Int J Adv Manuf Technol (2014) 72:1021–1037
DOI 10.1007/s00170-014-5619-8

ORIGINAL ARTICLE

Scheduling in iron ore open-pit mining

Maksud Ibrahimov · Arvind Mohais · Sven Schellenberg · Zbigniew Michalewicz

Received: 27 November 2011 / Accepted: 9 January 2014 / Published online: 7 March 2014

Abstract During the last few years, most production-based businesses have been under enormous pressure to improve their top-line growth and bottom-line savings. As a result, many companies are turning to systems and technologies that can help optimise their supply chain activities. In this paper, we discuss a real-world application of scheduling in the mining industry. This is a highly constrained problem with some of the constraints changing over time. The objective is to generate a plan that meets the quality and tonnage targets by utilising the provided equipment and machines. The mathematical model is described in detail, and a three-module complex algorithm based on computational intelligence is proposed. The main software functionality is also described. The developed software is currently in the production use.

Keywords Open-pit mining · Scheduling · Metaheuristics · Real-world application

M. Ibrahimov (✉) · A. Mohais · Z. Michalewicz
School of Computer Science, University of Adelaide, Adelaide, SA 5005, Australia
e-mail: maksud@ibragimov.org

A. Mohais
e-mail: am@solveitsoftware.com

Z. Michalewicz
e-mail: zbigniew.michalewicz@adelaide.edu.au

S. Schellenberg
School of Computer Science and IT, RMIT University, Melbourne, Australia
e-mail: sven.schellenberg@rmit.edu.au

Z. Michalewicz
Institute of Computer Science, Polish Academy of Sciences, ul. Ordona 21, 01-237 Warsaw, Poland

Z. Michalewicz
Polish-Japanese Institute of Information Technology, ul. Koszykowa 86, 02-008 Warsaw, Poland

1 Introduction

Due to the high level of complexity, it becomes virtually impossible for deterministic systems or human domain experts to find an optimal solution for many real-world problems, especially in the mining industry—not to mention that the term "optimal solution" loses its meaning in multi-objective environment, as often we can talk only about trade-offs between different solutions. Moreover, the manual iteration and adjustment of scenarios (what-if scenarios and trade-off analysis), which is needed for strategic planning, becomes an expensive, if not unaffordable, exercise. Many texts on advance planning and supply chain management (e.g. [1]) describe several commercial software applications (e.g. AspenTech, aspenONE; i2 Technologies, i2 Six.Two; Oracle, JDEdwards EnterpriseOne Supply Chain Planning; SAP, SCM; and many others), which emerged mainly in 1990s. However, it seems that the areas of supply chain management in general, and advanced planning in particular, are ready for a new genre of applications which are based on computational intelligence methods. Many supply chain-related projects run at large corporations

worldwide failed miserably (projects that span a few years and cost many millions). In [1], the authors wrote:

> In recent years since the peak of the e-business hype Supply Chain Management and especially Advanced Planning Systems were viewed more and more critically by industry firms, as many SCM projects failed or did not realise the promised business value.

The authors also identified three main reasons for such failures:

- the perception that the more you spend on IT (e.g. APS), the more value you will get from it,
- an inadequate alignment of the SCM concept with the supply chain strategy, and
- the organisational and managerial culture of industry firms.

Whilst it is difficult to argue with the above points, it seems that the fourth (and unlisted) reason is the most important: maturity of technology. Small improvements and upgrades of systems created in 1990s do not suffice any longer for solving companies' problems of the twenty-first century. A new approach is necessary which would combine seamlessly the forecasting, simulation and optimisation components in a new architecture. Further, many existing applications are not flexible enough in the sense that they cannot cope with any exceptions, i.e. it is very difficult, if not impossible, to include some problem-specific features—and most businesses have some unique features which need to be included in the underlying model—and are not adequately captured by off-the-shelf standard applications. Thus, the results are often not realistic, and the team of operators return to their spreadsheets and whiteboards rather than to rely on unrealistic recommendations of the software.

There are two main branches that tackle optimisation of complex problems: operations research and computational intelligence. Operations research uses techniques such as linear programming, branch and bound, dynamic programming, etc. In this paper, our methods, however, will be based on the methods of computational intelligence.

An interesting question, which is being raised from time to time, asks for guidance on the types of problems for which computational intelligence methods are more appropriate than, say, standard operation research methods. From our perspective, the best answer to this question is given in a single phrase: complexity. Let us explain. Real-world problems are usually difficult to solve for several reasons and include the following:

- The number of possible solutions is so large as to forbid an exhaustive search for the best answer.
- The evaluation function that describes the quality of any proposed solution is noisy or varies with time, thereby requiring not just a single solution but an entire series of solutions.
- The possible solutions are so heavily constrained that constructing even one feasible answer is difficult, let alone searching for an optimum solution.

Note that every time we solve a problem, we must realise that we are, in reality, only finding the solution to a model of the problem. All models are a simplification of the real world—otherwise they would be as complex and unwieldy as the natural setting itself. Thus, the process of problem solving consists of two separate general steps: (i) creating a model of the problem and (ii) using that model to generate a solution [2]:

$$Problem \rightarrow Model \rightarrow Solution.$$

Note that the "solution" is only a solution in terms of the model. If our model has a high degree of fidelity, we can have more confidence that our solution will be meaningful. In contrast, if the model has too many unfulfilled assumptions and rough approximations, the solution may be meaningless, or worse. So, in solving real-world problems, there are at least two ways to proceed [2]:

1. we can try to simplify the model so that traditional methods might return better answers, or
2. we can keep the model with all its complexities, and use non-traditional approaches, to find a near-optimum solution.

The system that is described in this paper shows a real-world application of computational intelligence in mining industry. The problem of mine planning has enormous amount of possible solutions and is very highly constrained with constraints changing over time. Because of these characteristics, algorithms based on computational intelligence were employed to find solutions to this problem. Unfortunately, due to the commercial and proprietary nature of this system, it is not possible to reveal all implementational details needed to replicate problem. However, enough details is provided to explain underlying concepts of the algorithms used. The system is currently in production use.

The rest of the paper is organised as follows. Section 2 provides a review of relevant literature. The following section explains the problem and challenges. Section 4 describing the mathematical model. An algorithmic approach to the problem is presented in Section 5. The functionality of the developed system is described in Section 6. The last Section 8 concludes the paper.

2 Literature review

Mining is the process of extraction for profit of valuable minerals or other geological materials from the earth. The following major types of minerals are commonly mined.

- *Metallic ores*. These include the *ferrous* metals (iron, manganese), the *base* metals (copper, lead, zinc), the *precious* metals (gold, silver, platinum) and the *radioactive* metals (uranium, radium).
- *Nonmetaillic minerals*. These include phosphate, limestone and sulphur.
- *Fossil fuels*. These include coal, petroleum and natural gas. Extraction of petroleum and gas which have different physical characteristics requires a different mining technology.

Mining is amongst the most profitable of industries, and therefore optimised mine planning and scheduling can have a great impact for optimal mine exploitation. There are two main ways to recover materials from the ground: *surface mining* and *underground mining*. In this paper, we will be focusing on the former. Many problems arise in open-pit mining: orebody modelling, creation of the life-of-mine (LOM) schedule, determination of mine equipment requirements and optimal operating layout, and the optimal transportation of ore from pit to port. Each one of these problems is very complex, so the operation of a large open-pit mine is an enormously difficult task.

In terms of the time horizon, there can be *operational*, *short-term*, *medium-term*, *long-term planning* (LTP), and LOM planning. The operational and short-term planning involve a very detailed scheduling of processes and equipment, where it is important to know what each piece of equipment should be doing during each day and even each hour. It also operates based on fairly precise data. In contrast, LTP involves looking at a global picture, working with estimated data, various uncertainties, risk analysis and evolving economic criteria. This research is mostly focused on the LTP, but future developments may include short-term planning as well.

Caccetta and Hill [3] discuss some techniques for LOM optimal production schedule. They describe mixed integer linear programming (MILP) model and give a survey of several methods to address the problem, including parametrisation method and a method based on the Lagrangian relaxation. In their later work, Caccetta and Hill [4] use the *branch and bound* method to solve the problem based on MILP. They use a combination of best-first search and depth-first search to achieve a "good spread" of possible pit schedules whilst benefiting from using depth-first search. Bley et. al [5] use the same model formulation as that of Caccetta and Hill whilst strengthening the problem by adding inequalities derived by combining the precedence (knapsack substructure) and production constraints. The goal is to maximise the net present value (NPV).

Lizotte and Elbrond [6] and Yun and Yegulalp [7] tackle the mine scheduling problem with techniques based on the dynamic programming. Smith and Tao [8] address the multi-objective phosphate mine scheduling problem via goal programming techniques.

