UNIVERSIDADE ESTADUAL DE CAMPINAS

Instituto de Matemática, Estatística e Computação Científica

ALEXANDRE BARBOSA ANDRADE

Cut-off grade and scheduling optimization for underground mines

Otimização do teor de corte e do sequenciamento de minas subterrâneas

Campinas

2020

Alexandre Barbosa Andrade

Cut-off grade and scheduling optimization for underground mines

Otimização do teor de corte e do sequenciamento de minas subterrâneas

Dissertação apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Matemática Aplicada e Computacional.

Dissertation presented to the Institute of Mathematics, Statistics and Scientific Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computational and Applied Mathematics.

Supervisor: Priscila Cristina Berbert Rampazzo

Campinas

2020

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

An24c

Andrade, Alexandre Barbosa, 1989-
Cut-off grade and scheduling optimization for underground mines / Alexandre Barbosa Andrade. – Campinas, SP : [s.n.], 2020.

Orientador: Priscila Cristina Berbert Rampazzo.
Dissertação (mestrado profissional) – Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Pesquisa operacional. 2. Programação linear. 3. Algoritmos genéticos. 4. Minas e recursos minerais. 5. Otimização. I. Rampazzo, Priscila Cristina Berbert, 1984-. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Otimização do teor de corte e do sequenciamento de minas subterrâneas
Palavras-chave em inglês:
Operations research
Linear programming
Genetic algorithms
Mines and mineral resources
Optimization
Área de concentração: Matemática Aplicada e Computacional
Titulação: Mestre em Matemática Aplicada e Computacional
Banca examinadora:
Priscila Cristina Berbert Rampazzo [Orientador]
Cristiano Torezzan
Anibal Tavares de Azevedo
Data de defesa: 27-02-2020
Programa de Pós-Graduação: Matemática Aplicada e Computacional

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-3518-1839
- Currículo Lattes do autor: http://lattes.cnpq.br/6962232809881181

Dissertação de Mestrado Profissional defendida em 27 de fevereiro de 2020 e aprovada pela banca examinadora composta pelos Profs. Drs.

Prof(a). Dr(a). PRISCILA CRISTINA BERBERT RAMPAZZO

Prof(a). Dr(a). CRISTIANO TOREZZAN

Prof(a). Dr(a). ANIBAL TAVARES DE AZEVEDO

A Ata da Defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria de Pós-Graduação do Instituto de Matemática, Estatística e Computação Científica.

Agradecimentos

Gostaria de deixar meus agradecimentos a todos aqueles que participaram de forma direta ou indireta deste trabalho:

Aos meus pais, Almerinda e Ademar, que não mediram esforços e me deram inúmeras oportunidades ao longo de minha vida, me colocando no caminho deste mestrado há muito tempo.

À minha irmã, Adriana, que muito me inspirou e encorajou a fazer este mestrado.

À minha amada esposa, amiga e revisora, Daniela. Agradeço pelas ideias, compreensão, incentivo e companheirismo ao longo deste trabalho.

À minha orientadora, Professora Dr. Priscila Rampazzo, agradeço pela orientação, disponibilidade, paciência e entusiamo. Sou extremamente grato por todo o apoio durante a elaboração desta dissertação na qual você foi fundamental!

Aos professores Dr. Aníbal e Dr. Cristiano, membros da banca de avaliação, agradeço pela presteza, comentários e análise deste trabalho. Aos professores Dr. Akebo e Dr. Washington agradeço à disponibilidade.

Ao professores que ministraram as disciplinas do curso, representantes da Unicamp, e que fomentam o programa de Mestrado Profissional em Matemática Aplicada e Computacional, deixo o meu muito obrigado por proporcionarem momentos tão gratificantes e de enorme aprendizado.

Aos amigos que fiz durante curso, em especial ao Rogério, João, Saris e Rômulo pelo companheirismo e colaboração.

À Vanessa pelo apoio com a revisão.

À AngloGold Ashanti que, por meio de Ricardo Assis, Diogo Costa e Heath Gerritsen, apoiaram este mestrado e mostraram acreditar no potencial da educação continuada, me ajudando à conciliar o trabalho e o estudo durante este período.

Aos amigos planejadores, em especial Corey, Ruy, Cáiron e Rodiney que participaram de inúmeras discussões sobre o problema e me ajudaram a amadurecer diversos conceitos de planejamento de mina.

Resumo

Métodos de lavra subterrânea são aplicados na extração de vários metais e minerais. O planejamento de métodos subterrâneos difere do planejamento de métodos de superfície pelo fato de que não é necessário extrair todas as áreas de produção dentro dos limites econômicos finais para se ter uma sequência factível, ou seja, nos métodos subterrâneos é fisicamente possível que algumas áreas permaneçam não lavradas mesmo estando dentro do limites econômicos finais. Neste contexto, o planejamento estratégico é a área central do planejamento de longo prazo de uma mina e visa definir estratégias de escala de produção, métodos de lavra e de beneficiamento mineral, selecionar as áreas que serão lavradas e otimizar a sequência de lavra destas áreas de produção. Para garantir a viabilidade econômica do empreendimento, o planejamento estratégico deve considerar as características-chave dos empreendimentos de mineração, que são: a necessidade de capital intensivo, o longo período de retorno do investimento e o ativo (reserva) limitado. Essas características devem ser consideradas durante o processo de valoração de um empreendimento mineiro, que normalmente é feito através do cálculo do VPL, valor presente líquido. Dentre as principais alavancas do planejamento estratégico, o teor de corte utilizado na seleção dos blocos que serão lavrados e o sequenciamento de mina são os que geram maior número de opções, fazendo com que avaliações de cenários demandem muito tempo e se tornem inviáveis na prática, dada a necessidade de respostas rápidas para tomadas de decisão. Neste trabalho, três diferentes modelos matemáticos são propostos para abordar, de forma conjunta, o problema da seleção dos blocos de lavra de uma mina subterrânea e a otimização do sequenciamento destes blocos. Tais modelos consideram o VPL como principal objetivo a ser maximizado e resultam no uso do teor de corte como fator que equilibra as capacidades de produção dos diferentes estágios de um sistema de mineração. A abordagem matemática adapta a modelagem clássica de problemas de sequenciamento considerando os blocos de lavra como tarefas e as atividades de escavação de galerias (desenvolvimento de acessos) e de produção de minério (lavra) como máquinas. Os modelos propostos são testados com base em casos reais, utilizando-se métodos de solução exata e um algoritmo genético. Os resultados computacionais mostram que o algoritmo genético é mais eficiente do que os métodos exatos, sobretudo para instâncias maiores, mais próximas da realidade.

Palavras-chave: Pesquisa operacional. Programação linear. Algoritmos genéticos. Mineração subterrânea. Otimização. Teor de corte. Sequenciamento.

Abstract

Underground mining methods are used at the extraction of many metals and minerals. Underground mining planning differs from surface mining planning mainly because, in the first case, it is not necessary to extract all mining blocks within the ultimate economic limits to have a feasible sequence, i.e., it is physically possible to an underground mine to have some areas left *in situ* even if they are inside the ultimate economic limits. In this context, strategic planning is the core area of long-term mine planning and aims to define the scale of production, mining and processing methods, to select areas that will be mined, and to optimize the mining sequence. To guarantee the economic feasibility of a mining asset, strategic planners must also consider the key aspects of mining businesses, which are: capital-intensive requirements, long-term payback, and limited asset (reserves) life. These characteristics must be considered during the valuation process of a mining asset, which is normally conducted through NPV, net present value, calculations. Among the main strategic planning levers, cut-off grades (used at the selection of blocks that will be mined), and the mine sequencing are the ones that generate the greatest number of options. As scheduling multiple scenarios requires a great deal of time, this is infeasible in real situations given the need for quick responses. In this dissertation, three mathematical models are proposed to tackle, at the same time, two problems: the selection of the mining blocks in an underground mine, and the optimization of their sequence. These models consider NPV as the main objective to be maximized and result in using cut-off grades as a factor that balances the main capabilities of a mining system. The mathematical approach adapts classical scheduling models considering mining blocks as jobs; and tunnel excavation (development of accesses) and ore production (mining) activities as machines. The proposed models are tested, with real cases, using exact-solution methods and a genetic algorithm. Results show that the genetic algorithm is more efficient than the exact methods, especially for greater instances that are similar to real problems.

Keywords: Operations research. Linear programming. Genetic algorithms. Underground mining. Optimization. Cut-off grades. Scheduling.

List of Figures

List of abbreviations and acronyms

COG	Cut-off grade
GA	Genetic algorithm
NPV	Net present value
OR	Operations research

Contents

Introduction

Development in engineering, computation, administration, and economy has been characterized by the increasing use of optimization models with paradigms to represent and solve decision-making problems. Grounded in mathematics, statistics, and computation, operations research (OR) (TAHA, 2007; WINSTON, 2004) is dedicated to solving problems aiming to select, according to criteria, the best choice among a set of alternatives.

Countless industries and companies face problems related to task scheduling, which may have been caused by resource misallocation and poorly defined processes. Processes can be optimized by executing careful task scheduling planning, which results in a better control of the production flow: meeting deadlines and scheduling tasks to the best use of available resources.

The mining industry has its own procedures and differs from other sectors mainly in three aspects: large capital investment, long-term payback, and limited asset life. Thus, planning is a crucial task in mining. (VICKERS, 1962), even in the early years of OR applications in mining, lists six areas of application that are still relevant these days:

- **Allocation of resources:** being it capital, manpower, equipment, etc.;
- **Scheduling:** in regards to optimize the sequence that a ore deposit will be mined;
- **Inventory:** understanding of different supplies inventory needed to a mine operation balancing availability and inventory size;
- **Replacement and maintenance:** decisions over new investments and maintenance cycles of equipment;
- **Queuing:** in regards to equipment interactions within a mining operation;
- **Simulation:** use of simulation (OR technique) to evaluate new processes.

If the problem translation into mathematical functions adjusts to one of the available mathematical programming techniques, a solution can be found (TAHA, 2007). The OR problem solution proposal starts with the model definition and an exploratory study of the mathematical and computational tools available that would better suit the presented situation (TAHA, 2007). There are many possible classifications to an optimization problem, fundamentally depending upon the decision variables and parameters considered. While exact methods can be efficient at solving some problems, other problems will require heuristic methods, once an exact solution would lead to an unmanageable complexity.

Scheduling problems represent the decision-making process used by the manufacturing and services industries (PINEDO, 2016). In general, they aim to solve resource allocation (machines) for a given set of tasks (jobs). They are crucial in any procedural industry, being no different to mining, where there is an extra benefit: the possibility of profit anticipation depending on the scheduling strategy.

As explained by (BRUCKER, 2006), some scheduling problems can be solved by reducing them to combinatorial optimization problems, others by using standard techniques, such as dynamic programming, and branch-and-bound methods. However, as shown by (PINEDO, 2016), most of the scheduling problems are *NP-hard* problems, meaning that they are complex enough to take polynomial time to be solved. This difficulty is also the situation for mine scheduling problems, where we have to consider multiple machines, constraints, task weights (grades, benefit, and so on), and activities. In some cases, it is necessary to abandon the search for the optimal solution and simply look for a quality solution in reasonable computational time through heuristic procedures.

In this context, genetic algorithms can play an important role: they are great approaches for hard optimization problems, where classic optimization methods fail due to difficult characteristics, (KRAMER, 2017). Genetic Algorithms can adapt to different problems and solution spaces and have the capability of finding near-optimum solutions in a reasonable time.

Objectives and research relevance

This dissertation aims to explore the underground mining long-term-scheduling problem, implementing different algorithms and solvers to optimize cut-off grades (mine blocks selection) and sequence of an underground mine.

Consequently, three areas are explored: computation, considering the implementation of different algorithms; mathematics, with the study of the optimization methods and the proposition of models; and engineering, considering a practical problem that is significant to both the operations research and mining industry.

Additionally, there is an emerging awareness in technology and innovation areas for the Industry 4.0 topic (FAPESP, 2017). Industry 4.0 covers mathematical programming strategies, data mining, artificial intelligence, and machine learning applied to production processes focusing on turning the decision-making processes more efficient. This subject overlaps the objectives of this dissertation.

Dissertation outline

The first part of this dissertation comprises of the whole theoretical framework, presenting and discussing the mining industry, the class of scheduling problems within Operational Research, and solving methods available.

The second part of the dissertation defines the problem, proposes different mathematical models, tests different integer linear programming solvers, and proposes an alternative algorithm for treating the problem.

Thus, this dissertation is organized following this structure:

- **Part I: Theoretical framework**:
  - **Chapter 1 - Mining industry** introduces the theoretical framework of the mining industry and strategic mining planning, giving the technical basis for the understanding of the problem situation herein researched;
  - **Chapter 2 - Scheduling problems** reviews operational research techniques with regards to scheduling problems and their linear programming approaches;
  - **Chapter 3 - Solving methods** provides some background on techniques for solving the models proposed in Chapter 2, giving basis to support the development of the proposed solution;
- **Part II: Project development**:
  - **Chapter 4 - Problem definition and literature review** defines the problem situation herein studied, presents a literature review of similar problems with solutions that have been applied and, proposes mathematical models to the problem of this dissertation;
  - **Chapter 5 - Implementation of the solving methods** discusses the methodology with the implementation of the proposed models using different integer linear programming solvers, besides the proposition and implementation of a genetic algorithm as a resolution alternative;
  - **Chapter 6 - Case study** presents some case studies, solving a real problem, summarizing the findings, analysis, and discussion;
  - **Chapter 7 - Conclusions** finishes with conclusions and subjects for future work.

Part I

Theoretical framework

1 Mining industry

In a broad sense, mining can be defined as the act of extracting mineral substances from the Earth and making them available to human use in a profitable way. Thus, a mineral should only be extracted (and consequently being considered ore) if it carries value.

Mining is probably the second venture of humankind as a civilization, given that agriculture was the first. From prehistoric times to the present, mining has played an important role in civilization's history being, together with agriculture, the primary industries that provide all basic resources needed to humanity, (HARTMAN, 2002).

The historical importance of mining and its evolution alongside human civilization is highlighted by the historical periods: from the Stone Age, through Bronze, Iron, Steel, and Nuclear Ages.

Mining products (metals and other minerals) consumption increases as human civilization develops. According to data from the Minerals Education Coalition, (MEC, 2019), an average person in US consumes about $18t$ of minerals every year.

The current mining industry is procedural and capital intensive, comprising all sorts of challenges from productivity vs. selectivity balance, utility placement, resource allocation, production schedule, and so on. Furthermore, the mining industry depends on non-renewable resources, which also have huge uncertainty attached to their quantification.

As a result, the mining industry has vast optimization potential. When facing these challenges, making the best decision will enable the extraction of the highest value from all stakeholders' perspective, making the most sustainable use of non-renewable resources.

1.1 Mining methods

Each mineral deposit is mined following a precise mining method that varies according to geological composition, geological geometry (dip, depth, etc.), geotechnical characteristics, and economic valuation. The most known mining methods classifications, as used by (HUSTRULID, 1982) and (HARTMAN, 2002), separate them in surface and underground.

Surface mining is the mechanical excavation of outcropping or near-surface mineral deposits. It can happen as open pits or open cast (stripping mining). Surface mining also includes aqueous methods such as placer and solution mining (HARTMAN, 2002, p.11).

As an example, the next figure illustrates the pushbacks (or phases) that will be mined in each year numbered from 1 to 3, the mineralization contour (orebody), and the waste rock that will be moved, in order to reach the ore.

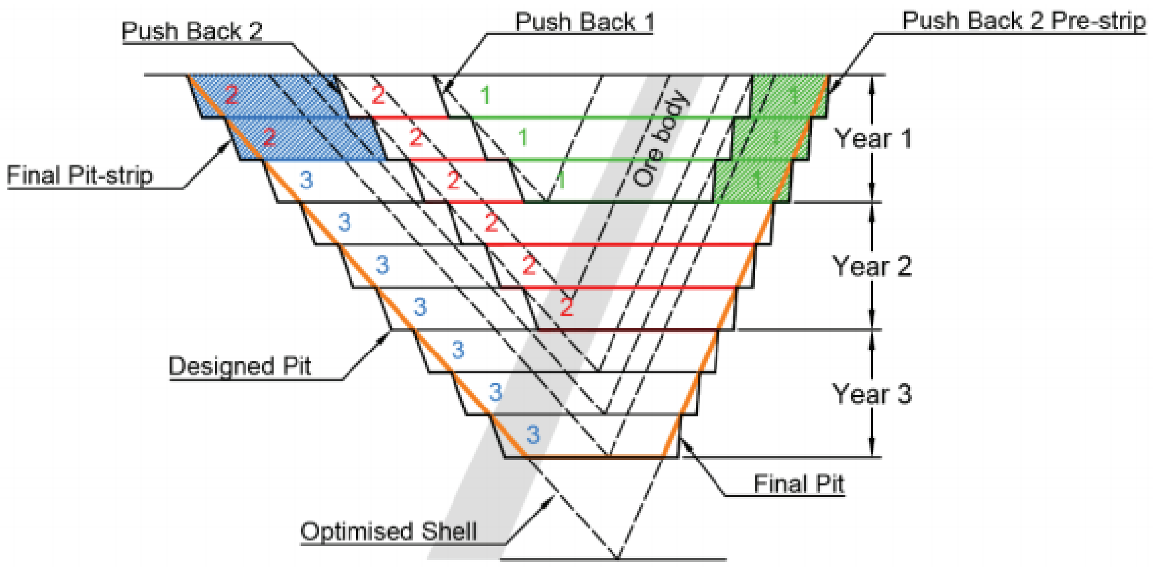


Figure 1 – Open pit vertical section.

(SABANOV; BEARE, 2015)

(HUSTRULID, 1982, p.88) categorizes underground mining methods according to the support required:

- **naturally supported methods (self-supported):** such as room and pillar and sublevel stopping.
- **artificially supported methods:** cut and fill and longwall mining.
- **caving methods:** sublevel caving and block caving.

However, as also mentioned by (HUSTRULID, 1982), due to the uniqueness of each ore deposit, the variations of each method are nearly limitless. Here, the main methods mentioned, as seen in (HUSTRULID, 1982), aim to provide the required understanding of the sequence of each one.

- **Longwall mining:** extensively applied in coal mining; it is a method more adequate to thin-bedded deposits of uniform thickness. The ore is excavated in slices (in coal, usually with continuous mining machines). The face close to the excavated area is artificially supported to provide space for drilling and ore removal. At some distance from the face, the roof may be allowed to cave, as seen in figure 2.

