Hence, in each period, an extracted block is sent either to one of the processors, or to one of the stockpiles, or to the waste dump. If blocks are sent to the stockpile, unit transportation and handling costs are incurred. Costs are also incurred when reclaiming material from the stockpiles. Thus, an optimal solution maximizes the NPV and indicates the set of blocks that should be extracted in each period, the destination of these blocks, and the amount of material to take from the stockpiles in each period to feed the processors. That is, compared to the optimization problem described in the previous section (SMPS), this problem (SMPS+) incorporates the material flow aspect in addition to the mining sequence design.

SMPS+ can also be formulated as a two-stage stochastic program. The first-stage consists of designing the mining sequence, and the second-stage consists of processing and stockpiling (i.e.; material flow decisions). The second stage decisions are made based on the first-stage decisions (i.e, the mining sequence) and on the realized metal scenario. A detailed description of the mathematical model is available in Lamghari and Dimitrakopoulos (2016a), and an outline of it is provided in Appendix B.

3. Hyper-heuristic solution approaches

As mentioned in Section 1, the three hyper-heuristic solution approaches considered in this paper fall under the category of *heuristic selection* (Burke et al., 2010); that is, they are methodologies for choosing existing heuristics. Burke et al. (2010) further categorize hyper-heuristics according to the nature of the heuristic search space (constructive heuristics versus perturbation heuristics) and the source of feedback during learning (online learning, offline learning, and no learning). With respect to this classification, the three approaches can be seen as approaches based on perturbation low-level heuristics with online learning. The general framework can be summarized as follows: The algorithm starts by generating an initial solution (a random feasible solution in this paper), and then tries to iteratively improve it using different local search heuristics (low-level heuristics). These heuristics are described in Section 3.4. To determine the appropriate heuristic to apply at a given iteration, the algorithm relies on a score-based learning mechanism. This means that a periodically updated score $S(h_j)$ is assigned to each heuristic h_j to measure how well h_j has performed during the search, and the heuristics are selected based on these scores. The selected heuristic is applied once to the current solution to obtain a new solution. Another decision that has to be made at this point is whether or not to accept this new solution. In this paper, the new solution is always accepted, independently of its quality. This means that this new solution becomes the new current solution. It also replaces the incumbent solution if it has a better objective value. This procedure is iterated until the stopping criterion is met. In this paper, the procedure terminates when a specified number of iterations, $\Upsilon_{\max}$, has elapsed.

Various learning mechanisms for selecting heuristics can be used in the framework outlined above. Three are considered in this paper, and each leads to a different hyper-heuristic. The first two that we will describe are previously-developed hyper-heuristics, introduced for comparison purposes, while the third is a new hyper-heuristic that we propose in this paper.

3.1. Tabu search hyper-heuristic (HH1)

The first hyper-heuristic, henceforth referred to as HH1, has been proposed by Burke et al. (2003b) and tested on timetabling and rostering problems. To guide the heuristic selection process, HH1 uses principles of reinforcement learning and tabu search metaheuristic. Let H be the number of low-level heuristics. Initially, each heuristic has a score equal to 0. As the search progresses, the scores increase and decrease within the interval [0, H], and the heuristics are selected according to the updated scores. Not all heuristics are available for selection at a given iteration. A dynamic tabu list of heuristics is maintained to temporarily exclude some of them. Details of a typical iteration are as follows: HH1 selects the non tabu low-level heuristic having the highest score. It applies it once and then compares the value of the current solution to the value of the new solution. If the new solution is better than the current solution, the heuristic is rewarded by incrementing its score by one. If the new solution and the current solution have the same value, the heuristic is punished by decrementing its score by one and making it tabu. Finally, if the new solution is worse than the current solution, HH1 proceeds as in the previous case except that the tabu list is first emptied before adding the heuristic.

Burke et al. (2003b) justify emptying the tabu list by claiming that there is no point in keeping a heuristic tabu once the current solution has been modified. However, there is a potential drawback to this strategy: The tabu status is revoked too soon, which might lead to choosing a relatively poor heuristic too often, thereby missing the opportunity to apply other better performing heuristics. To clarify, since the heuristics are rewarded the same way, independently of the magnitude of improvement they can achieve, a heuristic that is able to slightly improve the solution but also deteriorates it occasionally can gain large rewards. It might have the highest score and making it non tabu as soon as one single other heuristic has also been found deteriorating leads to using the same heuristic again and again (cycling behavior). This implies that the other heuristics, which might be better performing and more appropriate at the current decision point, have little or no chance to be selected. Strategies that allow a better exploration of the heuristic search space are discussed in Section 3.3, where we describe how the new proposed hyper-heuristic, HH3, overcomes the weakness of HH1.

3.2. Choice-function hyper-heuristic (HH2)

The second hyper-heuristic considered in this paper, henceforth referred to as HH2, has been proposed by Drake et al. (2012) and tested on personnel scheduling problems. It extends the hyper-heuristic developed in Cowling et al. (2001) and differs from HH1 in that i) it does not incorporate any mechanism at the high level to prevent choosing heuristics that did not perform well recently; and ii) it uses a more complex score update scheme. To be more specific, rather than incrementing and decrementing the score based on the heuristic's ability to improve the solution as HH1 does, HH2 calculates the scores as a weighted sum of the three following measures:

- The first measure, f_1, keeps track of the performance of the heuristics. It accounts for the improvement that each heuristic has achieved so far as well as the time it has required. Following the same notation as in Drake et al. (2012), let $I_n(h_j)$ be the change in the objective function value obtained the n^{th} last time h_j was called (applied), and let $T_n(h_j)$ be the time required. Denote by $\phi \in]0, 1[$ a weight adjustment parameter, defining the importance given to recent performance. The value of $f_1(h_j)$ is computed using the following formula:

$$f_1(h_j) = \sum_n \phi^{n-1} \frac{I_n(h_j)}{T_n(h_j)}. \quad (1)$$

- The second measure, f_2, seeks to capture any pairwise dependencies between heuristics. Whenever h_j is called immediately

after h_k, the value of $f_2(h_k, h_j)$ is updated as follows:

$$f_2(h_k, h_j) = \sum_n \phi^{n-1} \frac{I_n(h_k, h_j)}{T_n(h_k, h_j)} \quad (2)$$

where, ϕ is as defined previously, and $I_n(h_k, h_j)$ and $T_n(h_k, h_j)$ are respectively the change in the objective function value and the time required by heuristic h_j at the n^{th} last call following a call to h_k.

- The last measure, f_3, accounts for the time elapsed since each heuristic was last selected. If we denote this time by $\tau(h_j)$, then:

$$f_3(h_j) = \tau(h_j). \quad (3)$$

The score associated with heuristic h_j is a weighted sum of the three measures above:

$$S(h_j) = \phi f_1(h_j) + \phi f_2(h_k, h_j) + \delta f_3(h_j). \quad (4)$$

Note that the purpose of using measures f_1 and f_2 is to intensify the search by favoring heuristics that have shown good performance, while measure f_3 aims to give all heuristics a chance to be selected, thus providing an element of diversification. The weights ϕ and δ are parameters in the interval]0, 1[used to provide a balance between intensification and diversification. They are dynamically adjusted during the search process based on reinforcement learning principles. This is done as follows: Once the selected heuristic has been applied, the new solution is compared to the current solution. If it is better, then ϕ is increased to $\phi_{\max}$, a maximum value close to the upper bound 1, while δ is decreased to $\delta_{\min}$, a minimum value close to the lower bound 0, thus promoting intensification but reducing diversification. Otherwise, the value of ϕ is decreased by a linear factor κ and δ is increased by the same factor to gradually favor diversification over intensification.

HH2 has three drawbacks:

- It might select a heuristic that largely deteriorates the solution and requires long computational time rather than a heuristic that slightly deteriorates the solution and requires short computational time. To clarify, consider the following scenario with only two low-level heuristics, h_1 and h_2. Assume that h_1 was first applied and resulted in $I_1(h_1) = -20$ and $T_1(h_1) = 10$. At the second iteration, the heuristics are selected based on their scores. The scores of h_1 and h_2 are $S(h_1) = -2\phi$ and $S(h_2) = 10\delta$, respectively. Since h_2 has the highest score, it will be selected and applied. Assume that h_2 resulted in $I_1(h_2) = -1$ and $T_1(h_2) = 1$. Now, the scores of h_1 and h_2 are $S(h_1) = -2\phi + \delta$ and $S(h_2) = -2\phi$, respectively. Clearly, $S(h_1) > S(h_2)\ \forall \phi, \delta \in]0, 1[$. So, despite the fact that h_2 showed a relatively better performance compared to h_1, the latter has the highest score and will be selected.
- Although measures f_1 and f_2 are defined so as to give a greater importance to recent performance, the way the parameter ϕ is adjusted can lead to early performance dominating recent performance. Consider two heuristics h_j and h_k. Assume that h_j obtained large improvements in the early stages of the search but has exhibited a poor performance recently, while h_k has yielded small improvements since the beginning of the search. Assume also that the last iteration resulted in an improvement of the current solution and, consequently, the value of ϕ was increased to $\phi_{\max}$ to favor heuristics that showed good performance and intensify the search. Now recall that the parameter ϕ defines not only the importance given to good performing heuristics but also the importance given to previous performance; i.e, large ϕ values put more emphasis on previous performance. So, looking back to the example above, while one would want to favor h_k because the improvements it achieves, although small, are more significant at the current stage of the search, HH2 will select h_j because its recent performance is dominated by its early performance and thus has a small impact on the score value.
- A particularity of the problems addressed in this paper is that their objective functions take very large values in the order of hundreds of millions. Preliminary tests showed that, in general, the change in the objective function resulting from applying a given heuristic is in the order of tens of thousands, and that this change is obtained in a fraction of a second. Consequently, in Eq. (4), the values of the intensification components f_1 and f_2 have an order of magnitude much larger than that of the diversification component, f_3. This means that the latter component is dominated by the first two. It becomes obsolete when calculating the scores and thus does not provide diversification, as it is supposed to do.

The three aforementioned issues of HH2 are addressed in the new proposed hyper-heuristic presented in the next section.

3.3. New proposed hyper-heuristic (HH3)

In this section, the new proposed hyper-heuristic, referred to as HH3 in the rest of the paper, is described. HH3 uses some of the ideas of HH1 and HH2 but also includes new features aimed to overcome their weaknesses, outlined in the previous sections.

HH3 proceeds in two stages. In the first stage, the algorithm randomly picks a heuristic h_j, applies it, returns the resulting change in the objective function value ($\Delta f(h_j)$) as well as the time required ($T(h_j)$), and computes the heuristicÃnitial score ($S(h_j)$). The initial scores reflect the order of importance given to the heuristics. Of first importance are heuristics that are able to improve the solution. The larger the improvement rate per unit of time is, the more important the heuristic is considered. Because heuristics that deteriorate the solution help getting out of local optima, they are considered more important than heuristics that cannot modify the objective function value. However, not all of them are equally important. The more a heuristic deteriorates the solution and the more computational time it requires, the less important it is considered. Accordingly, the initial scores are computed using the following formula:

$$S(h_j) = \begin{cases} \frac{\Delta f(h_j)}{T(h_j)} & \text{if } \Delta f(h_j) \geq 0, \\ \frac{1}{|\Delta f(h_j)| T(h_j)} & \text{otherwise.} \end{cases} \quad (5)$$

To ensure that a chance is given to all heuristics to improve the solution, each heuristic is selected only once. The first stage of the algorithm terminates when all low-level heuristics have been considered.

In the second stage, HH3 selects the heuristics based on two factors: the heuristics' scores and their tabu status. While heuristics are declared tabu in the same way they are in HH1 (i.e., whenever a heuristic is not able to improve the current solution, it is made tabu to temporarily exclude it from the selection pool), the strategy for managing the tabu list is different. Rather than using a tabu list of fixed length and emptying the tabu list whenever a solution worse than the current one is obtained as HH1 does, HH3 chooses random tabu tenures (γ) generated in the interval $[\Gamma_{\min}, \Gamma_{\max}]$ at each iteration and empties the tabu list only if all heuristics are tabu. This strategy allows a better exploration of the heuristic search space as it significantly reduces the probability of repeatedly choosing the same heuristics during the search; for example, heuristics that have a high score as a result of good performance at the early stages of the search but have recently exhibited a poor performance. Recall that this is one of the weaknesses of HH1 and HH2. In addition, unlike HH1 and HH2, where the choice of the heuristic to be applied at a given iteration is done in a greedy manner; that is, the (non tabu) heuristic having the highest

score is selected, HH3 uses a roulette-wheel strategy. It associates with each non tabu heuristic h_j a selection probability p_j calculated by dividing its score by the total score of the non tabu heuristics ($p_j = \frac{S(h_j)}{\sum_{k:\, h_k \text{ non tabu}} S(h_k)}$). It then randomly selects a heuristic based on these probabilities.