In the current paper, we will mostly focus on the extended version of the *block sequencing problem*, i.e what blocks to dig at what time. Dagdelen and Johnson [9] propose an algorithm based on the Lagrangian relaxation for maximising NPV with constraints. The extension of this work has been done by Akaike and Dagdelen [10] by iteratively changing the values of Lagrangian multipliers, until the solution satisfies the constraints. Another extension has been done by Kawahata [11] in his doctoral thesis. He added variable cutoff grades, stockpiling and wastedump restrictions.

Busnach et al. [12] propose a heuristic algorithm that addresses a block sequencing problem for a phosphate mine in Israel. Samanta et. al [13] apply a genetic algorithm to the problem of *grade control planning* in a bauxite mine. The problem that is discussed in this paper has a grade control planning problem as its part. The goal of the problem is to minimise quality deviations of two elements: silicon and aluminium.

Osanloo et al. [14] has a good review of long-term optimisation models for open-pit mining. A very good review on operations research methods in mine planning (both surface and underground) is provided by Newman et al. [15].

Armstrong and Galli [16] discuss the problem of mine scheduling and pit optimisation when the mine resources become better known through time whilst economic parameters such as commodity prices and costs evolve over time. As the number of blocks in the mine can be enormous, the authors propose to work with larger aggregations of blocks called macro-blocks. They take advantage of convexity of block sequences to solve the problem with the two-step procedure to select the best sequence.

Halatchev [17] describes an important contemporary criteria for long-term production scheduling. These include technological, economical and probabilistic criteria.

Sattarvand and Niemann-Delius [18] propose a new algorithm based on the ant colony that tackles the problem of optimisation long-term planning. The produced schedule is 34 % better than the initial generated by the Lerchs Grossman and parametrisation algorithms.

Chicoisne et al. [19] look at integer programming formulation of mine planning problem C-PIT. The authors propose

a new decomposition method for the linear programming model with the single capacity constraint per time period. After the linear programming step, a heuristic based on local search is run to find even better solution.

Many problems in the mining industry can be classified as scheduling problems. The problem that is discussed in this paper also belongs to this category. Several different types of scheduling problems exist, such as job-shop scheduling, flow-shop, flexible flow-shop and open-shop. These are described in detail in [20]. Many other algorithms for different types of scheduling problems were developed, for example are those in [21–32].

3 Problem description

Since prehistoric times, people have been mining different ores and minerals to produce various goods for their lives. Mining goes hand in hand with human history, and so even major cultural eras are called by various metals mined: the Bronze Age (4000 to 5000 BC), the Iron Age (1500 BC to 1780 AD), the Steel Age (1780 to 1945) and the Nuclear Age (1945 to the present). In the modern world, mining is still one of the most important and advanced fields of industry. There are several main types of mining, and in this case study, the open-pit type of mining problem will be described. In particular, a metal ore mining system will be discussed.

A typical mine has a hierarchical structure. At the top level, the mine is divided into several sub-mines called *models*. Typically, each model is an island separated by non-ore material. Each model consists of one or several *pits*. Each pit in turn is sliced into horizontal layers called *benches*. Each bench contains multiple *blocks* (which is the last level of hierarchy). Each block has specific information about its coordinates, tonnage, characteristics, percentage of different metals and nonmetals (e.g. iron, aluminium, phosphorus) and waste. A block can fully consist of waste, or it can contain certain metals of high or low grade.

There are two main types of plants on the mining site: *crushers* and *washplants* (sometimes referred to as a *beneficiation plant*). Depending on the nature of the mine, there can be other types of plants; however, in this chapter, only these two are considered.

A *crusher* is a plant designed to reduce large block chunks into smaller rocks for further processing. Different crushers have different crushing speeds measured in tonnes crushed per hour. Two main types of crushers considered in this chapter are high-grade crushers and low-grade crushers that are designed to operate with high- and low-grade materials, respectively.

In order to improve low-grade ore, it needs to be processed by the washplant. The process of washing removes contaminations, impurities and other things that lower the quality of the ore. Several methods are known to achieve this task: magnetic separation, advanced gravity separation, jigging, washing and others.

There are two types of mobile equipment available for mining: diggers and trucks. *Trucks* are used to transport material from one place to another. One or several fleets of trucks can be available. Each fleet has a number of trucks of the same type with same speed and load capacity. *Diggers* are used to excavate material and load it to trucks. Each digger has its own digging rate measured in tonnes of material per hour. They normally have a very low speed (2 km/h).

Mine models, pits, benches, crushers, washplants, stockpiles and wastedumps are connected by the road network. Materials are transported by a means of trucks through this network. Each pit has one or several pit exits through which trucks enter the pit. Each bench has one or several starting blocks, called *toe blocks*, with the road connected to them.

The mining process is as follows. First, a block is blasted by explosives, and then a *digger* enters the blast area and excavates the blasted material. It loads the material onto trucks which drive it to the next destination. The destination can be different depending on the type and quality of material. All high-grade materials from the excavated block are transported to high-grade crushers, and then the crushed product is shipped to customers. The low-grade material is transported first to the washplant to purify it and then to the low-grade crusher and only after that shipped to customers. All excavated waste is taken to *wastedumps* which are basically piles of waste material. Along with the destinations mentioned above, the excavated material can also go to *stockpiles* which can be thought of as temporary places to put material that can be used in the future.

The current problem is a long-term planning and scheduling problem. Decisions have to be made the order in which blocks of the mine should be excavated and how to utilise the equipment, crushers, washplants, stockpiles and truck fleets. In order to understand the objectives of the problem, a mining time period concept needs to be discussed. As a long-term scheduling problem, operation decisions should be made in a time horizon which is a few times smaller than the life of the mine term (which is the long-term planning horizon). The whole decision time horizon is divided into smaller *time periods*. The number of diggers and digging rates, and the crusher and washplant operating rates, and capacities of wastedumps and stockpiles can be different across time periods. In addition, each time period has its *targets* which can be understood as a milestone to be reached at the end of the period.

- *Tonnage target.* As has been described earlier, each block contains a certain tonnage of desired metal. A

block can also consist of total waste, which will not be accounted for in reaching the targets.
- *Quality target.* All mining blocks have metal ore of a different quality. However, clients want ore of a predefined quality. This means that during a time period, a set of blocks should be excavated such that, after the blending of all the excavated blocks during the time period, the blended quality should be close to reaching the desired quality. It is very hard to match the quality target exactly; this is a reason a quality tolerance is introduced, measured in percentage from the target. All qualities within this tolerance level are considered to be acceptable.

The objective is to generate a plan that meets these targets by utilising the provided equipment and machines. However, this problem is even more complicated than is described so far, due to additional time-varying constraints and time-varying objectives.

4 The model

This section presents the mathematical model of the problem proposed above.

4.1 General notations

The following basic sets are defined for the mine Mine:

- Blocks $B = \{B_1, \ldots, B_{|B|}\}$ in the orebody. Each block is determined by the following characteristics: geometrical coordinates of the block are given; tonnage(B_i, el, gr) is a tonnage of material el of grade gr.
- Benches $\text{Ben} = \{\text{Ben}_1, \ldots, \text{Ben}_{|\text{Ben}|}\}$. A bench is represented as a set of blocks $\text{Ben}_i = \{x : x \in B\}$. Each bench has a *toe block* defined as $b_0(\text{Ben}_i)$.
- Pits $P = \{P_1, \ldots, P_{|P|}\}$. A pit consists of set of benches $P_i = \{x : x \in \text{Ben}\}$.
- Model $M = \{M_1, \ldots, M_{|M|}\}$. A model contains a set of pits $M_i = \{x : x \in P\}$. The top level hierarchy is the mine which contains a set of models $\text{Mine} = \{x : x \in M\}$.
- Areas $A = \{A_1, \ldots, A_{|A|}\}$. An area is a set of blocks $A_i = \{x : x \in B\}$. Additionally, an area may have an earliest start date esd(A_i) and latest end date led(A_i). Each area has a set of dependent areas defined as $\text{Dep}(Ai) = \{x : x \in A\}$. If area A_j is dependent on area A_i, then

$$\forall B_k, B_l : (\alpha(B_k) = A_i, \alpha(B_l) = A_j) \Rightarrow t_e(B_k) < t_s(B_l) \quad (1)$$

where $t_s(B_i)$ is a start time of digging block, B_i, $t_e(B_i)$ is an end time of digging block B_i and $\alpha(B_i)$ is an area to which the block belongs. This dependency relation is expressed as $A_i \prec A_j$.
- Time periods $T = \{T_1, \ldots, T_{|T|}\}$. Each time period is determined by its start date and end date. Each period has information about various aspects such as the capacities of diggers, crushers, targets and limits.
- Diggers $D = \{D_1, \ldots, D_{|D|}\}$. Each digger has own specific characteristics. cost(D_i) is a cost of operating a digger. speed(D_i) is a digger tramming speed. location$(D_i) \in B$ is an initial block of the digger. rate(D_i, T_j) denotes how many tonnes of ore a digger can excavate per hour. utilisation(D_i, T_j) means effective utilisation, which determines the percentage of the maximum rate the digger can operate in the real world due to day/night shifts, maintenance and other factors. Speed and cost remain fixed during the whole process, but utilisation and rate change across time periods.
- Crushers $C = \{C_1, \ldots, C_{|C|}\}$. Each crusher has its own characteristics. Its location is given by geometrical coordinates. capacity(C_i, T_j) is a feed capacity, i.e. the number of tonnes it can process per hour. utilisation(C_i, T_j) means effective utilisation, which determines what percent of the maximum rate the crusher can operate in the real world.