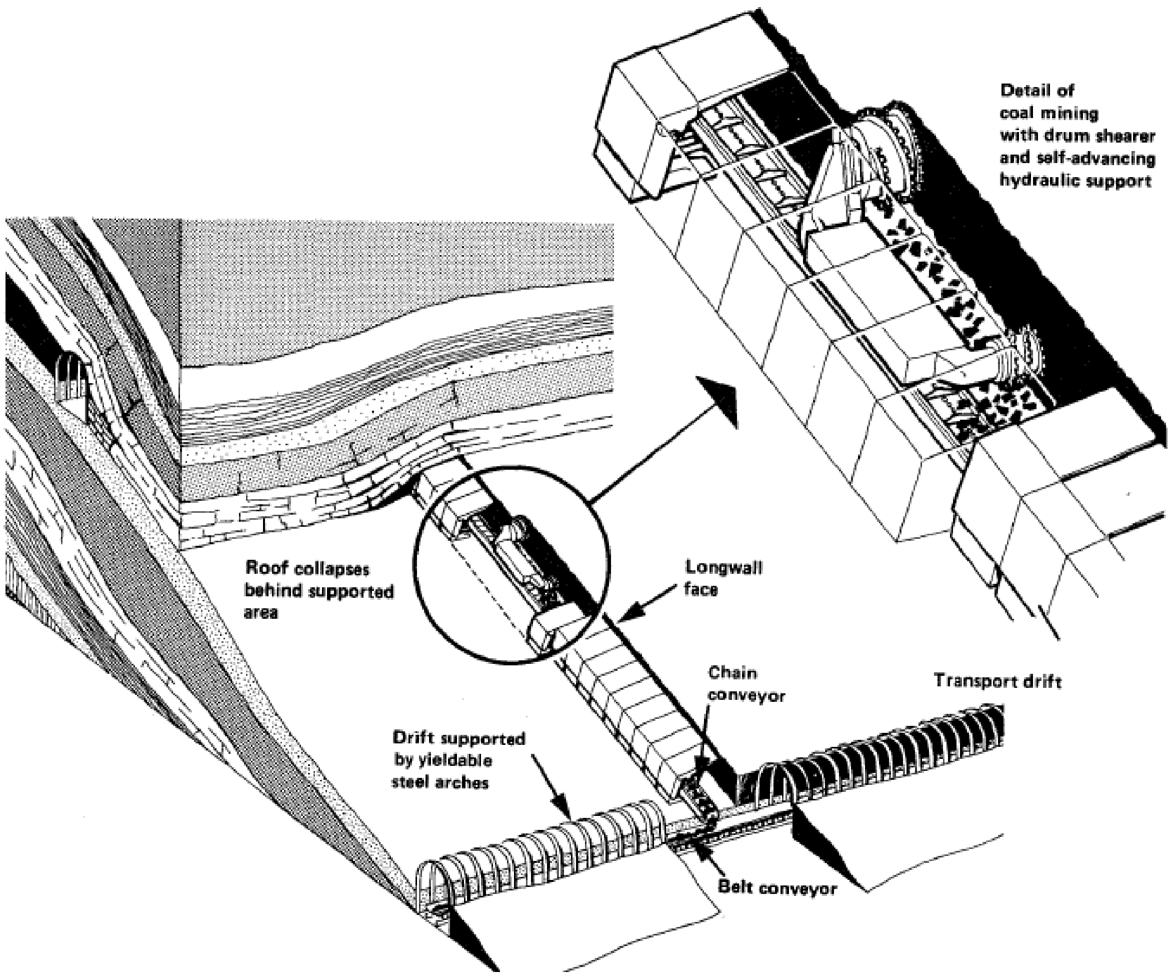


Figure 2 – Longwall mining example.

(HUSTRULID, 1982)

- **Room and pillar mining:** in this method the orebody is excavated leaving ore in pillars to support the roof (hanging wall). As shown in figure 3, rooms are usually paralleled excavated and then mined from one to another, leaving the pillars in between. It is more applicable to flat-lying orebodies. Mining might happen at different levels.

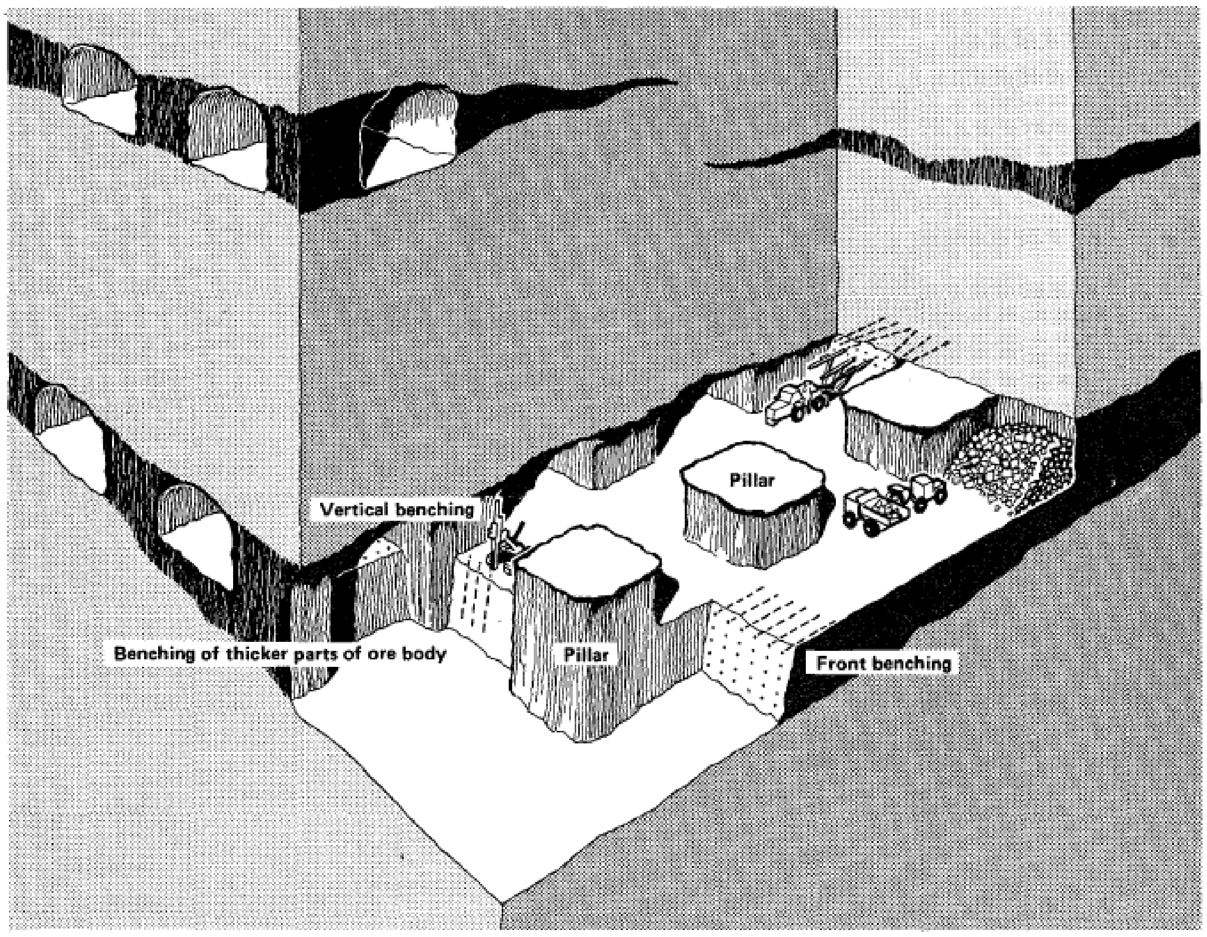


Figure 3 – Room and pillar mining design.

(HUSTRULID, 1982)

- **Cut and fill mining:** in this case, figure 4, the ore is excavated in horizontal slices from the bottom of a panel. After ore removal of one slice, the open void is filled with waste material (broken rock from waste development tunnels) or with hydraulic fill that uses plant tailings. The filling will provide wall support and also the working platform for the next cut.

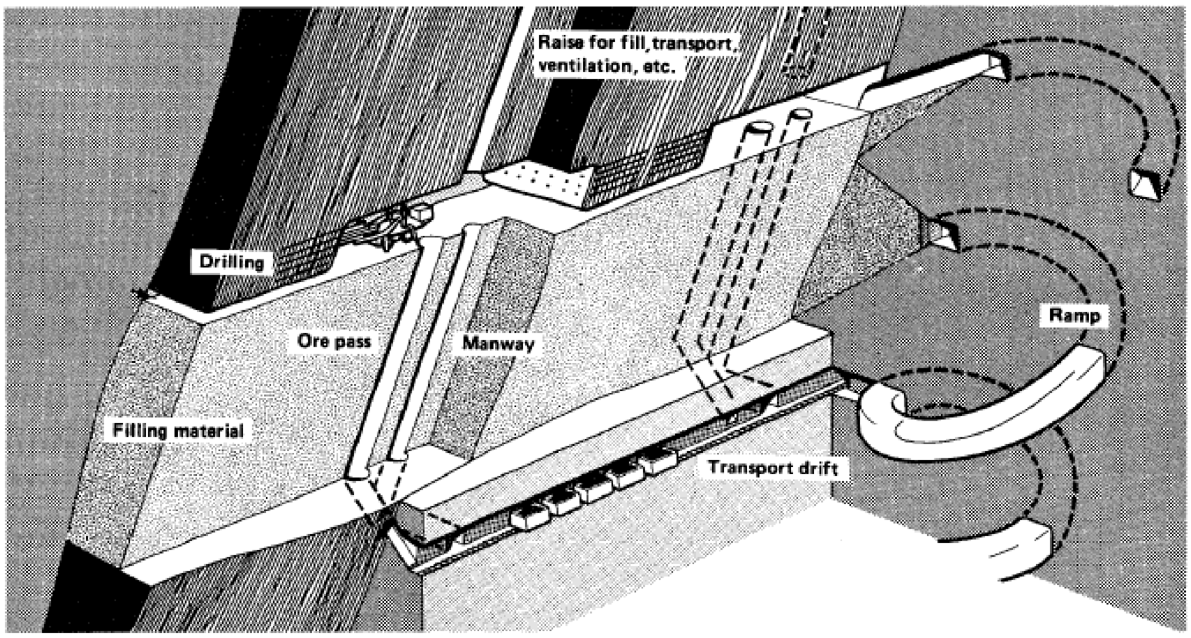


Figure 4 – Cut and fill mining method diagram.

(HUSTRULID, 1982)

- **Sublevel stopping mining:** as illustrated in figure 5, this method makes use of long-hole drilling (in a fan-shape) between tunnels excavated in ore (oredrives). Each production block (stope) from the same panel must be mined following a sequence that allows the material to flow to the drawpoint. It is applicable to more vertical orebodies since material flow only will happen in inclinations steeper than 40°.

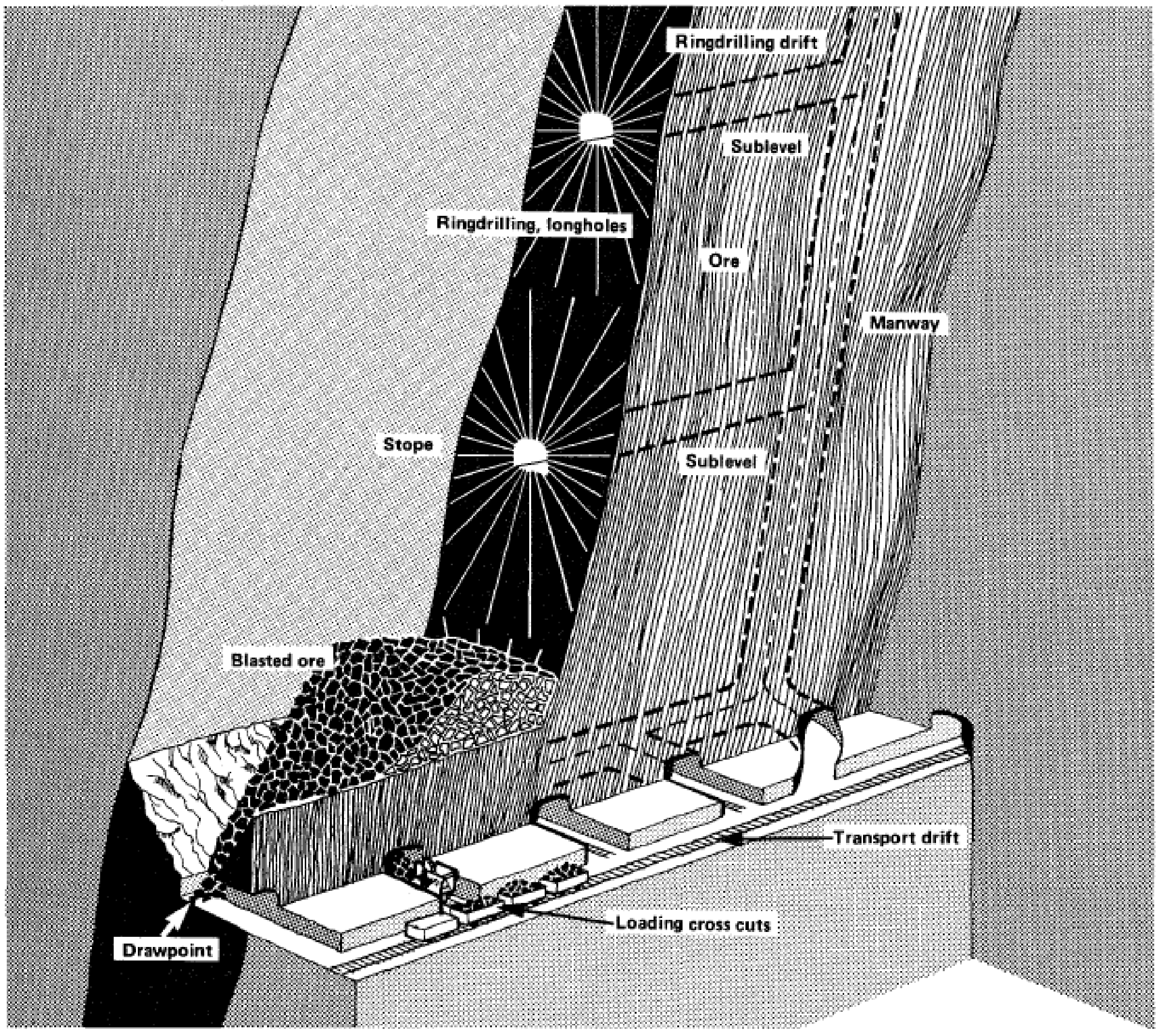


Figure 5 – Sublevel stoping mining representation.

(HUSTRULID, 1982)

- **Block caving mining:** in this method, the rock is allowed to break by itself (cave) as a result of induced stress (minimum drill and blast required), as seen in figure 6. It is applicable to large massive orebodies.

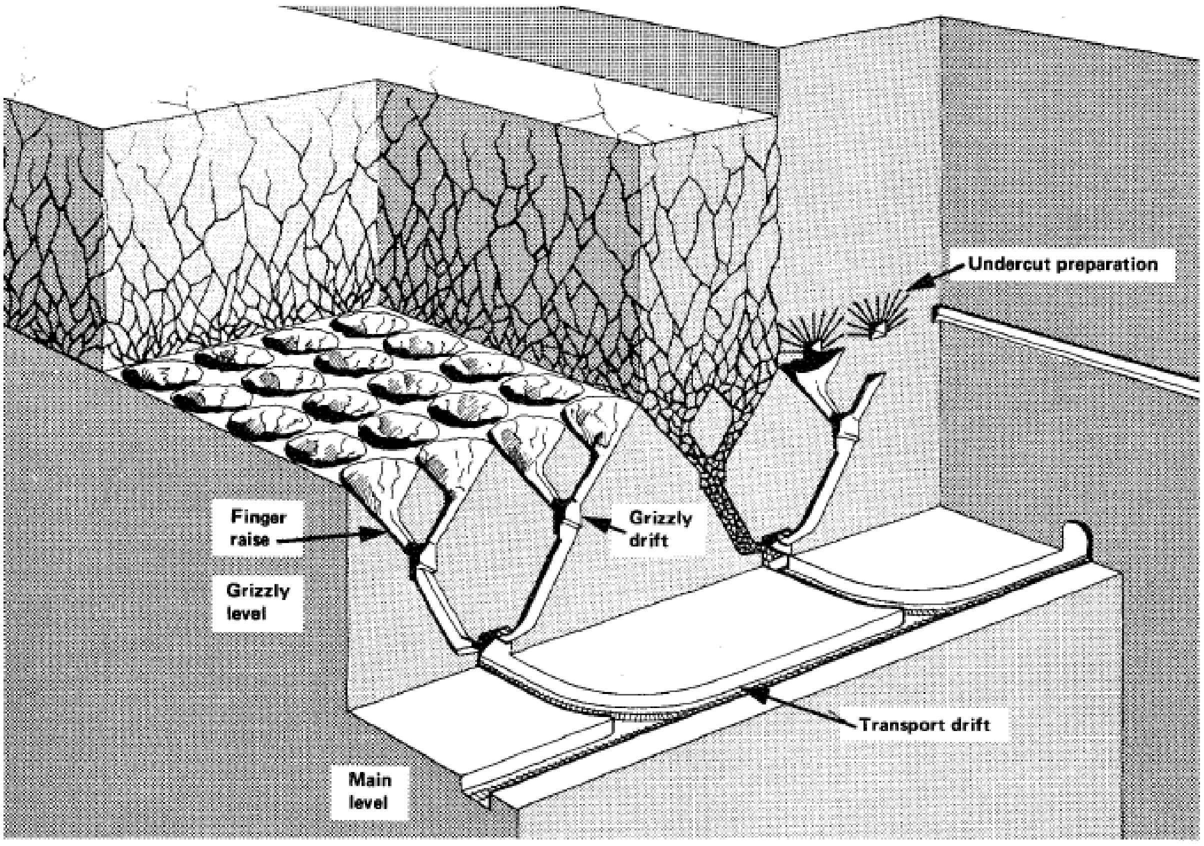


Figure 6 – Block caving schematic diagram.

(HUSTRULID, 1982)

Although the main idea of the algorithm proposed herein is its applicability to optimize value by selecting and scheduling any group of chained tasks (thus, any mining method), the work herein studied will be focused on the sublevel stoping method. There are numerous planning tools specially developed for open pit mining while just a small number for underground (which may not fully achieve the purpose of this work).

Therefore, for examples and case studies, sublevel stoping will be used. Nonetheless, the same reasoning behind the algorithms could be applied to other underground methods and even to surface mining methods.

1.2 Mine planning

Similarly to the planning of other industries, mine planning aims to provide a production schedule but, oppositely, dealing with engineering evaluations of limited lifespan assets. Consequently, besides challenges such as production scheduling and resource allocation in uncertain environments, mine planners also must to do engineering considerations such as design projects of mining workings and excavations considering geotechnical and operational constraints.

