# A hybrid heuristic algorithm for the open-pit-mining operational planning problem

M. J. F. Souza[*,1], I. M. Coelho[**,2], S. Ribas[**,3], H. G. Santos[**,4], L. H. C. Merschmann[**,5]

*Federal University of Ouro Preto, Department of Computer Science, Ouro Preto, Minas Gerais, Brazil 35400-000*

**Abstract**

This paper deals with the Open-Pit-Mining Operational Planning problem with dynamic truck allocation. The objective is to optimize mineral extraction in the mines by minimizing the number of mining trucks used to meet production goals and quality requirements. According to the literature, this problem is NP-hard, so a heuristic strategy is justified. We present a hybrid algorithm that combines characteristics of two metaheuristics: Greedy Randomized Adaptive Search Procedures and General Variable Neighborhood Search. The proposed algorithm was tested using a set of real-data problems and the results were validated by running the `CPLEX` optimizer with the same data. This solver used a mixed integer programming model also developed in this work. The computational experiments show that the proposed algorithm is very competitive, finding near optimal solutions (with a gap of less than 1%) in most instances, demanding short computing times.

*Key words:* Open-pit-mining, Metaheuristics, `GRASP`, Variable Neighborhood Search, Mathematical Programming

## 1. INTRODUCTION

This work deals with the Open-Pit-Mining Operational Planning (OPMOP) problem, which involves to determine extraction rate of material from ore and waste rock pits, and to assign the equipments (shovels and mining trucks) to these pits. The objective is to determine the extraction rate at each pit in a way that production and quality goals are satisfied, and to minimize the number of trucks needed for the production process.

We are considering dynamic truck allocation in the OPMOP problem, that is, the trucks are not fixed to specific pits/or shovels. Instead, a truck can be

[*]Principal corresponding author
[**]Corresponding author
[1]marcone@iceb.ufop.br
[2]imcoelho@iceb.ufop.br
[3]sabir@iceb.ufop.br
[4]haroldo@iceb.ufop.br
[5]luizhenrique@iceb.ufop.br

assigned to different pits, which increases the fleet productivity, allowing smaller fleets to perform the operations.

The problem in focus has the Multiple Knapsack Problem (MKP) as a sub-problem. In fact, the analogy can be made by considering each shovel like a knapsack and the loads (ore or waste rock) of the trucks as the items. In this analogy, the goal is to determine which loads are the most attractive to allocate to each knapsack, respecting its capacity (productivity). Thus, as MKP belongs to the NP-hard class (Papadimitriou and Steiglitz, 1998), OPMOP does too. Since in real cases the decision must be fast and it is unlikely that optimal solutions would be obtained by exact techniques in a short space of time, it is proposed to find sub-optimal solutions for the problem by means of heuristic techniques. The proposed heuristic algorithm is based on the procedures Greedy Randomized Adaptive Search Procedures - `GRASP` (Resende and Ribeiro, 2010) and General Variable Neighborhood Search - `GVNS` (Hansen et al., 2008b,a; Hansen and Mladenovic, 2001; Mladenovic and Hansen, 1997).

These algorithms have been applied with success to solve several hard combinatorial problems (Glover and Kochenberger, 2003). We propose here a hybrid heuristic with the aim of combining good features found in each one of these metaheuristics. From `GRASP` we used the construction phase to quickly produce good quality solutions and accelerate the improvement phase. `GVNS` was chosen due to its simplicity, efficiency and the natural capacity of its local search (`VND` method) for handling different neighborhoods.

To test the efficiency of the proposed heuristic, its results were also validated by using the state-of-the-art commercial optimization software `CPLEX` 11.0.1 applied to a mathematical programming model also proposed in this work.

The contribution of this work is the presentation of a more complete mathematical programming model of OPMOP than those found in literature. This model seeks to more faithfully depict a real operational mining industry environment. Moreover, it presents a new heuristic model not yet found in literature in order to solve the problem in focus.

The remainder of this paper is organized as follows. Section 2 shows the related work. Section 3 describes the problem considered in this work. Section 4 presents a mathematical programming formulation to OPMOP, while Section 5 presents a heuristic approach to the problem in focus. The testing scenarios are described in Section 6, while in the following section, the computational experiments are presented and analyzed. Section 8 concludes the work.

## 2. RELATED WORKS

White and Olson (1986) proposed an algorithm that is the basis for the `DISPATCH` System, which operates in many mines around the world. A solution is obtained in two steps. The first, based on linear programming, handles the problem of ore mixture optimizing by minimizing costs considering the mining rate, the quality of the mixture, the ore feed rate to the plant for beneficiating, and the material handling. The restrictions of the model are related to the production capacity of the shovels, the quality of the mixture and the minimum feeding rate to the processing plant. The second stage of the algorithm, which is solved by dynamic programming, uses a model similar to White et al. (1982), differing from this by using a decision variable for the volume of material transported per hour on a given route, instead of the truck working rate per hour.

Also considered is the presence of storage piles. In this second stage of the algorithm, the objective is to minimize material transportation in the mine.

Sgurev et al. (1989) described an automated system for real-time control of truck haulage in open-pit mines. This system is called `TRASY` and it is designed towards the improvement of the technical-economical indices of the loading-unloading process in open-pit mines where trucks are used as vehicles. The authors described the two ways of organizing the trucks work: on a closed-circuit system and on an open-circuit system, so called dynamic allocation system. The benefits of the open-circuit system are shown and the authors described the four modules of the `TRASY` system: configuration, control, monitoring and report. The authors concluded that the increase of the operation productivity in open-pit mines may be achieved by improving the effectiveness of the loading-haulage process control, so the introduction of automated systems for haulage vehicles control is one way to accomplish this goal. However, this system does not take into account the quality goals of the ore control parameters.

Chanda and Dagdelen (1995) developed a linear programming model that solves the problem of mixed minerals in the short-term planning of a coal mine. The objective function of this model is the weighted sum of three distinct objectives: to maximize an economic criterion, to minimize production deviations, and to minimize quality deviations from the desired values of the control parameters. No allocation for the loading and transport equipment was considered in this model.

Ezawa and Silva (1995) developed a system for dynamic truck allocation with the objective of reducing variability in the levels of the ore and increasing transport productivity. The system uses a heuristic to sequence the trucks in order to minimize changes in the levels. To validate it, the authors used a simulation and the theory of graphs for the mathematical modeling of the mine. Deploying this system transport productivity increased by 8% and management obtained more accurate data in real time.

Alvarenga (1997) developed a program for the optimal dispatch of trucks in the iron mining of an open pit mine, with the objectives of minimizing the queue time of the trucks in the fleet, increasing productivity and improving the quality of the extracted ore. In the work, which is the basis of the `SMART MINE` system widely used in various Brazilian mines, a technique of stochastic optimization was applied, using the genetic algorithm with parallel processing. Basically, the problem is to indicate the best point of tipping or loading and the trajectory for the movement, when there is a situation of choice to be made. The author pointed to productivity gains of 5% to 15%, proving the validity of the proposal.

Merschmann (2002) developed an optimization system and simulation for analyzing the production scenario in open pit mines. The system, called `OTISIMIN` (Simulator and Optimizer for Mining), was developed in two modules. The first is the optimization module where a linear programming model is constructed and solved, while the second is a simulation module that allows the user to use the results obtained by solving the linear programming model as input for the simulation. The optimization module was developed with the aim of optimizing the process of mixing the ores from the mining of several pits in order to meet the quality specifications imposed by the treatment plant and allocating equipments (trucks, shovels and / or excavators) to pits, considering both static and dynamic truck allocation. The developed model does not consider production optimization and quality targets, or reduction of the number of trucks required

by the production system.

Godoy and Dimitrakopoulos (2004) dealt with the open pit mine design and production scheduling problem, with a view to find the most profitable mining sequence over the life of a mine. According to the authors the dynamics of mining ore and waste and the spatial grade uncertainty make predictions of the optimal mining sequence a challenging task. The authors show a risk-based approach to life-of-mine production scheduling, including the determination of optimal mining rates for the life of mine, whilst considering ore production, stripping ratios, investment in equipment purchase and operational costs; and the generation of a detailed mining sequence from the previously determined mining rates, focusing on spatial evolution of mining sequences and equipment utilization. The production scheduling stage uses a specially-developed combinatorial optimization algorithm based on the Simulated Annealing metaheuristic. A new risk-based, multistage optimization process for long-term production scheduling is presented, and the results show the potential to considerably improve the valuation and forecasts for life-of-mine schedule.