$Q^k(T_i)$ is a k-th quality target for time period T_i with the tolerance q_i. For each time period T_i, there are total of $K(T_i)$ targets. $\Omega(T_i)$ is a tonnage target for time period T_i.

4.2 Objectives of the optimiser

Taking into account all the dependency, capacity and other constraints described above, the problem becomes a highly complex combinatorial optimisation problem. The system that addresses the described mining problem has been implemented with an optimisation component based on a population-based metaheuristic. This system was built for a large mining company, and it is currently in the process of being integrated into the existing IT structure. The key features of the system are as follows:

- *Meeting targets*. The system is taking into account all constraints, configuration and targets and on that basis produces a 5-year plan for the way to utilise diggers, plants and other equipment, and for the block at which each digger should operate at a given time.
- *Trade-off analysis*. There are some configurations where it is not possible to satisfy both tonnage and quality targets. Then, the system produces a set of solutions which show different trade-offs in tonnage and quality.

– *Manual changes and what-if scenarios*. Numerous business rules are a predominant feature of many real-world systems. It is not always possible to incorporate all business rules into the software. That is why for any given solution produced by the system, the operator can manually override decisions made by the system and see the impact of the changes instantly. Apart from the flexibility of manual changes, this also reveals another interesting feature of the application: the ability to analyse various what-if scenarios. In a matter of minutes, the user can see the result of adding or removing a digger or building another crusher on the mining site. Mathematically, the goal is to find a vector $X = \{X_1, \ldots, X_{|B|}\}$ that minimises

$$\min \Delta_T = \sum_{T_i \in T} |\Omega_T(T_i) - \Omega(T_i)| \tag{2}$$

and

$$\min \Delta_Q = \sum_{T_i \in T} \sum_{1 \le k \le K(T_i)} \times \rho(k)\chi(|Q_T(T_i) - Q^k(T_i)| - q_i) \tag{3}$$

Here, each element of X is a tuple X_i = (st, digger, block) corresponding to digging start time, digger and block to be dug, respectively. The end time can be calculated in the following way:

$$\text{end}T(X_i) = \text{start}T + \text{weight(block)}/\text{rate(digger}, T_k) \tag{4}$$

where weight(block) corresponds to the physical weight of the block, and rate(digger, T_k) defines the rate (in tonnes per hour) in time period T_k during which the digging of the block occurs.

$\Omega_T(T_k)$ is a total tonnage of ore excavated during time period T_k and defined as follows: for each time period T_k,

$$\Omega_T(T_k) = \sum_{X_i \in T_k} \text{tonnage}(X_i.\text{block, el, grade}) \tag{5}$$

Thus, the first objective is to minimise total deviation from the desired tonnage. The second objective defined by Eq. 3 is to minimise total deviation from the desired quality across all time periods. In Eq. 3, $\chi(x)$ is a positive indicator function which returns 0 if x is negative and the actual argument x otherwise. Since there are several quality targets, each target has its own priority which is determined by the coefficient $\rho(k)$.

4.3 Constraints

This subsection describes different types of problem-specific dependencies and business rules. Most of them fall into one of two categories: dependency constraints and capacity constraints. The current formulation of the problem recognises two types of dependency constraints: *block dependencies* and *area dependencies*. Block dependency is the basic constraint of the problem. There are three types of block dependencies:

– *Clear above dependencies*. Due to the physical nature of the mine, a block cannot be excavated before the block on top of it is cleared. It can be thought of as a vertical dependency.
– *Clear ahead dependencies*. This is a horizontal type of dependency. When digging the bench, all side blocks should be excavated first before getting to the inner ones.
– *User-defined dependencies*. Sometimes, due to certain business rules or specific circumstances, there is a need to define custom dependencies between blocks.

Another type of dependency constraint is area dependency. An area is basically an arbitrary collection of user-defined blocks. It can be an arbitrary collection of blocks, but usually, it involves a few adjacent pits that share a certain characteristic. The concept of an area gives a flexibility to the define custom dependencies. To indicate that one part of the mine should be excavated before another one, these two parts should be marked as different areas and connected by the dependency.

Capacity constraints, on the other hand, are the type of constraints that deal with the physical workload of machines and equipment. All of them are hard constraints, meaning that violation of any of them will render a solution infeasible. A number of these hard constraints are as follows:

– *Digger capacity constraint*. Each digger can excavate only a certain tonnage per time period.
– *Truck constraints*. There are speed and capacity constraints for each type of truck. Furthermore, the speed of the truck depends on whether it is loaded with ore, waste or unloaded and also on the slope of the road.
– *Crusher constraints*. As mentioned before, crushers can operate with a limited speed and have processing capacity limits.
– *Stockpile constraints*. Each stockpile has a limited capacity.
– *Wastedump constraints*. Each wastedump has a limited capacity.

There are also some other constraints defined in this application, for example:

- *Pit tonnage limits*. There is a limit on how much maximum tonnage can be excavated from each pit. Some pits may not have any limits.
- *Digger proximity constraint*. Due to safety reasons, two diggers cannot operate too close to each other.

Mathematically, these constraints are the following:
$\lambda_{AH}(B_i, B_j)$—clear ahead dependency, 1 if dependency violated, 0 otherwise
$\lambda_{AB}(B_i, B_j)$—clear above dependency, 1 if dependency violated, 0 otherwise

The solution made with the vector X should comply with the clear ahead and clear above constraints, so

$$\sum_{i,j} \lambda_{AH}(B_i, B_j) + \sum_{i,j} \lambda_{AB}(B_i, B_j) = 0$$

Also, no two blocks could be executed simultaneously on the same digger:

$$\forall i, j | X_i.\text{digger} = X_j.\text{digger} X_i \cap X_j = \emptyset$$

Here, $X_i \cap X_j$ operation returns the intersection between time intervals X_i and X_j.

All the constraints described above are hard dependencies.

5 Approach

This section describes a proposed algorithm for the described problem. Firstly, the representation of the solution is described, then the evaluation quality score is explained and, lastly, the three stages of the proposed optimiser are shown.

5.1 Structure of a solution

The structure of a solution for the optimiser is as follows. For each time period, a table stores a set of benches that are scheduled to be dug in this time period. If only a part of the bench is scheduled during the selected time period, then the fraction of the bench is recorded. Table 1 illustrates this concept. The first column in this table lists all available benches. The other columns represent time periods when a particular bench is scheduled. For example, bench B3 is scheduled for digging in the third time period, and bench B4 is scheduled as follows: 60 % during the fourth and 40 % during the fifth time periods. We refer to this data structure as a tonnage schedule (TS). During the execution of the optimiser, the tonnage schedule is initialised and later modified at every iteration of the process; special operators make sure that clear above and area dependencies are not violated (i.e. that the solution is feasible from the perspective of these dependencies).

The size of the search space is quite significant; for example, for 20 time periods (this would correspond to 5 years, assuming quarterly time periods), with 500 benches available in total and 150 benches available per time period (due to various dependencies) and at least 10 benches fully allocated for a time period, the number of possible solutions is in the order of 10^{240}. Note that if the number of allocated benches is higher (say, 25), the size of the search space grows dramatically much further. Note also that, to compare the number of solutions in the search space, the current calculations of the number of atoms in the observable universe is close to 10^{80}.