Whether during new projects, expansions evaluations, or if in operation, mining requires a high level of detail in planning. The mine planning process's main input is a three-dimensional block model representation of the deposit (geological resource) generated from the interpolation of the mineral samples gathered. Each block of this model is a discretization of the deposit and has unique attributes such as volume, density, mass, product grades, rock mass characteristics, etc. This model, such as shown in figure 7, is designed and scheduled respecting constraints such as geotechnical, operational, mineral processing, production costs, and commodity prices.

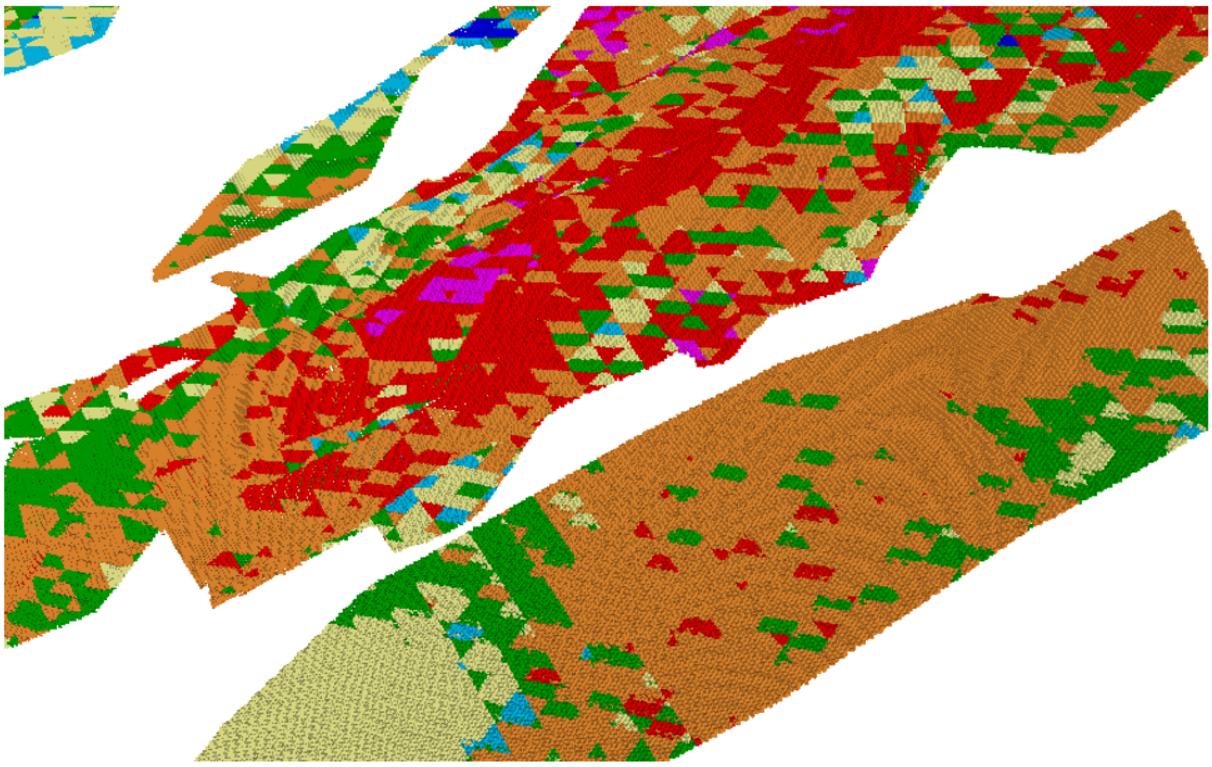


Figure 7 – Example of a block model representation of a mineral deposit. Colors represent different gold (Au) grades (concentration) in this case.

After a project has been developed to the operational level, mine planning generally follows three different stages where we have different focuses:

- **Long term:** here the focus is to optimize the return (profit) on investment. The main objective is to set up the strategic plan of the asset. Thus, the emphasis is on the whole life of mine, and the main decisions are about mine method and layout definitions, capital investments, production capabilities balance, net present value (NPV) potential of the asset, etc.

- **Medium term:** this is a tactical plan that aims to detail the strategy. It usually comprises a time window of 1 to 5 years, and the main challenges are to define production, revenues, and costs targets (the budget of a mining company) up to a monthly basis.
- **Short term:** breaking down the medium-term guidance, this stage requires an operational plan that will look to production capacities of the whole mining cycle from a monthly to a shift perspective. The focus is to allocate resources (machines and teams) to comply with the medium-term plan but also keeping the highest productivity and lowest cost of the resources.

The crucial challenge is to align business' strategic objectives (long term) with short term adaption needs due to internal and external changes (plan compliance, geological model changes, commodity price, etc.).

1.3 Strategic planning

As stated by (KENNEDY, 1990), business or strategic planning is the determination of where the enterprise wants to go and how it expects to get there. It is the core of long-term planning and aims to guarantee fulfillment of the most important management responsibilities: survival and continued profitable growth of the business. As per the main characteristics of mining business (capital intensive, long-term payback, and limited asset life), it is usually said that only a few industries require more strategic planning than mining.

(NEHRING, 2016) divides strategic mining planning decisions into five different levers:

- **Scale of operation:** a trade-off between large-scale production (bigger equipment, higher productivity, and lower costs) against higher selectivity mining (less mixing between ore and waste, less material movement);
- **Mining method selection:** choice of the method that best fits the deposit, given the pros and cons of each method;
- **Processing route:** choice of the ore processing route and marketable products, given mineral characteristics;
- **Cut-off grade:** selection of the minimum mining grade above which material will be mined or processed.
- **Mining sequence:** choice of the particular sequence of mining activities within the applied method.

Generally, scale, method, and processing route are decisions with a smaller degree of freedom than cut-off grade and sequence. Consequently, trade-offs (scenarios comparison) can be applied to guide these decisions. In addition, with mine equipment and plant facilities in place, it becomes harder to change scale, method, and processing route. On the contrary, even at the operational level, cut-off grade policy and mining sequence still have a high-profit impact and are manageable strategic levers.

Cut-off grade (COG) is the grade above which the mined material should be considered ore and thus be processed; and below which the material should be left *in situ* or dumped as waste. More simplistic early approach (CUMMINS; GIVEN; AIME., 1973 apud HALL, 2014), considered just one COG, usually with a mindset that "each ton of ore has to pay for itself" (HALL, 2014).

The following equation (1.1) can calculate this break-even COG:

$$\textit{Break-even COG} = \frac{\textit{costs}}{\textit{product price} \times \textit{recovery}} \tag{1.1}$$

As the cost structure is better understood and each of its component is detailed, there is a possibility that material below this break-even COG increase value for the company in specific circumstances.

Given its economic aspect, COG is also the decision point between ore and waste; it will be applied not only to *in situ* rock decisions but also to broken rock from development (and any other kind of material that will be moved anyway). In this latter case, only the downstream costs are considered for the economic calculation. There may be even limiting grades between different processing routes, denominated cut-over grades.

Thus, instead of just one fixed COG, a set of different COGs is defined (i.e., a COG policy) and will have each COG that is applied for each case. Therefore, what may seem an easy concept, become more complex and intricate. In addition, usually, a COG policy is set at the beginning of the planning process, and its objective function (NPV in most of the cases) is not assessed at this point.

For the mining sequence, there will be as many different ones as the multiplication of the number of decision possibilities of each time period. Since there may be more activities to plan (tunnel development or stope ore) than the resource constraints, prioritization is needed to decide which tasks will be considered at each period (day, month, or year of a schedule). Normally, there are some rules of thumb (greedy algorithms from the mathematical perspective) that are used to guide this process. These step-by-step procedures (such as anticipating high-grade zone tasks) cannot guarantee that optimum (or close enough to optimum) decisions are made.

(LANE, 2015) may be the main reference for cut-off grade optimization with his book "The economic definition of ore" first published in 1988 and with articles of the same subject published before.

Besides production costs, (LANE, 2015) introduced three new aspects to cut-off grade calculations:

- time factor and, therefore, NPV as the main objective of the optimization (instead of cash flow previously used);
- deposit remaining grades and volumes;
- production capabilities at different stages of a mining system.

Although is straightforward, NPV as optimization objective turns the calculation more complex and introduces the COG calculation step to the scheduling step.

Deposit remaining grades and volumes mean the sensitiveness of the mineral deposit with regards to the COG applied. This can be better expressed by a grade-tonnage curve (or ton-vs.-grade graph), as shown in figure 8. Thus, in general, (LANE, 2015) proposes a higher COG at the beginning of a mining operation (higher remaining deposit) than in its later years.

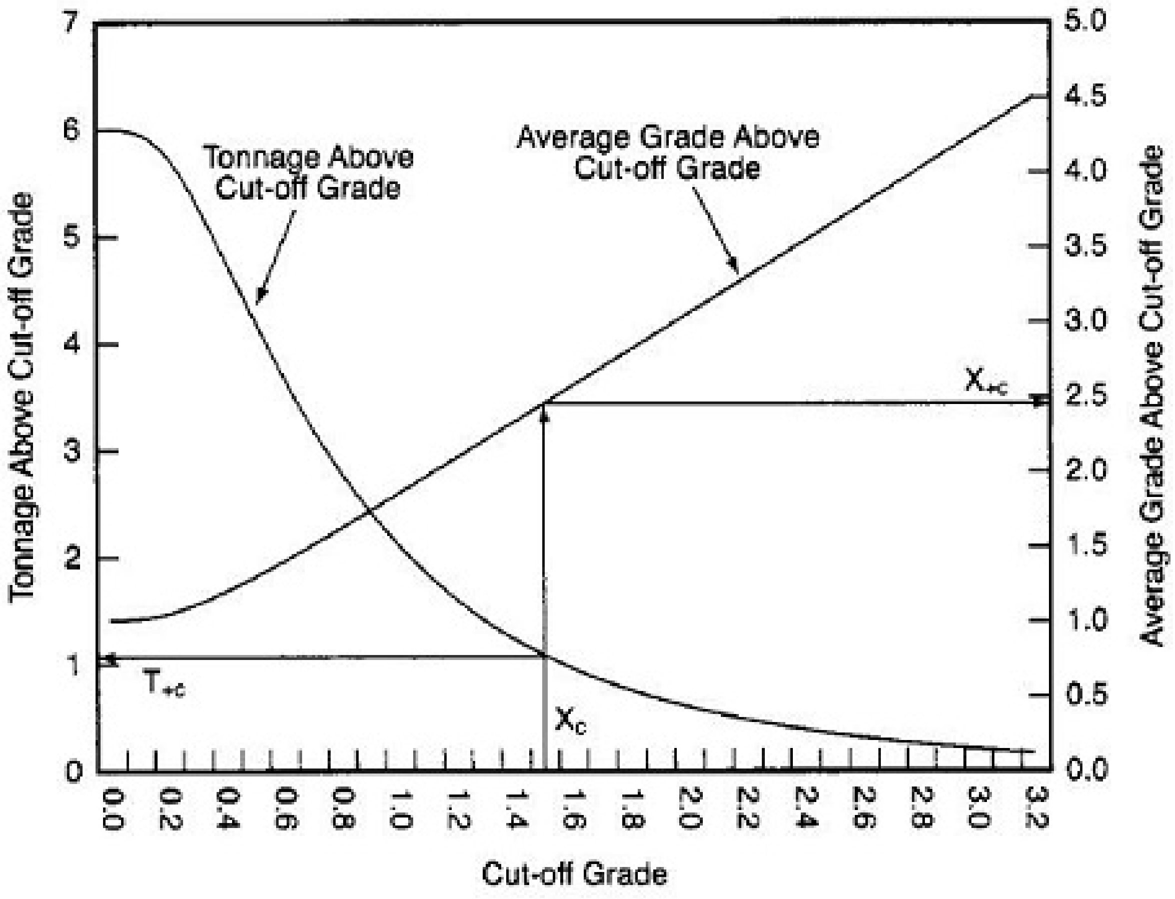


Figure 8 – A grade-tonnage curve showing a deposit sensitivity to cut-off grade.

(UNIVERSITY, 2019)

Production capabilities refer to the use of COG to balance the stages of the mining system. Thus, if a particular deposit/mine system has more capacity to treat (process) the ore than to access (developed) it, the COG used would be lower than in the opposite situation. (LANE, 2015) particularly considered three main production capacities:

- **Mining:** this refers to the capacity of accessing the mineralized zone. In an underground mine is the development of tunnels to reach mineralized zones.
- **Treatment:** this is the capacity of ore production (mining itself) and treat ore in processing plants;
- **Marketing:** this is the capacity of refineries, smelters, and selling the final product.

Summing up, (LANE, 2015) optimum cut-off grades aim to optimize NPV while making the best use of the remaining resource and balancing mine production capabilities.

On one hand, this approach set the theoretical standard for life-of-mine optimization. On the other, due to its iterative approach, it may be an impractical algorithm: hard to apply (especially in underground mining) and time-consuming. Also, while the three production stages might well cover a mining system, some cases may need the introduction of new ones (e.g. exploration).

(HALL, 2014) proposes an updated version of (LANE, 2015) theory, but aiming to "account for everything", i.e., all value drivers of a mining asset (including method, COG, sequence, etc.). The final outcome of such approach is a 3D graph that plots cut-off grades vs. production rates vs. values (usually NPV) of different mining strategies, the so-called hill of value. An example is given in figure 9. However, it ends up being a framework for a discrete analysis (a trade-off between some possible cases of COG and production rate).

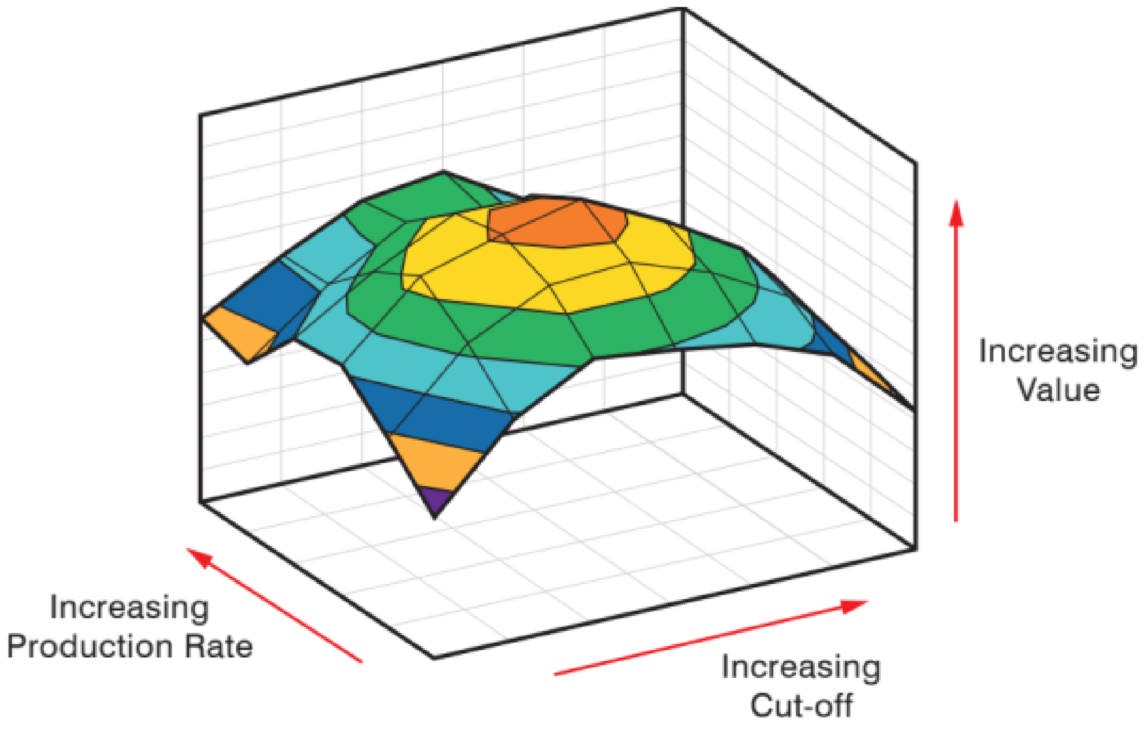


Figure 9 – A hill-of-value graph.

(HALL, 2014)

In both cases, the cost definition (its projection and also its allocation) is a critical input. (KING, 2004) proposes an approach for classifying costs that is more suitable to strategic evaluations and will be used in this work. He divides costs into four classes:

- **foundation costs:** costs that incur at specific times regardless of processing rates, such as exploration of new areas, clean of contaminated areas, or feasibility studies.

This type of costs may not interfere in planning decisions, since they may occur independently of the strategy chosen;

- **activity costs:** costs related to production (mining, development, processing, refining, etc.) thus being expressed by $\$/t$ or $\$/m$ of the respective driver or $\$/h$ for equipment, etc.;
- **reserve timing costs:** costs incurred as reserves are mined. They are costs related to the decommissioning of mining workings, main infrastructures, dewatering, etc. These costs are better expressed as $\$M$;
- **period costs:** costs incurred annually while the mining asset operates not varying if more or less material is processed. This is best described as $\$/year$.

1.4 Strategic planning workflow

All these strategic evaluations, discussed in section 1.3, are based on operational life-of-mine designs and the schedule of the deposits of a company. Similarly to the workflow seen in (WANG, 2019), and regardless of the mining method, commodity, or any other specificity, a general workflow can be defined as follows:

- **Mining blocks definition:** in this first step, geotechnical and operational parameters are considered as geometrical constraints to the design of the minimum mining blocks. The objective is to have a file, where each solid (three-dimensionally located) represents a mining activity and has attributes such as volume, density, mass, grades, etc. A cut-off grade is defined, i.e., the commodity grade that represents the break-even value between revenues and costs. Any activity with a grade below this value is considered waste and, in the opposite case, is considered ore (a mineral that can be profitably extracted). For instance, stopes and development solids can be broken in sections that represent operational blasts, as seen in figure 10.