Guimaraes et al. (2007) presented a computational simulation model to validate the results obtained by applying a mathematical programming model to determine the mining rate in open pit mines. `LINGO` solver, version 7.0, was used for optimizing the problem and `ARENA`, version 7.0, simulated the solver's solution. Contrary to belief, the modeling demonstrated that by increasing the number of vehicles, the production goal was not met and was further deterred due to increased queue time. Thus, increasing the number of vehicles does not necessarily optimize mining operations.

Boland et al. (2009) dealt with the open pit mining production scheduling problem (OPMPSP). The treated problem consists of finding the sequence in which the blocks should be removed from pits, over the lifetime of the mine, such that the net present value (NPV) of the operation is maximized. Due to the large number of blocks and precedence constraints linking them, blocks are aggregated to form larger scheduling units. The authors investigated the characteristics of the problem and showed how the aggregates can be systematically divided into bins (groups of blocks) so that the solution of the linear programming (LP) relaxation with all processing decision variables fully disaggregated to block level (D-LP) can be recovered from the solution of our compactly disaggregated LP relaxation (B-LP) with processing decisions made at the level of bins. As the number of bins is much smaller than the number of blocks, using their binning approach, D-LP can be solved to optimality for much larger data instances than by a direct disaggregation approach. They showed that their approach can lead to significant improvements in NPV.

## 3. The OPMOP Problem

In the Open-Pit-Mining Operational Planning (OPMOP) problem there are ore pits and waste rock pits. The material extracted by shovels from the ore and waste rock pits is transported by trucks to unloading points (e.g., crusher and waste rock deposit). For the waste rock pits is necessary to meet a recommended rate of mining, while for the ore pits, besides satisfying a recommended rate of mining, we need to fulfill quality requirements of the ore mixture (formed by ore mass extracted from ore pits). These quality requirements correspond to percentages of several ore control parameters (e.g., % $Fe$, % $SiO_2$ and % $P$).

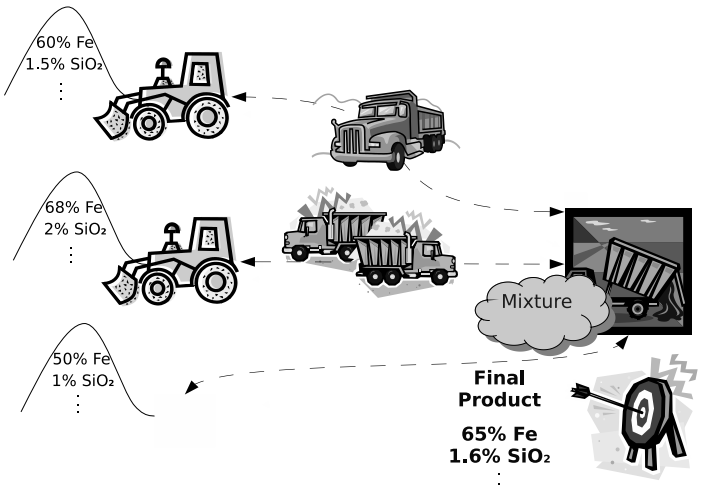


Figure 1: Example of mining operations in a open-pit mine.

It is considered that there are shovels of different productivities and their set is smaller than the number of pits they can be allocated to. Given the high cost of a shovel, a minimum productivity is required to justify its use. Also, the trucks used to transport the material (ore and waste rock) may have different capacities.

This work deals with dynamic truck allocation in the OPMOP problem. In the dynamic allocation system, the trucks are not fixed to specific pits/or shovels. A truck can be directed to different pits, which increases the fleet productivity, reducing the amount of equipment needed to maintain a certain level of production. In this system it is also possible to decrease the time of the queue, since the truck can be allocated to different loading points. The disadvantages of dynamic vehicle allocation are: the demand for a greater number of operations; and a computerized dispatching system for the mining trucks.

In this problem the objective is to determine the extraction rate at each pit in a way that production and quality goals are satisfied, and to minimize the number of trucks needed for the production process. The Figure 1 shows a typical production scenario for the problem here described. In this figure, there are equipments assigned to only two pits. The quantity extracted from each pit defines the quality of the final product (ore mixture), since each pit has a known composition.

## 4. MATHEMATICAL MODEL

This section presents a new mixed integer programming (MIP) model based on goal programming (Romero, 2004) to solve OPMOP. This model refers to production planning for one hour, replicated while there isn't any exhausted pit and operational conditions of the mine remain the same. The objective is to minimize the deviations of the production and quality goals and to reduce the number of vehicles required for the operation.

Let the parameters be:

$O$ : Set of ore pits;
$W$ : Set of waste rock pits;
$F$ : Set of ore and waste rock pits, i.e., $F = O \cup W$;
$P$ : Set of control parameters analyzed in the ore (% Fe, $SiO_2$, etc);
$S$ : Set of shovels;
$T$ : Set of mining trucks;
$O_r$ : Recommended rate of mining for ore (ton/h);
$O_l$ : Minimum rate of mining for ore (ton/h);
$O_u$ : Maximum rate of mining for ore (ton/h);
$W_r$ : Recommended rate of mining for waste rocks (ton/h);
$W_l$ : Minimum rate of mining for waste rocks (ton/h);
$W_u$ : Maximum rate of mining for waste rocks (ton/h);
$\alpha^-$ : Penalty for negative deviation from the production of ore;
$\alpha^+$ : Penalty for positive deviation from the production of ore;
$\beta^-$ : Penalty for negative deviation from the production of waste rocks;
$\beta^+$ : Penalty for positive deviation from the production of waste rocks;
$p_{ij}$ : Percentage of the control parameter $j$ in pit $i$ (%);
$pr_j$ : Recommended percentage for the control parameter $j$ in the mixture (%);
$pl_j$ : Minimum allowable percentage for the control parameter $j$ in the mixture (%);
$pu_j$ : Maximum allowable percentage for the control parameter $j$ in the mixture (%);
$\lambda_j^-$ : Penalty for a negative deviation of the control parameter $j$ in the mixture;
$\lambda_j^+$ : Penalty for a positive deviation of the control parameter $j$ in the mixture;
$\omega_l$ : Penalty for use of the $l$-th truck;
$Qu_i$ : Maximum rate of mining for pit $i$ (ton/h);
$Tx_l$ : Maximum rate of use for truck $l$ (%);
$Sl_k$ : Minimum productivity for shovel $k$ (ton/h);
$Su_k$ : Maximum productivity for shovel $k$ (ton/h);
$cap_l$ : Capacity of truck $l$ (ton);
$ct_{il}$ : Total cycle time of truck $l$ in pit $i$ (min);
$g_{lk}$ : 1, if truck $l$ is compatible with shovel $k$; and 0, otherwise.

Consider also the following variables of decision:

$x_i$ : Mining rate of pit $i$ (ton/h);
$y_{ik}$ : 1, if shovel $k$ operates in pit $i$; and 0, otherwise.
$n_{il}$ : Number of trips that truck $l$ performs to pit $i$;
$D_o^-$ : Negative deviation from the recommended ore production (ton/h);
$D_o^+$ : Positive deviation from the recommended ore production (ton/h);
$D_w^-$ : Negative deviation from the recommended waste rock production (ton/h);
$D_w^+$ : Positive deviation from the recommended waste rock production (ton/h);
$d_j^-$ : Negative deviation of the control parameter $j$ in the mixture (ton/h);
$d_j^+$ : Positive deviation of the control parameter $j$ in the mixture (ton/h);
$U_l$ : 1, if truck $l$ is being used; and 0, otherwise.