Table 1 Solution representation

Bench	TP1	TP2	TP3	TP4	TP5	TP6	TP7
B1	1						
B2		1					
B3			1				
B4				0.6	0.4		
B5						1	
B6						0.5	0.5
B7							1

5.2 The quality score of the solution

The quality score of a solution is based on a combination of various penalty functions:

- Quality violation penalty. This penalty is calculated as a sum of deviations from the desired qualities of each quality target.
- Tonnage violation penalty. This penalty is calculated as a deviation of actual tonnage from the desired tonnage.
- Digger workload violation penalty. This penalty represents how much total digger workload exceeds the maximum digger workload. If total digger workload does not exceed the maximum workload, the penalty is 0.
- Haulage workload violation penalty. This penalty represents how much total haulage workload exceeds the maximum haulage workload. If total haulage workload does not exceed, the maximum workload the penalty is 0.

5.3 The optimiser

The optimiser used in the system consists of three modules:

- Module 1: initialisation
- Module 2: problem-specific evolutionary algorithm
- Module 3: decoder

which are executed sequentially. These three modules are discussed in turn.

5.3.1 Module 1: initialisation

As the size of the search space for the planning problem is prohibitive, module 1 is responsible for creating the initial

set of candidate solutions. The main parameters that control the execution of this algorithm are n, population size; m, the number of offspring produced by a parent; k, the elitism factor (the number of the best individuals to maintain); and g, the diversity factor (to generate a pool of diverse solutions).

This module initialises the pool of candidate solutions in a few stages. It starts with a pre-distribution that considers the frozen time period, where all parts of the solution that fall into the frozen time period are copied from a previously generated solution (if it exists) into the new one. Then, a *pre-distribution of dependency chains* is performed by analysing the pits' dependency chains for possible bottlenecks. A certain number of pits are selected and pre-distributed over several time periods to release bottleneck dependencies early. Many different arrangements (e.g. a different number of selected pits, different distributions over time periods) are possible. At this stage, we have a population of size n, which consists of n (incomplete) solutions created during the first two stages. During the third stage of initialisation, the module allocates benches to all considered time periods taking into account (a) all available benches from all pits that are not held by any dependencies (area dependencies and clear above dependencies), (b) pit limits (if specified), (c) the target tonnage and (d) the target quality. Such allocation is run in a loop whilst at least one of the following conditions holds:

1. Actual tonnage is greater than that desired.
2. Haulage workload capacity is exceeded for the benches scheduled for the current time period.
3. Digger workload capacity is exceeded for the benches scheduled for the current time period.

By eliminating these violations, the algorithm makes sure that, after exiting the loop, none of the hard constraints are violated, i.e. the solutions are feasible. Further, spare digger capacity is assigned to dig waste benches by rescheduling some waste benches—during such rescheduling, the algorithm checks that no haulage workload and digger workload limits are violated.

The above steps are performed m times for each individual in the current population to produce an array of solutions. Elitist selection is then performed on this array with the elitism factor of k, i.e. k individuals with the best fitness are then selected from this array as a population for the next time period. This means that for all time periods except for the first, k solutions are selected from $m \times k$ candidate solutions for further processing. The quality measure of an individual solution is based on penalties (as discussed earlier). The selection algorithm maintains the diversity of the population by eliminating solutions that are too similar to other solutions—this is controlled by the diversity factor g.

At the final stages of the initialisation, reclaiming from stockpiles is performed in order to bring the quality closer to the desired levels. If the actual quality is within the desired quality tolerance range, this stage is skipped. The tonnage to be reclaimed is calculated in such a way that digger workload and haulage workload are not violated.

5.3.2 Module 2: problem-specific evolutionary algorithm

The main parameters that control the execution of this algorithm are the following: n_{ea}, population size; m_{ea}, the number of offspring per parent; k_{ea}, the elitism factor (the number of the best individuals to maintain); and g_{ea}, diversity factor. A population of solutions generated during the initialisation phase is fed into the evolutionary algorithm that improves the population during the iterative process by applying variation operators to existing candidate solutions thus generating new offsprings. Sample variation operators include the following:

- *Move right operator.* This operator reschedules a selected bench to the next time period. The operator is performed in such a way that no clear above rule is violated.
- *Move left operator.* This operator does the same job as the previous one except that it reschedules benches to the previous time period instead.
- *Repair operators.* These operators restore the feasibility of the solution after modification.

The evolutionary algorithm uses steady-state tournament selection with operators that have the adaptive probabilities of being applied. The diversity of population is preserved by ignoring solutions that lie within a certain distance of each other.

5.3.3 Module 3: decoder

The decoding phase consists of two parts:

- converting from tonnage schedule into bench schedule
- converting from bench schedule into block schedule

The module uses solutions generated by the previous module for making decisions on digger assignments and further material movements. During this stage, only deterministic decisions are made. For example, the module decides to which crusher, stockpile or wastedump the excavated material (low grade, high grade, and waste) should be sent.

6 Functionality

This section describes the main software functionality of the system. It has been developed in Java. The developed

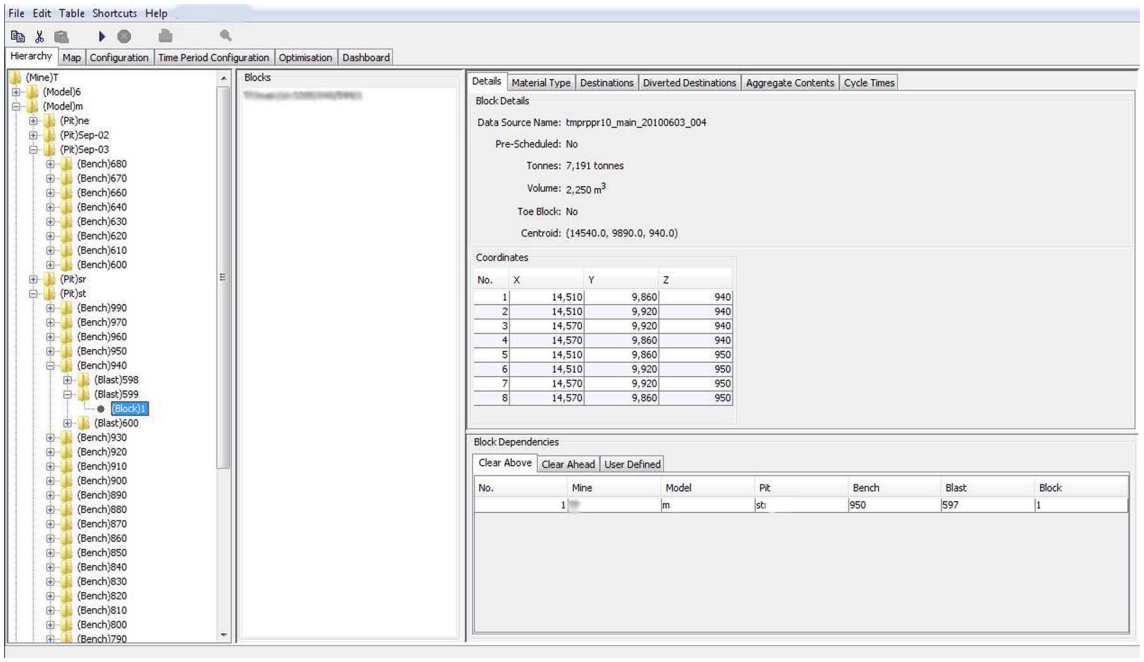


Fig. 1 Screenshot presents the hierarchy tab with coordinates and tonnages of a selected block and its dependencies

software has several main tabs: Hierarchy, Map, Configuration, Optimisation and Dashboard. These will be described in the following subsections.

6.1 Hierarchy tab

The hierarchy tab presents the mine in the hierarchical view: mine-model-pit-bench-block. As has been described in the previous sections, the basic unit of the mine is a block. Each block is specified by its geometrical coordinates and tonnages of certain material type and grade (for example, high-grade iron ore), which are exported from file. From the geometrical coordinates, clear above and clear ahead dependencies are calculated.

The road network of the whole mine can be exported from the file as well. Each road consists of road segments which are represented by the geometrical coordinates. All parts of mine, plants, crushers, stockpiles and wastedumps are connected by road network, therefore it is possible to calculate the shortest distances between each block and destinations. The user of the software can manually override the shortest path destination. These kind of business rules of overriding decision of the system are very common for the real-world problems, and they are normally not considered in the classical research problems. Figure 1 shows the screenshot of the hierarchy tab with one block selected. On the right side of the screen, the user is presented with all characteristics of the block including its dependencies.

6.2 Map tab

The map tab presents the user with the interactive 3D map of the mine, road network and ore destinations fully constructed from the raw geometrical coordinates. The software gives the ability to "fly" over the mine, zoom in and out, and explore the structure. It gives a user-friendly way of assigning toe blocks, setting destinations and visually exploring contents of the block. Different colour schemes visualise the mine by concentration of iron in the blocks (the higher concentration, the more intense is the colour), by pit, by model, by time period it is planned to be mined in, and depending if the block is going to the crusher of stockpile. Figure 2 shows the screenshot of the map tab. The main part of the screen is a described 3D interactive map, and the right panel allows the user to hide or show various elements of the map, change colour scheme, filter specific parts of the mine, etc.