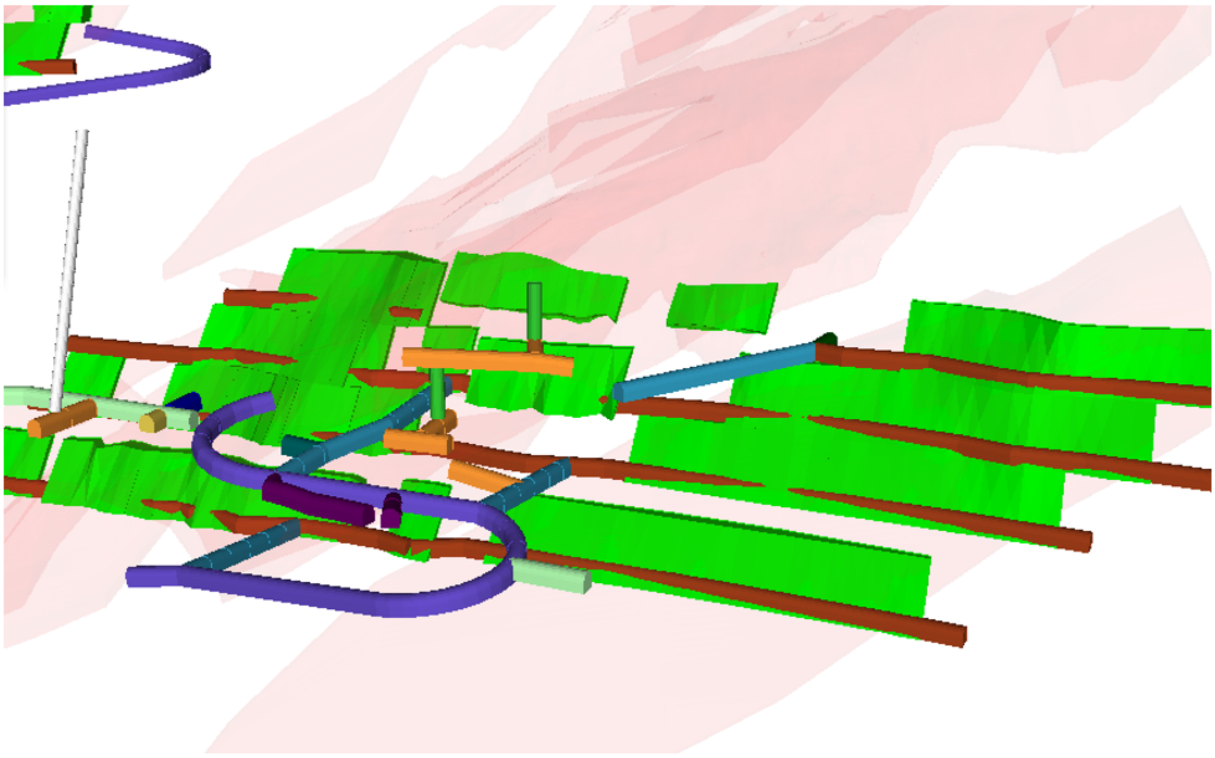


Figure 10 – Example of a sublevel stoping design.

Green solids represent operational shapes of ore stopes; Brown solids, oredrives tunnels. Blue ones, crosscuts tunnels; and purple ones the main decline. Light red contour represents the orebody.

- **Logical sequence:** in the second step, dependencies between the solids (mining tasks) are created accordingly to the specific mining method. Links between predecessors and successors tasks are established, creating, between activities, a mandatory sequence that can be represented in a Gantt chart.

  As an example, figure 11, a dependency from a development tunnel and its related mining excavation (stope), may be created, meaning that the stope will only be mined after the development is excavated.

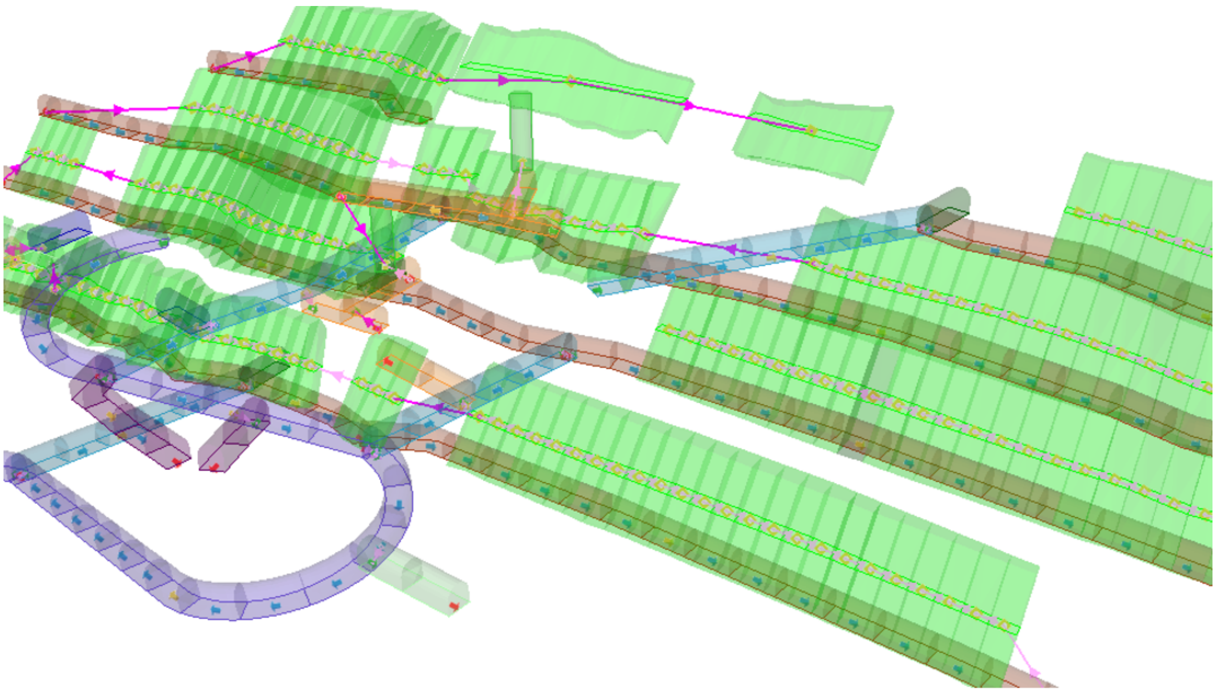
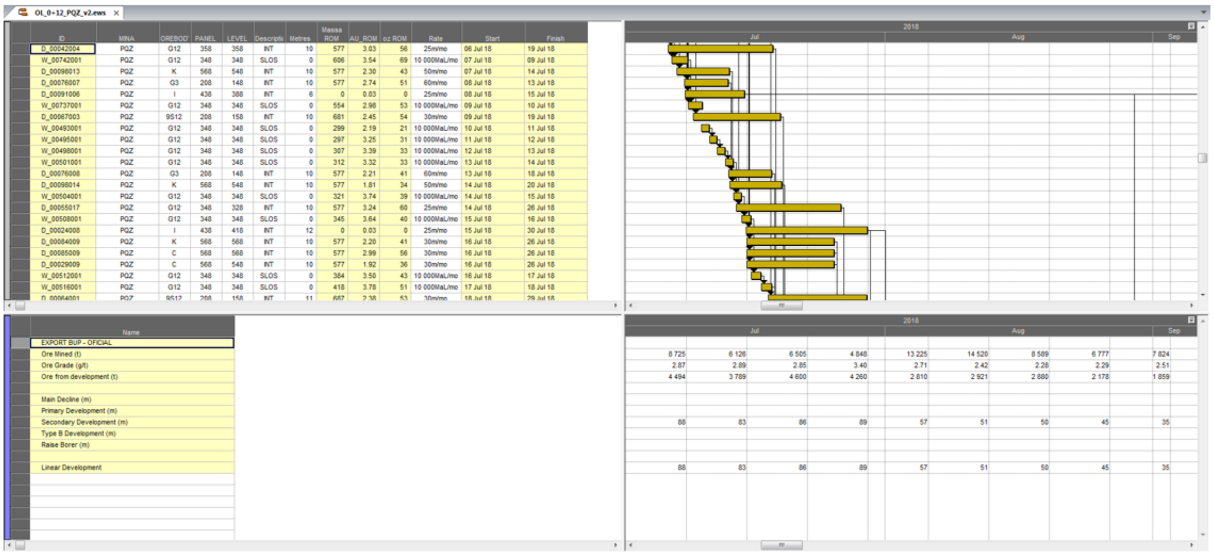


Figure 11 – Example of the logical sequence.

Using the same design shown in figure 10 we have pink arrows representing precedencies. Thus, crosscuts can only start after a nearby decline segment is developed, an oredrive has to be developed before sublevel stoping starts in retreat, and so on.

- **Operational sequence:** in this last step, the logical sequence is leveled regarding production capabilities constraints (such as equipment production capabilities, total ore produced or hoisted, total development meters, total ore processed by a plant, contaminant grades, etc.). This leveling step aims to optimize one or more previously defined attributes, such as total profit (cash flow), NPV, grade, total production, etc. while managing the risks of each assumption considered along with the three steps. Figure 12 illustrates that.


OL_0+12_PQZ_v2.ews

ID	MINA	OREBOD'	PANEL	LEVEL	Descriptic	Metres	Massa ROM	AU_ROM	oz ROM	Rate	Start	Finish
D_00042004	PQZ	G12	358	358	INT	10	577	3.03	56	25m/mo	06 Jul 18	19 Jul 18
W_00742001	PQZ	G12	348	348	SLOS	0	606	3.54	69	10 000MaL/mo	07 Jul 18	09 Jul 18
D_00098013	PQZ	K	568	548	INT	10	577	2.30	43	50m/mo	07 Jul 18	14 Jul 18
D_00076007	PQZ	G3	208	148	INT	10	577	2.74	51	60m/mo	08 Jul 18	13 Jul 18
D_00091006	PQZ	I	438	388	INT	6	0	0.03	0	25m/mo	08 Jul 18	15 Jul 18
W_00737001	PQZ	G12	348	348	SLOS	0	554	2.98	53	10 000MaL/mo	09 Jul 18	10 Jul 18
D_00067003	PQZ	9S12	208	158	INT	10	681	2.45	54	30m/mo	09 Jul 18	19 Jul 18
W_00493001	PQZ	G12	348	348	SLOS	0	299	2.19	21	10 000MaL/mo	10 Jul 18	11 Jul 18
W_00495001	PQZ	G12	348	348	SLOS	0	297	3.25	31	10 000MaL/mo	11 Jul 18	12 Jul 18
W_00498001	PQZ	G12	348	348	SLOS	0	307	3.39	33	10 000MaL/mo	12 Jul 18	13 Jul 18
W_00501001	PQZ	G12	348	348	SLOS	0	312	3.32	33	10 000MaL/mo	13 Jul 18	14 Jul 18
D_00076008	PQZ	G3	208	148	INT	10	577	2.21	41	60m/mo	13 Jul 18	18 Jul 18
D_00098014	PQZ	K	568	548	INT	10	577	1.81	34	50m/mo	14 Jul 18	20 Jul 18
W_00504001	PQZ	G12	348	348	SLOS	0	321	3.74	39	10 000MaL/mo	14 Jul 18	15 Jul 18
D_00055017	PQZ	G12	348	328	INT	10	577	3.24	60	25m/mo	14 Jul 18	26 Jul 18
W_00508001	PQZ	G12	348	348	SLOS	0	345	3.64	40	10 000MaL/mo	15 Jul 18	16 Jul 18
D_00024008	PQZ	I	438	418	INT	12	0	0.03	0	25m/mo	15 Jul 18	30 Jul 18
D_00084009	PQZ	K	568	568	INT	10	577	2.20	41	30m/mo	16 Jul 18	26 Jul 18
D_00085009	PQZ	C	568	568	INT	10	577	2.99	56	30m/mo	16 Jul 18	26 Jul 18
D_00029009	PQZ	C	568	548	INT	10	577	1.92	36	30m/mo	16 Jul 18	26 Jul 18
W_00512001	PQZ	G12	348	348	SLOS	0	384	3.50	43	10 000MaL/mo	16 Jul 18	17 Jul 18
W_00516001	PQZ	G12	348	348	SLOS	0	418	3.78	51	10 000MaL/mo	17 Jul 18	18 Jul 18
D_00064001	PQZ	9S12	208	158	INT	11	687	2.38	53	30m/mo	18 Jul 18	29 Jul 18

Name	2018								
	Jul				Aug				Sep
EXPORT BUP - OFICIAL									
Ore Mined (t)	8 725	6 126	6 505	4 848	13 225	14 520	8 589	6 777	7 824
Ore Grade (g/t)	2.87	2.89	2.85	3.40	2.71	2.42	2.28	2.29	2.51
Ore from development (t)	4 494	3 789	4 600	4 260	2 810	2 921	2 880	2 178	1 859
Main Decline (m)									
Primary Development (m)									
Secondary Development (m)	88	83	86	89	57	51	50	45	35
Type B Development (m)									
Raise Borer (m)									
Linear Development	88	83	86	89	57	51	50	45	35

Figure 12 – Gantt chart representation of a leveled plan.

The leveled plan is then economically evaluated. Generally, the sum of all the discounted future cash flows (the net present value, NPV) is the measure that will be used to balance all sorts of inputs. Therefore, all decisions made during method selection, parameters, and assumptions considered only will have their values assessed at the very end of the workflow.

Different strategies (scale of operation, mining method, processing routes, cut-off grades, and mine sequence) are usually evaluated by repeating the whole workflow with different assumptions (being them in steps 1, 2, or 3) and doing trade-off analyses (from simple scenarios comparison to more complex hill-of-value graphs).

In addition, as inputs and assumptions are dynamic, this is an iterative workflow. With new exploration information, new geological models will be generated; with different market perspectives, new price assumptions; with operational improvements and new technologies, new operational parameters; and thus, this planning cycle and the decisions attached to it need to be reviewed.

In summary, usually, cut-off grades and mine sequence are solved separately. Some of the current algorithms solve one of them using the other as an assumption. Furthermore, this process is, most of the time, too sophisticated (requiring numerous scenarios) and, thus, not fully applied in the current mining industry. Underground mining adds extra complexity: mine costs are usually higher, and, differently from an open pit, it is not required to remove the underground stopes to access adjacent ore.

2 Scheduling problems

A scheduling problem consists of assigning a set of tasks (jobs, processes, actions, etc.) to a given set of resources (machines, people, etc.) while satisfying allocation constraints and optimizing an objective function, (BRUCKER, 2006). It is a whole area of study within the area overlapped by Operations Research and Production Planning.

Different objective functions can be used. (LEUNG, 2004) and (PINEDO, 2016) highlight:

- minimization of makespan (the time that elapses from the start of work to the end of all tasks);
- minimization of total tardiness (the sum of all delays in delivering tasks);
- minimization of the number of tardy jobs (delivered after due date);
- minimization of weighted tardiness (e.g., time-related cost);
- minimization of the total weighted completion time;
- minimization of the total waiting time (i.e., the sum of the differences between starting and releasing time for each task);
- optimization of multiple objectives, and others.

General parameters considered in a scheduling problem include:

- job release time;
- job starting time;
- job processing time;
- due dates (committed finish time for a job);
- machine or job preparation time;
- number of machines;
- machines processing rates;
- machines processing flows;
- precedencies between jobs

- possibility of preemption (job splitting);

Solutions for scheduling problems are not obtained as a closed-form expression. They are, instead, determined by algorithms: a sequence of procedures that are applied multiple times to the problem until the best solution is found.

According to the adopted formulation (fundamentally dependent upon constraints, parameters, and the number of machines), a scheduling problem can be solve for an exact solution in polynomial time.

These problems are classified in the Complexity Theory as **P problems** (*Polynomial Time*) since they can be solved by deterministic algorithms which complexity is polynomial determined. Other examples are calculating the greatest common divisor and determining if a number is prime.

However, there are other problems belong to the class of **NP problems** (*Non-Deter-ministic Polynomial Time*): problems whose solutions are not found in polynomial time by algorithms known. NP problems are considered hard to solve. Those are also problems that belong to the **NP-hard** class (i.e., they are at least as hard as an NP problem). These add expressive difficultness on the development of efficient algorithms that find optimum solutions.

Usually, scheduling problems have exponential complexity and can not be solved in polynomial time, given the input size. In these cases, it is required to abandon the optimum solution and simply look for a high-quality solution ("good enough") through heuristics procedures.

Classic scheduling problems are single machines, parallel machines, flow shop, job shop, and open shop. A simple definition based on (PINEDO, 2016) follows:

- **single machine:** in this configuration, a set of jobs (or tasks) have to be processed by a unique machine. Usually, jobs have a release time (i.e., when jobs are available to the machine), and optimization objective function is to minimize the makespan.
- **parallel machines:** in this environment, a set of jobs have to be processed by any of a set of machines. These machines can have the same processing rate or different ones, process all or a subset of jobs;
- **flow shop:** in this case, there are a number of machines in series. Each job has to be processed on each of the machines following the same processing route (unidirectional flow).
- **job shop:** in this case, each job has its own predetermined route of machines to follow (machines in series with non-unidirectional job flow).

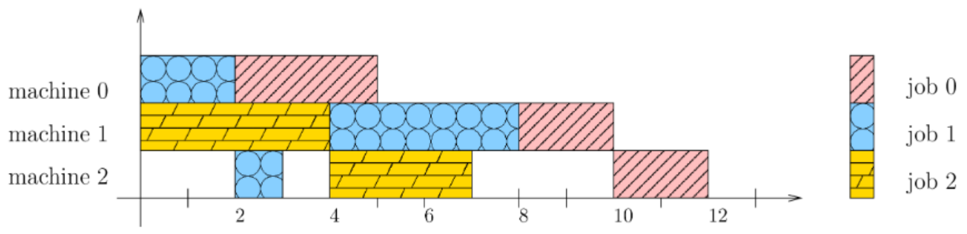


Figure 13 – Job shop solution example.

(OR-TOOLS, 2019)

- **open shop:** in this situation, there are, again, a number of machines in series, and each job has to be processed on each of the machines. However, differently from the flow shop problem, in open shops, there are no restrictions with regard to the routing of each job. Thus different jobs can have different processing routes. In addition, some of the processing times can be zero. Part of the schedule solution is to determine the best machine sequence for each job.

Key problems to the development of this dissertation (single machine, parallel machines and flow shop) are detailed in the next sections.

2.1 Single Machine

The single machine environment is very simple and a special case of all other environments, such as machines in parallel or machines in series. In this section, two different single machine models are analyzed. The minimization of total weighted completion time is considered as the objective function (Equations 2.1 and 2.6). Even though these problems are quite easy and can be solved by simple priority rules, these problems still serve to assist in modeling more complex problems.

For the first model, the decision variables are defined as:

$$x_i \quad = \quad \text{starting time that a job } i \text{ is processed at the planning horizon.}$$

$$w_{ik} \quad = \quad \begin{cases} 1 & \text{if job } i \text{ precedes job } k. \\ 0 & \text{otherwise.} \end{cases}$$

And the integer linear programming model considering $i = 1, ..., n$ jobs, p_i the processing time of job i and w_i the weight of job i (a priority factor) is:

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n} w_i.(x_i + p_i) && (2.1)\\
\text{subjected to} \quad & x_i + p_i \leqslant x_k + M(1 - w_{ik}) \quad \forall i, \forall k, \quad i \neq k && (2.2)\\
& w_{ik} + w_{ki} = 1 \quad \forall i, \forall k, \quad i \neq k && (2.3)\\
& x_i \geqslant 0 \quad \forall i && (2.4)\\
& w_{ik} \in \{0, 1\} \quad \forall i, \forall k && (2.5)
\end{aligned}$$

The first two sets of constraints (Equations 2.2 and 2.3) ensures no overlap of jobs. The second set of constraints (Equation 2.3) ensures that only one job can be processed at any point in time. Equations 2.4 and 2.5 contain the integrality constraints on the variables.