Next, the equations (1)-(26) present the MIP model for the problem in focus.

$$\min \sum_{j \in P} \lambda_j^- d_j^- + \sum_{j \in P} \lambda_j^+ d_j^+ + \alpha^- D_o^- + \alpha^+ D_o^+ + \beta^- D_w^- + \beta^+ D_w^+ + \sum_{l \in T} \omega_l U_l \qquad (1)$$

$$\sum_{i \in O} (p_{ij} - pu_j) x_i \leq 0 \qquad \forall j \in P \tag{2}$$

$$\sum_{i \in O} (p_{ij} - pl_j) x_i \geq 0 \qquad \forall j \in P \tag{3}$$

$$\sum_{i \in O} (p_{ij} - pr_j) x_i + d_j^- - d_j^+ = 0 \qquad \forall j \in P \tag{4}$$

$$\sum_{i \in O} x_i \leq O_u \tag{5}$$

$$\sum_{i \in O} x_i \geq O_l \tag{6}$$

$$\sum_{i \in O} x_i + D_o^- - D_o^+ = O_r \tag{7}$$

$$\sum_{i \in W} x_i \leq W_u \tag{8}$$

$$\sum_{i \in W} x_i \geq W_l \tag{9}$$

$$\sum_{i \in W} x_i + D_w^- - D_w^+ = W_r \tag{10}$$

$$x_i \leq Qu_i \qquad \forall i \in F \tag{11}$$

$$x_i \geq 0 \qquad \forall i \in F \tag{12}$$

$$d_j^-, d_j^+ \geq 0 \qquad \forall j \in P \tag{13}$$

$$D_o^-, D_o^+ \geq 0 \tag{14}$$

$$D_w^-, D_w^+ \geq 0 \tag{15}$$

$$\sum_{k \in S} y_{ik} \leq 1 \qquad \forall i \in F \tag{16}$$

$$\sum_{i \in F} y_{ik} \leq 1 \qquad \forall k \in S \tag{17}$$

$$y_{ik} \in \{0, 1\} \qquad \forall i \in F,\ \forall k \in S \tag{18}$$

$$x_i - \sum_{k \in S} Su_k\ y_{ik} \leq 0 \qquad \forall i \in F \tag{19}$$

$$x_i - \sum_{k \in S} Sl_k\ y_{ik} \geq 0 \qquad \forall i \in F \tag{20}$$

$$n_{il} ct_{il} - 60 \sum_{k \in S,\ g_{lk}=1} y_{ik} \leq 0 \qquad \forall i \in F,\ \forall l \in T \tag{21}$$

$$x_i - \sum_{l \in T} n_{il}\, cap_l = 0 \qquad \forall i \in F \tag{22}$$

$$\frac{1}{60} \sum_{i \in F} n_{il} ct_{il} \leq Tx_l \qquad \forall l \in T \tag{23}$$

$$U_l - \frac{1}{60} \sum_{l \in T} n_{il} ct_{il} \geq 0 \qquad \forall i \in F \tag{24}$$

$$n_{il} \in Z^+ \qquad \forall i \in F,\ \forall l \in T \tag{25}$$

$$U_l \in \{0, 1\} \qquad \forall l \in T \tag{26}$$

The objective function (1) seeks to minimize the differences with regard to production goals of ore and waste rock, quality targets of the mixture, as well

as to reduce the number of trucks used. The constraints (2)-(15) model the classic problem of blending with goals. Constraints (2) and (3) assure that the maximum and minimum limits for the control parameters must be verified, respectively. Constraints (4), together with the objective function, aim to meet the recommended percentage for the control parameters. Constraints (5) and (6) guarantee that the maximum and minimum production of ore are verified. The constraints (8) and (9) model the same, but considering waste rock. Constraints (7) and (10) relate respectively to the care of the production targets of ore and waste rock, while the constraints (11) limit the maximum mining rate defined by the user for each pit.

The other constraints which complement the model can be divided into two groups. The first concerns the allocation of shovels and productivity range in order to justify the equipment use. The second is related to the allocation of trucks for material transport in the mine.

For the first group, constraints (16) define that at most one shovel can be allocated to each pit, while constraints (17) define that each shovel can be allocated to one pit at most. Constraints (18) define that the variables $y_{ik}$ are binary. Each constraint (19) and (20) limits, respectively, the maximum and minimum mining rate defined by shovel $k$ allocated to pit $i$.

In the second group of constraints, each constraint (21) forces the truck to only perform trips where there is compatible shovel allocated. The constraints (22) are such that the mining rate of a pit is equal to the total production of the trucks allocated to that pit. The constraints (23) ensure that each truck $l$ is in operation for at most $Tx_l\%$ in one hour. The constraints (24), together with the objective function, force the number of trucks used be penalized. The constraints (25) force the number of trips that a truck performs to a pit to be a positive integer value. Constraints (26) indicate that the variables $U_l$ are binary.

## 5. HEURISTIC MODEL

### *5.1. Representation of a Solution*

A solution is represented by a matrix $R = [Y \mid N]$, where $Y$ is a matrix $|F| \times 1$ and $N$ is a matrix $|F| \times |T|$.

Each cell $y_i$ of the matrix $Y_{|F|\times 1}$ represents the shovel $k$ allocated to pit $i$. A value $-1$ means that there isn't any shovel allocated to pit $i$. If there aren't any trips made to pit $i$, the shovel $k$ associated to that pit is considered *inactive* and it isn't penalized for a production below the minimum for a shovel.

In the matrix $N_{|F|\times|T|}$, each cell $n_{il}$ represents the number of trips that each truck $l \in T$ performs to a pit $i \in F$. A value 0 (zero) means that there aren't any trips allocated to the truck $l$ to the pit $i$ , while a value $-1$ indicates that the truck and the shovel allocated to that pit aren't compatible.

Figure 2 illustrates a solution involving 4 trucks, 4 pits and 3 shovels. In this figure, for example, the truck 1 makes 2 trips to pit 1 and 1 trip to pit 3. The shovel 2 is assigned to pit 4 and the truck 1 is incompatible with it. In this solution, pit 2 is available.

From $Y$, $N$ and the cycle times from the matrix $CT$ ($|F| \times |T|$ dimensional) the extraction rate at each pit is determined, as well as the sum of the cycle times for each truck.

*5.2. Neighborhoods*

To explore the solution space of the problem, eight movements were developed. Each movement defines a neighborhood $N(.)$, which are presented in sections 5.2.1 to 5.2.8.

*5.2.1. Movement Number of Trips - $N^{NT}(s)$*

This movement increases or decreases the number of trips of truck $l$ to pit $i$ where there's an allocated compatible shovel. Thus, in this movement, a cell $n_{il}$ of the matrix $N$ has its value increased or decreased by one trip.

*5.2.2. Movement Load - $N^{L}(s)$*

Consists of changing two separate cells $y_i$ and $y_k$ of the matrix $Y$, i.e. exchanging the shovels that operate in the pits $i$ and $k$, if both pits have allocated shovels. If only one of the two pits has an allocated shovel and the other is available, this movement will relocate the shovel to the available pit. In order to maintain compatibility between shovels and trucks, the trips made to that pits are relocated along with the shovels.

*5.2.3. Movement Relocate Trip from a Truck - $N^{TT}(s)$*

Consists of choosing two cells $n_{il}$ and $n_{kl}$ from the matrix $N$ and passing one trip from $n_{il}$ to $n_{kl}$. Thus, in this movement, the truck $l$ cancels one trip to pit $i$ and does it at another pit $k$. Compatibility restrictions between equipment are respected in this movement, so the trip relocation is only done when there's compatibility between them.

*5.2.4. Movement Relocate Trip from a Pit - $N^{TP}(s)$*

Two cells $n_{il}$ and $n_{ik}$ from the matrix $N$ are chosen and a unit of $n_{il}$ is relocated to $n_{ik}$. So this movement consists of relocating one trip from truck $l$ to truck $k$ which are both working at pit $i$. Compatibility restrictions between equipment are respected in this movement, so the trip relocation is only done when there's compatibility between them.

*5.2.5. Movement Pit Operation - $N^{PO}(s)$*

Consists of removing from operation the shovel that is allocated to pit $i$. The movement removes all the trips made to this pit, leaving this shovel *inactive*. The shovel is again put in operation as soon as a new trip is associated to it.