Another noticeable difference between HH3 and the two hyper-heuristics described in the previous sections is the frequency at which the scores are updated and the score update scheme. In HH3, the scores are updated every ζ iterations, accounting for the average performance of the heuristics during these iterations instead of updating them at each iteration. For this purpose, the algorithm maintains for each heuristic h_j two measures, $\pi_1(h_j)$ and $\pi_2(h_j)$. Such measures are initially equal to 0. Whenever h_j is applied, either $\pi_1(h_j)$ or $\pi_2(h_j)$ is increased. The increase is related to the obtained change in the objective function value ($\Delta f(h_j)$). Specifically, if $\Delta f(h_j) > 0$ (i.e., if h_j improves the current solution), $\frac{\Delta f(h_j)}{T(h_j)}$ is added to $\pi_1(h_j)$, $T(h_j)$ being the time h_j required; if $\Delta f(h_j) < 0$ (i.e., if h_j deteriorates the current solution), $\frac{1}{|\Delta f(h_j)| T(h_j)}$ is added to $\pi_2(h_j)$; and if $\Delta f(h_j) = 0$ (i.e., if h_j cannot modify the value of the current solution), both $\pi_1(h_j)$ and $\pi_2(h_j)$ remain unchanged. When ζ iterations of the algorithm are completed, the scores are recalculated as follows:

$$S(h_j) := \begin{cases} S(h_j) & \text{if } \eta(h_j) = 0, \\ (1-\alpha) S(h_j) + \alpha \dfrac{\beta \pi_1(h_j) + (1-\beta)\pi_2(h_j)}{\eta(h_j)} & \text{otherwise} \end{cases} \tag{6}$$

where, $\eta(h_j)$ is the number of times heuristic h_j has been selected in the last ζ iterations, and α and β are two weight adjustment parameters in [0, 1]. Clearly, α defines the importance given to recent performance, while β defines the importance given to heuristics that were recently able to improve the solution. In this paper, the value of α is set to 0.7 to decrease the weight of previous performance (recall that one of the weaknesses of HH2 is that early performance might dominate recent performance). On the other hand, the parameter β is self-adjusted during the search process. The value of β is initially set equal to 0.5, and it is modified every ζ iterations based on whether or not a new incumbent solution has been found during the last segment of search: If a new solution better than the incumbent has been found during the last ζ iterations, β is increased to 1; otherwise, it is decreased to $\max(\beta - 0.1, 0)$. This way of proceeding ensures that emphasis is put on intensification if a new incumbent is found, while focus is gradually shifted to diversification otherwise. Indeed, when the value of β is increased, the score of heuristics that were recently able to improve the solution is also increased, so such heuristics are more likely to be selected, leading to an intensification of the search. As the value of β decreases, the effect is the opposite, leading to a diversification of the search.

Once the value of β is updated, the previous scores as well as π_1 and π_2 are normalized to a value in the interval [1, 100], and the scores are updated using Eq. (6). Afterwards, $\pi_1(h_j)$, $\pi_2(h_j)$ and $\eta(h_j)$ are reset to zero for each h_j, the tabu list is emptied, and a new segment of search is initiated for another ζ iterations. This process is repeated until the stopping criterion is met. An outline of the pseudocode of HH3 is given in Algorithm 1.

3.4. Low-level heuristics

To produce new solutions, the three hyper-heuristics described in the previous section use 27 simple perturbative low-level heuristics, each of which examines a subset of one of the following four neighborhoods, previously proposed in the literature:

Algorithm 1 HH3.

Initialization

Generate an initial solution x
Set $x^* := x$ (best-known or incumbent solution)

Improvement

Stage I: Giving a chance to all heuristics
Add all heuristics to a list (list of not yet selected heuristics)
while the number of heuristics in the list > 0 **do**
  Choose a heuristic h_j randomly from the list
  Generate a new solution x' from x using h_j
  if x' is better than x^* **then**
    Set $x^* := x'$
  end if
  Calculate the initial score of h_j using Eq. (5)
  Remove h_j from the list
  Set $x := x'$
end while

Stage II: Selecting heuristics based on their score and tabu status
Set $iter := 1$, $\alpha := 0.7$, $\beta := 0.5$, and $newIncumbent :=$ **false**
for each heuristic h_j **do**
  Set $\pi_1(h_j) := 0$, $\pi_2(h_j) := 0$, and $\eta(h_j) := 0$
end for
while stopping criterion not met **do**
  Choose a heuristic h_j from among the heuristics that are not tabu (using roulette-wheel selection based on scores)
  Set $\eta(h_j) := \eta(h_j) + 1$
  Generate a new solution x' from x using h_j. Let $\Delta f(h_j)$ and $T(h_j)$ be the obtained change in the objective function value and the solution time required, respectively
  if x' is better than x^* **then**
    Set $x^* := x'$ and $newIncumbent :=$ **true**
  end if
  if $\Delta f(h_j) > 0$ **then**
    Set $\pi_1(h_j) := \pi_1(h_j) + \frac{\Delta f(h_j)}{T(h_j)}$
  else
    if $\Delta f(h_j) < 0$ **then**
      Set $\pi_2(h_j) := \pi_2(h_j) + \frac{1}{|\Delta f(h_j)| T(h_j)}$
    end if
    Generate a random number γ in $[\Gamma_{\min}, \Gamma_{\max}]$
    Make h_j tabu for γ iterations
  end if
  Set $x := x'$
  if $iter < \zeta$ **then**
    Set $iter := iter + 1$
  else
    if $newIncumbent =$ **true then**
      Set $\beta := 1$
    else
      Set $\beta := \max(\beta - 0.1, 0)$
    end if
    for each heuristic h_j **do**
      Normalize $S(h_j)$, $\pi_1(h_j)$, and $\pi_2(h_j)$
      Update the score of h_j using equation (6)
      Set $\pi_1(h_j) := 0$, $\pi_2(h_j) := 0$, and $\eta(h_j) := 0$
    end for
    Revoke the tabu status of all heuristics
    Set $iter := 1$ and $newIncumbent :=$ **false**
  end if
end while

return x^*.

- *Single-Shift* (Lamghari and Dimitrakopoulos, 2012): This neighborhood involves moving a single block from its current period t to another period $t' \neq t$.
- *Swap* (Lamghari et al., 2014): This neighborhood allows exchanging blocks between periods and can be seen as two simultaneous changes associated with the *Single-Shift* neighborhood. More specifically, it involves moving two blocks: block i from its current period t to another period $t' \neq t$ and another block i' from t' to t.
- *Shift-Before* (Lamghari et al., 2014): Here multiple blocks; , a block i and its predecessors mined in the same period, are moved from their current period $t \neq 1$ to the preceding period $(t-1)$. Recall that a predecessor of block i is a block that must be extracted to have access to i. In what follows, we will refer to the set formed by a block i and its predecessors mined in the same period as the *inverted cone* whose base is i.
- *Shift-After* (Lamghari et al., 2014): This neighborhood also allows moving multiple blocks. A block and its successors mined in the same period are moved from their current period t to the next period $(t+1)$. Note that j is a successor of i if and only if i is a predecessor of j. In what follows, we will refer to the set formed by a block i and its successors mined in the same period as the *cone* whose apex is i.

Not only do the proposed low-level heuristics examine different subsets of the four neighborhoods described above (different sub-neighborhoods), but they also use different functions to evaluate solutions in these sub-neighborhoods and different strategies to select one of these solutions to become the new current solution. Generating only subsets of the neighborhoods and using different evaluation functions and selection strategies serves three main purposes: to reduce the computational effort, to drive the search to interesting parts of the search space, and to ensure intensification and diversification.

To simplify the discussion, the heuristics are classified in three different groups. The first group contains heuristics that select block(s) from a random period and move them either earlier or later. The second group contains heuristics that select block(s) from a specific period, as opposed to a random period, and move them either earlier or later. Clearly, heuristics in these two groups do not allow for any changes in the set of extracted blocks. Heuristics in the third group allow for such changes by either dropping block(s) from the schedule or adding unscheduled block(s) to the schedule. Heuristics in the three groups consider only moves that yield a feasible solution. Below, additional details about the heuristics are provided.

Heuristics that choose blocks from a random period

h_1: This heuristic explores a subset of the *Single-Shift* neighborhood. It starts by randomly selecting a period t. It then identifies blocks currently scheduled in t that can be moved either earlier or later without violating the hard constraints. Moves are evaluated based on the change produced in the objective function value, and the best move is selected.

h_2: Similar to h_1 except that it considers only blocks that can be moved earlier. Furthermore, the evaluation of a move is based on the total economic value of the block and all its successors. This evaluation function can be seen as a measure of attractiveness used to identify potential blocks that if advanced will entail advancing high-grade ore blocks. Thus, h_2 explores a smaller subset of the *Single-Shift* neighborhood compared to h_1, and to orient the search, it does not use the objective function of the problem but an auxiliary function; namely, the value of the block and all its successors.

h_3: Unlike the two previous heuristics, which move a single block, this heuristic simultaneously moves multiple blocks; more specifically, it advances the extraction of an inverted cone whose base block is currently scheduled in t, from t to $t-1$. Only inverted cones having a positive economic value are considered, and the heuristic selects the one with the highest unit economic value (i.e., highest value per unit of weight). Thus, h_3 explores a subset of the *Shift-Before* neighborhood, and to orient the search, it uses another auxiliary function, the unit economic value.

h_4: Similar to h_3 except that all inverted cones are considered, among which one is chosen at random. This heuristic induces some form of diversification.

h_5: Similar to h_4, but the moves are evaluated based on the change produced in the objective function value, and the best move is selected.

h_6: This heuristic also allows simultaneously moving multiple blocks. However, rather than changing the period of an inverted cone from t to $t-1$, it changes the period of a cone whose apex is currently scheduled at t, from t to $t+1$, which means that it explores a subset of the *Shift-After* neighborhood. Only cones having a non-positive economic value are considered, and the heuristic selects the one with the smallest unit economic value.

h_7: Similar to h_6 except that all cones are considered, among which one is chosen at random. Like h_4, h_7 is used for diversification purposes.

h_8: Similar to h_7, but the moves are evaluated based on the change produced in the objective function value, and the best move is selected.

h_9: Similar to h_8, but the blocks to be moved are not necessarily related to each other via precedence, and they are moved sequentially. At each iteration, a single block is selected and moved from t to $t+1$, and this process is repeated as long as there is improvement in the objective function value. Therefore, h_9 explores a subset of the *Single-Shift* neighborhood using a best-improvement descent. It acts as a trimming mechanism to free some capacity in t for hopefully more interesting blocks.

h_{10}: This heuristic also moves blocks that are not related to each other via precedence. It exchanges blocks i and i' currently scheduled in periods t and $t+1$, respectively. That means that h_{10} explores a subset of the *Swap* neighborhood. Moves are evaluated based on the change produced in the objective function value and selected using a first-improving strategy.