In this first model, M is a large coefficient. Big-M constraints are typically used to represent the implications of an "on-off" decision. Although quite useful to the model, this constant can propagate computational inaccuracies. The second proposed model eliminates the use of M.

For the second model, the decision variables are indexed in time and are defined as:

$$x_{it} = \begin{cases} 1 & \text{if job } i \text{ starts at time } t. \\ 0 & \text{otherwise.} \end{cases}$$

In this case, we have an integer programming formulation with time-indexed variables proposed by (PINEDO, 2016). Considering $i = 1, ..., n$ jobs, p_i the processing time of job i and $t = 1, ..., l$, with $l = \sum_i p_i - 1$, the model is described as:

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n} \sum_{t=0}^{l} w_i.(t + p_i).x_{it} && (2.6)\\
\text{subjected to} \quad & \sum_{t=0}^{l} x_{it} = 1 \quad \forall i && (2.7)\\
& \sum_{i=1}^{n} \sum_{s=max(t-p_i+1,0)}^{t} x_{is} = 1 \quad \forall t && (2.8)\\
& x_{it} \in \{0, 1\} \quad \forall i, \forall t && (2.9)
\end{aligned}$$

The first set of constraints (Equation 2.7) ensures that a job i can start only at one point in time. The second set of constraints (Equation 2.8) ensures that only one job can be processed at any point in time. Equation 2.9 contains the integrality constraints on the variables. The major disadvantage of this formulation is the number of variables required (PINEDO, 2016).

2.2 Parallel Machines

Parallel machine scheduling problems deal with the decision-making of allocating jobs in any of the available machines. It is irrelevant which machine will process the job, but it cannot be processed on more than one machine at the same time neither be processed by more than one machine (there is no processing route).

(PINEDO, 2016) classifies this problem according to the machines:

- **identical machines:** job j requires a single operation and may be processed on any of the m machines **or on any that belongs to a given subset**. Thus, it can happen that a job j can only be processed on machines that belong to a specific subset M_1;
- **uniform machines:** parallel machines with different processing rates (speeds). Thus, if all machines have the same processing rate then this situation will be identical to the previous one;
- **unrelated machines:** in this case, the processing rate of each machine are dependent on the job being processed. Therefore, this is a further generalization of the previous.

The integer linear programming model for the identical machine's problem is described as seen in (BARBOSA, 2014). Decision variables in a classic optimization model that minimizes the makespan are defined as:

$$x_i = \text{starting time that a job } i \text{ is processed at the planning horizon.}$$

$$z_{ij} = \begin{cases} 1 & \text{if job } i \text{ is allocated at machine } j. \\ 0 & \text{otherwise.} \end{cases}$$

$$w_{ik} = \begin{cases} 1 & \text{if job } i \text{ precedes job } k. \\ 0 & \text{otherwise.} \end{cases}$$

And the integer linear programming model with objective function (minimize makespan $Cmax$, Equation 2.10) and constraints, considering $i = 1, ..., n$ jobs, $j = 1, ..., m$ machines and p_i the processing time of job i is:

$$
\begin{aligned}
\text{minimize} \quad & Cmax && (2.10)\\
\text{subjected to} \quad & \sum_j z_{ij} = 1 \quad \forall i && (2.11)\\
& z_{ij} + z_{kj} - w_{ik} - w_{ki} \leqslant 1 \quad \forall i, \forall k, \forall j, \quad i \neq k && (2.12)\\
& z_{ij} + z_{kh} + w_{ik} + w_{ki} \leqslant 2 \quad \forall i, \forall k, \forall j, \forall h, \quad i \neq k, \quad j \neq h && (2.13)\\
& x_i + p_i - (1 - w_{ik})M \leqslant x_k \quad \forall i, \forall k, \quad i \neq k && (2.14)\\
& Cmax \geqslant x_i + p_i \quad \forall i, \forall j && (2.15)\\
& x_i \geqslant 0 \quad \forall i && (2.16)\\
& w_{ik}, z_{ij} \in \{0, 1\} \quad \forall i, \forall k, \forall j && (2.17)
\end{aligned}
$$

Constraint 2.11 ensures that each job will be allocated in only one machine. 2.12 shows that if jobs i and k are allocated on the same machine, job i precedes k or vice-versa. Complimentary, equation 2.13 reinforces that variable w_{ik} only exists for jobs that are allocated on the same machine. Constraint 2.14 guarantees that jobs are not overlapping on the planning horizon. 2.15 defines the makespan $Cmax$. Last two equations (2.16 and 2.17) specify the type of each decision variable (positive and binary).

In order to consider the machine speed, is is required to replace p_i in constraint 2.14 by $z_{ij}p_i/s_j$ (where s_j is machine j speed) considering this constraint for all machines. Objective function, the minimization of makespan, would also have to be modified to adapt to the other parameters of the model. Thus, equation 2.10 would become $min\ max\{\sum_{i=1}^{n} x_i + z_{ij}p_i/s_j | j = 1, ..., n\}$.

2.3 Flow Shop

A flow shop is a processing system in which the processing sequence of each job is fully specified, and all jobs visit the workstations in the same order (processing route), (EMMONS; VAIRAKTARAKIS, 2012). The target is to optimize one or more objectives. Resources (workstations) and jobs can take different forms. They can be machines in a factory, products in an assembly line, baggage conveyor belts in airports, CPU in computational operating systems, etc.

There many variants of the classic problem, most relevant are:

- **permutation flow shop:** in this particular case, all jobs are processed in "first in, first served" strategy. Thus, if job j_1 is the first to be processed in machine m_1, j_1 will also be the first in m_2 and so on.

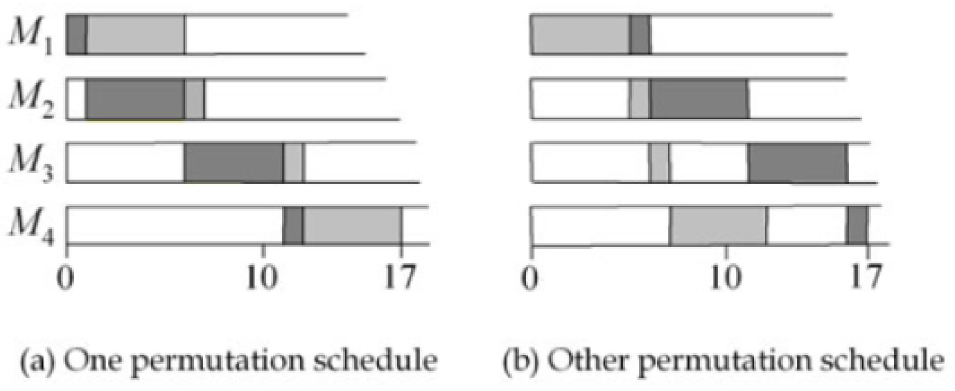


Figure 14 – Example of two permutation schedules.

Adapted from (EMMONS; VAIRAKTARAKIS, 2012)

- **flexible flow shop:** is a generalization of the flow shop and parallel machine cases. There are stages in series (processing flow) executed by parallel machines in each stage. This is also called a hybrid flow shop or multi-processor flow shop (PINEDO, 2016).

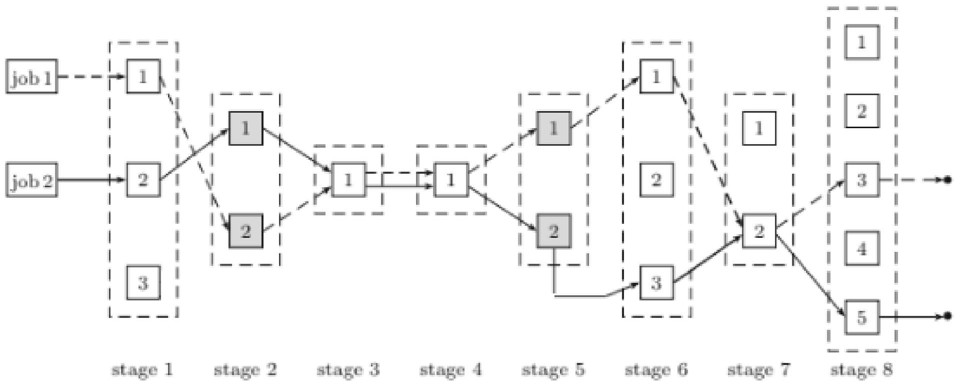


Figure 15 – Example of a flexible flow shop.

Adapted from (SCHULZE J. RIECK, 2016)

The following integer linear programming model depicts a classic flow-shop problem.

Considering n jobs and m machines. Each job i has m O_{ij} operations with p_{ij} processing times. Operation O_{ij} has to be processed in machine M_j and precedencies constraints $O_{ij} \rightarrow O_{i,j+1}$ have to be satisfied for all job i. The objective of the problem is to find the job processing order for each machine j.

Decision variables in a flow shop optimization model that minimizes the makespan are:

$$
\begin{aligned}
x_{ij} &= \text{starting time for job } i \text{ in machine } j. \\
w_{ikj} &= \begin{cases} 1 & \text{if job } i \text{ precedes job } k \text{ in machine } j. \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
$$

And the integer linear programming model with objective function (minimize makespan $Cmax$) and constraints, considering $i = 1, ..., n$ jobs, $j = 1, ..., m$ machines, p_{ij} the processing time of job i in machine j:

$$
\begin{aligned}
\text{minimize} \quad & Cmax && (2.18) \\
\text{subjected to} \quad & Cmax \geqslant x_{ij} + p_{ij} \quad \forall i, \forall j && (2.19) \\
& x_{ij} + p_{ij} \leqslant x_{i(j+1)} \quad \forall i, \quad \forall j = 1, ..., m-1 && (2.20) \\
& x_{ij} + p_{ij} \leqslant x_{kj} + M(1 - w_{ikj}) \quad \forall i, \forall k, \quad i \neq k && (2.21) \\
& x_{kj} + p_{kj} \leqslant x_{ij} + M w_{ikj} \quad \forall i, \forall k, \quad i \neq k && (2.22) \\
& x_{ij} \geqslant 0 \qquad \forall i, \forall j && (2.23) \\
& w_{ikj} \in \{0, 1\} \quad \forall i, \forall k, \forall j && (2.24)
\end{aligned}
$$

Objective function 2.18 minimizes the makespan $Cmax$. Constraint 2.19 ensures that makespan $Cmax$ is greater than the finishing time of all jobs in all machines, thus greater than the finish of all work. Constraint 2.20 guarantees that, for each job, a $j + 1$-th operation starts after the j-th operation concludes. Constraints 2.21 and 2.22 ensure that if two jobs i and k are processed by the same machine j, job i is processed before job k or vice-versa. Last two equations (2.23 and 2.24) specify the type of each decision variable (positive and binary).

2.4 Precedence constraints

An important feature for the work presented here in this dissertation is the existence of precedencies between jobs. Precedencies compel a specific sequence between jobs to be followed. Therefore, this has to be modeled as an extra constraint equation, since there is no flexibility to avoid or bypass these precedencies.

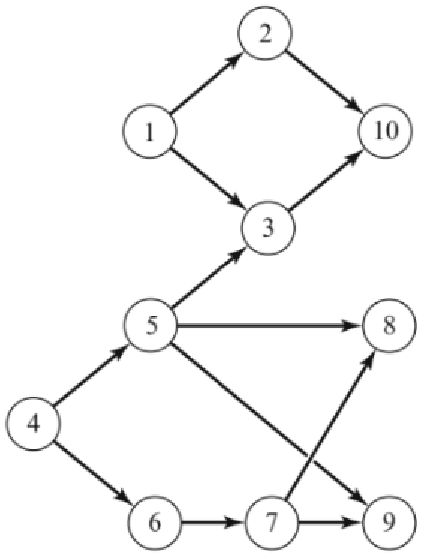


Figure 16 – A graph representation of precedencies between jobs.

In this example, job 6 has to wait job 4 to be finished before it can starts.

Adapted from (PINEDO, 2016)

For the long-term underground mine scheduling problem here studied, these precedencies are represented by the logical sequence (mining method) depicted in section 1.4. In practical terms, this is a 3D file with arrows connecting a point from a task to other (showing the connection and dependency between them) but also a table where each task has an ID and a list of predecessors tasks and other of successors.

3 Solving methods

3.1 Mathematical programming

Computational tools that solve mathematical programming problems have an algebraic modeling language for writing the objective function to be optimized and the constraints considered in the model. After the model implementation, the program is executed by a *solver*. There are several commercial and free packages available to entirely solve linear programming problems. In general, they differ from each other in the methods implemented and types of problems they are able to solve.

3.2 Heuristics

Beside the exact methods, there are alternative methods for solving optimization problems.

Heuristics are techniques that use approximating-strategy algorithms that do not ensure an exact solution, but that return a reasonable one. Therefore, a simple heuristic algorithm could be a rule based on guess ("rules of thumb"), trial and error, the process of elimination, and so on. For a scheduling problem, for example, a heuristic could be prioritizing longer duration tasks or the highest value tasks on more efficient machines.

Therefore, for some problems that are too difficult for integer linear programming or another exact approach, (e.g. scheduling) heuristics can be an interesting approach to turn the decision-making simpler and faster through shortcuts and "good-enough" calculations. In addition, heuristic algorithms are usually easier to model and implement compared to most of the exact algorithms.

(MICHALEWICZ; FOGEL, 2004) mentions some reasons that make problems challenging to solve. Some of them apply to the long-term underground mine scheduling problem here studied:

- size of the search space
- size of input data
- challenges in modeling the problem
- constraints
- algorithm proving

However, when using heuristics, there is, most of the time, a trade-off. The accuracy-effort trade-off mentioned by (JOHNSON, 1985) means that if a less effort algorithm is chosen (a simpler and faster heuristic, for example), it would mean a less accurate result and vice-versa. Consequently, when choosing a heuristic strategy to solve a problem, optimality, solution space coverage, solution accuracy, and execution time should be carefully balanced.

There are many frameworks that classify heuristic strategies. However, since it is an area in constant evolution, it is still an open discussion. A common classification follows:

- **standard heuristics:** are simpler algorithms and straightforward rules focused on improving a single solution;
- **metaheuristics:** consists of a set of combined heuristics that aims to find and improve the best solutions within a solution space;
- **hyper-heuristics:** are search methods that consist of higher-level heuristics that aims to automate the elaboration process of lower-level heuristics (components of the higher-level one) to solve a problem.

Therefore, **standard heuristics** are rules that generate and improve one single solution; **metaheuristics** search within the space of problem solutions; and **hyper-heuristics** search within a space of heuristics.

(GLOVER; KOCHENBERGER, 2003) explain metaheuristics as solution methods that "orchestrate an interaction between local improvement strategies and higher-level strategies to create a process capable of escaping from local optima and performing a robust search of a solution space".

In other words, a metaheuristic could also be understood as a template that defines how multiple low-level heuristic bits (solution generation procedures, local searches, solutions recombinations, etc.) interact in the search for the optimized solution.

A comprehensive framework, (FONTBONA, 2015) and (DRÉO et al., 2006), classifying metaheuristics strategies, is depicted in image 17. This classification, however, will depend also on algorithm implementation.

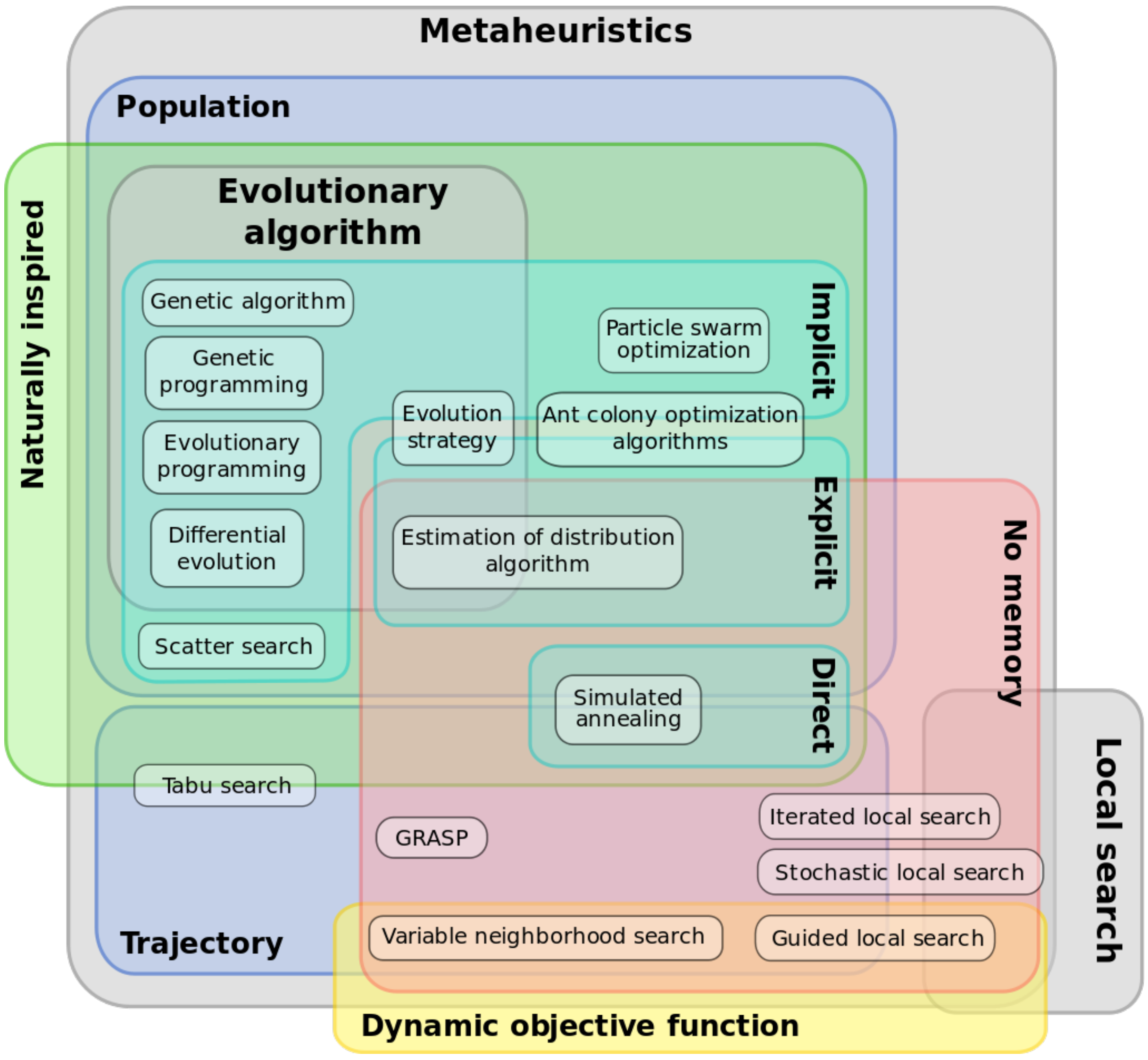


Figure 17 – Classification of metaheuristic methods.