*5.2.6. Movement Truck Operation - $N^{TO}(s)$*

Consists of selecting a cell $n_{il}$ from the matrix $N$ and zero-fill its content, meaning that the truck $l$ doesn't operate in pit $i$ anymore.

*5.2.7. Movement Swap Trips - $N^{ST}(s)$*

Two cells of the matrix $N$ are selected and one trip is relocated from one to another. This movement can occur in any cell of the matrix $N$ if compatibility restrictions between equipments are respected.

*5.2.8. Movement Swap Shovels - $N^{SS}(s)$*

Consists of swapping two separate cells $y_i$ and $y_k$ from the matrix $Y$, i.e. exchanging the shovels that operate in pits $i$ and $k$. This movement is similar to the movement Load (Neighborhood $N^L$), because the shovels are also exchanged, but the trips made to these pits are not exchanged. To maintain compatibility between the shovels and trucks, the trips made by incompatible equipment are removed.

Figure 2 shows examples of these movements, where $m$ is a movement that belongs to the neighborhood $N(s)$. In this figure, a signal $^*$ close to a number means the modification made in the solution $s$. For example, solution $s \oplus m^{TT}$ differs of $s$ in relation to the trips of the truck 2 for the pits 1 and 3. This neighbor was obtained from $s$ by reassigning one trip of the truck 2 from the pit 1 to the pit 3. Now, this truck realizes 3 trips to the pit 1 and 1 trip to the pit 3.

$$s = \left[\begin{array}{r|rrrr} 1 & 2 & 4 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 3 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array}\right]$$

$$s \oplus m^{NT} = \left[\begin{array}{r|rrrr} 1 & 2 & 4 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & {}^*4 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array}\right] \qquad s \oplus m^{L} = \left[\begin{array}{r|rrrr} {}^*3 & {}^*1 & {}^*0 & {}^*3 & {}^*2 \\ -1 & 0 & 0 & 0 & 0 \\ {}^*1 & {}^*2 & {}^*4 & {}^*3 & {}^*0 \\ 2 & -1 & 4 & 2 & 1 \end{array}\right]$$

$$s \oplus m^{TT} = \left[\begin{array}{r|rrrr} 1 & 2 & {}^*3 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & 1 & {}^*1 & 3 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array}\right] \qquad s \oplus m^{TP} = \left[\begin{array}{r|rrrr} 1 & {}^*1 & {}^*5 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 3 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array}\right]$$

$$s \oplus m^{PO} = \left[\begin{array}{r|rrrr} 1 & 2 & 4 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & {}^*0 & {}^*0 & {}^*0 & {}^*0 \\ 2 & -1 & 4 & 2 & 1 \end{array}\right] \qquad s \oplus m^{TO} = \left[\begin{array}{r|rrrr} 1 & 2 & 4 & {}^*0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 3 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array}\right]$$

$$s \oplus m^{ST} = \left[\begin{array}{r|rrrr} 1 & {}^*1 & 4 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & {}^*4 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array}\right] \qquad s \oplus m^{SS} = \left[\begin{array}{r|rrrr} {}^*3 & 2 & 4 & 3 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ {}^*1 & 1 & 0 & 3 & 2 \\ 2 & -1 & 4 & 2 & 1 \end{array}\right]$$

Figure 2: Examples of the proposed movements

*5.3. Evaluation of a Solution*

As the developed movement can generate infeasible solutions, a solution is evaluated by a mono-objective function $f : S \to \mathbb{R}$, where $S$ represents the set of all possible solutions $s$ generated from the movements presented in the previous section. This function $f$, defined by Equation (27), to be minimized, consists of two parts: first, the objective function itself (Equation (1) from the mathematical programming model) and second, a group of functions that penalize the occurrence of infeasibility in current solution.

$$f(s) = f^{MP}(s) + f^{p}(s) + \sum_{j \in P} f_j^q(s) + \sum_{l \in T} f_l^u(s) + \sum_{k \in S} f_k^c(s) \tag{27}$$

In Equation (27), $f^{MP}(s)$ is the objective function from the mathematical programming model given by Equation (1), i.e. $f^{MP}(s)$ evaluates $s \in S$ considering production and quality goals, as well as the number of trucks used; $f^p(s)$ evaluates $s$ considering unmet production goals for ore and waste rock; $f_j^q(s)$ evaluates $s$ considering the infeasibility of the $j$-th control parameter; $f_l^u(s)$ evaluates $s$ regarding disrespect of the maximum use rate of the $l$-th truck; and $f_k^c(s)$ evaluates $s$ for disrespect of the productivity limits of the shovel $k$.

*5.4. Initial Solution Generation*

An initial solution to the problem is built in two steps. First, the allocation of the shovels and the distribution of trips are realized for the waste rock pits; secondly, for the ore pits. This strategy is adopted because in the waste rock pits it is important to meet production and not necessary to observe the quality of the control parameters.

In the first step a greedy heuristic is used (Algorithm 1). In this algorithm, we define the "best" choice according to our greedy criterion as follows: for waste rock pits, the best is the one with the greatest mass; for shovels, the best is the one with the greatest production and for trucks, the largest one is the best.

**Algorithm 1**: BuildWasteSolution

**Input**: $S, T, W, W_r$
**Output**: Solution $s_W$
$T \leftarrow$ Set of available trucks ordered by their capacities (the first is the truck that has the greatest capacity);
$S \leftarrow$ Set of available shovels ordered by their maximum productivities (the first is the shovel that has the greatest productivity);
$W \leftarrow$ Set of available waste rock pits ordered according to their maximum rates of mining (the first is the pit that has the greatest rate);
**while** *the waste rock production is less than the recommended one* ***and*** *there are available waste rock pits* **do**
  Select the first pit $i$ from $W$;
  **if** *there is no shovel at pit $i$* **then**
    **if** *All shovels are assigned* **then** Remove pit $i$ from $W$ **else** Update $s_W$ assigning the best available shovel to pit $i$;
  **end**
  **if** *Pit $i$ was not removed from $W$* **then**
    Find a truck $l \in T$ such that: a) it is compatible with the shovel assigned to pit $i$; b) it can do one more trip; c) its capacity does not violate the shovel's maximum production;
    **if** *truck $l$ exists* **then** Update $s_W$ assigning the maximum number of trips of the $l$-truck to pit $i$;
    **else** Remove pit $i$ from $W$;
  **end**
**end**
**return** $s_W$;

For the second step, a heuristic based on GRASP is used. In its original form (Feo and Resende, 1995), GRASP is an iterative method that has two phases: construction and local search. The construction phase builds a feasible solution, whose neighborhood is explored by local search. The best solution over all $GRASPmax$ iterations is returned as the result.

For the ore pits, the classification of the candidate elements to be inserted in the solution is made considering that: a) the best pit is the one that has the least deviation of the control parameter levels in relation to the targets; b) the best shovel is the one that provides the greatest production and c) the best truck is the one that has the smallest capacity.

In order to select the ore pits in the second step, a guide function $g$, which measures the deviation values of the quality goals, is used. According to this function, it is more likely to choose the ore pit that best helps to minimize the deviations from the quality targets. First, all candidate pits ($CL$) are sorted with respect to the function $g$, where $CL$ is the set of available pits. From $CL$, the construction phase creates a restricted candidate list ($RCL$) using the best qualified ore pits according to the guide function. The parameter $\gamma \in [0, 1]$ defines the size of this restricted list. The procedure includes the best $\lceil \gamma \times |CL| \rceil$ pits in the $RCL$.

Afterwards, the procedure chooses a pit randomly from this list using a strategy proposed by Bresina (1996), and adds it to the partial solution. The strategy consists in assigning a rank-based probability for each candidate pit in $RCL$. The bias function $bias(r) = 1/(r)$ is associated to the $r$-th best classified pit. Then, each candidate pit is chosen with probability $p(r) = bias(r)/\sum_{i=1,\cdots,|RCL|} bias(i)$. The construction phase ends when the ore production goal is reached or when there are no more pits or shovels available. In each iteration of this construction, the shovel with the greatest production and the truck that has the smallest capacity is chosen. The Algorithm 2 outlines the second step of the construction phase.