Heuristics that choose blocks from a specific period

Unlike the previous heuristics ($h_1 - h_{10}$), the following six heuristics do not select blocks from a random period but rather from a specific period in an attempt to reduce either soft constraints violations or the tightness of the hard constraints. The first heuristic explores a subset of the *Single-Shift* neighborhood, while the remaining heuristics explore subsets of the *Shift-After* and/or *Shift-Before* neighborhoods. The way these subsets are chosen and explored is explained below.

h_{11}: This heuristic first identifies the period t with the highest penalty cost (incurred by violation of the soft constraints). It moves a single block currently mined in t either later or earlier. The moves are evaluated based on the change produced in the objective function value, and the best move is selected.

h_{12}: This heuristic first identifies the period with the highest mining utilization, t (the mining utilization is calculated as the total amount mined in period t in the current solution divided by the mining capacity). It then determines the adjacent period with the most residual capacity, t'. If $t' = t-1$, then the heuristic selects an inverted cone whose base is currently scheduled in t; otherwise (i.e., if $t' = t+1$), it selects a cone whose apex

is currently scheduled in t. The (inverted) cone is selected at random and its period is changed from t to t'.

h_{13}: Similar to h_{12} except for the way t is selected. Here t is selected among the periods with high penalty cost, not among those with high mining utilization. Also, t is not chosen in a greedy manner but using roulette wheel selection.

h_{14}: Similar to h_{13}, but the moves are evaluated based on the change produced in the objective function value, and the best move is selected.

h_{15}: Unlike the previous heuristics where t is chosen first, this heuristic starts by selecting t'. The choice of t' is based on a probability distribution biased towards the periods with the most residual capacity. The heuristic then examines all the candidates that could be moved to t' (cones and inverted cones currently mined in periods adjacent to t') and evaluates them using the objective function to choose the best move to be performed.

h_{16}: Similar to h_{15} except that the (inverted) cone to move from t to t' is randomly selected.

Heuristics that modify the set of scheduled blocks

All heuristics described above change the periods of scheduled blocks. However, they do not allow for any changes in the set of extracted blocks. Such changes are obtained with the heuristics presented below.

h_{17}: This heuristic starts by identifying blocks that are currently unscheduled then adds one of them to the schedule. Moves are evaluated based on the change produced in the objective function value, and the best one is selected.

h_{18}: Similar to h_{17}, but only improving moves are considered.

h_{19}: Similar to h_{17} but more aggressive in the sense that it induces greater solution changes than h_{17}. It adds inverted cones (a block and its predecessors) to the schedule rather than a single block.

h_{20}: Similar to h_{19}, but only improving moves are considered.

h_{21}: This heuristic considers only inverted cones having a positive economic value, among which it selects the one with the highest unit economic value.

The following heuristics drop block(s) from the schedule.

h_{22}: This heuristic drops a single block from the schedule. Moves are evaluated based on the change produced in the objective function value. Only improving moves are considered, and the best one is selected.

h_{23}: Similar to h_{22}, but it does not terminate after dropping a single block. The process is repeated until no improvement is possible.

h_{24}: Like h_{23}, this heuristic also drops multiple blocks from the schedule. However, these blocks are related to each other by precedence. It removes a cone whose apex is currently scheduled in the last period of the horizon (a block and its successors). Moves are evaluated based on the change produced in the objective function value, and the best one is selected.

h_{25}: Similar to h_{24} except that only improving moves are considered.

h_{26}: Similar to h_{24} but evaluates moves based on the unit economic value of the cones. Only cones having a non-positive value are considered, and the one with the smallest value is selected.

h_{27}: Similar to h_{22}, but also simultaneously drops a single block from the schedule. In other words, it exchanges an unscheduled block with a scheduled block. The neighborhood is explored using a first-improving strategy.

4. Numerical results

To assess the efficiency and the robustness of the three hyper-heuristic approaches described in Section 3, numerical experiments have been performed on five benchmark test sets for the two strategic mine planning problems considered in this paper (SMPS and SMPS+). These benchmark datasets include a total of 43 instances of different sizes and characteristics. They are briefly described below and summarized in Table 1.

- The first set of benchmark instances, L1, was introduced by Lamghari and Dimitrakopoulos (2012). It consists of 10 small- to large-size instances from a copper and a gold deposit that all contain one processor and one waste dump; that is, 10 instances for the SMPS (c.f. Section 2.1). Each period is one year long, and it is assumed that the production capacities are identical in all periods. For each instance, it is possible to extract a total of $\lceil 1.20\frac{\sum_{i=1}^{N} w_i}{T}\rceil$ tonnes per year,[5] of which the waste is sent to the waste dump having an unlimited capacity, and the ore is sent to a processor p having a capacity of $\lceil 1.05\frac{\sum_{i=1}^{N}\sum_{s=1}^{S}\theta_{ips}w_i}{ST}\rceil$;[6] that is, $\frac{\text{Expected amount of ore in the deposit}}{\text{Number of periods}} + 5\%$. Lower bounds on mining and processing are set to $\lceil 0.80\frac{\sum_{i=1}^{N} w_i}{T}\rceil$ and $\lceil 0.95\frac{\sum_{i=1}^{N}\sum_{s=1}^{S}\theta_{ips}w_i}{ST}\rceil$, respectively. Finally, lower and upper bounds on metal production are set to $\lceil 0.95\frac{\sum_{i=1}^{N}\sum_{s=1}^{S}\theta_{ips}m_{is}}{ST}\rceil$ and $\lceil 1.05\frac{\sum_{i=1}^{N}\sum_{s=1}^{S}\theta_{ips}m_{is}}{ST}\rceil$;[7] that is, $\frac{\text{Expected amount of metal in the deposit}}{\text{Number of periods}} \pm 5\%$.
- The following four benchmark test sets (S1-S4) include 33 instances for the SMPS+ (c.f. Section 2.2), in which several destinations for the extracted material are considered instead of only two.[8] The first three sets are those used by Lamghari and Dimitrakopoulos (2016a), while the fourth one is a new dataset that contains larger instances. Details about these sets are as follows.
  - The set S1 consists of 10 small- to large-size instances from a copper and a gold deposit that all contain one processor, one stockpile, and one waste dump. These instances are the same as those in L1, except that a stockpile has been added. Moreover, lower bounds on mining and processing are not imposed and there are no requirements for metal production levels.
  - The set S2 consists of three instances representing three different real deposits: two copper deposits and a gold deposit. The size of these instances is larger than those in the previous benchmark set. Furthermore, the instances in this set contain two processors, two stockpiles, and a waste dump (as opposed to one processor, one stockpile, and one waste dump in S1). Finally, the processing capacities are set to a value 5% smaller than for the instances in S1 so as to make the satisfaction of the processing constraints more difficult and thus force the use of the stockpiles.
  - The set S3 consists of 10 medium-size instances from a copper deposit with two processors, two stockpiles, and a waste dump. While the processing capacities are similar to those in the previous set, S2, the mining capacities are much

[5] N and T represent the number of blocks in the deposit and the number of periods over which the blocks are being scheduled, respectively. w_i denotes the tonnage of block i.

[6] S is the number of scenarios used to model metal uncertainty and θ_{ips} is a parameter equal to 1 if block i is classified as an ore block under scenario s (can be sent to the processor p) and 0 otherwise.

[7] m_{is} is the metal content of block i under scenario s

[8] Recall that the SMPS instances in the previous set (L1) consider only two destinations for the extracted material: a processor and a waste dump.

Table 1
Overview of the instances in the five benchmark datasets.

Dataset	L1	S1	S2	S3	S4
Number of instances	10	10	3	10	10
Number of blocks (N)	$4,273 \leq N \leq 40,762$	$4,273 \leq N \leq 40,762$	$14,118 \leq N \leq 48,821$	$21,965 \leq N \leq 22,720$	40,900
Number of periods (T)	$3 \leq T \leq 13$	$3 \leq T \leq 13$	$6 \leq T \leq 16$	$11 \leq T \leq 12$	21
Number of scenarios (S)	20	20	$20 \leq S \leq 25$	20	20
Number of processors (P)	1	1	2	2	2
Number of stockpiles	0	1	2	2	2
Number of waste dumps	1	1	1	1	1
Metal type	Copper and Gold	Copper and Gold	Copper and Gold	Copper	Copper
Block weight (w_i) in tonnes	$5,625 \leq w_i \leq 10,800$	$5,625 \leq w_i \leq 10,800$	$5625 \leq w_i \leq 10,800$	10,000	10,000
Lower bound on mining	$\lceil 0.80 \frac{\sum_{i=1}^{N} w_i}{T} \rceil$	0	0	0	0
Upper bound on mining	$\lceil 1.20 \frac{\sum_{i=1}^{N} w_i}{T} \rceil$	$\lceil 1.20 \frac{\sum_{i=1}^{N} w_i}{T} \rceil$	$\lceil 1.20 \frac{\sum_{i=1}^{N} w_i}{T} \rceil$	$\lceil \frac{\sum_{i=1}^{N} w_i}{T} \rceil$	$\lceil \frac{\sum_{i=1}^{N} w_i}{T} \rceil$
Lower bound on processing	$\lceil 0.95 \frac{\sum_{i=1}^{N} \sum_{s=1}^{S} \theta_{ips} w_i}{ST} \rceil$	0	0	0	0
Upper bound on processing	$\lceil 1.05 \frac{\sum_{i=1}^{N} \sum_{s=1}^{S} \theta_{ips} w_i}{ST} \rceil$	$\lceil 1.05 \frac{\sum_{i=1}^{N} \sum_{s=1}^{S} \theta_{ips} w_i}{ST} \rceil$	$\lceil \frac{\sum_{i=1}^{N} \sum_{s=1}^{S} \theta_{ips} w_i}{ST} \rceil$	$\lceil \frac{\sum_{i=1}^{N} \sum_{s=1}^{S} \theta_{ips} w_i}{ST} \rceil$	$\lceil \frac{\sum_{i=1}^{N} \sum_{s=1}^{S} \theta_{ips} w_i}{ST} \rceil$
Lower bound on metal production	$\lceil 0.95 \frac{\sum_{i=1}^{N} \sum_{s=1}^{S} \theta_{is} m_{is}}{ST} \rceil$	0	0	0	0
Upper bound on metal production	$\lceil 1.05 \frac{\sum_{i=1}^{N} \sum_{s=1}^{S} \theta_{is} m_{is}}{ST} \rceil$	∞	∞	∞	∞

tighter here. They are set to a value 20% smaller than for the instances in S1 and S2 (i.e., they are set to $\lceil \frac{\sum_{i=1}^{N} w_i}{T} \rceil$).

- The last set of instances, S4, consists also of 10 instances from a copper deposit with two processors, two stockpiles, and a waste dump. They are similar to those in S3, except that they are larger.

All algorithms were coded in C++ and the experiments were run on an Intel(R) Xeon(R) CPU X5675 computer (3.07 GHz) with 96 Go of RAM running under Linux.

4.1. Results on the SMPS instances (dataset L1)

In this section, we examine how the three hyper-heuristic approaches (HH1, HH2, and HH3), described in Section 3, perform on the ten benchmark instances in the set L1. We compare the hyper-heuristics to each other and also to another problem-specific method tailored for the SMPS; namely, the tabu search heuristic (TS) proposed in Lamghari and Dimitrakopoulos (2012), which has previously achieved the best solution quality for the instances in L1.

All four methods (HH1, HH2, HH3, and TS) contain some user-controlled parameters. Recall that the three hyper-heuristics terminate when a specified number of iterations, $\Upsilon_{\max}$, has elapsed. We varied the value of $\Upsilon_{\max}$ and analyzed the tradeoff between solution quality and solution time. The best results were obtained with the value $\Upsilon_{\max} = 1000 + 0.5N$, N being the number of blocks. Consequently, this value is used in all further experiments. HH1 does not have any other parameters, while HH2 and HH3 have each three more parameters. For HH2, we used the same parameter settings as in the original paper by Drake et al. (2012); that is, 0.99, 0.01, and 0.01 for the parameters $\phi_{\max}$, $\delta_{\min}$, and κ, respectively. To tune the parameters of HH3, we have chosen five instances at random, run tests using different values for $\Gamma_{\min}$, $\Gamma_{\max}$, and ζ.[9] The best values found for these parameters are $\Gamma_{\min} = 0.5H$, $\Gamma_{\max} = H$, and $\zeta = 5H$, H being the number of low-level heuristics (27 in this paper). These settings are thus used in all further experiments. Finally, for TS, we used the same parameter settings as in the original paper (Lamghari and Dimitrakopoulos, 2012).

[9] Recall that $[\Gamma_{\min}, \Gamma_{\max}]$ is used to choose the tabu tenure; that is, the number of iterations during which a heuristic that has not performed well recently is made tabu, while ζ limits the number of iterations performed with the same score values.