(DRÉO et al., 2006)

In this context, population heuristics is characterized by making use of several current solutions (i.e., populations of solutions) and to combine them together to generate new solutions.

3.3 Genetic Algorithms

As illustrated in figure 17, a genetic algorithm is a metaheuristic on the group of population and evolutionary algorithms, meaning that genetic algorithms make use of the improvement (evolution) of multiple solutions (population) based on biological evolution knowledge.

Evolutionary algorithms are problem-solving computational procedures built on heuristics techniques based on the following fundamental sequence: reproduction with genetic heritage, random variable introductions, competition, and selection among individuals of a given population (MICHALEWICZ, 1996).

This process is repeated a fixed number of times, allowing it to flexibly promote the search for solutions in a huge space of possibilities (COELLO, 2005; KNOWLES; CORNE; DEB, 2008).

Although genetic algorithms are biologically-inspired optimization algorithms, the iterative process that mimics the evolution of species used to optimize solutions can be extremely time-efficient (oppositely to the evolution of species) since it makes use of simple calculations.

This general iterative process is presented in figure 18, (KRAMER, 2017), showing the main operators of a genetic algorithm.

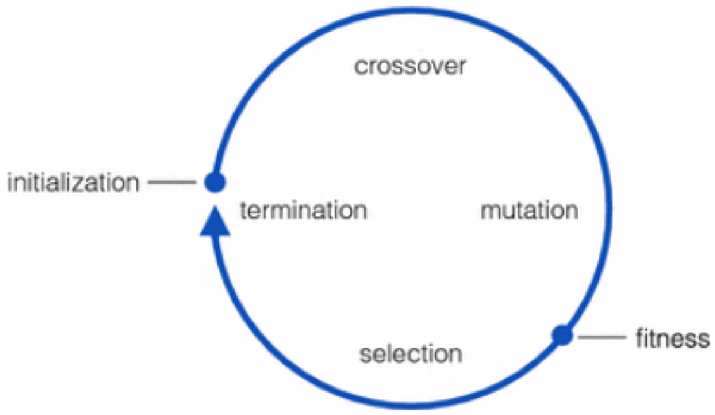


Figure 18 – Genetic algorithm iterative cycle.

(KRAMER, 2017)

Initialization is the first step of a genetic algorithm. This operator creates the first population of solutions to initialize the iterative algorithm cycle. Each individual of a population is a solution that is usually represented by a numerical sequence that has a physically real representation of the problem. These solutions (individuals) have only to be feasible solutions to the problem (no optimization required at this point).

Crossover operator comprises of operations that will promote the combination of the "genetic material" (part of the solution, i.e., pieces of the numerical sequence) of two or more solutions (KRAMER, 2017). Similarly to the biological chromosome crossover, this combination of parents' genes aims to create diverse individuals in a population, thus, exploring the solution space.

Mutation operator changes solutions (individuals) by disturbing them. These random changes imitate the biological mutation with the intention to create even more diverse individuals than crossover, however, not guaranteeing that these changes will only be positive.

There are three main requirements for mutation operators: reachability (creation of the chance that each point in the solution space can be reached), unbiasedness (not inducing a drift of the search to a particular direction) and scalability (a degree of freedom of its strength, i.e., rate, adaption) (KRAMER, 2017).

Fitness computation of the solution is evaluating each individual of a population on a fitness function (analog to an objective function). Thus, each individual is translated to a fitness value that can be used to compare them according to the optimization objective of the problem.

Selection of parental population, is the process that will sort and select the best fitness value individuals (highest for maximization and lowest for minimization). Selected individuals will form the next generation population, which is usually a mix of selected parent individuals with crossover and mutated individuals (offspring).

There are different types of selection, such as comma selection (best solutions from offspring solutions), plus selection (best solutions from offspring and parents solutions), etc. (KRAMER, 2017).

Another usual strategy to improve solutions is selecting only the best-fitness solutions (elitist selection). However, this strategy can also rapidly lead to local optima. In addition, strategies such as forgetting some of the best solutions may help to overcome local optima.

Termination condition is the evaluation of the stop point of the iterative cycle. If this condition is satisfied, that means the the genetic algorithm found a good-enough solution, and the process can be stopped. Typical termination conditions are related to convergence (fitness value difference among population individuals) or the runtime of the genetic algorithm.

Constraints can be considered in initialization and crossover: in this case, these two operators can only generate feasible solutions. The other option is to deal with them in the selection step making use of penalties for infeasible solutions ("death penalty").

Initialization, crossover, mutation and selection operators are governed by global parameters that define global properties such as population size, mutation rate, and selection pressure. These global parameters can be tuned to hone the genetic algorithm aiming to find better solutions in a shorter time. An algorithm that allows tuning during the process (adaptive algorithm) usually performs better than the opposite.

Ultimately, genetic algorithms quality analysis that supports this tuning will focus on the two properties: runtime and convergence. The performance of a genetic algorithm in solving a problem can be measured in terms of the number of required fitness function evaluations until the optimum (or approximated with the desired accuracy) is found.

Part II

Project development

4 Problem definition and literature review

4.1 Problem definition

Summarizing the theoretical framework, it can be said that in order to evaluate and decide over strategic mining possibilities, schedule scenarios are developed with different assumptions, so as to do life-of-mine trade-offs.

Although a guideline exists, there is no standard algorithm to optimize strategic planning cycle, and usually, it works by evaluating one lever (value driver) at time, based on the assumption that everything besides it is fixed.

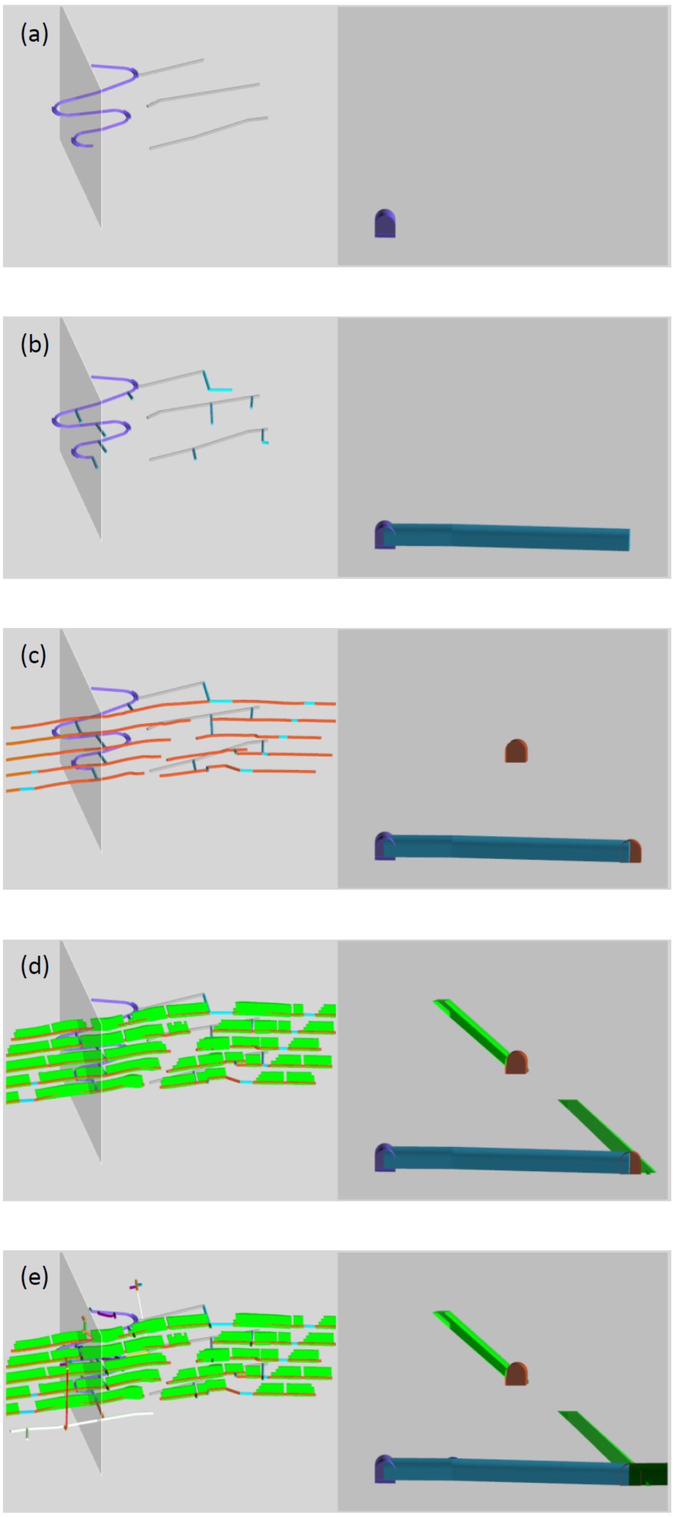
Summarizing, a common workflow to generate one of the strategic planning schedule scenarios, is shown in figure 19:

$$\begin{tikzcd}
\begin{array}{c}\text{COG}\\\text{definition}\end{array} \arrow[r] & \begin{array}{c}\text{Mining blocks}\\\text{design}\end{array} \arrow[r] & \begin{array}{c}\text{Logical}\\\text{sequence}\end{array} \arrow[r] & \begin{array}{c}\text{Operational}\\\text{sequence}\end{array} \arrow[r] & \begin{array}{c}\text{NPV}\\\text{assessment}\end{array}
\end{tikzcd}$$

Figure 19 – Typical strategic planning workflow

Schedule scenarios can be effective to evaluate scale of operation, mining methods and processing routes. However, cut-off grades and mining sequence result in multiple options. Scenarios covering all possibilities for these two levers would not be practical.

Figure 20 shows, step-by-step, each production cycle for the Sublevel Open Stope method herein considered. Right side of the figure is a vertical section at the plane shown in gray on left side of the picture. Each step is shown herein separately, but in practice, they happen most of the times in parallel: some levels in decline development, others in crosscut development, others in oredrive development, and others in stope production.



Declines development

Declines (purple and gray solids) are excavated (developed) from surface or other existing decline. They are positioned far from the orebody as they will be permanent access to the multiple sublevels.

Crosscuts development

Perpendicular to the decline, crosscut tunnels (blue solids) are developed. They connect declines to the orebody.

Oredrives development

Perpendicular to the crosscuts, oredrive tunnels (brown solids) are developed. They are excavated along the ore length, exposing it for production drilling, blasting, and hauling.

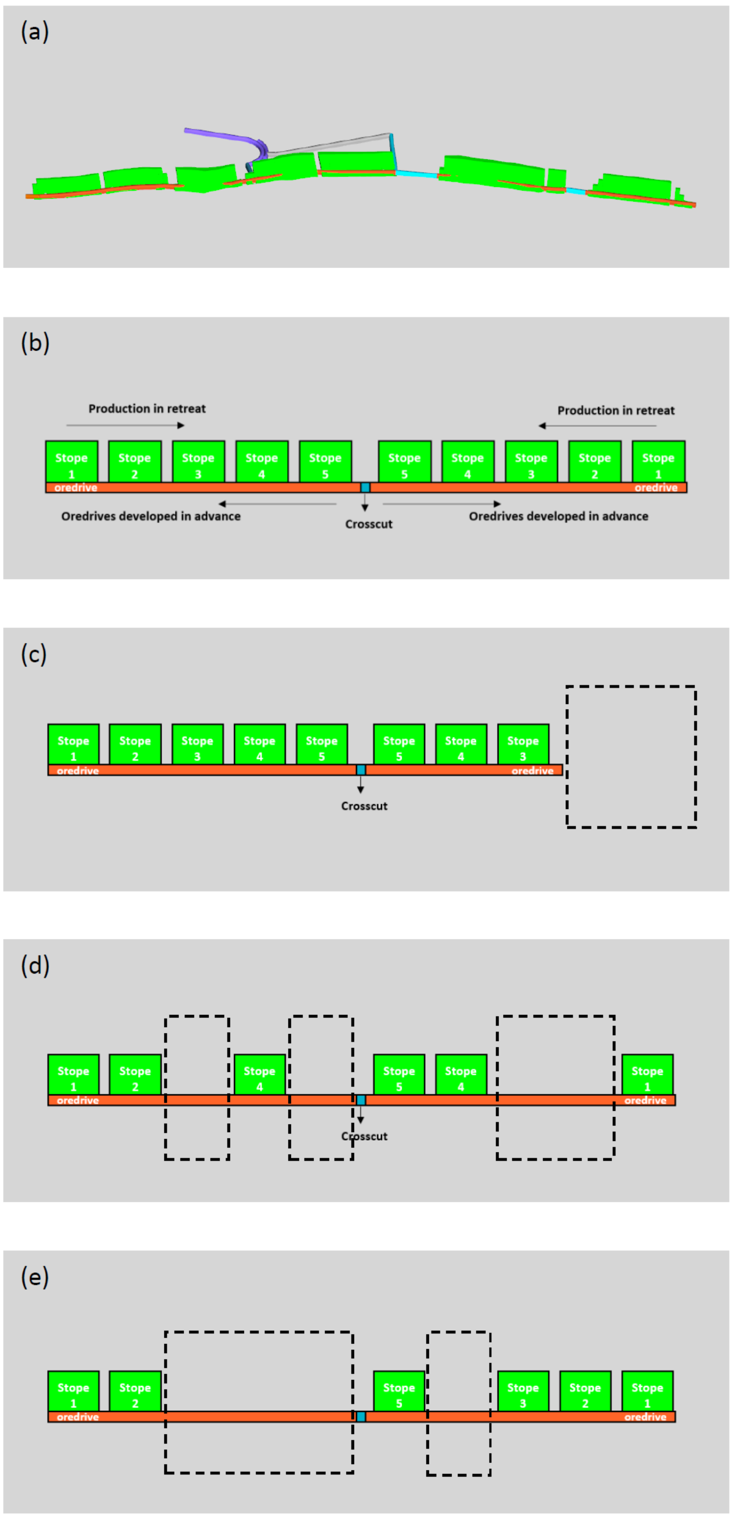
Stope production

Production happens in retreat through cycles of drilling and blasting using long holes in sub-vertical direction. These sub-vertical ore blocks (green) are called stopes.

Auxiliary development
These tunnels are required to place structures that will support development and production operations such as infrastructure (electric, pumping, etc.), and safety ones (refugee chambers and emergency routes).

Figure 20 – Examples of a Sublevel Open Stope mining sequence.

Besides selecting which blocks are considered, each activity (development and production) could happen at different times according to equipment available (schedule leveling) and rates applied (schedule priorities). A schematic design of figure 20 is depicted in figure 21 explaining how numerous schedules are possible.



Detail of one production sublevel showing its decline (purple and gray), crosscuts (blue), oredrive (brown) and stopes (green).

Vertical section of production level above showing the mining sequence: after the **crosscut** reach the orebody, **oredrives** are developed in advance and after they finish, **production** starts in retreat.

Since designs are made only considering the "activity costs", a sequence leaving some stopes *in situ* could reduce development and could be more profitable than mining the whole sublevel.

Other feasible solutions would be to develop the whole oredrives but without mining all the stopes available.
That could anticipate high-value stopes consistently and lead to better NPV.

In fact, in this example with 10 stopes, there are numerous stope selection combinations. When considered that the sequence generates different solutions from the same set of stopes, we have even more possible solutions

Figure 21 – Schematic analysis of block selection (cut-off decision) possibilities.

Being more specific, figure 22 shows an example of one out of nine orebodies from a mine in production. This particular orebody contributes to about 35% of annual production. Three panels were selected (colored portion of figure 22(a)). They have about 2000 tasks (development and production) and 9 sublevels. From that, 40 tasks from one sublevel were taken, circled in figure 22 (b). Three options are showed: (c), mine all designed area; (d) leave the farthest portion (left block) after the waste connection; and (e), leave part of the left block. Therefore, this ends as a combinatorial problem with multiple options only regarding the selection of blocks that will be scheduled (cut-off lever).

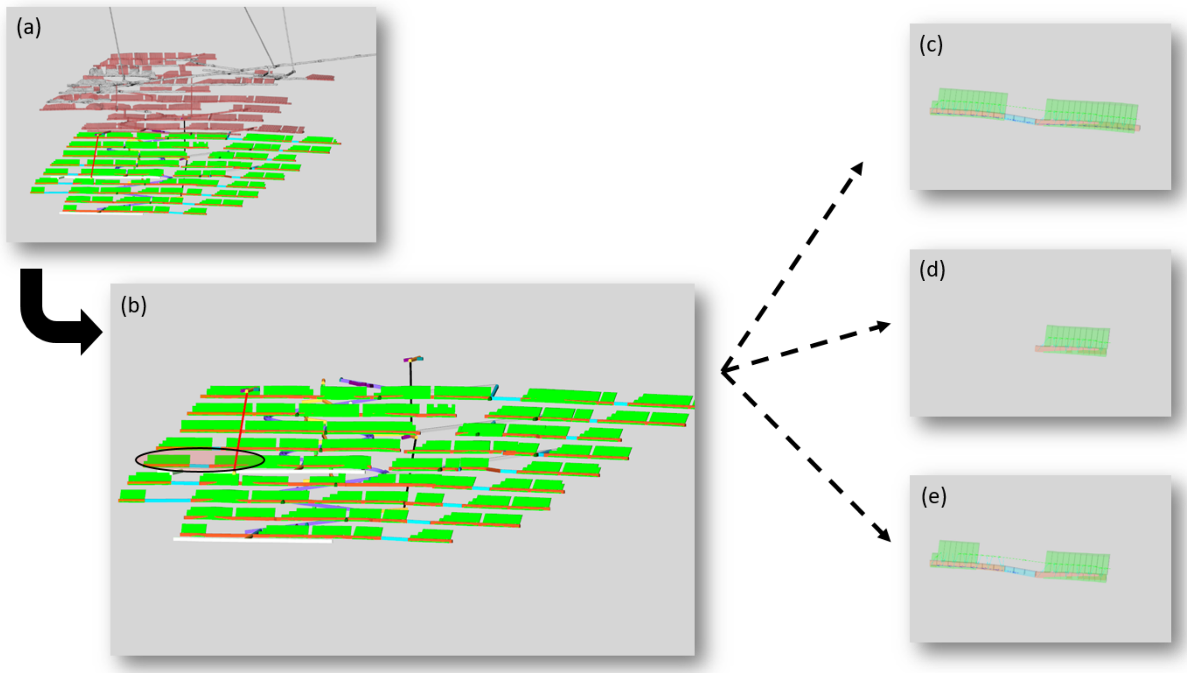


Figure 22 – Examples of block selection (cut-off) possibilities of a small area.