According to Lourenço et al. (2003), the initial solution is certainly important to achieve high quality solutions in the first instants of the search. Since the construction phase of GRASP is often able to produce solutions close to some local optimum (Resende and Ribeiro, 2010) and our local search procedures are very expensive (Section 5.2), we opted for executing a number of iterations of the construction procedure alone before proceeding do the next phase.

**Algorithm 2**: BuildOreSolution

**Input**: $s_W$, $\gamma$, $g$, $O$, $S$, $T$, $O_r$
**Output**: Solution $s_0$
$s_0 \leftarrow s_W$;
$T \leftarrow$ Set of available trucks ordered by their capacities (the first is the truck that has the smallest capacity);
$S \leftarrow$ Set of available shovels ordered by their maximum productivities (the first is the shovel that has the greatest productivity);
**while** *the ore production is less than the recommended one* ***and*** *there are available ore pits* **do**
    $CL \leftarrow$ Set of available ore pits $i \in O$ ordered according to function $g$;
    $|RCL| = \lceil \gamma \times |CL| \rceil$;
    Select $i \in RCL$ according to the bias function;
    **if** *there is no shovel at pit* $i$ **then**
        **if** *All shovels are assigned* **then** Remove pit $i$ from $CL$ **else** Update $s_0$ assigning the best available shovel to pit $i$;
    **end**
    **if** *Pit* $i$ *was not removed from* $CL$ **then**
        Find a truck $l \in T$ such that: a) it is compatible with the shovel assigned to pit $i$; b) it can do one more trip; c) its capacity does not violate the shovel's maximum production;
        **if** *truck* $l$ *exists* **then** Update $s_0$ assigning one $l$-truck trip for the pit $i$;
        **else** Remove pit $i$ from $CL$;
    **end**
**end**
**return** $s_0$;

### *5.5. Proposed Algorithm*

The proposed algorithm, called `GGVNS`, combines ideas from `GRASP` (Resende and Ribeiro, 2010) and *General Variable Neighborhood Search* - `GVNS` (Hansen et al., 2008b) procedures. Algorithm 3 outlines the steps.

**Algorithm 3**: GGVNS

**Input**: sets O, W, P, S, T, $\cdots$ (See parameters in Section 4)
**Input**: $\gamma$, *GRASPmax*, *IterMax*
**Output**: Solution $s$

```
1  s_W ← BuildWasteSolution()
2  s_0 ← best solution from GRASPmax calls to BuildOreSolution(s_W, γ)
3  s ← VND(s_0)
4  p ← 0
5  while stop criterion not satisfied do
6  |  iter ← 0
7  |  while iter < IterMax and stop criterion not satisfied do
8  |  |  s' ← s
9  |  |  for i = 1 to p + 2 do
10 |  |  |  k ← SelectNeighborhood()
11 |  |  |  s' ← Shake(s', k)
12 |  |  end
13 |  |  s'' ← VND(s')
14 |  |  if f(s'') < f(s) then
15 |  |  |  s ← s''
16 |  |  |  p ← 0
17 |  |  |  iter ← 0
18 |  |  end
19 |  |  iter ← iter + 1
20 |  end
21 |  p ← p + 1
22 end
23 return s
```

Building an initial solution $s_0$ (lines 1 and 2 of Algorithm 3) is made by the procedure described in Subsection 5.4. The local search (lines 3 and 13 of Algorithm 3), in turn, uses the VND procedure (see the pseudo-code in Algorithm 4) with the movements described in Subsection 5.2.

Whenever a given number of iterations without improvement is reached, the GGVNS algorithm applies $p + 2$ times the *Shake* procedure, using a previously selected neighborhood. The procedure *SelectNeighborhood* (line 10 of the Algorithm 3) works as follows. We randomly select a neighborhood $k$ from the list $\{N^{SS}, N^{TO}, N^{PO}, N^{ST}, N^{NT}, N^{L}\}$ with probabilities $\{10\%, 10\%, 10\%, 20\%, 30\%, 20\%\}$, respectively . We observed that some neighborhoods are more likely to contain solutions which are significantly different of the current solution. These probabilities reflect this observation. Each $Shake(s', k)$ call (line 11 of Algorithm 3) performs a random movement from neighborhood $k$ of the shaken solution $s'$. After *IterMax* iterations without improvement, we increment $p$ in order to generate solutions which become increasingly distant from the current location in the search space.

The local search applied on the solution returned by the *Shake* procedure is based on the VND procedure (line 13 of Algorithm 3). If VND finds a better solution, the variable $p$ returns to the lowest value, that is, $p = 0$.

**Algorithm 4**: `VND`

**Input**: $r$ neighborhoods in random order: $N^L$, $N^{NT}$, $N^{TT}$ and $N^{TP}$
**Input**: Initial solution $s$
**Output**: Solution $s$

```
1  k ← 1
2  while k ≤ r do
3  |  Find the best neighbor s' ∈ N^(k)(s)
4  |  if f(s') < f(s) then
5  |  |  s ← s'
6  |  |  k ← 1
7  |  end
8  |  else
9  |  |  k ← k + 1
10 |  end
11 end
12 return s
```

As in the preliminary tests some neighborhoods did not produced good quality solutions or spent too much processing time to achieve a good one, only a small group of neighborhoods was used in the local search. Thus, the `VND` used the following neighborhoods: $N^L$, $N^{NT}$, $N^{TT}$ and $N^{TP}$. Furthermore, the `VND` procedure (see Algorithm 4) operates in the neighborhoods in a random order, which can be different at each `VND` call (more details in the Section 7).

## 6. SCENARIOS DESCRIPTION

The scenarios utilized for the tests refer to an iron mining company located in the state of Minas Gerais, Brazil and are available at http://www.iceb.ufop.br /decom/prof/marcone/projects/mining.html.

Table 1 describes some characteristics of the instances. The columns "# pits, # shovels, # trucks and par." indicate the number of pits, shovels, trucks and control parameters (chemical and/or granulometric), respectively. The column "characteristics" shows the number and the truck capacity or the shovel productivity. For example, the pair (15, 50t) means there are 15 trucks (or shovels) of 50 ton of capacity (or maximum productivity).

The following weights were adopted in the evaluation function: $\alpha^- = \alpha^+ = \beta^- = \beta^+ = 100$, $\lambda_j^- = \lambda_j^+ = 1 \ \ \forall j \in T$, $\omega_l = 1 \ \ \forall l \in V$, $Tx_l = 75\% \ \ \forall l \in V$.

## 7. COMPUTATIONAL EXPERIMENTS AND ANALYSIS

The proposed algorithm, so-called `GGVNS`, was coded in C++ programming language and compiled with the GNU Compiler Collection version 4.0. The mathematical programming model was written in AMPL language (Fourer et al., 1990) and solved by the `ILOG CPLEX` optimizer version 11.01 (ILOG, 2008), using default parameters. Both heuristic and exact models were tested in a PC Pentium Core 2 Quad (Q6600), 2.4 GHz, with 8 GB of RAM, running Windows Vista.

Table 1: Characteristics of the instances

| | | shovels | | | trucks | |
|---|---|---|---|---|---|---|
| inst. | # pits | # shovels | characteristics | # par. | # trucks | characteristics |
| opm1 | 17 | 8 | (4, 900t), (2, 1000t) (2, 1100t) | 10 | 30 | (15, 50t), (15, 80t) |
| opm2 | 17 | 8 | (4, 900t), (2, 1000t) (2, 1100t) | 10 | 30 | (15, 50t), (15, 80t) |
| opm3 | 32 | 7 | (2, 400t), (2, 500t) (1, 600t), (1, 800t) (1, 900t) | 10 | 30 | (30, 50t) |
| opm4 | 32 | 7 | (2, 400t), (2, 500t) (1, 600t), (1, 800t) (1, 900t) | 10 | 30 | (30, 50t) |
| opm5 | 17 | 8 | (4, 900t), (2, 1000t) (2, 1100t) | 5 | 30 | (15, 50t), (15, 80t) |
| opm6 | 17 | 8 | (4, 900t), (2, 1000t) (2, 1100t) | 5 | 30 | (15, 50t), (15, 80t) |
| opm7 | 32 | 7 | (2, 400t), (2, 500t) (1, 600t), (1, 800t) (1, 900t) | 5 | 30 | (30, 50t) |
| opm8 | 32 | 7 | (2, 400t), (2, 500t) (1, 600t), (1, 800t) (1, 900t) | 5 | 30 | (30, 50t) |

All the experiments considered the following parameters: $IterMax = 5,000$, $GRASPmax = 10,000$ and $\gamma = 0.3$.