6.3 Configuration tab

This is one of the most important tabs in the software as the configuration of all different parts of the mine and optimiser is taken place here (Fig. 3). Configuration is split into several categories: Dependencies, Plants, Mobile Equipment, Wastedumps, Stockpiles, Scenario and Time Period Configuration. This part of the system contains most of the constraints and business rules. The Time Period Configuration category defines dynamic constraints, i.e. they

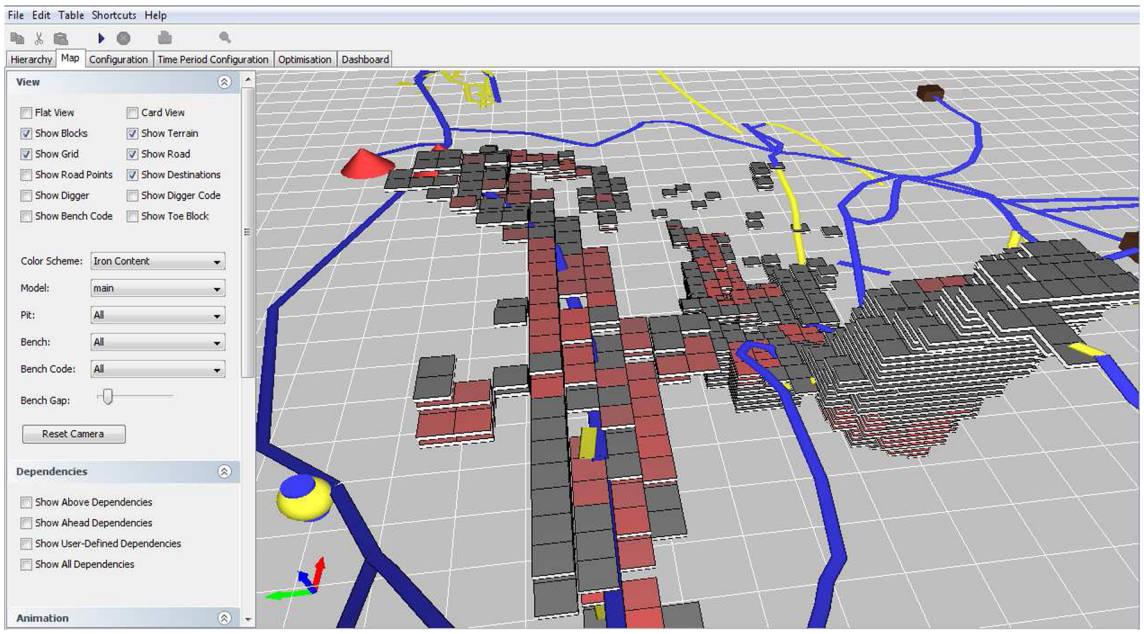


Fig. 2 Screenshot presents the 3D view of the mine and map controls

change over the time periods. These two types of constraints configuration described below.

6.3.1 Static configuration

The *Dependencies* in the system are Area dependences, Clear Above and Clear Ahead dependencies which are introduced in Section 3. The first step in configuring an area is to create a master record in the area configuration screen. Once the area has been defined, the planner can then configure the area within the map screen by adding a set of blocks, a bench, a pit or even a model. The next step is to configure the area dependencies based on the defined areas. The Dependencies category allows also to change settings on values of clear above and clear ahead dependencies (for example, how many blocks should be cleared before a certain block can be mined in clear ahead dependencies screen). Additionally, each area can be set to be mined after

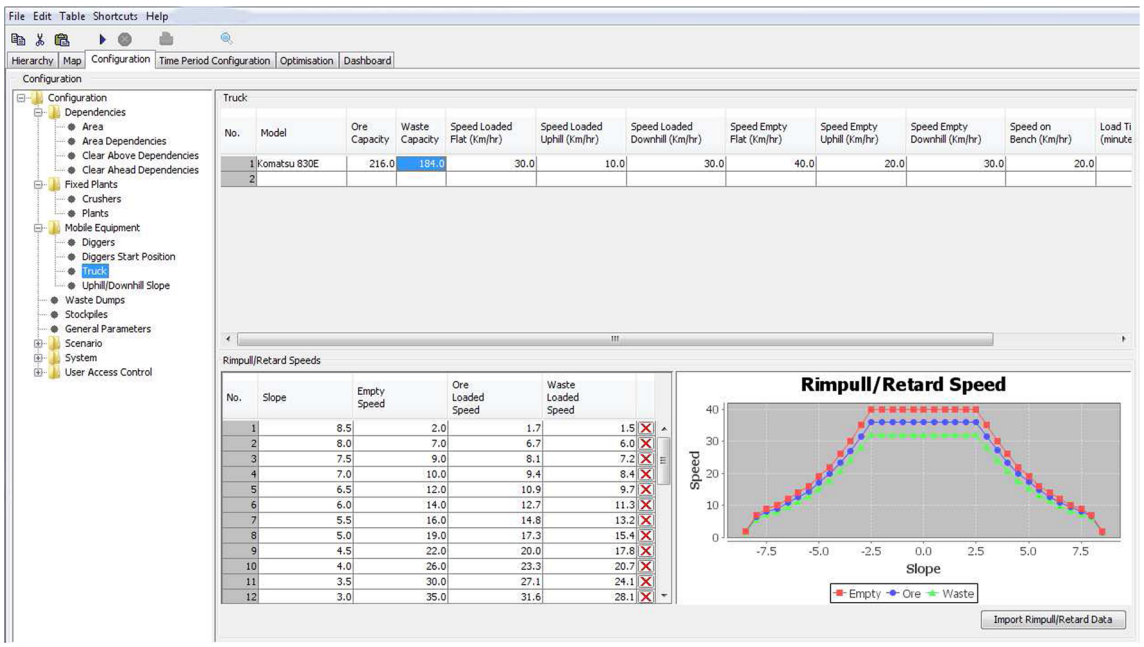


Fig. 3 Screenshot presents the configuration tab

a certain time period, before the certain time period or in between two time periods due to a certain business rule. The dependencies concept in mining is very essential and lets schedulers to restrict excavation of one part of the mine before the other one is excavated or excavate it before or during certain time periods. From the optimisation point of view, it significantly constraints the search space and, in many cases, makes finding even one feasible solution a very challenging problem.

The *Plant* category lets adding or removing plants and crushers available in the system. At the static stage of the configuration, plants and crushers are defined by their coordinates on the plane.

The *Mobile equipment* configuration category lets the scheduler set the number of diggers, their type, speed and operating cost. Each digger may have different settings, so the configuration is not homogeneous. Additional constraint for diggers is their starting position on a certain block. As the travelling speed of each digger is very limited (typically around 2 kph), each digger is limited to excavate only a certain part of the mine because it is normally not optimal to spend most of time in time period for transporting the digger.

Transportation of the ore from the mine to crushers, plants and stockpiles is done via trucks over the road network. This is represented in the system as fleets of homogeneous trucks. Each fleet defines parameters for each of the trucks in the fleet. Some of the parameters include capacity of truck loaded with ore and capacity loaded with waste (as ore and waste have different densities). The speed of each truck depends on the slope of the road and type of material loaded on the truck. The calculation of speed is based on rimpull curves—a mathematical curve used to lookup the speed based on the slope and load of the truck. As road segments have geometrical coordinates, the slope can be easily calculated. The time it takes to travel a certain route of the network can be calculated summing travel times for road segments of that route. Note that the speed of the truck over a road segment is different if the truck is loaded with ore or waste, or it drives empty.

The *Wastedumps and Stockpiles* category lets defining wastedumps and stockpiles that exist in the mine. Apart from their geographical coordinates, the total capacity and opening balance of the structure are given.

The *Scenario* category lets defining various optimiser configurations. If the *Continue Equipment Utilisation* after targets achieved flag is blank, then the optimiser will not attempt to plan any unused capacity that is available once the specified targets have been achieved. If this flag is selected, then it will attempt to utilise diggers and trucks on waste blocks which will not affect the tonnage targets. If the *Allow stockpile reclamation* flag is not selected, then the system will not reclaim any tonnage from the stockpile to the crusher in generating a plan. If the *Constrain optimisation by haulage capacity* is blank, then the optimiser will not attempt to constrain the optimiser by exceeding specified truck capacity. If this flag is selected, then the optimiser will not exceed truck capacity for each time period. If the *Use top ranked quality target* is selected, the optimiser will only focus on achieving the top weighted quality target for each time period. If this flag is not selected, then it will attempt to reach all quality targets in priority wet by the weighting settings in time period configuration. If the *Enforce Area Dates* is selected, then the optimiser will respect the area dates set in the configuration screen. If this flag is not selected, then it will ignore the area dates in the optimisation process. If the *Enforce Pit Tonnage Limits* is selected, then the optimiser will respect the specified limits set in the time period configuration screen. If this flag is not selected, then it will ignore the limits in the optimisation process.