All four methods start with a random initial solution generated using the heuristic in Lamghari and Dimitrakopoulos (2012) and also make other random choices during the improvement phase. Hence, each of them was applied to each instance 30 times. The results are summarized inTables 2–4. Table 2 presents statistics regarding the values of the best solutions found by each method for each of the test instances (Z^*), namely the average solution value (Avg.) and its standard deviation (St. Dev.). Tables 3 and 4 provide statistics regarding the optimality gaps and the solution times, respectively. The formula used to calculate the gaps is $\%Gap = \frac{Z^* - Z_{LR}}{Z_{LR}} \times 100$, where Z_{LR} is the linear relaxation optimal value, computed using CPLEX 12.2. The time required by CPLEX is given in the last column of Table 4 (column LR). A dash ("-") indicates that CPLEX was not able to solve the linear relaxation of the instance within the time limit, set to four weeks. The best results obtained for each instance are indicated in bold. The name of the instances and their size (number of blocks (N), number of periods (T), and number of scenarios modelling geological uncertainty (S)) are given in the first four columns of each table.

The results show that among the three hyper-heuristic methods, HH3 is the best in terms of solution quality. On average, the optimality gap for HH3 is 1.40% as opposed to 5.68% and 20.92% for HH1 and HH2, respectively. Even though HH3 is outperformed by the problem-specific method, TS, on some instances (L1-C4 and L1-C5), overall, it finds better quality solutions than does TS (on average, the optimality gap for TS is 1.70%). It is worth noting that although the differences in the gap appear small, for the context of the strategic mine planning problems studied in this paper, they are meaningful because they represent millions of dollars, as can be seen from the values of Z^* in Table 2. With respect to solution time, HH3 is less time consuming than TS for copper instances (L1-C1 to L1-C5), while the opposite is true for gold instances (L1-G1 to L1-G5). On average, HH3 is slightly better than TS (52.30 min versus 64.65 min). Both TS and HH3 are more time consuming than HH1 and HH2, whose average CPU are 36.45 and 30.52 min, respectively. All four methods require significantly less time than the time required by CPLEX to solve the linear relaxation of the instances. In addition, they are less sensitive to the size of the instances compared to CPLEX.

Another interesting feature that can be observed in Tables 2–4 is related to the standard deviation of the solution values, optimality gaps, and solution times (St. Dev.). The results suggest that TS and HH3 achieve the smaller deviations. This means that their per-

Table 2
Result summary of solution values (in dollars) - instances in the benchmark dataset L1..

Name	*N*	*T*	*S*	Z^* ($)							
				TS		HH1		HH2		HH3	
				Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.
L1-C1	4273	3	20	162,256,000	26,501	161,535,000	2,663,613	157,621,000	12,020,986	**162,410,000**	25,341
L1-C2	7141	4	20	194,469,000	108,576	191,603,000	10,057,748	189,089,000	12,634,045	**195,047,000**	141,983
L1-C3	12,627	7	20	**220,535,000**	480,192	207,897,000	20,571,856	186,129,000	38,084,152	220,463,000	984,132
L1-C4	20,626	10	20	**237,980,000**	836,537	195,994,000	50,954,399	128,446,000	71,997,376	235,160,000	1,693,122
L1-C5	26,021	13	20	**221,326,000**	1,142,693	153,675,000	69,819,128	96,968,000	79,201,659	217,000,000	2,861,557
L1-G1	18,821	5	20	404,673,000	163,686	401,917,000	10,756,738	328,560,000	103,065,978	**407,027,000**	190,413
L1-G2	23,901	7	20	434,273,000	281,340	429,536,000	13,100,921	324,647,000	137,244,411	**438,047,000**	222,009
L1-G3	30,013	8	20	468,095,000	414,450	460,163,000	14,961,717	393,538,000	98,980,867	**473,027,000**	340,525
L1-G4	34,981	9	20	474,361,000	616,784	455,281,000	34,636,817	331,171,000	160,159,746	**477,789,000**	2,899,283
L1-G5	40,762	11	20	449,591,000	1,336,320	420,278,000	53,870,050	241,897,000	182,354,677	**450,090,000**	4,600,160

Table 3
Result summary of optimality gaps - instances in the benchmark dataset L1..

Name	*N*	*T*	*S*	*%Gap*							
				TS		HH1		HH2		HH3	
				Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.
L1-C1	4273	3	20	0.24	0.02	0.68	1.64	3.09	7.39	**0.14**	0.02
L1-C2	7141	4	20	0.69	0.06	2.15	5.14	3.44	6.45	**0.40**	0.07
L1-C3	12,627	7	20	**1.86**	0.21	7.48	9.16	17.17	16.95	1.89	0.44
L1-C4	20,626	10	20	**3.80**	0.34	20.77	20.63	48.08	29.07	4.94	0.71
L1-C5	26,021	13	20	-	-	-	-	-	-	-	-
L1-G1	18,821	5	20	1.16	0.04	1.83	2.63	19.75	25.17	**0.59**	0.05
L1-G2	23,901	7	20	1.71	0.06	2.79	2.97	26.52	31.06	**0.86**	0.05
L1-G3	30,013	8	20	1.97	0.09	3.63	3.13	17.58	20.73	**0.94**	0.07
L1-G4	34,981	9	20	2.19	0.13	6.13	7.14	31.72	33.02	**1.49**	0.60
L1-G5	40,762	11	20	-	-	-	-	-	-	-	-

Table 4
Result summary of computational times (in minutes) - instances in the benchmark dataset L1..

Name	*N*	*T*	*S*	*Time(minutes)*								
				TS		HH1		HH2		HH3		LR
				Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	
L1-C1	4273	3	20	4.28	0.00	**1.10**	0.73	1.13	0.85	1.74	0.12	8.97
L1-C2	7141	4	20	9.53	0.00	1.91	1.10	**1.81**	1.12	3.89	0.47	73.64
L1-C3	12,627	7	20	29.47	0.00	**3.62**	1.91	4.25	3.64	8.19	0.82	1457.05
L1-C4	20,626	10	20	68.77	0.00	**7.73**	5.80	7.80	7.17	16.41	1.67	12115.63
L1-C5	26,021	13	20	112.77	0.00	**9.02**	5.19	15.25	9.52	22.64	2.29	-
L1-G1	18,821	5	20	31.38	0.00	49.97	21.58	**41.03**	37.31	65.63	6.13	855.50
L1-G2	23,901	7	20	55.78	0.00	59.72	22.09	**37.02**	28.27	75.76	5.18	3786.73
L1-G3	30,013	8	20	80.05	0.00	83.71	40.00	**65.00**	44.29	106.82	11.81	7902.27
L1-G4	34,981	9	20	104.95	0.00	90.03	36.92	**71.52**	47.11	112.60	6.78	15230.50
L1-G5	40,762	11	20	149.47	0.00	**57.70**	35.41	60.43	51.66	109.33	9.17	-

formance is less affected by particular characteristics of the initial solutions (recall that the initial solutions are generated randomly). HH1 and HH2, on the other hand, seem to be much less robust in terms of solution value and solution time variation.

4.2. Results on the SMPS+ instances (datasets $S1 - S4$)

In this section, we report results obtained on the SMPS+ instances. Again, we compare the three hyper-heuristic methods to each other and to a heuristic specifically designed for the problem addressed here. Although different methods have been developed for the SMPS+, we have chosen to restrict our comparisons to the diversified local search (DLS) presented in Lamghari and Dimitrakopoulos (2016a), which combines a variable neighborhood descent heuristic and a very large neighborhood search heuristic based on network flow techniques, as it is the most recent and best problem-specific heuristic proposed in the literature. Similar to the previous tests, each method (HH1, HH2, HH3, and DLS) was applied to each instance 30 times, each time starting from a different random initial solution. The parameters for HH1, HH2, and HH3 were set as in Section 4.1, while those for DLS were set as in Lamghari and Dimitrakopoulos (2016a).

4.2.1. Results for the first set of benchmark instances (dataset S1)

Recall that the SMPS+ instances are divided into four datasets. Tables 5–7 show the results for the first dataset, S1, which contains ten instances. Similar to the previous section, the characteristics of each instance are given in the first four columns of the tables. For each instance and each method, we report the following: the average value of the best solutions found and its standard deviation (Z^*), the average optimality gap (*%Gap*) as defined above and its standard deviation, and the average solution time in minutes (Time) and its standard deviation. We use boldface symbols to indicate the best results obtained for each instance. The linear relaxation optimal values, used to calculate the gaps, have been

Table 5
Result summary of solution values - instances in the benchmark dataset S1..

Name	N	T	S	Z^* ($)							
				DLS		HH1		HH2		HH3	
				Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.
S1-C1	4273	3	20	**165,638,000**	20,719	146,682,000	28,535,996	131,467,000	2,439,761	165,618,000	41,477
S1-C2	7141	4	20	199,358,000	100,660	162,466,000	39,332,702	143,314,000	36,436,056	**199,393,000**	101,650
S1-C3	12,627	7	20	229,183,000	156,615	175,930,000	52,922,646	123,343,000	38,044,780	**229,408,000**	199,579
S1-C4	20,626	10	20	251,355,000	263,697	139,941,000	62,990,444	109,042,000	47,737,213	**251,410,000**	565,524
S1-C5	26,021	13	20	242,366,000	435,232	153,237,000	72,390,791	101,945,000	64,016,211	**244,827,000**	460,462
S1-G1	18,821	5	20	411,085,000	68,543	301,233,000	114,092,674	237,009,000	122,237,304	**411,138,000**	123,600
S1-G2	23,901	7	20	443,449,000	120,091	299,115,000	156,632,340	238,013,000	151,333,545	**443,548,000**	165,700
S1-G3	30,013	8	20	479,205,000	132,878	344,217,000	164,367,556	187,633,000	127,390,329	**479,345,000**	170,186
S1-G4	34,981	9	20	487,199,000	123,886	379,611,000	151,088,770	191,437,000	154,552,595	**487,212,000**	251,682
S1-G5	40,762	11	20	465,884,000	211,491	288,697,000	157,886,556	230,382,000	148,649,780	**466,386,000**	337,414

Table 6
Result summary of optimality gaps - instances in the benchmark dataset S1..

Name	N	T	S	%Gap							
				DLS		HH1		HH2		HH3	
				Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.
S1-C1	4273	3	20	**0.04**	0.01	11.48	17.22	20.66	18.04	0.05	0.02
S1-C2	7141	4	20	0.11	0.05	18.60	19.71	28.19	18.26	**0.10**	0.05
S1-C3	12,627	7	20	0.36	0.07	23.51	23.01	46.37	16.54	**0.26**	0.09
S1-C4	20,626	10	20	1.13	0.10	44.95	24.78	57.11	18.78	**1.11**	0.22
S1-C5	26,021	13	20	1.77	0.18	37.89	29.34	58.68	25.95	**0.77**	0.19
S1-G1	18,821	5	20	0.31	0.02	26.95	27.67	42.52	29.64	**0.30**	0.03
S1-G2	23,901	7	20	0.48	0.03	32.87	35.15	46.58	33.96	**0.45**	0.04
S1-G3	30,013	8	20	0.50	0.03	28.53	34.13	61.04	26.45	**0.47**	0.04
S1-G4	34,981	9	20	0.52	0.03	22.49	30.85	60.91	31.56	**0.52**	0.05
S1-G5	40,762	11	20	0.82	0.05	38.54	33.61	50.96	31.65	**0.71**	0.07

Table 7
Result summary of computational times (in minutes) - instances in the benchmark dataset S1..

Name	N	T	S	Time(minutes)								
				DLS		HH1		HH2		HH3		LR
				Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	
S1-C1	4273	3	20	2.18	0.19	1.04	0.83	**0.72**	0.68	2.08	0.11	0.23
S1-C2	7141	4	20	6.38	0.76	2.27	1.94	**1.31**	1.40	5.27	0.57	5.68
S1-C3	12,627	7	20	34.89	9.27	5.56	3.11	**2.50**	3.04	9.30	0.67	139.01
S1-C4	20,626	10	20	128.32	34.96	11.71	10.19	**7.08**	9.79	23.03	1.67	1540.61
S1-C5	26,021	13	20	222.85	62.66	18.88	14.49	**17.96**	29.36	41.09	9.51	3470.63
S1-G1	18,821	5	20	50.93	10.22	37.38	23.75	**19.49**	18.37	66.66	7.57	187.77
S1-G2	23,901	7	20	90.03	17.73	52.19	36.34	**32.48**	40.20	69.05	5.69	323.75
S1-G3	30,013	8	20	159.25	32.01	69.69	44.83	**49.33**	67.05	106.42	8.06	1947.79
S1-G4	34,981	9	20	255.21	54.60	103.71	74.79	**91.07**	233.77	141.78	8.32	1179.05
S1-G5	40,762	11	20	366.29	100.55	97.46	73.66	**79.73**	137.60	168.18	31.70	2394.03

obtained using CPLEX 12.5. The time required by CPLEX is given in the last column of Table 7 (column LR).