This dissertation aims to the development of an algorithm that, with given mine design and logical mining sequence, could optimize both cut-off selection and schedule of the selected mining blocks, solving the herein called "underground mine scheduling problem". In other words:

- optimize NPV;
- determine the best operational (leveled) sequence respecting given production constraints;
- decide cut-off grades and thus decide whether mining tasks will be excavated as ore, waste or be left *in situ*;
- best allocate a given set of development and production rates to tasks;

- consider fixed costs as a fixed parcel on the top of marginal (unit) costs, in a similar approach as (KING, 2004), instead of considering them proportional to ore tonnes (as unit costs);
- automate and standardize these steps of the workflow, reducing time to execute them.

From the operations research perspective, the underground mine scheduling problem will be modeled as a parallel machine problem.

In a given design, each solid will be a development activity (tunnel) or a stoping activity (production) linked with precedencies that represent possible logical sequences of the mining method. Therefore, there are two sets of unrelated parallel machines (machine whose rates depends on the activity being processed) representing the number of simultaneous development and production (stoping) activities that can happen in that mine.

Although all machines could process all tasks, a null allocation value is forced for production tasks in machines that should only be considered for development and vice-versa, using that as a workaround to guide the solver to allocate development machines to development tasks and mining machines to production tasks.

To better model this problem, precedencies will be divided in two types. The first type will be the **hard precedencies** that represents precedencies needed to access a given task. In other words, as a development tunnel is needed to access a production stope physically, this will be a hard precedence.

The second type will be the **soft precedencies**, which will represent precedencies that could be ignored (tasks that could be abandoned) and still result in a feasible schedule. An example could be a production stope in a retreat sequence: the first stope could be abandoned, and production started from the second stope and on. These are depicted in figure 23.

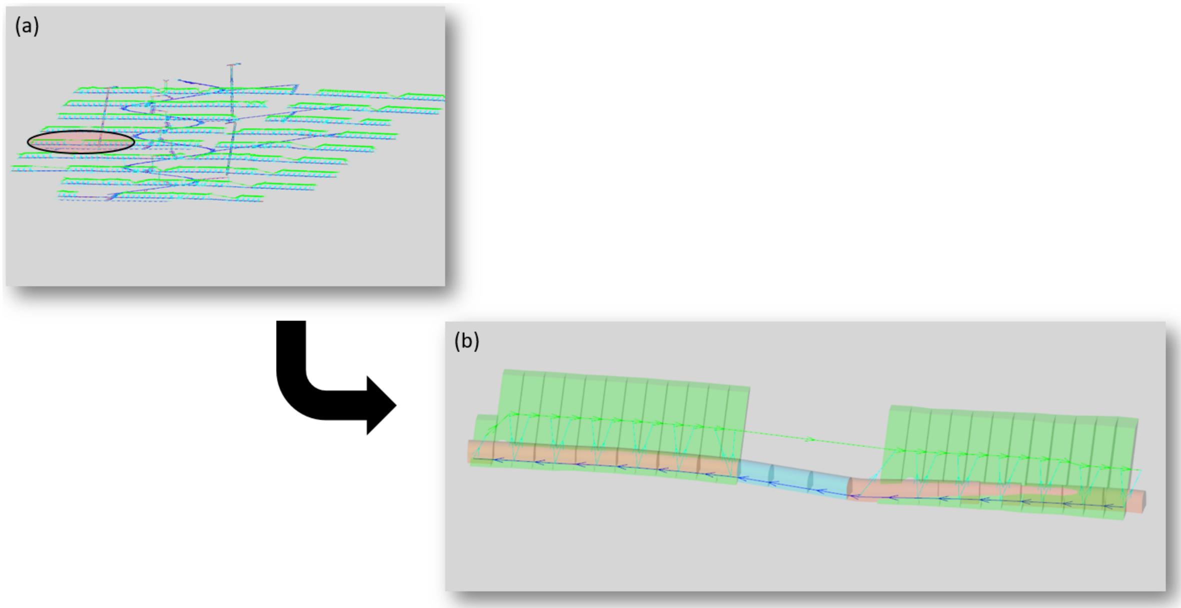


Figure 23 – Examples of precedencies that will be considered in the algorithm.

Access (development tunnels) will be hard precedencies (dark and light blue arrows) to other activities (development or production). Tasks that can be left while still generating feasible schedules are soft precedencies (green arrows between stopes).

Figure 23 depicts the same area shown in figure 22 but highlights the arrows that represents the precedence relationships between tasks. Dark blue arrows on figure 23 (*b*) show the mandatory connection between a development task and others. Light blue shows the mandatory connection between a development task and the relative stope. Green arrows show the soft precedence between stopes (which are not access to each other, thus, not imperative to a feasible schedule).

The aim of this precedencies is to model possible options behind cut-off decisions and mining blocks selections. In other words, if a task's successor is already done, there are only two options: the task was done before the successor, or the task was abandoned.

Constraint dates will also be considered as they appear in mining environments representing dependency on infrastructure or any other restriction to access a given area on a specific date.

Alternatively, the underground mine scheduling problem could also be represented by a flexible flow shop scheduling with precedencies. In this case, there would be a flow with two stages (development and production) that could be performed by uniform parallel machines (different rates for development and production). Development activities would have 0 duration time in the production stage and vice-versa. The parallel machine model is deemed easier to apply while also representing the problem as good as a flexible flow shop model.

Genetic algorithm is chosen as an alternative solver technique because of its successful track record in solving complex optimization problems, while also balancing computational effort (processing time), and coding easiness, (KRAMER, 2017).

The case study presented use a sublevel open stopping design from a real gold mine that has only one processing route; thus market demand will not be one of the limiting factors and COG/sequence might only balance the other production capabilities (development, mining, and processing).

All production tasks (stopes) considered have grades above the marginal cut-off, i.e., generates revenue higher than the minimum variable cost to mine them. This is done since negative marginal stopes would never increase NPV (objective function). Thus, cut-off decisions will be whether to keep or left a block (never to add a block).

4.2 Literature review

Some different approaches have been applied to the underground mine scheduling problem. Most common approach on mine sites is to do a cash flow analysis in each panel or sublevel aiming to consider only full grade ore in plans (i.e., "tonnes that fully pay for themselves") and then schedule with levelling tools (i.e., greedy algorithms) focused on grade or value without optimizing the whole life-of-mine value.

Other more advanced propositions treat the problems separately, still not considering different rates, time value of money, capabilities balancing, and mining blocks selection (the possibility to abandon blocks) at the same time.

On the one hand, (BENNETT, 2018) uses a pseudo flow algorithm (usually applied for open-pit analysis) to choose underground areas that are economically feasible. (ANDRADE, 2018) presents and application of the generalized maximum-weight connected subgraph problem based on (LOBODA MAXIM N. ARTYOMOV, 2016) that optimize the cash value of the connected tree of mining activities. Both are examples of algorithms that select the best blocks (optimizing cash flow) but still not considering the time value or production capabilities balancing of scheduling.

On the other hand, scheduling tools, such as (SOT, 2019), optimize the NPV of an input schedule considering multiple constraints but yet not making decisions regarding cut-off grades (selection of mining blocks).

(SCHULZE J. RIECK, 2016) develop a specific scheduling algorithm for a phosphate mine while mentioning some other mine scheduling algorithms that cope with scheduling problems in different ways, but yet not considering cut-off decisions.

(OTA, 2017) propose a direct block scheduling methodology that does consider both cut-off and scheduling; however, for open-pit problems only.

Therefore, to the extent of the author's knowledge, the problem presented herein in this dissertation has not been fully studied and solved. Thus, no solution algorithm exists yet, being that the main motivation for this work.

4.3 Mathematical modeling

Three mathematical models for this problem are proposed herein. The purpose is to analyze the complexity of each one and understand how the differences between them affect the problem solution.

The three proposed models are inspired by the classic scheduling models presented in chapter 2, but also consider the particularities of the underground mine scheduling problem.

4.3.1 Model 1: makespan minimization

Makespan plays an important role as an upper-bound limit of variables and $big-M$ constraints and also in NPV optimization (generally related to makespan reduction given the same amount of tasks). With the aim of using this model result as an input of other models, the first approach is to optimize makespan, assuming that all tasks will be performed. Therefore, the objective function of this model is to minimize the makespan.

So, n tasks that can be processed on m uniform parallel machines. Tasks have a precedence order. For each task i, $PredSoft_i$ and $PredHard_i$ are the sets of activities that precede i optionally or mandatorily, respectively. In addition, p_{ij} represents the processing time of project i on machine j.

The decision variables of the problem are:

$$\begin{aligned}
Cmax &= \text{makespan.} \\
x_i &= \text{task } i \text{ processing start time.} \\
y_{ij} &= \begin{cases} 1 & \text{if activity } i \text{ is allocated at machine } j \\ 0 & \text{otherwise.} \end{cases} \\
w_{ik} &= \begin{cases} 1 & \text{if activity } i \text{ precedes activity } k \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}$$

Considering $i = 1, ..., n$, $j = 1, ..., m$, A_i the set of allowed machines for activity

i, and p_{ij} the processing time of activity i in machine j; we have the model:

$$\text{minimize} \quad z = Cmax \tag{4.1}$$

$$\text{subject to:} \quad \sum_{j \in A_i} y_{ij} = 1 \quad \forall i \tag{4.2}$$

$$\sum_{j \notin A_i} y_{ij} = 0 \quad \forall i \tag{4.3}$$

$$x_i \geqslant x_k + \sum_{j=0}^{m} p_{kj}.y_{kj} \quad \forall k \in \{PredHard_i \cup PredSoft_i\}, \quad \forall i \tag{4.4}$$

$$x_i + \sum_{j=0}^{m} p_{ij}.y_{ij} \leqslant x_k + M.(1 - w_{ik}) \quad \forall i, \forall k, i \neq k \tag{4.5}$$

$$y_{ij} + y_{kj} - w_{ik} - w_{ki} \leqslant 1 \quad \forall i, \forall k, i \neq k, \forall j \tag{4.6}$$

$$y_{ij} + y_{kh} + w_{ik} + w_{ki} \leqslant 2 \quad \forall i, \forall k, i \neq k, \forall j, \forall h, j \neq h \tag{4.7}$$

$$Cmax \geqslant x_i + \sum_{j=0}^{m} p_{ij}.y_{ij} \quad \forall i \tag{4.8}$$

$$x_i \geqslant 0 \quad \forall i \tag{4.9}$$

$$w_{ik} \in (0,1) \quad \forall i, \forall k \tag{4.10}$$

$$y_{ij} \in (0,1) \quad \forall i, \forall j \tag{4.11}$$

Constraints 4.2 and 4.3 ensure that each task will be allocated in only one allowed machine. Constraint 4.4 guarantees that each task i can only be started after its predecessors k (in this case, both , hard and soft, precedence types are considered mandatory). Constraint 4.5 shows that if jobs i and k are allocated on the same machine, job i precedes k or vice-versa. Constraint 4.6 guarantees that jobs are not overlapping on the planning horizon. Equation 4.7 reinforce that variable w_{ik} only exists for jobs that are allocated on the same machine. The last equations (4.9 4.10 and 4.11) specify the type of each decision variable (positive and binaries).

4.3.2 Model 2: time-indexed model

In this model, there is an integer programming formulation with time-indexed variables (x_{ijt}). The l parameter can be defined as the makespan obtained in the previous model, $l = Cmax$ (in which all activities are performed), or through some constructive heuristic that yields a workable solution to a time limit. In this second model, this l parameter is an upper-bound limit to variables and makespan itself.

Objective function models the project net present value (NPV) as discussed at the end of section 1.3: a (negative or positive) profit value calculated from activity costs ($revenue - costs$) is attached to each task as an input to the problem. These values are discounted by a discounting rate based (another input parameter) on each activity start time (time index, t). On the top of the sum of all discounted profit values, a fixed parcel based on the total time required to process the whole project (i.e., the makespan) is subtracted. As foundation and reserve timing costs are kept constants, they were not included in the model. This way, the time value of money is considered, i.e., revenue available at the present time is worth more than the identical sum in the future due to its potential earning capacity.

In this model, not all activities need to be performed, thus, options behind cut-off grade decisions mentioned in 1.3 are considered.

Decision variables of the problem are:

$$
\begin{aligned}
Cmax &= \text{makespan.} \\
x_{ijt} &= \begin{cases} 1 & \text{if activity } i \text{ starts on machine } j \text{ and time } t \\ 0 & \text{otherwise.} \end{cases} \\
w_i &= \begin{cases} 1 & \text{if the } i \text{ activity is not performed} \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
$$

The problem is modeled as follows, with $i = 1, ..., n$, $j = 1, .., m$, A_i the set of allowed machines for activity i, p_{ij} the processing time of activity i in machine j:

$$\text{maximize} \quad z = \sum_{t=0}^{l}\sum_{j=0}^{m}\sum_{i=0}^{n} \frac{(profit_i \;.\; x_{ijt})}{(1+rate)^t} - (period\ costs)\;.\;Cmax \tag{4.12}$$

$$\text{subject to:} \quad w_i + \sum_{t=0}^{l}\sum_{j=0}^{m} x_{ijt} = 1 \quad \forall i \tag{4.13}$$

$$\sum_{i=0}^{n} \sum_{s=max(t-p_{ij}+1,0)}^{t} x_{ijs} \leqslant 1 \quad \forall t, \forall j \tag{4.14}$$

$$QPredHard_i. \sum_{j=0}^{m}\sum_{t=0}^{t} x_{ijt} \leqslant \sum_{j=0}^{m} \sum_{k \in PredHard_i} \sum_{s=0}^{l} x_{kjs} \quad \forall i \tag{4.15}$$

$$\sum_{j=0}^{m}\sum_{t=0}^{l} t.x_{ijt} + lw_i \geqslant \sum_{j=0}^{m}\sum_{s=0}^{l} (s+p_{kj}).x_{kjs} \;\forall k \in PredHard_i, \;\forall i \tag{4.16}$$

$$\sum_{j=0}^{m}\sum_{t=0}^{l} t.x_{ijt} + lw_i \geqslant \sum_{j=0}^{m}\sum_{s=0}^{l} (s+p_{kj}).x_{kjs} \;\forall k \in PredSoft_i, \;\forall i \tag{4.17}$$

$$Cmax \geqslant \sum_{t=0}^{l}\sum_{j=0}^{m} t.x_{ijt} + \sum_{t=0}^{l}\sum_{j=0}^{m} p_{ij}.x_{ijt} \quad \forall i \tag{4.18}$$

$$\sum_{j \notin A_i}^{m}\sum_{t=0}^{l} x_{ijt} = 0 \quad \forall i \tag{4.19}$$

$$x_{ijt} \in (0,1) \;\forall i, \forall j, \forall t \tag{4.20}$$

$$w_i \in (0,1) \;\forall i \tag{4.21}$$

Constraint 4.13 ensures that all activity i starts on a single machine and in a single period t (or does not start at all). Constraint 4.14 does not allow more than one activity to be performed on machine j at the same time t.

Constraint 4.15 ensures that activity i will only be executed if all required predecessors $k \in PredHard_i$ are also executed ($QPredHard_i$ is the amount of required predecessors). On the other hand, it is possible that some predecessors of the activity i will be executed, but the activity i will not be. Constraint 4.16 complements constraint 4.15 to ensure that activity i starts only after its required predecessors. In this one, the upper-limit l value has to be calculated whether summing up all the processing times or applying an heuristic.

Constraint 4.17 ensures that activity i can only be started after its optional predecessors k: (activity i start time) $\geqslant$ (end time of any activity $k \in PredSoft_i$). However, it is possible that the i activity will perform even if its optional predecessors are not; in this case the constraint is: (activity i start time) $\geqslant 0$. Also, it is not possible for any $k \in PredSoft_i$ predecessor to be executed after the i activity has been executed.

Constraint 4.18 defines the makespan, $Cmax$. Constraint 4.19 eliminates the possibility of i activity being allocated on any machine that does not belong to the A_i set (set of allowed machines for activity i).

Equations (4.20 and 4.21) define the type of each decision variable (binaries).

If on one hand the time-indexed model was the best way to correctly model the objective function (NPV), on the other model size is the main disadvantage of such formulation.

4.3.3 Model 3: undiscounted objective function

The third proposition does not use time-indexed variables aiming to reduce the complexity of the formulation, thus achieving reduced runtime. However, the objective function is not the NPV, as shown in model 2, section 4.3.2. An approximation is required: undiscounted cash flow is used. The idea is to compare this model to model 2 in terms of NPV and runtime to evaluate if solutions of model 3 are "good enough" and close to the solutions of model 2.

The decision variables are the same as in the model 1, section 4.3.1, but in this model, not all activities need to be performed. Constraints resemble the ones seen in model 2, and the objective function is based on undiscounted cash flows (the time value of money is not considered in this case).

$$\text{maximize} \quad z = \sum_{i=0}^{n}\sum_{j=0}^{m}(profit_i \, . \, y_{ij}) - (period\ costs) \, . \, Cmax \tag{4.22}$$

$$\text{subject to:} \quad \sum_{j \in A_i} y_{ij} \leqslant 1 \quad \forall i \tag{4.23}$$

$$\sum_{j \notin A_i} y_{ij} = 0 \quad \forall i \tag{4.24}$$

$$x_i \leqslant l.\sum_{j=0}^{m} y_{ij} \quad \forall i \tag{4.25}$$

$$QPredHard_i.\sum_{j=0}^{m} y_{ij} \leqslant \sum_{j=0}^{m}\sum_{k \in PredHard_i} y_{kj} \quad \forall i \tag{4.26}$$

$$x_i \geqslant x_k + \sum_{j=0}^{m} p_{kj}.y_{kj} \quad \forall k \in \{PredHard_i \cup PredSoft_i\}, \forall i \tag{4.27}$$

$$x_i + \sum_{j=0}^{m} p_{ij}.y_{ij} \leqslant x_k + l.(1 - w_{ik}) \quad \forall i, \forall k, i \neq k \tag{4.28}$$

$$y_{ij} + y_{kj} - w_{ik} - w_{ki} \leqslant 1 \quad \forall i, \forall k, i \neq k, \forall j \tag{4.29}$$

$$y_{ij} + y_{kh} + w_{ik} + w_{ki} \leqslant 2 \quad \forall i, \forall k, i \neq k, \forall j, \forall h, j \neq h \tag{4.30}$$

$$Cmax \geqslant x_i + \sum_{j=0}^{m} p_{ij}.y_{ij} \quad \forall i \tag{4.31}$$

$$x_i \geqslant 0 \quad \forall i \tag{4.32}$$

$$w_{ik} \in (0, 1) \quad \forall i, \forall k, \tag{4.33}$$

$$y_{ij} \in (0, 1) \quad \forall i, \forall j, \tag{4.34}$$

Constraints 4.23 and 4.24 ensure that each job will be allocated in only one allowed machine or will not be allocated. Constraint 4.25 creates the relationship between variables x_i and y_{ij}. 4.26 ensures that activity i will only be executed if all required predecessors $k \in PredHard_i$ are also executed (i.e. there is an amount equal to $QPredHard_i$ required predecessors). Conversely, it is possible that some predecessors of the activity i will be executed, but the activity i will not be.