As mentioned in Hansen et al. (2008b), one important decision to build an efficient `VND` procedure is to select an application order of the different neighborhoods. In a preliminary set of experiments (10 runs for each instance) we tried to discover the optimal sequence of neighborhood application, that is, the one which, in average, produces better solutions in a limited amount of time when running the `GGVNS` algorithm. To accomplish this objective, we adopted the following strategy. Firstly we executed experiments considering each one of the neighborhoods as the first neighborhood in the sequence, and the remaining ones were chosen in a random order, at each `VND` call (Phase I columns of the Table 2). For simplicity, the neighborhoods $N^L, N^{NT}, N^{TT}, N^{TP}$ are denoted by $L, NT, TT, TP$, respectively, in the Table 2. Considering these results, we observed better results in relation to the average gap (the last row in the Table 2) when selecting $N^L$ as the first neighborhood. Therefore, this neighborhood was kept as the first in the sequence.

After that, we ran experiments to define the second neighborhood of the sequence. From the remaining neighborhoods $\{N^{NT}, N^{TT}, N^{TP}\}$, $N^{TT}$ produced the best results in relation to the same metric used previously (Phase II columns in Table 2), being selected to occupy the second position.

Finally, additional experiments (Phase III columns in Table 2) indicated that it would be better to search in the neighborhood $N^{TP}$ before proceeding to search in $N^{NT}$. In these experiments, we observed that our "best" sequence of neighborhoods does not outperform many of the results produced when the neighborhood application order was partially random (Columns 2-8 in Table 2). This motivated us to perform an additional experiment in which the neigh-

borhood application sequence was completely random at each `VND` call. This experiment produced the best results, indicating that the random selection of neighborhoods to search is the best option (Random column in the Table 2). The results in Table 3 were produced using this last strategy.

Table 2: Results of the preliminary experiments

| | Phase I | | | | Phase II | | | Phase III | | Random |
|---|---|---|---|---|---|---|---|---|---|---|
| Instance | *L* | *NT* | *TT* | *TP* | *L-NT* | *L-TT* | *L-TP* | *L-TT-NT-TP* | *L-TT-TP-NT* | |
| opm1 | 232.80 | 1,816.40 | 235.54 | 236.85 | 232.66 | 232.36 | 233.11 | 1,696.35 | 236.20 | 230.12 |
| opm2 | 335.37 | 340.70 | 2,686.72 | 326.78 | 350.27 | 340.93 | 327.30 | 332.68 | 338.18 | 256.56 |
| opm3 | 164,058.99 | 164,054.62 | 164,057.97 | 164,067.40 | 164,059.08 | 164,057.56 | 164,054.60 | 164,057.96 | 164,054.29 | 164,064.68 |
| opm4 | 164,138.64 | 164,143.77 | 164,126.98 | 164,123.92 | 164,118.64 | 164,158.45 | 164,123.00 | 164,135.01 | 164,133.37 | 164,153.92 |
| opm5 | 229.86 | 1,692.72 | 1,690.83 | 229.07 | 232.01 | 229.58 | 228.93 | 450.71 | 231.05 | 228.09 |
| opm6 | 326.60 | 330.51 | 308.14 | 2,703.00 | 319.09 | 325.50 | 2,672.08 | 308.21 | 312.88 | 237.97 |
| opm7 | 164,021.50 | 164,021.65 | 164,021.67 | 164,021.58 | 164,021.55 | 164,021.86 | 164,021.59 | 164021.56 | 164021.66 | 164021.89 |
| opm8 | 164,024.32 | 164,024.36 | 164,024.53 | 164,023.89 | 164,023.90 | 164,023.58 | 164,024.43 | 164,024.08 | 164,023.62 | 164,027.29 |
| | | | | | GAP (%) | | | | | |
| opm1 | 2.501 | 699.753 | 3.707 | 4.284 | 2.440 | 2.309 | 2.639 | 646.894 | 3.996 | 1.321 |
| opm2 | 30.814 | 32.894 | 947.986 | 27.465 | 36.628 | 32.985 | 27.669 | 29.766 | 31.913 | 0.074 |
| opm3 | 0.019 | 0.017 | 0.019 | 0.025 | 0.019 | 0.019 | 0.017 | 0.019 | 0.017 | 0.023 |
| opm4 | 0.050 | 0.053 | 0.043 | 0.041 | 0.038 | 0.062 | 0.040 | 0.048 | 0.047 | 0.059 |
| opm5 | 1.240 | 645.560 | 644.726 | 0.892 | 2.189 | 1.120 | 0.834 | 98.516 | 1.768 | 0.462 |
| opm6 | 38.052 | 39.702 | 30.247 | 1,042.531 | 34.876 | 37.586 | 1,029.463 | 30.278 | 32.251 | 0.588 |
| opm7 | 0.002 | 0.003 | 0.003 | 0.003 | 0.002 | 0.003 | 0.003 | 0.003 | 0.003 | 0.003 |
| opm8 | 0.003 | 0.003 | 0.004 | 0.003 | 0.003 | 0.003 | 0.004 | 0.003 | 0.003 | 0.005 |
| Average | 9.085 | 177.248 | 203.342 | 134.405 | 9.524 | 9.261 | 132.583 | 100.691 | 8.750 | 0.317 |

In the first set of experiments we evaluated GGVNS considering its ability to produce good solutions in a short amount of time. Considering the needs of decision makers, we limited the execution time to 2 minutes, which is a typical value for the maximum tolerance in a real case. The GGVNS algorithm was applied 30 times for each instance. For CPLEX, we also allowed longer execution times for searching for the optimal solution.

Results of this set of experiments appear in Table 3. In this table, column "best known" refers to the best known cost found in all our experiments. In column "opt." we indicate by "$\surd$" instances in which CPLEX succeeded in proving the optimality of the best known cost. Columns "gap" are computed as follows: consider that $f_i^*$ is the best known cost for instance $i$ (optimal cost for some instances), $f_i^{CPLEX}$ is the upper bound obtained at the end of CPLEX execution for instance $i$ and $\bar{f}_i^{\texttt{GGVNS}}$ is the average value found in the thirty executions of GGVNS algorithm, gap is computed for each instance $i$ for the CPLEX optimizer ($gap_i^{CPLEX}$) and for GGVNS ($gap_i^{\texttt{GGVNS}}$) in equations (28) and (29), respectively.

$$gap_i^{CPLEX} = \frac{f_i^{CPLEX} - f_i^*}{f_i^*} \tag{28}$$

$$gap_i^{\texttt{GGVNS}} = \frac{\bar{f}_i^{\texttt{GGVNS}} - f_i^*}{f_i^*} \tag{29}$$

Table 3: Experimental results: Mathematical Programming Model in CPLEX and GGVNS heuristic.

| | | | CPLEX | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | best known | | 2 hours | | 2 minutes | | GGVNS | | | |
| instance | cost | opt.† | cost | gap | cost | gap | best | average | std. dev. | gap |
| opm1 | 227.12 | | 227.12 | 0.00 | 230.65 | 1.55 | 230.12 | 230.12 | 0.01 | 1.32 |
| opm2 | 256.37 | | 257.66 | 0.50 | 4858.39 | > 100.00 | 256.37 | 256.56 | 0.26 | 0.07 |
| opm3 | 164,027.15 | √ | 164,027.15 | 0.00 | 164,042.60 | 0.01 | 164,039.12 | 164,064.68 | 17.24 | 0.02 |
| opm4 | 164,056.68 | √ | 164,056.68 | 0.00 | 164,061.80 | 0.00 | 164,099.66 | 164,153.92 | 29.43 | 0.06 |
| opm5 | 227.04 | | 227.04 | 0.00 | 7,229.07 | > 100.00 | 228.09 | 228.09 | 0.00 | 0.91 |
| opm6 | 236.58 | | 236.58 | 0.00 | 236.58 | 0.00 | 236.58 | 237.97 | 2.38 | 0.59 |
| opm7 | 164,017.46 | √ | 164,017.46 | 0.00 | 164,017.46 | 0.00 | 164,021.38 | 164,021.89 | 0.34 | 0.00 |
| opm8 | 164,018.65 | √ | 164,018.65 | 0.00 | 164,018.65 | 0.00 | 164,023.73 | 164,027.29 | 1.60 | 0.00 |

† Considering CPLEX mipgap tolerance $\leq 10^{-5}$, except for opm3, which used mipgap tolerance $\leq 10^{-4}$.