The following observations can be derived from Tables 5–7:

- In terms of solution quality, again, HH3 outperforms HH1 and HH2. On average, the optimality gap for HH3 is 0.47% as opposed to 28.58% for HH1 and 47.30% for HH2. It is worth noting that the differences between the three hyper-heuristics are more pronounced for this dataset (S1) than they are for the previous dataset (L1).
- HH3 is outperformed by the problem-specific method, DLS, on only one instance out of 10. Moreover, the differences are not significant. In general, both methods find quite comparable solutions (average gaps for HH3 and DLS are 0.47% and 0.60%, respectively).
- HH3 requires less computational time than does DLS (on average, 63.29 min versus 131.63 min). Both methods are outperformed by HH1 (39.99 min, on average) and HH2 (30.17 min, on average).
- Among the four methods, HH2 is the one that requires the least computational time, but it was not successful in solving any of the ten instances. It can compete neither with the well-performing HH3 and DLS nor with HH1.
- Whether in terms of solution quality or in terms of solution time, DLS and HH3 are more robust and exhibit smaller variability compared to HH1 and HH2 (c.f. columns St. Dev.).
- As expected, all four methods outperform CPLEX in terms of solution time. The differences are more pronounced as the size of the instances increases.

4.2.2. Results for the second set of benchmark instances (dataset S2)

Tables 8–10 compare the results obtained for the second SMPS+ set of benchmark instances, S2. In these tables, a dash "-" indicates that CPLEX was not able to solve the linear relaxation of the problem within the time limit (four weeks), and thus neither the computational time of CPLEX nor the linear relaxation optimal value used to compute the gap are known. Recall that the instances in

Table 8
Result summary of solution values - instances in the benchmark dataset S2..

Name	*N*	*T*	*S*	Z^* ($)							
				DLS		HH1		HH2		HH3	
				Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.
S2-C1	14,118	6	25	27,296,100	180,931	(-316,780)	16,707,412	1,932,820	18,668,793	**27,773,400**	104,167
S2-C2	28,154	16	20	**225,575,000**	689,167	112,376,000	65,047,685	98,525,500	67,969,412	223,129,000	1,859,936
S2-G1	48,821	14	20	**471,814,000**	478,251	270,195,000	179,500,283	201,011,000	190,022,357	471,046,000	1,513,583

Table 9
Result summary of optimality gaps - instances in the benchmark dataset S2..

Name	*N*	*T*	*S*	*%Gap*							
				DLS		HH1		HH2		HH3	
				Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.
S2-C1	14,118	6	25	8.38	0.61	101.06	56.08	93.51	62.66	**6.77**	0.35
S2-C2	28,154	16	20	-	-	-	-	-	-	-	-
S2-G1	48,821	14	20	-	-	-	-	-	-	-	-

Table 10
Result summary of computational times (in minutes) - instances in the benchmark dataset S2..

Name	*N*	*T*	*S*	*Time(minutes)*								
				DLS		HH1		HH2		HH3		LR
				Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	
S2-C1	14,118	6	25	79.21	8.90	10.83	16.00	**8.40**	7.29	22.16	3.01	932.00
S2-C2	28,154	16	20	765.07	140.87	**14.86**	14.23	15.34	27.76	111.21	49.98	-
S2-G1	48,821	14	20	1246.14	175.10	305.30	898.26	**106.95**	142.95	305.25	127.54	-

S2 are larger than the instances in the first set (S1) and also more difficult to solve.

From Table 8, it appears that among the four methods, HH1 and HH2 are the ones that provide the worst results. Moreover, their performance is far inferior to the other methods and so is their robustness in terms of solution value and solution time variation. HH2 is not dominated by HH1 as was the case for the instances in the previous sets, L1 and S1. In particular, for the smallest instance, S2-C1, HH2 is significantly better than HH1. The solutions found by HH3 are again generally comparable to those obtained by the problem-specific method, DLS, but they are obtained much faster as can be seen from the results in Table 10. On average, the solution time is reduced by a factor of 5 when HH3 is used. DLS is also more time consuming than HH1 and HH2. On average, it runs 6 and 16 times longer than do HH1 and HH2, respectively. It should be noted that, the differences between DLS and the three hyper-heuristics are more pronounced for these larger and more computationally demanding instances in S2 than they are for the instances in the previous datasets. As was the case for the instances in the first datasets, HH2 is the fastest method. However, its short computational times do not compensate for the poor quality of the solutions it provides.

4.2.3. Results for the third set of benchmark instances (dataset S3)

We next compare the four methods on the ten instances of the third benchmark dataset, S3. As in the previous experiments, Tables 11–13 summarize the statistical data obtained from the performed experiments, showing the average solution value, the average optimality gap, and the average solution time (columns Avg.) over the 30 runs as well as their standard deviations (columns St. Dev.). Again, the best results obtained for each instance are indicated in bold.

Some of the observations made in the previous sections can be confirmed from the results in Tables 11–13. First, all four methods solve the problems in a very reasonable time, in the order of a few minutes to a few hours, which is significantly smaller than the 36.44 hours that CPLEX requires on average to solve the linear relaxation. Second, although HH1 is the fastest method, requiring 15.30 min on average, and although it is more effective than HH2, the quality of the solutions obtained with this hyper-heuristic is far from the quality obtained by HH3 and DLS. On average, the optimality gap for HH1 is 40.10% as opposed to 1.12% and 1.74% for HH3 and DLS, respectively. When comparing HH3 and DLS, it appears that not only does HH3 reach better solutions than does DLS, finding new best solutions for many instances in at least one of the 30 runs, but it also requires much less computational time. On average, the CPU time is reduced by a factor of 10.5 when HH3 is used. We can then conclude that, for the instances in this third SMPS+ benchmark dataset, regarding both solution quality and solution time, the new proposed hyper-heuristic HH3 seems to be the best choice.

4.2.4. Results for the fourth set of benchmark instances (dataset S4)

Finally, we compare the four methods on the instances of the fourth SMPS+ benchmark dataset, S4, which are larger than those in the third dataset, S3. Tables 14–16 summarize this comparison.

The results seem to be in line with what has been observed in the previous experiments. HH3 significantly outperforms DLS in terms of solution time. On average, the CPU time for DLS exceeds 37 hours, while the CPU time for HH3 is in general less than 5 hours. In terms of solution quality, for the ten tested instances, HH3 is slightly dominating. Both methods produce excellent solutions very close to optimality, with an average gap of 0.28% and 0.59%, respectively. These solutions are significantly better than those obtained by HH1 and HH2. The latter again gives very poor results and ranks last in terms of solution quality, but first in terms of solution time. In terms of robustness, DLS and HH3 present a similar behavior as in the previous experiments, exhibiting smaller solution value and solution time variation.

Table 11
Result summary of solution values - instances in the benchmark dataset S3..

Name	*N*	*T*	*S*	*Z** ($)							
				DLS		HH1		HH2		HH3	
				Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.
S3-C1	22,549	12	20	248,314,000	837,267	139,865,000	73,695,138	95,478,900	36,863,792	**249,518,000**	1,405,838
S3-C2	22,388	12	20	246,517,000	859,979	142,344,000	68,398,626	130,530,000	65,529,563	**248,031,000**	1,485,804
S3-C3	22,285	12	20	246,593,000	741,710	143,023,000	63,634,875	117,577,000	59,329,082	**248,308,000**	1,419,006
S3-C4	22,302	12	20	245,996,000	712,002	138,220,000	65,099,850	95,892,000	36,673,013	**247,759,000**	1,417,730
S3-C5	21,965	11	20	251,982,000	718,838	158,080,000	65,494,764	118,489,000	62,900,268	**253,424,000**	1,507,897
S3-C6	22,246	12	20	245,583,000	535,267	164,023,000	67,104,039	120,865,000	62,427,884	**247,727,000**	1,332,192
S3-C7	22,716	12	20	249,533,000	779,508	158,471,000	76,659,337	108,276,000	57,122,618	**250,552,000**	1,437,669
S3-C8	22,529	12	20	250,272,000	810,756	138,498,000	71,532,160	115,132,000	63,158,593	**251,641,000**	1,563,798
S3-C9	22,253	12	20	250,071,000	780,406	173,468,000	70,534,791	102,101,000	52,464,983	**251,485,000**	1,331,241
S3-C10	22,720	12	20	247,060,000	788,470	157,062,000	69,577,407	116,011,000	66,297,402	**249,106,000**	1,308,200

Table 12
Result summary of optimality gaps - instances in the benchmark dataset S3..

Name	*N*	*T*	*S*	*%Gap*							
				DLS		HH1		HH2		HH3	
				Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.
S3-C1	22,549	12	20	1.75	0.33	44.66	29.16	62.22	14.59	**1.27**	0.56
S3-C2	22,388	12	20	1.77	0.34	43.28	27.25	47.99	26.11	**1.17**	0.59
S3-C3	22,285	12	20	1.77	0.30	43.02	25.35	53.16	23.63	**1.08**	0.57
S3-C4	22,302	12	20	1.76	0.28	44.80	26.00	61.70	14.65	**1.06**	0.57
S3-C5	21,965	11	20	1.67	0.28	38.31	25.56	53.76	24.55	**1.11**	0.59
S3-C6	22,246	12	20	1.87	0.21	34.46	26.81	51.70	24.95	**1.01**	0.53
S3-C7	22,716	12	20	1.62	0.31	37.52	30.22	57.31	22.52	**1.22**	0.57
S3-C8	22,529	12	20	1.65	0.32	45.57	28.11	54.76	24.82	**1.11**	0.61
S3-C9	22,253	12	20	1.77	0.31	31.86	27.71	59.89	20.61	**1.21**	0.52
S3-C10	22,720	12	20	1.75	0.31	37.54	27.67	53.87	26.36	**0.94**	0.52

Table 13
Result summary of computational times (in minutes) - instances in the benchmark dataset S3..

Name	*N*	*T*	*S*	*Time(minutes)*								
				DLS		HH1		HH2		HH3		LR
				Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	
S3-C1	22,549	12	20	310.89	111.51	**21.29**	65.80	50.17	139.69	30.51	7.15	3586.66
S3-C2	22,388	12	20	299.76	97.83	**13.31**	14.30	15.46	32.51	29.97	6.53	2421.27
S3-C3	22,285	12	20	312.79	86.81	**13.58**	14.75	29.95	80.68	31.25	7.17	1420.44
S3-C4	22,302	12	20	334.31	101.76	24.98	57.40	**11.47**	30.01	30.41	8.45	3518.42
S3-C5	21,965	11	20	268.35	75.41	**14.34**	14.97	35.90	103.97	31.81	5.03	1464.72
S3-C6	22,246	12	20	284.28	69.51	12.85	12.59	**11.42**	9.81	34.72	8.31	1593.52
S3-C7	22,716	12	20	401.18	103.81	**10.20**	8.52	28.23	71.42	28.93	4.34	1986.17
S3-C8	22,529	12	20	348.02	98.07	**11.22**	9.03	15.10	31.49	29.56	5.23	1878.28
S3-C9	22,253	12	20	303.21	90.08	17.86	14.24	**16.54**	34.00	27.57	5.56	1807.55
S3-C10	22,720	12	20	338.66	98.84	**13.36**	11.76	26.93	69.15	31.29	7.01	2184.68

Table 14
Result summary of solution values - instances in the benchmark dataset S4..