There is the capability of setting a time where the optimiser will not optimise prior to this date. Therefore, the mine sequence that exists in the plan will remain unchanged up until the date specified. Note that there is an option of No Freeze where the optimiser will reschedule from the current date and not apply any freeze during the optimisation process.

There is the capability of setting a time where the optimiser will not optimise prior to this date. Therefore the mine sequence that exists in the plan will remain unchanged up until the date specified. Note, there is an option of No Freeze where the optimiser will reschedule from the current date and not apply any freeze during the optimisation process.

6.3.2 Dynamic constraints configuration

The Dynamic Constraints configuration can be done by configuring each time period with its own parameters (Fig. 4).

The following parameters can be changed over the time periods: rate and utilisation of diggers, feed capacity (in tonnes per hour) and utilisation of crushers, input capacity (in tonnes per hour) and effective utilisation of plants, capacity of wastedump (it is limited per time period and different from the total capacity of the wastedump), maximum and minimum capacity of stockpiles, number of trucks in the fleet and their effective utilisation. Additionally, each pit has a minimum and maximum tonnage that should be excavated from it.

The main part of the time period configuration is the configuration of targets. As has been described in Section 3, two types of targets exist: tonnage targets and quality targets. Each time period has its own setting for both.

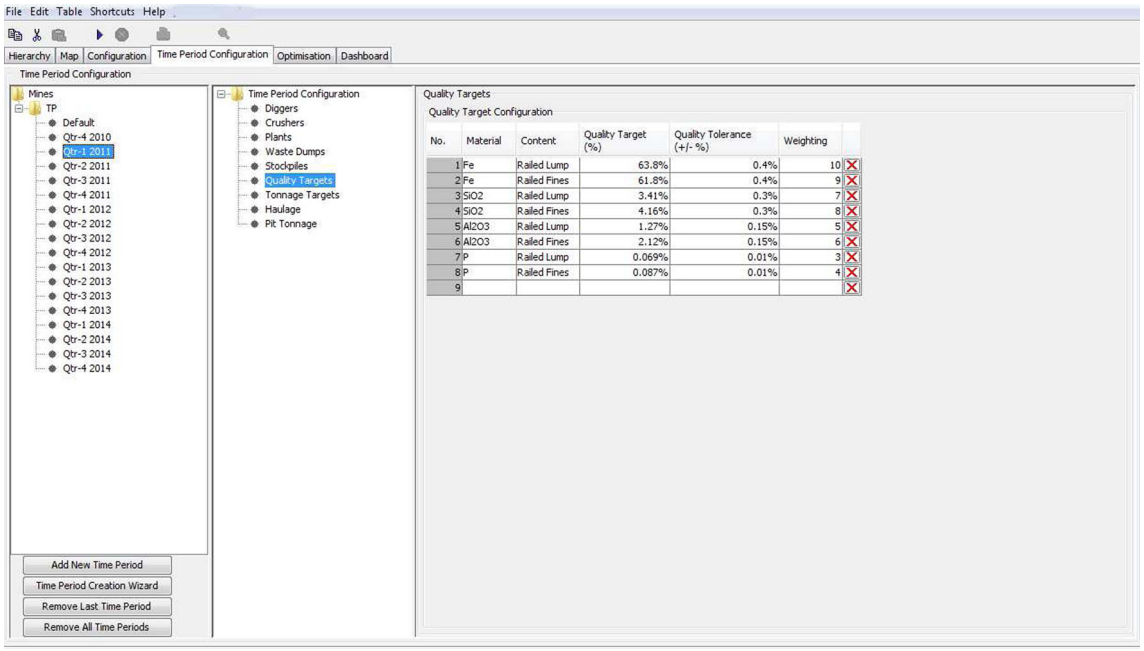


Fig. 4 Screenshot presents the dynamic configuration by time period tab

6.4 Optimisation tab

The Optimisation tab presents optimisation results. It has two main graphs that show actual and desired tonnage and quality targets. The quality graph has also two tolerance lines, so it is very obvious if the solution is within the tolerance range or not. Figure 5 shows the screenshot of this tab. The top part shows detailed breakdown of parts of the mine to be dug during each quarter, while the bottom part displays desired and actual graphs on tonnage and quality.

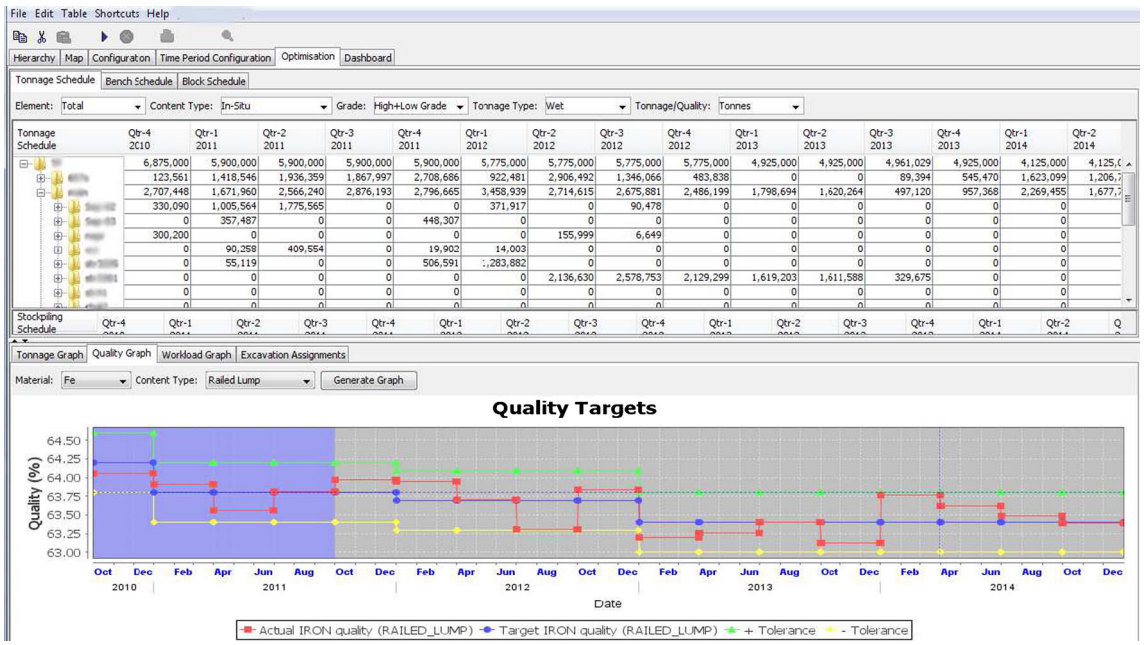


Fig. 5 Screenshot of the optimisation tab with the performance graphs and detailed schedule