As can be seen in Table 3, considering the time limit of two minutes, CPLEX was able to prove the optimality of the solution only in two of the eight instances. In addition, for another two instances (opm2 and opm5), CPLEX presented very high gap. For the other hand, `GGVNS` presented near best known solutions (gap $< 1.5\%$) in all instances, even with the time limit constraint. A remarkable result for `GGVNS` appeared in the hard instances opm2 and opm5. In these instances, CPLEX could not provide a solution satisfying production goals within two minutes, while `GGVNS` always produced solutions satisfying this requirement in the restricted time. In instances 3, 4, 7 and 8 the solutions presented a very high cost. We observed that this happens due to a waste production goal which cannot be satisfied, generating a constant in the objective function. We decided to not change this goal to maintain compatibility with previous works.

One important result would be the discovery of the optimal solution for the remaining instances 1 to 6. This motivated us to perform the longer runs (two hours) of the CPLEX optimizer. Within this time limit, CPLEX found the optimal solution for two additional instances: omp3 and opm4. In Figures 3, 4 and 5 we plotted the evolution of the lower and upper bounds during CPLEX search for some instances. As can be seen, although CPLEX heuristics managed to improve the upper bounds, the lower bounds remained stable. After a certain amount of time, both lower and upper bounds stagnated, which led us to believe that longer execution times would not suffice to produce optimal solutions using our formulation.

Below we analyze the results of the proposed algorithm with regard to the quality of the control parameters. For each instance, considering the 30 executions of `GGVNS`, we calculated the largest absolute error between the recommended percentage $pr_j$ for the control parameter $j$ in the blending and the encountered percentage $ep_{ji}$ for this control parameter in all executions $i$ of the `GGVNS`. For this calculation we chose the solution of the $i^{th}$ execution of the `GGVNS` in which the percentage $ep_{ji}$ is the farthest from the recommended percentage $pr_j$. The symbol + in the Figures 6 and 7 represents the biggest absolute error for the instances opm2 and opm3, respectively. We also calculated the absolute error between the recommended percentage $pr_j$ for the control parameter $j$ in the blending and the average of the encountered percentages $\overline{ep}_j$ for this control parameter in the `GGVNS` solutions. The symbol $\square$ in the Figures 6 and 7 represents this error for the instances opm2 and opm3, respectively.

While the absolute errors for the instance opm2 (Figure 6) vary from 0.32% up to 11.49%, they reach 40.53% in the instance opm3 (Figure 7). This is because the minimum and maximum allowable percentages for the control parameters in the mixture vary from one instance to another. In the instance opm3 these values are, respectively, 0% and 100%, that is, the percentage of each control parameter can vary from 0% to 100% in the solution. On the other hand, in the instance opm2, the difference between the minimum and maximum allowable percentages is smaller, so forcing the encountered percentages for the control parameters to be closer to the recommended percentages. For the remaining instances, the error behavior is similar to the one obtained for opm2 or opm3.

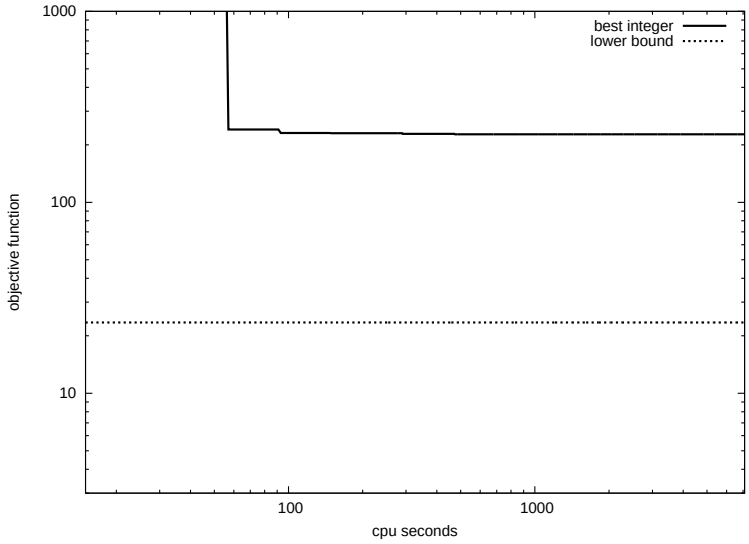


Figure 3: Evolution of upper and lower bounds in CPLEX - instance opm1.

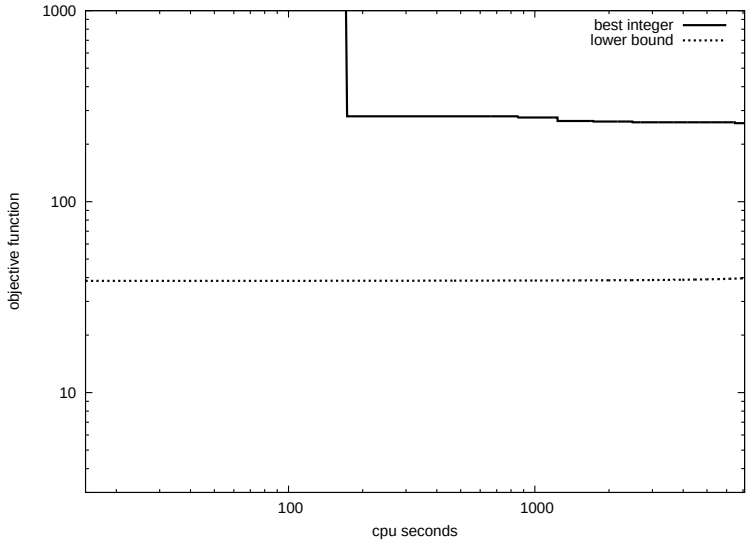


Figure 4: Evolution of upper and lower bounds in CPLEX - instance opm2.

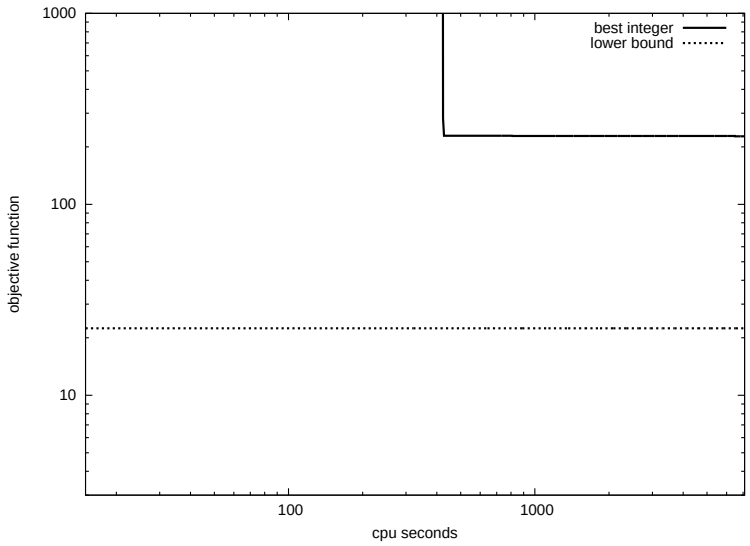


Figure 5: Evolution of upper and lower bounds in CPLEX - instance opm5.