Name	*N*	*T*	*S*	*Z** ($)							
				DLS		HH1		HH2		HH3	
				Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.
S4-C1	40,090	21	20	210,730,000	320,782	86,005,700	64,300,494	73,531,000	56,703,353	**211,173,000**	100,287
S4-C2	40,090	21	20	208,489,000	923,802	98,409,000	69,131,440	68,332,200	52,404,860	**209,752,000**	118,277
S4-C3	40,090	21	20	209,376,000	393,760	93,777,900	68,189,998	87,115,900	66,576,356	**210,215,000**	73,937
S4-C4	40,090	21	20	208,569,000	432,349	101,288,000	70,612,871	70,077,900	51,986,177	**209,308,000**	119,086
S4-C5	40,090	21	20	207,998,000	379,795	97,373,300	61,762,622	72,009,800	55,840,810	**208,465,000**	118,933
S4-C6	40,090	21	20	209,166,000	451,314	80,653,400	61,349,789	77,818,500	57,103,052	**209,654,000**	114,897
S4-C7	40,090	21	20	210,808,000	374,605	107,495,000	72,701,674	64,365,000	50,480,597	**211,433,000**	99,874
S4-C8	40,090	21	20	211,990,000	181,598	90,113,400	63,957,765	89,415,500	65,223,018	**212,313,000**	137,690
S4-C9	40,090	21	20	212,863,000	49,042	103,977,000	70,070,664	80,788,000	62,344,086	**213,247,000**	131,179
S4-C10	40,090	21	20	209,022,000	205,059	72,271,300	61,480,358	76,607,700	53,837,539	**209,850,000**	125,853

Table 15
Result summary of optimality gaps - instances in the benchmark dataset S4..

Name	*N*	*T*	*S*	*%Gap*							
				DLS		HH1		HH2		HH3	
				Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.
S4-C1	40,090	21	20	0.47	0.15	59.38	30.37	65.27	26.78	**0.26**	0.05
S4-C2	40,090	21	20	0.90	0.44	53.22	32.86	67.52	24.91	**0.30**	0.06
S4-C3	40,090	21	20	0.65	0.19	55.50	32.36	58.66	31.59	**0.25**	0.04
S4-C4	40,090	21	20	0.66	0.21	51.76	33.63	66.62	24.76	**0.31**	0.06
S4-C5	40,090	21	20	0.50	0.18	53.42	29.55	65.55	26.71	**0.27**	0.06
S4-C6	40,090	21	20	0.53	0.21	61.64	1.15	62.99	27.16	**0.29**	0.05
S4-C7	40,090	21	20	0.59	0.18	49.31	34.28	69.65	23.81	**0.29**	0.05
S4-C8	40,090	21	20	0.44	0.09	57.68	30.04	58.01	30.63	**0.29**	0.06
S4-C9	40,090	21	20	0.44	0.02	51.37	32.77	62.21	29.16	**0.26**	0.06
S4-C10	40,090	21	20	0.70	0.10	65.67	29.21	63.61	25.58	**0.31**	0.06

Table 16
Result summary of computational times (in minutes) - instances in the benchmark dataset S4..

Name	*N*	*T*	*S*	*Time(minutes)*								
				DLS		HH1		HH2		HH3		LR
				Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	
S4-C1	40,090	21	20	2520.62	550.46	48.31	47.01	**23.44**	30.95	322.58	114.46	10868.69
S4-C2	40,090	21	20	1648.16	693.77	38.33	40.46	**28.39**	40.29	345.80	142.42	4724.07
S4-C3	40,090	21	20	2070.65	524.77	39.90	47.17	**25.32**	32.07	320.33	113.81	7501.91
S4-C4	40,090	21	20	2037.65	709.55	56.26	58.67	**23.06**	25.40	338.26	223.94	12267.54
S4-C5	40,090	21	20	2360.88	518.49	38.37	37.66	**23.76**	35.65	285.36	117.25	7376.82
S4-C6	40,090	21	20	2392.38	569.52	41.91	40.13	**24.35**	32.52	301.76	137.33	6229.74
S4-C7	40,090	21	20	2321.52	582.40	47.94	44.79	**18.59**	26.56	273.12	112.79	10308.31
S4-C8	40,090	21	20	2462.85	233.24	41.44	45.65	**25.90**	29.30	271.20	114.55	7906.48
S4-C9	40,090	21	20	2503.03	125.06	46.44	42.04	**33.71**	43.48	334.33	171.54	9102.34
S4-C10	40,090	21	20	1937.58	446.52	40.47	57.24	**18.84**	32.13	285.86	160.42	6329.43

4.3. Summary

Overall, the new proposed hyper-heuristic, HH3, outperforms all other algorithms from the literature considered in the comparisons in this paper. When comparing it to the best specialized heuristics for the SMPS and SMPS+ (TS and DLS), the results show that HH3 finds high-quality solutions, comparable to or better than those produced by these problem-specific methods, in addition to being faster ; the performance differences being more significant for the largest and most difficult instances. The superiority of HH3 over TS and DLS can be explained by the fact that it relies on a mixture of simple and fast, yet efficient heuristics selected based on an effective learning mechanism that combines elements from reinforcement learning and tabu search. In comparison with the two general hyper-heuristic approaches from the literature (HH1 and HH2), HH3 is relatively more time consuming. However, the increase in solution time is offset by the excellent quality of the solutions HH3 finds and the consistency of the results, independent of the particularities of a given problem or a given instance and/or a given initial solution. Considering the 38 tested instances for which it was possible to solve the linear relaxation within the time limit, the average gap of the solutions provided by HH3 over the 1140 runs is 0.79% as opposed to 35.01% for HH1 and 48.34% for HH2, with a standard deviation of 0.88 compared to 33.06 and 30.52, for HH1 and HH2, respectively. The poor performance of HH1 and HH2 compared to HH3 is due to their weaknesses outlined in Sections 3.1 and 3.2; in particular, they tend to make "myopic" choices and fail to thoroughly explore the heuristic search space. As explained in Section 3.3, HH3 includes various strategies to overcome these weaknesses. These strategies have proven to be beneficial and yielded a better and more effective learning mechanism, as supported by the excellent results obtained for all tested instances in all runs.

To assess whether these differences in performance are statistically significant, for each pair of hyper-heuristic methods, we carried out Wilcoxon signed-rank tests with a 5% level of confidence. In terms of solution quality, the results of these tests, obtained with R (http://www.R-project.org), confirmed not only the superior performance of the new proposed hyper-heuristic HH3 that we have observed before, but also that HH1 is statistically better than HH2. With regards to solution time, Wilcoxon test results also indicate significant differences between the three hyper-heuristics. HH2 is the fastest method, followed by HH1, while HH3 ranks last.

5. Conclusions

Strategic mine planning involves solving very large stochastic mixed-integer programming problems. In the last decade, there has been a sustained development of methods capable of producing high-quality solutions to these complex real-world problems within short computing times. However, not all methods work equally well for each situation, and when faced with a new problem specific to a given mining operation, it is difficult if not impossible to know in advance which method will work best. At the outset, we asked if there is a methodology that, given a particular problem and a number of solution methods, will help determine which method or combination of methods is the best for the given problem. To attempt to answer this question, this paper investigated three hyper-heuristic approaches and applied them to two strategic mine planning problems: the stochastic open-pit mine production scheduling problem with one processing stream (SMPS) and the stochastic open-pit mine production scheduling problem with multiple processors and stockpiles (SMPS+). We proposed, and we conclude, that hyper-heuristics offer a practical alternative to the problem-specific state-of-the-art search methodologies. Not only can they tackle large instances, but also, being self-managed, they do not need to be tuned for particular problem characteris-

tics. Because they operate on a search space of heuristics rather than a search space of problem solutions, they are more generally applicable to a variety of problems.

The three hyper-heuristic approaches considered in this paper fall under the category of perturbative hyper-heuristics with online learning; that is, they use a set of simple perturbative low-level heuristics to improve a candidate solution, and a score-based learning mechanism to decide which low-level heuristic should be applied at a given step of the search process. Two of the proposed approaches (HH1 and HH2) are approaches previously proposed in the literature, while the third one (HH3) is a novel approach that uses some of the ideas of the first two but also includes new features aimed to overcome their weaknesses. To assess the performance of the three hyper-heuristics, extensive numerical experiments were performed on 43 benchmark instances of various sizes and characteristics. The three approaches were compared to each other and to two problem-specific search methodologies from the literature; namely, a tabu search heuristic (TS) and a diversified local search (DLS) that combines a variable neighborhood descent heuristic and a very large neighborhood search heuristic based on network flow techniques. The major conclusions of this study are that i) HH1 and HH2, although being the fastest of the methods, cannot compete with any of the other three in terms of solution quality; ii) HH1 outperforms HH2 in terms of solution quality, and vice versa in terms of solution time; iii) TS and DLS perform as well as HH3 or slightly better on some instances, but HH3 is substantially better than TS and DLS on the larger and most difficult instances; iv) HH3 requires shorter computational times than do TS and DLS and is less sensitive to the size of the instances; and v) HH3 is the most robust approach, exhibiting consistent performance for different problems, instances, and initial solutions.

We believe that hyper-heuristic approaches hold much promise in the field of mine planning, as they have in other fields, and that they can, according to our results, efficiently determine which heuristic to apply at each step of the search process, thereby automating the design of solution methods. They do not require problem-specific knowledge and therefore can address different classes of problems instead of solving just one problem. In line with the work in this paper, the next step is to further explore single-point perturbative hyper-heuristics with online learning. Machine learning, data mining, and data analytics techniques will be investigated to design new mechanisms to choose low-level heuristics. This should enable the generation of better learning schemes and thus more efficient hyper-heuristics. Since it has been observed empirically that one of the weaknesses of HH1 and HH2 is that early performance might dominate recent performance and that the two methods are unable to adequately put emphasis on recent performance, one approach to further explore this issue could be to use a decaying function to define and update the scores.

Developing frameworks to enable the use of multi-point-based search methodologies and metaheuristics as low-level heuristics (meta-hyper-heuristics) will also be examined. Meta-hyper-heuristics are particularly promising as they provide a more diverse and powerful set of algorithms to the high-level strategy. In a second stage of development, heuristic generation hyper-heuristics, which are approaches that create new heuristics from a set of other existing heuristics, will be investigated. The hybridization of heuristic selection and heuristic generation hyper-heuristics will also be explored.

Acknowledgement

The work in this paper was funded by NSERC CRDPJ 411270-10, NSERC Discovery Grant 239019-06, and the industry members of the COSMO Stochastic Mine Planning Laboratory: Anglo-Gold Ashanti, Barrick, BHP, De Beers, Kinross, Newmont, and Vale. This support is gratefully acknowledged.

Appendix A. Two-stage SMPS formulation

The model is described in detail in Lamghari and Dimitrakopoulos (2012), and we only briefly recall it here. The following notation is used to formulate the first stage of the problem:

N: the number of blocks considered for scheduling.
i: block index, $i = 1, \ldots, N$.
T: the number of periods over which blocks are being scheduled (horizon).
t: period index, $t = 1, \ldots, T$.
$Pred(i)$: the set of predecessors of block i; i.e., blocks that should be removed before i can be mined.
w_i: the weight of block i (in tonnes).
$\underline{W}_t$: lower bound on mining (minimum amount that should be extracted during period t considering both ore and waste blocks).
$\overline{W}_t$: upper bound on mining (maximum amount that can be mined during period t — mining equipment capacity).

To formulate the second stage of the problem, the following notation is used for each scenario:

S: the number of scenarios used to model metal uncertainty.
s: scenario index, $s = 1, \ldots, S$.
θ_{is}: parameter indicating the group of block i under scenario s

$$\theta_{is} = \begin{cases} 1 & \text{if block } i \text{ is an ore block under scenario } s, \\ 0 & \text{otherwise (i.e., if } i \text{ is a waste block under scenario } s). \end{cases}$$

m_{is}: the metal content of block i under scenario s.
v_{its}: the discounted economic value of block i if mined during period t, and if scenario s occurs. If we denote by d_1 the discount rate and by v_{is} the economic value of block i under scenario s, then v_{its} is given by the following formula:

$$v_{its} = \frac{v_{is}}{(1+d_1)^t}.$$

The economic value of a block is defined as being the net profit associated with it. The net profit differs according to whether the block is ore or waste. In the first case, it is equal to the value of the metal content of the block less the mining, processing, and selling costs. In the second case, it is equal to minus the cost of mining the block. It is assumed that ore blocks are processed during the same period when they are mined and that the profit is also generated during that period.