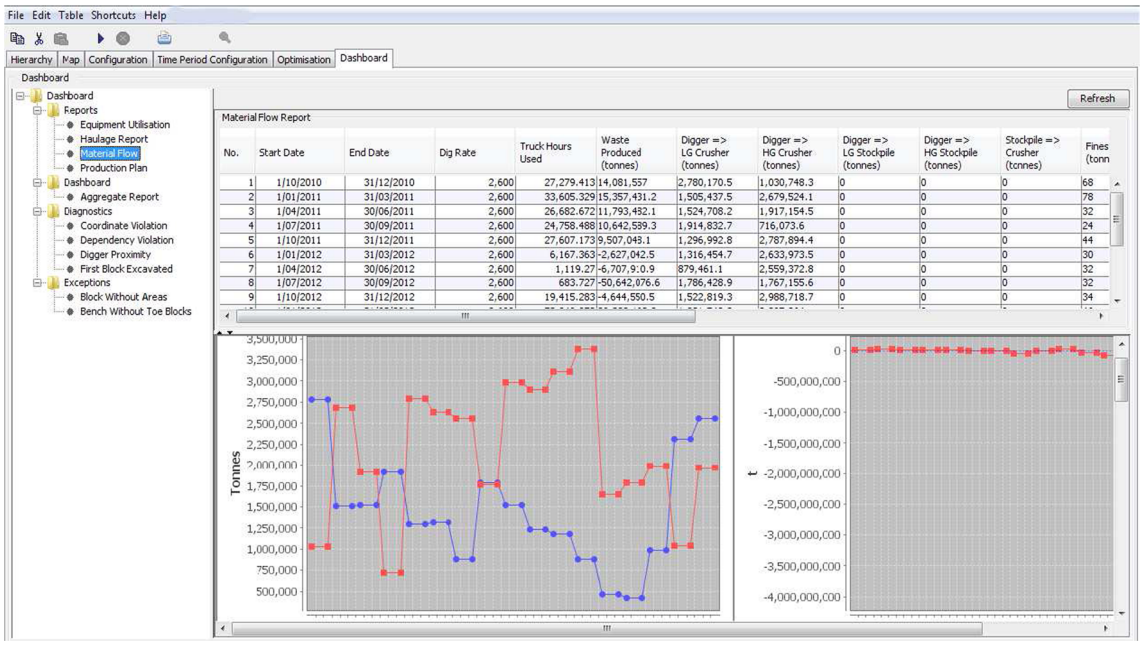


Fig. 6 Screenshot of the dashboard tab

Fig. 7 Result of the optimiser

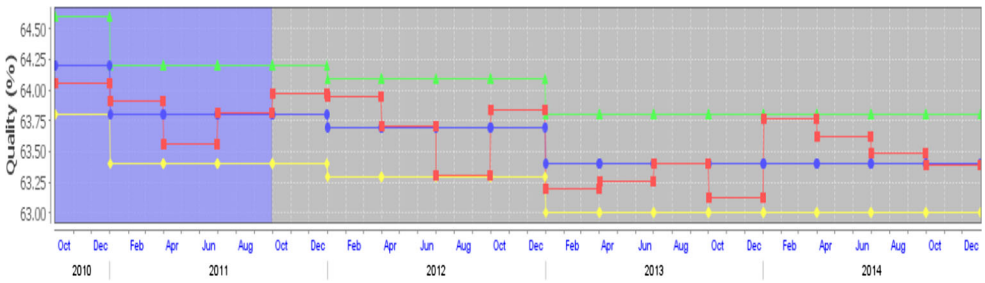
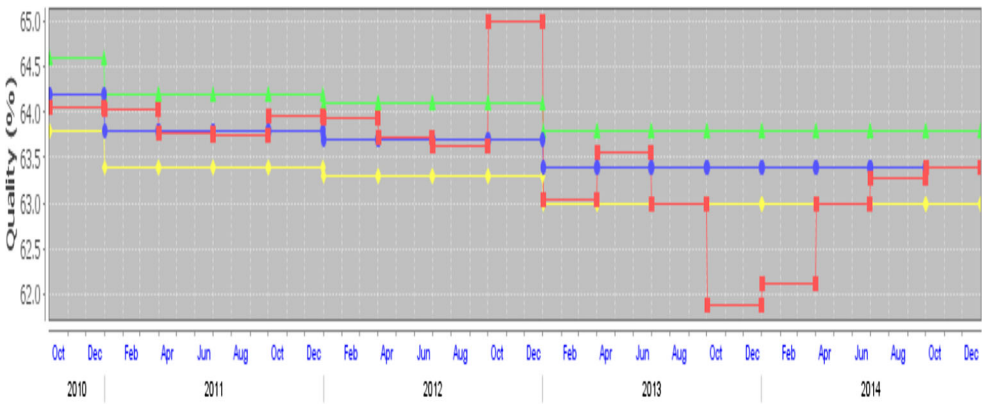


Fig. 8 Result of the optimiser. Quality of the first objective with the limited truck capacity



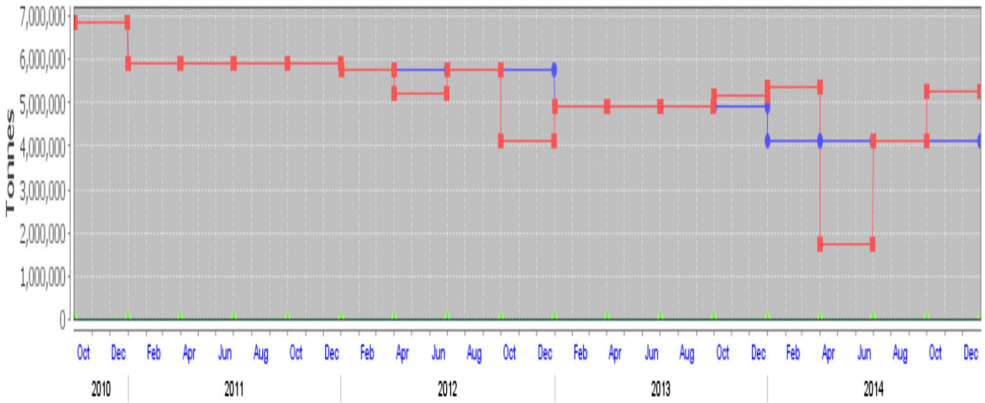


Fig. 9 Result of the optimiser. Tonnage graph with the limited truck capacity

6.5 Dashboard tab

The last tab, Dashboard, presents to the user with the various kinds of reports, showing different KPIs, error and diagnostic information, for example, equipment utilisation, haulage report, various aggregated and material flow reports, coordinate, dependency and other violations and data exceptions (Fig. 6).

6.6 What-if functionality

The Software is allowing the user to configure hypothetical mine and equipment configurations and then run sequencing optimisations and view KPI reports, thereby enabling an evaluation of what would happen if those scenarios were actually implemented. The factors that the user would be able to experiment with under what-if scenarios will be as follows:

- change the location of a stock pile or dump
- change equipment capacity and/or availability
- change quality and tonnage targets
- reaction to events such as slope failure or flooding

7 Results

The implemented optimiser can be run in different modes, with certain optimiser options chosen to be used or not:

1. Continue equipment utilisation after targets achieved, i.e. use all spare digger capacity to dig more blocks and send to stockpiles.
2. Allow or deny stockpile reclamation.
3. Use total truck capacity as a hard constraint.
4. Use top ranked quality target only.
5. Enforce area dates.
6. Enforce or ignore pit tonnage limits.

Each one of these points can be selected or deselected independently of others. This gives a flexibility in varying different types of limitations during the run (total number of choices $2^6 = 64$).

Each of the optimiser executions has been run on the live mine data provided by the mining company. The production schedule developed by the company's expert schedulers team has been analysed as well. However, a much more narrow quality tolerance boundaries of 0.4 % has been enforced to our system in contrast with the expert quality tolerance

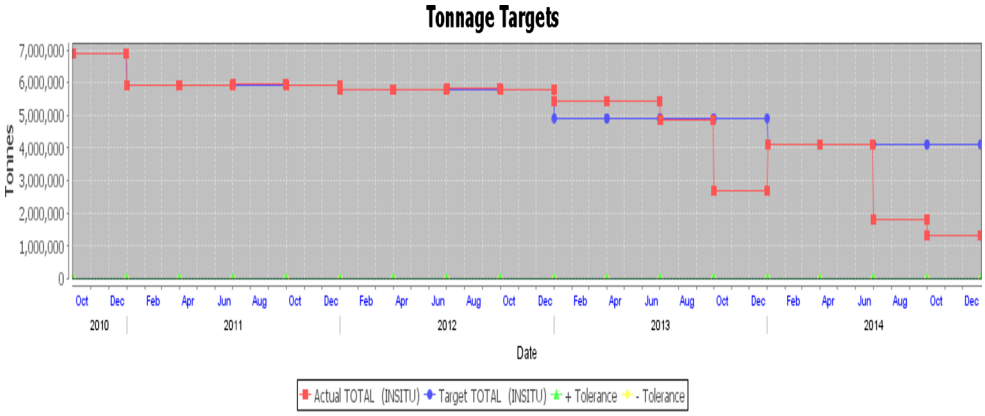


Fig. 10 Result of the optimiser. Tonnage graph

Fig. 11 Result of the optimiser. Graph of quality of iron

of 1.5 %. The software produced a valid schedule within approximately 5 min and has been evaluated as fully working by the expert team. The expert team used their own software suite based on the XPAC mining system that has various scripts which help to find a solution. To build a solution with their system, it takes effort of several systems from the suite to produce even one solution. An approximate time to build one solution can be around 1 day which is significantly slower than the software based on our metaheuristics. Any variation in the schedule, change of business rules or testing what-if scenarios would incur a full schedule creation process.

Figure 7 shows the quality results of the first quality target with optimiser options 1, 2 and 4 selected. Tonnage targets were matched with zero total deviation from the desired tonnage. This configuration produced best results on both tonnage and quality objectives. The blue line on the graph represents the desired quality, the red line shows the actual quality of the schedule, and green and yellow lines show higher and lower tolerance levels (of 0.4 %).