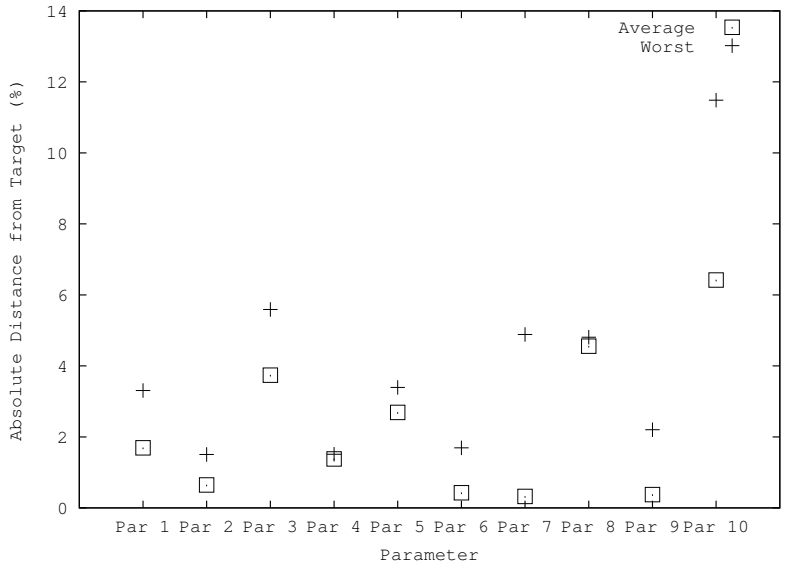


Figure 6: Deviation of the control parameters in the mixture for the instance opm2

## 8. CONCLUSIONS

This work dealt with the operational planning of mines considering the dynamic allocation of trucks. Because of the complexity of this combinatorial problem, we proposed a hybrid heuristic algorithm, called `GGVNS`, which combines the heuristic procedures `GRASP` and General Variable Neighborhood Search to solve it.

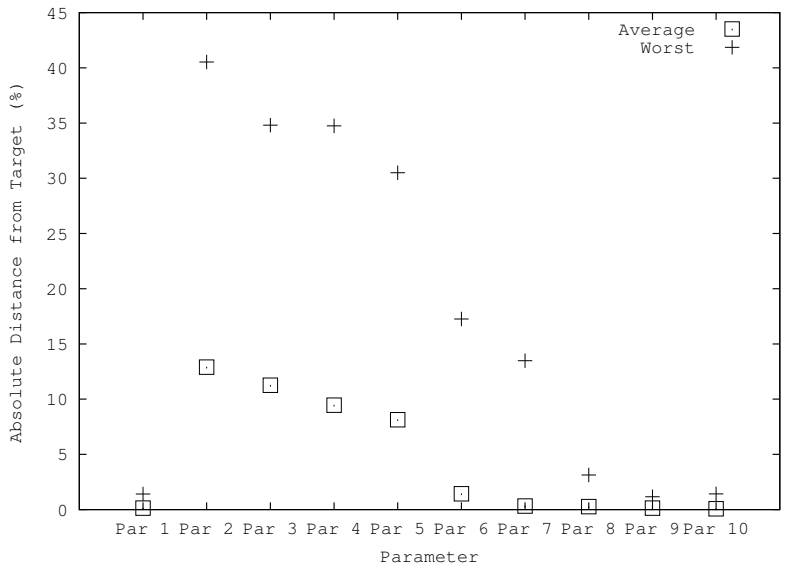


Figure 7: Deviation of the control parameters in the mixture for the instance opm3

Using instances from literature, the proposed heuristic algorithm was compared to the optimizer CPLEX 11.0.1 applied to a mathematical programming model, also developed in this work. It was found that the `GGVNS` algorithm is competitive with CPLEX solver, since `GGVNS` is able to find good quality solutions quickly with low variability. Since staff decisions have to be made quickly, the results validate the use of the proposed algorithm as a tool for decision support.

As future works we consider to integrate the mathematical programming solver with `GGVNS`, with the aim of combining the fast solution times of `GGVNS` with the systematic exploration of the search tree of exact solvers. We are also studying the improvement of `GGVNS` by adding a Path Relinking strategy to work with an elite pool of solutions.

## 9. AKNOWLEGEMENTS

The authors acknowledge FAPEMIG (grants CEX 2991-06.1/07, CEX 357-09 and CEX 01201-09) and CNPq (grant 474831/2007-8) for supporting the development of this research. We also thank the anonymous referees for their constructive comments, leading to an improved version of this paper.

## References

Alvarenga, G. B., 1997. Optimal dispatch of trucks in an iron mine using genetic algorithms with parallel processing (in portuguese). Master's thesis, Programa de Pós-Graduação em Engenharia Elétrica, Escola de Engenharia, UFMG, Belo Horizonte, Minas Gerais, Brazil.

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. Computers and Operations Research 36, 1064–1089.

Bresina, J. L., 1996. Heuristic-biased stochastic sampling. In: Proceedings of the 13th National Conference on Artificial Intelligence, AAAI Press. Portland, pp. 271–278.

Chanda, E. K. C., Dagdelen, K., 1995. Optimal blending of mine production using goal programming and interactive graphics systems. International Journal of Surface Mining, Reclamation and Environment 9, 203–208.

Ezawa, L., Silva, K. S., 1995. Dynamic allocation of trucks aiming quality (in portuguese). In: Proceedings of the VI Congresso Brasileiro de Mineração. Salvador, Bahia, Brazil, pp. 15–19.

Feo, T. A., Resende, M. G. C., 1995. Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109–133.

Fourer, R., Gay, D. M., Kernighan, B. W., May 1990. A modeling language for mathematical programming. Management Science 36 (5), 519–554.

Glover, F., Kochenberger, G. (Eds.), 2003. Handbook of Metaheuristics. Kluwer Academic Publishers.

Godoy, M., Dimitrakopoulos, R., 2004. Managing risk and waste mining in long-term production scheduling of open-pit mines. SME Transactions 316, 43–50.

Guimaraes, I. F., Pantuza, G., Souza, M. J. F., 2007. A computational simulation model to validate results by dynamic allocation of trucks in open-pit mines (in portuguese). In: Proceedings of the XIV Simpósio de Engenharia de Produção (SIMPEP). Bauru, São Paulo, Brazil, 11 p.

Hansen, P., Mladenovic, N., 2001. Variable neighborhood search: Principles and applications. European Journal of Operational Research 130, 449–467.

Hansen, P., Mladenovic, N., Pérez, J. A. M., 2008a. Variable neighborhood search. European Journal of Operational Research 191, 593–595.

Hansen, P., Mladenovic, N., Pérez, J. A. M., 2008b. Variable neighborhood search: methods and applications. 4OR: Quarterly journal of the Belgian, French and Italian operations research societies 6, 319–360.

ILOG, 2008. CPLEX 11.0 User's Manual.

Lourenço, H. R., Martin, O. C., Stützle, T., 2003. Iterated local search. In: Glover, F., Kochenberger, G. (Eds.), Handbook of Metaheuristics. Kluwer Academic Publishers, Boston.

Merschmann, L. H. C., 2002. Development of an optimization and simulation system for the analysis of production scenarios in open-pit mines (in portuguese). Master's thesis, Programa de Engenharia de Produção/COPPE, UFRJ, Rio de Janeiro, Brazil.

Mladenovic, N., Hansen, P., 1997. A variable neighborhood search. Computers and Operations Research 24, 1097–1100.

Papadimitriou, C. H., Steiglitz, K., 1998. Combinatorial Optimization: Algorithms and Complexity. Dover Publications, Inc., New York.

Resende, M. G. C., Ribeiro, C. C., 2010. Grasp. In: Burke, E. K., Kendall, G. (Eds.), Search Methodologies, 2nd Edition. Springer (to appear), available at: http://www.ic.uff.br/∼celso/artigos/grasp.pdf.

Romero, C., 2004. A general structure of achievement function for a goal programming model. European Journal of Operational Research 153, 675–686.

Sgurev, V., Vassilev, V., Dokev, N., Genova, K., Drangajov, S., Korsemov, C., Atanassov, A., 1989. Trasy - an automated system for real-time control of the industrial truck haulage in open-pit mines. European Journal of Operational Research 43, 44–52.

White, J. W., Arnold, M. J., Clevenger, J. G., 1982. Automated open-pit truck dispatching at Tyrone. Engineering and Mining Journal 183 (6), 76–84.

White, J. W., Olson, J. P., 1986. Computer-based dispatching in mines with concurrent operating objectives. Mining Engineering 38 (11), 1045–1054.