Furthermore, for each period t, the following notation is used:

$\underline{\Theta}_t$: lower bound on processing (minimum amount of ore required to feed the processing plant during period t).
$\overline{\Theta}_t$: upper bound on processing (maximum amount of ore that can be processed in the plant during period t — processing plant capacity).
c_t^{o-}: unit shortage cost associated with the failure to meet $\underline{O}_t$ during period t ($c_t^{o-} = \frac{c^{o-}}{(1+d_2)^t}$ where c^{o-} is the undiscounted unit shortage cost, and d_2 represents the risk discount rate).
c_t^{o+}: unit surplus cost incurred if the amount of *ore* extracted during period t exceeds $\overline{O}_t$ ($c_t^{o+} = \frac{c^{o+}}{(1+d_2)^t}$).
$\underline{M}_t$: lower bound on metal production (minimum amount of metal that should be produced during period t).
$\overline{M}_t$: upper bound on metal production (maximum amount of metal that can be sold during period t).

c_t^{m-}: unit shortage cost associated with the failure to meet $\underline{M}_t$ during period t ($c_t^{m-} = \dfrac{c^{m-}}{(1+d_2)^t}$).

c_t^{m+}: unit surplus cost incurred if the metal production during period t exceeds $\overline{M}_t$ ($c_t^{m+} = \dfrac{c^{m+}}{(1+d_2)^t}$).

The variables used to formulate the problem are as follows:

- A binary variable is associated with each block i for each period t:
  $x_{it} = \begin{cases} 1 & \text{if block } i \text{ is mined during period } t, \\ 0 & \text{otherwise.} \end{cases}$
- The variables d_{ts}^{o-} and d_{ts}^{o+} are used to denote the shortage and the surplus in the amount of *ore* mined during period t if scenario s occurs, respectively.
- Finally, the variables d_{ts}^{m-} and d_{ts}^{m+} measure the shortage and the surplus in metal production during period t under scenario s, respectively.

Note that the x_{it} are the first-stage decision variables. They are scenario-independent since they must be fixed before knowing the values of the uncertain parameters. The deviation variables $d_{ts}^{o-}, d_{ts}^{o+}, d_{ts}^{m-}$, and d_{ts}^{m+} are the second-stage (recourse) decision variables. Their values depend both on the realization of the uncertain parameters and on the values of the first-stage decision variables.

Using this notation, the SMPS can be modeled as follows:

$$\max \frac{1}{S}\left\{\sum_{s=1}^{S}\sum_{t=1}^{T}\sum_{i=1}^{N} v_{its}x_{it} - \sum_{s=1}^{S}\sum_{t=1}^{T} \times (c_t^{o-}d_{ts}^{o-} + c_t^{o+}d_{ts}^{o+} + c_t^{m-}d_{ts}^{m-} + c_t^{m+}d_{ts}^{m+})\right\} \quad \text{(A.1)}$$

Subject to

$$\sum_{t=1}^{T} x_{it} \le 1 \qquad i = 1,\dots,N \quad \text{(A.2)}$$

$$x_{it} - \sum_{\tau=1}^{t} x_{j\tau} \le 0 \qquad i = 1,\dots,N,\ j \in Pred(i),\ t = 1,\dots,T \quad \text{(A.3)}$$

$$\sum_{i=1}^{N} w_i x_{it} \le \overline{W}_t \qquad t = 1,\dots,T \quad \text{(A.4)}$$

$$\sum_{i=1}^{N} w_i x_{it} \ge \underline{W}_t \qquad t = 1,\dots,T \quad \text{(A.5)}$$

$$\sum_{i=1}^{N} \theta_{is} w_i x_{it} + d_{ts}^{o-} \ge \underline{\Theta}_t \qquad t = 1,\dots,T,\ s = 1,\dots,S \quad \text{(A.6)}$$

$$\sum_{i=1}^{N} \theta_{is} w_i x_{it} - d_{ts}^{o+} \le \overline{\Theta}_t \qquad t = 1,\dots,T,\ s = 1,\dots,S \quad \text{(A.7)}$$

$$\sum_{i=1}^{N} \theta_{is} m_{is} x_{it} + d_{ts}^{m-} \ge \underline{M}_t \qquad t = 1,\dots,T,\ s = 1,\dots,S \quad \text{(A.8)}$$

$$\sum_{i=1}^{N} \theta_{is} m_{is} x_{it} - d_{ts}^{m+} \le \overline{M}_t \qquad t = 1,\dots,T,\ s = 1,\dots,S \quad \text{(A.9)}$$

$$x_{it} = 0 \text{ or } 1 \qquad i = 1,\dots,N,\ t = 1,\dots,T \quad \text{(A.10)}$$

$$d_{ts}^{o-}, d_{ts}^{o+}, d_{ts}^{m-}, d_{ts}^{m+} \ge 0 \qquad t = 1,\dots,T,\ s = 1,\dots,S. \quad \text{(A.11)}$$

The objective function (A.1) includes two components to maximize the expected net present value of the mining operation, and to minimize the expected recourse costs incurred whenever the stochastic constraints are violated due to metal uncertainty. We assume that all scenarios have an equal probability of occurrence and hence the coefficient $\frac{1}{S}$ represents the probability that scenario s occurs. Constraints (A.2)–(A.5) are related to the non-stochastic constraints and thus are scenario-independent. Constraints (A.2) guarantee that each block i is mined at most once during the horizon. The mining precedence is enforced by constraints (A.3). Constraints (A.4) and (A.5) ensure that the requirements on the mining levels are respected during each period of the horizon. Constraints (A.6)–(A.9) are related to the stochastic constraints and thus are scenario-dependent. Constraints (A.6) and (A.7) are related to the requirements on the processing levels. For each scenario s, the target is to have the total weight of *ore* blocks mined during any period t in the interval $[\underline{\Theta}_t, \overline{\Theta}_t]$. If it is equal to a value smaller than $\underline{\Theta}_t$ (respectively, larger than $\overline{\Theta}_t$), then the shortage penalty cost is equal to $c_t^{o-}d_{ts}^{o-}$ (respectively, the surplus penalty cost is equal to $c_t^{o+}d_{ts}^{o+}$). Finally, constraints (A.8) and (A.9) indicate that, during any period t, the target is to have the metal production in the interval $[\underline{M}_t, \overline{M}_t]$. Otherwise, the shortage penalty cost is equal to $c_t^{m-}d_{ts}^{m-}$ or the surplus penalty cost is equal to $c_t^{m+}d_{ts}^{m+}$.

Appendix B. Two-stage SMPS+ formulation

The model is described in detail in Lamghari and Dimitrakopoulos (2016a), and we only briefly recall it here. The following notation is used:

Indices:

N: the number of blocks considered for scheduling.
i: block index, $i = 1,\dots,N$.
T: the number of periods over which blocks are being scheduled (horizon).
t: period index, $t = 1,\dots,T$.
P: the number of processors. A stockpile is associated with each processor, thus the number of stockpiles is equal to the number of processors.
p: processor index, $p = 1,\dots,P$.
S: the number of scenarios used to model metal uncertainty.
s: scenario index, $s = 1,\dots,S$.

Variables:

$x_i^t = \begin{cases} 1 & \text{if } i \text{ is mined by period } t, \\ 0 & \text{otherwise.} \end{cases}$
This means that if block i is mined in period τ, then $x_i^t = 0$ for all $t = 1,\dots,\tau-1$ and $x_i^t = 1$ for all $t = \tau,\dots,T$. If i is not mined during the horizon, then $x_i^t = 0$ for all $t = 1,\dots,T$. To simplify the notation, we introduce a set of N dummy decision variables x_i^0 ($i = 1,\dots,N$), each having a fixed value equal to 0.

y_{ps}^{t+}: surplus in the amount of ore mined during period t that can be processed in p if scenario s occurs (i.e., the amount to send from the mine to the stockpile associated with p).
y_{ps}^{t-}: amount of ore to take in period t from the stockpile associated with processor p, if scenario s occurs (i.e., the amount to send from the stockpile to the processor).
y_{ps}^{t}: amount of ore in the stockpile associated with processor p at the end of period t under scenario s. It is assumed that the stockpile is empty at the beginning of the first period but might not be empty at the end of the planning horizon.

Parameters:

π_s: probability that scenario s occurs. The S scenarios are equiprobable, thus $\pi_s = \frac{1}{S}$.

$Pred(i)$: the set of predecessors of block i; i.e., blocks that have to be removed to have access to block i ($Pred(i) \subseteq \{1, \ldots, N\}$).

w_i: weight of block i in tonnes (tonnage).

W^t: maximum amount of material (ore and waste) that can be mined during period t (mining capacity in tonnes).

$\bar{c}$: undiscounted cost of mining a tonne of material.

c_p: undiscounted cost of processing a tonne of ore in processor p.

$\theta_{ips} = \begin{cases} 1 & \text{if } i \text{ is processed in } p \text{ under scenario } s, \\ 0 & \text{otherwise.} \end{cases}$

Θ_p^t: maximum amount of ore that can be processed in processor p during period t (processing capacity of p in tonnes).

r_{is}: undiscounted revenue of an already mined block i if sent immediately for processing (i.e., during the same period it is mined), and if scenario s occurs ($r_{is} = 0$ if i is a waste block under scenario s).

c_p^+: undiscounted cost of sending a tonne of ore to the stockpile associated with processor p (transportation cost plus handling cost). It is assumed that when a block arrives at the stockpile, it is mixed with the other material already there. The cost of this operation is included in c_p^+.

c_p^-: undiscounted cost of taking a tonne of ore from the stockpile associated with processor p (transportation cost plus loading cost).

$\tilde{r}_{ps}$: undiscounted revenue to be generated if a tonne of ore in the stockpile associated with p is processed, and if scenario s occurs.

d: the discount rate per period for cash flows.

The two-stage stochastic programming model can be summarized as follows:

$$\begin{aligned} \max\ & -\sum_{t=1}^{T} \frac{1}{(1+d)^t} \sum_{i=1}^{N} w_i \bar{c}(x_i^t - x_i^{t-1}) \\ & + \sum_{t=1}^{T} \frac{1}{(1+d)^t} \sum_{i=1}^{N} \sum_{s=1}^{S} \pi_s r_{is}(x_i^t - x_i^{t-1}) \\ & - \sum_{t=1}^{T} \frac{1}{(1+d)^t} \sum_{p=1}^{P} \sum_{s=1}^{S} \pi_s (\tilde{r}_{ps} + c_p^+) y_{ps}^{t+} \\ & + \sum_{t=1}^{T} \frac{1}{(1+d)^t} \sum_{p=1}^{P} \sum_{s=1}^{S} \pi_s (\tilde{r}_{ps} - c_p^-) y_{ps}^{t-} \end{aligned} \tag{B.1}$$

Subject to

$$x_i^{t-1} \leq x_i^t \qquad \forall\, i,\ t \tag{B.2}$$

$$x_i^t \leq x_j^t \qquad \forall\, i,\ j \in Pred(i),\ t \tag{B.3}$$

$$\sum_{i=1}^{N} w_i (x_i^t - x_i^{t-1}) \leq W^t \qquad \forall\, t \tag{B.4}$$

$$\sum_{i=1}^{N} \theta_{ips} w_i (x_i^t - x_i^{t-1}) - y_{ps}^{t+} + y_{ps}^{t-} \leq \Theta_p^t \qquad \forall\, t,\ p,\ s \tag{B.5}$$

$$y_{ps}^{t-1} + y_{ps}^{t+} - y_{ps}^{t-} = y_{ps}^t \qquad \forall\, t,\ p,\ s \tag{B.6}$$

$$x_i^t = 0 \text{ or } 1 \qquad \forall\, i,\ t \tag{B.7}$$

$$x_i^0 = 0 \qquad \forall\, i \tag{B.8}$$

$$y_{ps}^{t+}, y_{ps}^{t-}, y_{ps}^t \geq 0 \qquad \forall\, t,\ p,\ s \tag{B.9}$$

$$y_{ps}^0 = 0 \qquad \forall\, p,\ s. \tag{B.10}$$

The objective function (B.1) maximizes the expected NPV of the mine. It includes four different terms:

1. The first term ($-\sum_{t=1}^{T} \frac{1}{(1+d)^t} \sum_{i=1}^{N} w_i \bar{c}(x_i^t - x_i^{t-1})$) evaluates the total discounted cost of the extraction (discounted cost of the first stage solution).
2. The second term ($\sum_{t=1}^{T} \frac{1}{(1+d)^t} \sum_{i=1}^{N} \sum_{s=1}^{S} \pi_s r_{is}(x_i^t - x_i^{t-1})$) gives the total expected discounted profit generated if all the ore mined is sent immediately for processing during the period in which it is mined (first type of recourse). Note that waste blocks do not contribute to this term because, as mentioned earlier, if i is a waste block under scenario s, then $r_{is} = 0$.
3. The third term ($-\sum_{t=1}^{T} \frac{1}{(1+d)^t} \sum_{p=1}^{P} \sum_{s=1}^{S} \pi_s (\tilde{r}_{ps} + c_p^+) y_{ps}^{t+}$) gives the total expected discounted cost of sending ore to the stockpiles, including both the revenue lost because the ore is not processed in the period when it is available and the cost of transportation to the stockpiles (second type of recourse).
4. The fourth term ($\sum_{t=1}^{T} \frac{1}{(1+d)^t} \sum_{p=1}^{P} \sum_{s=1}^{S} \pi_s (\tilde{r}_{ps} - c_p^-) y_{ps}^{t-}$) represents the total expected discounted profit to be generated from processing ore taken from the stockpiles; that is, revenue minus loading and transportation costs (third type of recourse).