The addition of optimiser option 3 (enforce truck capacity) has an impact on the solution quality. Tonnage and quality graphs are shown on Figs. 8 and 9. This was achieved with the setting of 31 standard trucks working fully per time period. This result shows that it is very hard (if possible) to produce good enough schedule with this amount of trucks and the current configuration. Previous result of good schedule without constraining truck capacity and the current result should tell schedulers to change certain values, most likely increase number of trucks in the problematic time periods and then rerun the optimiser again to see the result. This experiment shows the strengths of the software of quickly analysing various what-if scenarios.

The next experimental run of the optimiser allows utilising the stockpiles, i.e. options 1 and 2 enabled. This optimiser configuration uses all quality targets prioritising them by rank. Here, we set priority to high quality and low quality of iron ore over other materials. This will assign the highest coefficient to iron ore quality deviation whilst evaluating the solution. In the real world, schedulers are interested in both optimisations: the optimisation that considers only the first quality target and the optimisation that considers all quality targets with priorities.

Results presented in Figs. 10, 11 and 12 show that considering all quality objectives makes the first objective worse whilst improving on other objectives. This converts the current problem from a two-objective problem into a nine-objective problem as the default configuration consists of eight quality targets per quarter.

Fig. 12 Result of the optimiser. Graph of quality of silicon

As with the previous optimiser run, the results give schedulers knowledge of what happens with the current configuration. By changing certain values and rerunning the optimiser, they were able to see the impact of the change within 5–10 min. Previously, it was taking at least 1 day to see the results of the change. Additionally, client utilised several systems to produce the result and had to feed the output of one system manually into another as well as manually configure each system for current settings. This, certainly, would increase the chance of making an error.

A very powerful tool for the guiding optimisation in the desired direction is an ability to set minimum and maximum pit limits. However, outcome of some unthoughtful settings can be very dangerous as slowing down certain parts of the mine can cause other parts of the mine to be blocked due to dependencies. A feature of using area dates is another one that lets an experienced scheduler to force the system for the certain result, but again, each of these configurations may cause complications of not meeting the targets.

The described results show two main strengths of the system that most of other systems do not have:

- quick optimisation
- powerful what-if analysis

The next section describes the main functionality and configuration settings of the system.

8 Conclusion and future works

In this paper, we considered a highly constrained mining problem. Firstly, the description of the problem has been presented, the approach based on metaheuristics followed that, and then description of functionality of the software along with its configuration and constraints has been described.

Each of the configuration categories presented in Section 6 present additional constraint to the problem which makes it extremely hard to find even a feasible solution. The complexity of the problem can be thought of from the several perspectives. If we take, for example, such NP-hard problems as travelling salesman problem, vehicle routing problem, various scheduling problems and other classical optimisation problems, these problems present very hard combinatorial complexity. However, normally, they are not very constrained which makes them hard to apply in practical applications. Real-world problems usually have an additional layer of complexity—complexity by constraints. In addition to the enormous number of constraints, they are also non-linear and very often change over time. The acceptance usage of the presented application by a technologically highly advanced top-tier enterprise shows that the methods of computational intelligence are appropriate to solve highly constrained problems such as mining.

This work concentrated on optimal scheduling within a single mine. However, the concepts described here can be extended to multiple mines. This problem is known as integrated planning in the mining industry. Our future work will focus on the extension of the current approach and investigate alternative methods of addressing the described problem.

Acknowledgments We thank the anonymous reviewers for their very insightful comments. This work was partially funded by the ARC Discovery Grant DP0985723 and by grants N 516 384734 and N N519 578038 from the Polish Ministry of Science and Higher Education (MNiSW).

References

1. Stadtler H, Kilger C (2008) Supply chain management. Springer, Berlin
2. Michalewicz Z, Fogel DB (2004) How to solve it: modern heuristics, 2nd edn. Springer, Berlin
3. Caccetta L, Hill SP (1999) Optimization techniques for open pit mine scheduling. In: International congress on modelling and simulation
4. Caccetta L, Hill S (2003) An application of branch and cut to open pit mine scheduling. J Glob Optim 27:349–365. doi:10.1023/A:1024835022186
5. 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 37:1641–1647. doi:10.1016/j.cor.2009.12.008
6. Lizotte Y, Elbrond J (1982) Optimum scheduling of overburden removal in open pit mines. CIM Bull 75:154–163
7. Yun Q, Yegulalp T (1982) Optimum scheduling of overburden removal in open pit mines. CIM Bull 75:22–31
8. Smith ML, Youa TW (1995) Mine production scheduling for optimization of plant recovery in surface phosphate operations. Int J Min Reclam Environ 9:41–46
9. Dagdelen K, Johnson T (1986) Optimum open pit mine production scheduling by lagrangian parameterization. In: 19th APCOM symposium of the society of mining engineers, AIME, New York
10. Akaike A, Dagdelen K (1999) A strategic production scheduling method for an open pit mine. In: Proceedings of the 28th international symposium on the application of computers and operations research in the mineral industry. Golden, Colorado
11. Kawahata K (2006) A new algorithm to solve large scale mine production scheduling problems by using the lagrangian relaxation method. Ph.D. dissertation, Colorado School of Mines
12. Busnach E, Mehrez A, Sinuany-Stern Z (1985) A production problem in phosphate mining. J Oper Res Soc 36:285–288
13. Samanta B, Bhattacherjee A, Ganguli R (2005) A genetic algorithms approach for grade control planning in a bauxite deposit. Taylor & Francis, London
14. Osanloo M, Gholamnejad J, Karimi B (2008) Long-term open pit mine production planning: a review of models and algorithms. Int J Min Reclam Environ 22:3–35
15. Newman AM, Rubio E, Caro R, Weintraub A, Eurek K (2010) A review of operations research in mine planning. Interfaces 40(3):222–245
16. Armstrong M, Galli A (2012) New approach to flexible open pit optimisation and scheduling. Min Technol 121(3):132–138

17. Halatchev RA (2011) Contemporary criteria of open pit long-term production scheduling. In: 35th APCOM symposium
18. Sattarvand J, Niemann-Delius C (2011) A new metaheuristic algorithm for long-term open pit production planning. In: 35th APCOM symposium
19. Chicoisne R, Espinoza D, Goycoolea M, Moreno E, Rubio E (2012) A new algorithm for the open-pit mine production scheduling problem. Oper Res 60(3):517–528. http://or.journal.informs.org/content/60/3/517.abstract
20. Pinedo M (2002) Scheduling: theory, algorithms, and systems. Springer, New York
21. Cheng R, Gen M, Tsujimura Y (1996) A tutorial survey of job-shop scheduling problems using genetic algorithms—I: representation. Comput Ind Eng 30(4):983–997
22. Garey MR, Johnson DS, Sethi R (1976) The complexity of flowshop and jobshop scheduling. Math Oper Res 1(2):117–129
23. Yamada T, Reeves CR (1998) Solving the c_{sum} permutation flowshop scheduling problem by genetic local search. In: The 1998 IEEE international conference on evolutionary computation proceedings, IEEE world congress on computational intelligence.
24. Nowicki E, Smutnicki C (1996) A fast tabu search algorithm for the permutation flow-shop problem. Eur J Oper Res 91(1):160–175. http://ideas.repec.org/a/eee/ejores/v91y1996i1p160-175.html
25. Wang L, Zheng D-Z (2002) A modified genetic algorithm for job-shop scheduling. Int J Adv Manuf Technol 20:72–76
26. Ponnambalam S, Mohan Reddy M (2003) A GA-SA multiobjective hybrid search algorithm for integrating lot sizing and sequencing in flow-line scheduling. Int J Adv Manuf Technol 21(2):126–137
27. Burke EK, Smith AJ (1999) A memetic algorithm to schedule planned maintenance for the national grid. J Exp Algorithmics 4:1
28. Ray T, Sarker RA (2007) Optimum oil production planning using an evolutionary approach. In: Evolutionary scheduling. Springer, Heidelberg, pp 273–292
29. Marchiori E, Steenbeek A An evolutionary algorithm for large scale set covering problems with application to airline crew scheduling. In: Scheduling, in real world applications of evolutionary computing. Lecture notes in computer science. Springer, Berlin, pp 367–381
30. Van Laarhoven PJM (1992) Job shop scheduling by simulated annealing. Oper Res 40:113
31. Astashkin N (1972) Scheduling of mining operations. J Mining Sci 8:424–429. doi:10.1007/BF02497857
32. Rendu J-M (2002) Geostatistical simulations for risk assessment and decision making: the mining industry perspective. Int J Mining Reclam Environ 16(2):122–133