Constraints (B.2)–(B.4) are scenario-independent. Constraints (B.2) guarantee that each block i is mined at most once during the horizon. The mining precedence is enforced by constraints (B.3). Constraints (B.4) impose an upper bound W^t on the amount of material (waste and ore) mined during each period t. Constraints (B.5) are related to the requirements on the processing levels and therefore are scenario-dependent. They stipulate that for each scenario s and each processor p, if the total weight of ore blocks mined during any period t is greater than the processing capacity at that period, Θ_p^t, then the surplus y_{ps}^{t+} is sent to the stockpile associated with the processor p, inducing a cost equal to $\frac{(\tilde{r}_{ps} + c_p^+)}{(1+d)^t} y_{ps}^{t+}$. If it is smaller than Θ_p^t and there is material in the stockpile, then an amount equal to y_{ps}^{t-} (maximum possible such that neither the capacity of the processor nor the amount available in the stockpile is exceeded) is taken from the stockpile and added to feed the processor, generating a profit equal to $\frac{(\tilde{r}_{ps} - c_p^-)}{(1+d)^t} y_{ps}^{t-}$. Finally, constraints (B.6) balance the flow at each stockpile and are also scenario-dependent. They ensure that for each scenario s, at the end of any period t, the amount of ore in the stockpile associated with each processor p is equal to the amount that was in the stockpile at the end of the previous period $(t-1)$ plus the amount added to the stockpile during t minus the amount taken from the stockpile during t (i.e., the amount sent from the stockpile for processing, if any). The initial amount in the stockpile, y_{ps}^0, is assumed to be equal to 0.

References

Albor, F., Dimitrakopoulos, R., 2009. Stochastic mine design optimization based on simulated annealing: pit limits, production schedules, multiple orebody scenarios and sensitivity analysis. IMM Trans., Min. Technol., 118(2): 80–91. 118 (2), 80–91.

Albor, F., Dimitrakopoulos, R., 2010. Algorithmic approach to pushback design based on stochastic programming: method, application and comparisons. IMM Trans., Min. Technol., 119(2): 88–101. 119 (2), 88–101.

Asad, M., Dimitrakopoulos, R., 2013. Implementing a parametric maximum flow algorithm for optimal open pit mine design under uncertain supply and demand. J. Oper. Res. Soc. 64, 185–197.

Behrang, K., Hooman, A., Clayton, D., 2014. A linear programming model for long-term mine planning in the presence of grade uncertainty and a stockpile. Int. J. Min. Sci.Technol. 24, 451–459.

Bienstock, D., Zuckerberg, M., 2010. Solving lp relaxations of large-scale precedence constrained problems. Lect. Notes Comput. Sci. 6080, 1–14.
Birge, J., Louveaux, F., 2011. Introduction to Stochastic Programming, Second Edition. Springer New York, Dordrecht Heidelberg London.
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.
Boland, N., Dumitrescu, I., Froyland, G., Gleixner, A.M., 2009. Lp-based disaggregation approaches to solving the open pit mining production scheduling problem with block processing selectivity. Comput. Oper. Res. 36, 1064–1089.
Boucher, A., Dimitrakopoulos, R., 2009. Block simulation of multiple correlated variables. Math. Geosci. 41 (2), 215–237.
Burke, E., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Qu, R., 2013. Hyperheuristics: a survey of the state of the art. J. Oper. Res. Soc. 64, 1695–1724.
Burke, E., Hart, E., Kendall, G., Newall, J., Ross, P., Schulenburg, S., 2003. Hyperheuristics: An Emerging Direction in Modern Search Technology. In: Handbook of Metaheuristics, Glover F and Kochenberger G (eds), Kluwer, pp. 457–474.
Burke, E., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J., 2010. A Classification of Hyper-Heuristic Approaches. In: Handbook of metaheuristics - Second Edition, Springer, New York, pp. 449–468.
Burke, E., Kendall, G., Soubeiga, E., 2003. A tabu-search hyperheuristic for timetabling and rostering. J. Heuristics 9 (3), 451–470.
Chanda, E., 2007. Network linear programming optimization of an integrated mining and metallurgical complex. In: Proceedings of Orebody Modelling and Strategic Mine Planning: Uncertainty and Risk Management Models, The Australasian Institute of Mining and Metallurgy Spectrum Series 14, 2nd Edition, pp. 149–155.
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, 517–528.
Chiles, J., Delfiner, P., 2012. Geostatistics: Modeling Spatial Uncertainty, Second Ed.. John Wiley & Sons., New Jersey.
Cowling, P., Kendall, G., Soubeiga, E., 2001. A hyperheuristic approach for scheduling a sales summit. Lect. Notes Comput. Sci. 2079, 176–190.
Cullenbine, C., Wood, R., Newman, A., 2011. A sliding time window heuristic for open pit mine block sequencing. Optim. Lett. 5, 365–377.
Denzinger, J., Fuchs, M., Fuchs, M., 1997. High performance ATP systems by combining several AI methods. In: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI 97), Morgan Kaufmann, CA, USA, pp. 102–107.
Dimitrakopoulos, R., 2011. Stochastic optimization for strategic mine planning: a decade of developments. J. Min. Sci. 84, 138–150.
Dimitrakopoulos, R., Farrelly, C., Godoy, M., 2002. Moving forward from traditional optimization: grade uncertainty and risk effects in open pit mine design. IMM Trans. 111, A82–A88.
Dowd, P., 1994. Risk assessment in reserve estimation and open-pit planning. Trans. Inst. Min.Metall. 103, A148–A154.
Drake, J., Ozcan, E., Burke, E., 2012. An improved choice function heuristic selection for cross domain heuristic search. Lect. Notes Comput. Sci. 7492, 307–316.
Ferland, J.A., Amaya, J., Djuimo, M.S., 2007. Application of a particle swarm algorithm to the capacitated open pit mining problem. In: Autonomous Robots and Agents, Mukhopadhyay S. and Sen Gupta G. Ed. Springer-Verlag, pp. 127–133.
Fisher, H., Thompson, G., 1963. Probabilistic learning combinations of local job-shop scheduling rules. In: Industrial Scheduling, Muth, J.F., Thompson, G.L. (eds.), Prentice- Hall, New Jersey, pp. 225–251.
Fricke, C., Velletri, P., Wood, R., 2014. Enhancing risk management in strategic mine planning through uncertainty analysis. In: Proceedings of Orebody Modelling and Strategic Mine Planning Symposium 2014, The Australasian Institute of Mining and Metallurgy, pp. 275–279.
Godoy, M., 2002. The Effective Management of Geological risk. University of Queensland, Qld, Australia.
Goodfellow, R., Dimitrakopoulos, R., 2016. Global optimization of open pit mining complexes with uncertainty. Appl. Soft. Comput. 40, 292–304.
Goodfellow, R., Dimitrakopoulos, R., 2017. Simultaneous stochastic optimization of mining complexes and mineral value chains. Math. Geosci. 49 (3), 341–360.
Goovaerts, P., 1997. Geostatistics for Natural Resources Evaluation. Oxford University Press, New York.
Hochbaum, D., Chen, A., 2000. Improved planning for the open-pit mining problem. Oper. Res. 48, 894–914.
Hoerger, S., Hoffman, L., Seymour, F., 1999. Mine planning at newmontś nevada operations. Min. Eng. 51 (10), 26–30.
Horta, A., Soares, A., 2010. Direct sequential co-simulation with joint probability distributions. Math. Geosci. 42 (3), 269–292.
Kawahata, K., Schumacher, P., Fein, M., 2015. Strategic mine planning and production scheduling optimization at Newmont's twin creeks operation. In: Proceedings of the 37th International Symposium Application of Computers and Operations Research in the Mineral Industry (APCOM), Fairbanks, Alaska, pp. 1052–1060.
Lamghari, A., 2017. Mine planning and oil field development: a survey and research potentials. Math. Geosci. 49 (3), 395–437.
Lamghari, A., Dimitrakopoulos, R., 2012. A diversified tabu search approach for the open-pit mine production scheduling problem with metal uncertainty. Eur. J. Oper. Res. 222, 642–652.
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, 273–290.
Lamghari, A., Dimitrakopoulos, R., 2016. Progressive hedging applied as a metaheuristic to schedule production in open-pit mines accounting for reserve uncertainty. Eur. J. Oper. Res. 253, 843–855.
Lamghari, A., Dimitrakopoulos, R., Ferland, J., 2014. A variable descent neighborhood algorithm for the open-pit mine production scheduling problem with metal uncertainty. J. Oper. Res. Soc. 65, 1305–1314.
Lamghari, A., Dimitrakopoulos, R., Ferland, J., 2015. A hybrid method based on linear programming and variable neighborhood descent for scheduling production in open-pit mines. J. Global Optim. 63, 555–582.
Maleki, M., Emery, X., 2015. Joint simulation of grade and rock type in a stratabound copper deposit. Math. Geosci. 47 (4), 471–495.
Marcotte, D., Caron, J., 2013. Ultimate open pit stochastic optimization. Comput. Geosci. 51, 238–246.
Menabde, M., Froyland, G., Stone, P., Yeates, G., 2007. Mining schedule optimization for conditionally simulated orebodies. Orebody Modelling and Strategic Mine Planning, The Australasian Institute of Mining and Metallurgy, Spectrum Series 14, 379–384.
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.
Montiel, L., Dimitrakopoulos, R., Kawahata, K., 2016. Globally optimising open-pit and underground mining operations under geological uncertainty. Min. Technol. 125 (1), 2–14.
Moreno, E., Espinoza, D., Goycoolea, M., 2010. Large-scale multi-period precedence constrained knapsack problem: a mining application. Electron. Note. Discrete Math. 36, 407–414.
Newman, A.M., Rubio, E., Caro, R., Weintraub, A., Eurek, E., 2010. A review of operations research in mine planning. Interfaces (Providence) 40, 222–245.
Ramazan, S., Dimitrakopoulos, R., 2013. Production scheduling with uncertain supply: a new solution to the open pit mining problem. Optim. Eng. 14, 361–380.
Ravenscroft, P., 1992. Risk analysis for mine scheduling by conditional simulation. Trans. Inst. Min.Metall., Sect. A A104–A108.
Ross, P., 2005. Hyper-heuristics. In: Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, Burke EK and Kendall G (eds), Springer, pp. 529–556.
Rossi, M., Deutsch, C., 2014. Mineral Resource Estimation. Springer, New York.
Whittle, G., 2007. Global asset optimization. In: Proceedings of Orebody Modelling and Strategic Mine Planning: Uncertainty and Risk Management Models, The Australasian Institute of Mining and Metallurgy Spectrum Series 14, 2nd Edition, pp. 331–336.
Whittle, J., 2009. The global optimizer works-what next? In: Proceedings of Advances in Orebody Modelling and Strategic Mine Planning: Old and New Dimensions in a Changing World, The Australasian Institute of Mining and Metallurgy Spectrum Series 17, 1st Edition, pp. 3–5.