Constraint 4.27 ensures that activity i can only be started after its optional and required predecessors k. Constraint 4.28 shows that if jobs i and k are allocated on the same machine, job i precedes k or vice-versa. Constraint 4.29 guarantees that jobs are not overlapping on the planning horizon. Equation 4.30 reinforce that variable w_{ik} only exists for jobs that are allocated on the same machine. Constraint 4.31 defines the makespan, $Cmax$. Equations (4.32, 4.33 and 4.34) specify the type of each decision variable.

5 Implementation of the solving methods

5.1 Mathematical programming

The three models proposed in the previous chapter were implemented using mathematical programming problem-solving packages. The models were implemented in Python programming language, importing the *ortools* package. OR-Tools (OR-TOOLS, 2019) is an open source software suite for writing the optimization models. After modeling, commercial solvers, such as CPLEX (IBM, 2019), or open-source solvers, such as CBC (CBC, 2019) and CP-SAT (CP-SAT, 2019), can be used to solve the problem.

5.1.1 CBC

The Coin-or Branch and Cut (CBC, 2019) is an open-source mixed-integer linear programming solver written in C++ included in OR-Tools (OR-TOOLS, 2019). Its algorithm is based on the branch and cut method, which consists of a branch and bound algorithm that uses cutting planes to tighten the linear programming relaxations.

5.1.2 CP-SAT

The CP-SAT is a solver for constraint programming, also included in OR-Tools (OR-TOOLS, 2019). CP-SAT stands for Constraint programming and SAT for Boolean satisfiability.

Constraint programming is a method for solving combinatorial problems; it uses a variety of techniques from artificial intelligence, computer science, and operations research. Constraints differ from other programming languages in that they do not follow a sequence of steps, but rather they define the properties of a solution to be found.

SAT refers to the Boolean satisfiability, which is the problem of determining if there are solutions that satisfy a given Boolean formula. That is, it evaluates whether the variables of a given Boolean formula can be replaced by the values TRUE or FALSE in a way that the formula evaluates to TRUE.

Integer optimization problems can be solved with either a MIP solver or the CP-SAT solver. Some problems have a highly non-convex feasible set: when the feasible set is defined by constraints of type "OR" (OR-TOOLS, 2019). In this case, the CP-SAT solver is often faster than the MIP solver as mentioned by (OR-TOOLS, 2019).

5.1.3 CPLEX

CPLEX solver (IBM, 2019) was named after the simplex method as implemented in the C language. Currently, under the full name of IBM ILOG® CPLEX® Optimization Studio, it operates as a stand-alone suite but also as a Python API where objective function, variables, and constraints can be declared in Python language, and only the solver can be called.

5.2 Genetic Algorithm

The Genetic Algorithm (GA) implementation begins with the **initialization** of a random population of individuals (chromosomes). Next, the individuals are decoded and evaluated, and a fitness value, representing the objective function (NPV as described in Section 4.3.2), is assigned to each of them. This first generation go through **crossover** and **mutation** operators. The resulted offspring will also be decoded and evaluated. A **selection** operator will select the individuals that will form the next generation. Apart from the initialization, this process is repeated by a given number of times, always using the previous generation as the input (parent).

The main loop of the implemented Genetic Algorithm is shown in Algorithm 1. Details on genetic operators are given in the next sections.

Algorithm 1 – Genetic Algorithm outline

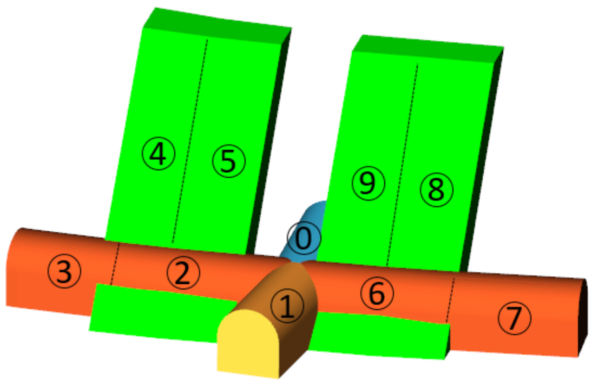
```
random initialization of population → $P_0$ ;
decode and allocation of $P_0$ ;
$g = 0$ ;
while $g <$ (number of generations) do
    crossover in selected individuals of $P_g$ → $Q_g$ ;
    mutation of $Q_g$ ;
    decode and allocation of $Q_g$ ;
    selection among $P_g$ and $Q_g$ → $P_{g+1}$;
    $g = g + 1$ ;
end
output last generation: Individuals, Allocations, and Fitness value (NPV);
local search to improve last generation best results;
output best individual (result) found: Allocation and Fitness value (NPV);
```

5.2.1 Codification

The representation of each individual in a genetic algorithm should allow random processes (such as initialization and mutation) and unconstrained crossover operations, avoiding repair mechanisms (when after each operator we need to manipulate the individuals to make them feasible solutions). Therefore, initialization, crossover, and mutation operators are flexible for tuning and indifferent to the particularities of the underground scheduling problem (hard and soft predecessors implications, for example).

The codification chosen is to represent each individual (also called chromosome) of a population as a priority list with indexes (IDs) of tasks (or activities) that will be allocated. The list order is considered for the prioritization of the tasks. Thus, $a = [4, 3]$ will try to allocate task 4 before task 3 and $b = [3, 4]$ the opposite. Required predecessors will be added during the decoding step respecting the list order. All precedencies' relationship will also be considered during the allocation step. This approach guarantees that an individual does not have to had any particular characteristic, i.e., it does not have to include predecessors nor to be in any specific order.

Figure 24 shows an example of the underground scheduling problem. It is a small instance with 10 tasks (IDs ranging from 0 to 9). In this case, a vector $a = [2, 1]$ or $b = [4, 5, 6, 8, 0, 1, 9]$ are both examples of individuals for this instance. In fact, any set $A = \{x_1, ..., x_n \mid n \leqslant 10, \quad x_i \in [0, 9]\}$ can be an individual.



Task ID	Hard predecessors	Soft predecessors
0		
1	0	
2	0	
3	2	
4	3	
5	2, 3	4
6	0	
7	6	
8	7	
9	6	8

Figure 24 – Underground scheduling problem instance with 10 tasks.

In Figure 24, numbers over each solid (task) represent their IDs. Tunnels, such as 0 and 1, have negative profit (only costs). Oredrives tasks, such as 2, 3, 6, and 7, usually will have positive profit but smaller than stopes (tasks 4, 5, 8, and 9) due to their mining cost. Development tasks are hard predecessors, and stope tasks are soft predecessors.

5.2.2 Decodification and Evaluation

The decoding step translates the individual representation to the allocation of activities to machines (a schedule). Then, based on this allocation, fitness function (NPV) can be calculated, as shown in Model 2, Section 4.3.2. This decode can be applied to any group of individuals (population), which can be the results of initialization, crossover, or mutation operators. The general algorithm is shown in Algorithm 2.

Algorithm 2 – Decodification operator

```
for each individual do
    while task list keeps changing do
        add hard predecessors of tasks in task list;
    end
    create available tasks list → tasks list that do not have predecessors at time 0;
    while available tasks list is not empty do
        get task duration for all machines;
        get future times for all possible machines;
        allocate first available task to the machine with minimum future time;
        calculate task starting time;
        remove processed task from available tasks list;
        update available tasks list with tasks that became available;
    end
    calculate NPV (fitness) of the resultant allocation;
    output allocation results:
        tasks start and finish times;
        tasks duration;
        allocated machine (ID) for each task;
        makespan for the allocation;
        NPV for the allocation;
end
```

First, hard predecessors of tasks on the individual (priority list) are added. As only direct predecessors are listed as input, after the first round, tasks added (predecessors) will also have their own predecessors and so on. Thus, this will be a loop until no extra task (predecessor) is added to the list. This inclusion of hard predecessors maintains the priority of the individual: predecessors of higher priority tasks are added with higher priorities.

Secondly, the algorithm goes through the list of tasks (in priority order) looking for the ones that do not have predecessors. These are possible starting points for the allocation.

The task is allocated to the machine with the minimum future time (i.e., the finish time for processing a given task on that machine). Task starting time will be the maximum between the time which the machine is available and the cumulative finish time of the task's predecessors.

After each allocation, processed tasks are removed from the available list, and tasks that would require the allocated activity as predecessor are now considered available and added to the available tasks list.

This loop follows, allocating tasks one-by-one until there are no tasks to be allocated. After that, outputs (including makespan and NPV) are calculated.

5.2.3 Initialization

The first generation in this genetic algorithm is formed by randomly generated individuals. They are arbitrary in a number of tasks and order of included tasks.

These individuals' fitness may be far from the optimized solution, but they represent feasible solutions and set up a population with good solution space coverage (since they are randomly selected).

5.2.4 Selection

The selection is carried out in two steps of the algorithm (Algorithm 1: (i) selection of individuals from P_g to perform the crossover and (ii) selection of individuals from $P_g \cup Q_g$ to update the new population.

For the selection of the individuals from P_g, two different operators (BÄCK; FOGEL; MICHALEWICZ, 2000a; BÄCK; FOGEL; MICHALEWICZ, 2000b) are proposed. The first one is known as the Roulette Wheel and the second, as Binary Tournament. Both prioritize fittest parents, but in different ways.

In Roulette Wheel (Algorithm 3), parents are selected randomly but based on a uniform distribution probability depending on their fitness, i.e., the fittest individual has the largest share of the wheel. The relative fitness is normalized with the sum of all fitness values in a population. This fraction of fitness can be understood as the probability for an individual to be selected as a parent, similarly to a roulette wheel (KRAMER, 2017).

Algorithm 3 – Roulette Wheel selection operator

```
normalize parents' fitness values → range between 0 and 1 (percentage);
sum these values cumulatively (from individual 1 to n) → wheel value;
for (half of population) number of times do
    pick a random value between 0 and 1 → match random value and wheel range to select parent 1;
    pick a random value between 0 and 1 → match random value and wheel range to select parent 2;
    crossover parent 1 and 2 → generate offspring 1 and 2;
    store offsprings in $Q_g$ ;
end
output all generated offspring individuals
```

In Binary Tournament (Algorithm 4), 4 parents are selected randomly. They compete in pairs (i.e., two tournaments), and the best one of each tournament will be crossed over to the other.

Algorithm 4 – Binary Tournament selection operator

```
for (half of population) number of times do
    randomly select 4 parents: parent 1-1, parent 1-2, parent 2-1 and parent 2-2;
    select the fittest between parent 1-1 and parent 1-2 → parent 1;
    select the fittest between parent 2-1 and parent 2-2 → parent 2;
    crossover parent 1 and 2 → generate offspring 1 and 2;
    store offsprings in $Q_g$ ;
end
output all generated offspring individuals
```

Comparisons showed that, for the number of individuals per population and number of generations considered, Binary Tournament was better in keeping population variability while Roulette Wheel was deemed to put excessive selective pressure.

For the selection of the individuals from $P_g \cup Q_g$, as outlined in algorithm 1 (Section 5.2), the input of this selection operator will be a decoded population, (after previous generation crossover and mutation). This means that the input population will have the double of individuals of a standard population (parents: previous generation individuals; and offsprings: new individuals). Therefore, this selection operator will choose the best half of individuals. In this proposed genetic algorithm, to avoid excessive selective pressure, only part of the selected individuals are chosen due to their high fitness value (NPV). The others are chosen randomly (Algorithm 5).

Algorithm 5 – Selection operator

```
sort input population by fitness value (NPV);
keep first (selective pressure) individuals;
randomly select individuals until new generation (population) is complete;
```

5.2.5 Crossover

Crossover generates new individuals through the recombination of characteristics of two or more individuals (genetic inheritance). The underlying principle is to keep traits (in our case, COG selection and scheduling traits) of best individuals but not doing excessive selective pressure that would reduce solution space representation.

As explained in Section 5.2.4, Roulette Wheel (Algorithm 3) or Binary Tournament (Algorithm 4), select pairs of parents and mix them using a middle point crossover: parents are split in half (with regards of number of tasks in list) and *parent 1* first part is combined to *parent 2* second, generating *offspring 1*; and *parent 2* first part is combined to *parent 1* second, generating *offspring 2*. In both cases, crossover occurs the number of times of half of a population. Since each crossover round generates two offsprings, the offspring population have the same amount of individuals than their parent population.

5.2.6 Mutation

In this algorithm, the mutation operator randomly adds or removes an arbitrary percentage of tasks from an input priority list (an individual). Two key parameters control mutation: the percentage of a population (number of individuals) that will be mutated and the percentage of an individual (number of tasks) that will be mutated (added or removed).

For each mutation round, a random integer between 1 and 10 is picked and, if it is greater than 5, an addition will happen. Otherwise, a removal will occur. For the first case, random tasks that are not already in the list are inserted in a random position of the list. For the second case, some tasks of the list are randomly removed.

Algorithm 6 – Mutation operator

```
for (mutation rate) times do
    randomly select an individual from the population;
    a = random integer in [1,10];
    if $a > 5$ then
        randomly select (mutation element rate) tasks not presented in individual;
        insert tasks on a random position of the individual;
    else
        randomly remove (mutation element rate) tasks from the individual;
    end
end
```

5.2.7 Local search

In this genetic algorithm, local search can be considered post-processing over the last generation of the genetic algorithm. The focus is to find individuals that are similar to the best solutions (neighborhoods) aiming to improve the solution while keeping computational requirements acceptable. It can be said that this search will work similarly to a guided mutation operator.

Three different searches are considered: highest profit tasks addition, lowest profit tasks removal, and both (addition and removal). Neighborhoods are then compared to the last generation results (selection), and the best individual is considered the problem solution.

Algorithm 7 – Local search

```
for i in (n last generation best solutions) do
    for (number of searches) times do
        search 1:
            add high profit value tasks that are not presented in the individual;
        search 2:
            remove low profit value tasks from the individual;
        search 3:
            add high profit value tasks that are not presented in the individual;
            remove low profit value tasks from the individual;
    end
end
selection over neighborhood solutions and last generation;
```

5.2.8 Parameters

One of the key concepts of a genetic algorithm is to keep the operators flexible enough to enable tuning, adapting the algorithm to the problem-solution space. With these parameters, the user can control the balance of selective pressure and solution space coverage, that is, the trade-off between rapid convergence to an optimized value versus local optima convergence risk (not intended in most of the cases).

In this algorithm, a good way to measure the selective pressure of the algorithm is to plot the fitness function (NPV) values for the individuals generated in all generations, as shown in figure 25.

Diverse parameters result in more varied solutions, while more selective parameters result in more similar solutions. Although that does not necessarily mean that the diverse parameters will result in fittest solutions, chances that this happens are higher in this case.

Besides, these parameters also control the computational time required to run the algorithm, which has to be reasonable even in problem instances with a huge number of tasks.

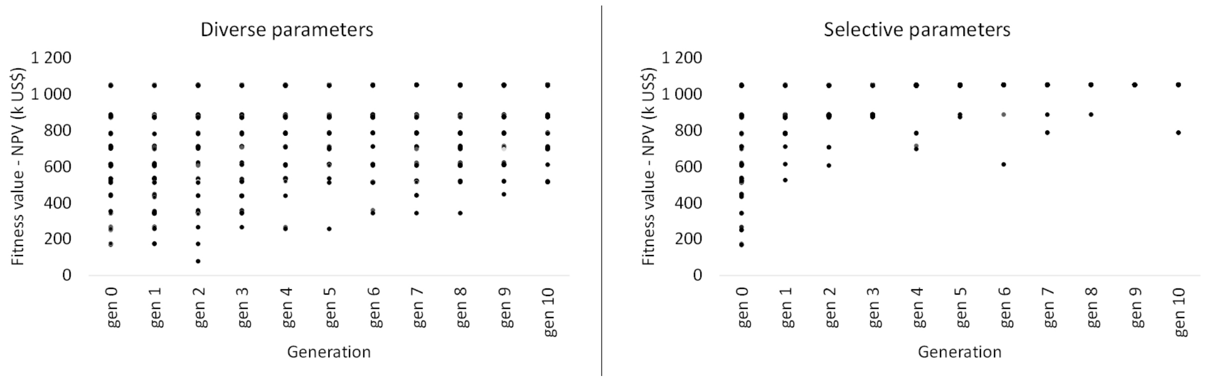


Figure 25 – Comparison of individuals variety according to parameters used.

Parameters considered in this genetic algorithm are:

- population size: number of individuals (chromosomes) in each generation. The greater the number of individuals, the greater the runtime and coverage of solution space.
- number of generations: number of cycles of crossover-mutation-selection that the initial population goes through. The rational is the same as the population size: the greater the number of generations, the greater the runtime and coverage of solution space.

- minimum number of elements in an individual: minimum number of tasks in a priority list (individual). This parameter aims to avoid artifacts such as one-task individuals or any too small solutions which may only consume processing time with poor solutions.
- mutation rate in a population: percentage of individuals in a population that will be mutated. This parameter balances the randomicity of the individuals: a more diverse population will better cover the solution space but will take longer to converge to an optimized solution.
- mutation rate in each individual: percentage of the tasks in an individual that will be removed or added. The rationale is similar to the previous parameter.
- selective pressure to update the population: percentage of elements in a population (parents plus offspring) that will be selected by their rank (fitness value). The remainder of the population (to complete the population size of the next generation) is selected randomly.
- selective pressure for crossover: the two possibilities are Roulette Wheel or Binary Tournament, as explained in section 5.2.5.
- number of individuals to search: individuals of the last generation that will undergo local search process. This is a trade-off between trying to optimize results versus computational time.
- number of searches per individual: Number of searches for each individual. Complementary to the previous parameter.

6 Case study

6.1 Problem instances - Inputs

A practical case study is evaluated to test each model presented in chapter 5. The full problem instance, showed in figure 26, is a mine area that represents 40% of yearly production during first 4 years of life of mine of a real Brazilian gold mine complex. The mine method here applied is sublevel open stoping on a top-down sequence. All levels are separated by sill pillars (horizontal pillars), therefore, the sequence in each level is independent. There is only one processing route, a carbon-in-leach circuit. This instance is formed by 2044 tasks among production (green solids) and development (all other colors representing different development tasks: main decline, crosscuts, oredrives, ventilation, etc.). This design considers at least marginally-feasible areas, i.e. areas which revenue offsets the activity costs (section 1.3). Time is discretized in days for all cases evaluated.

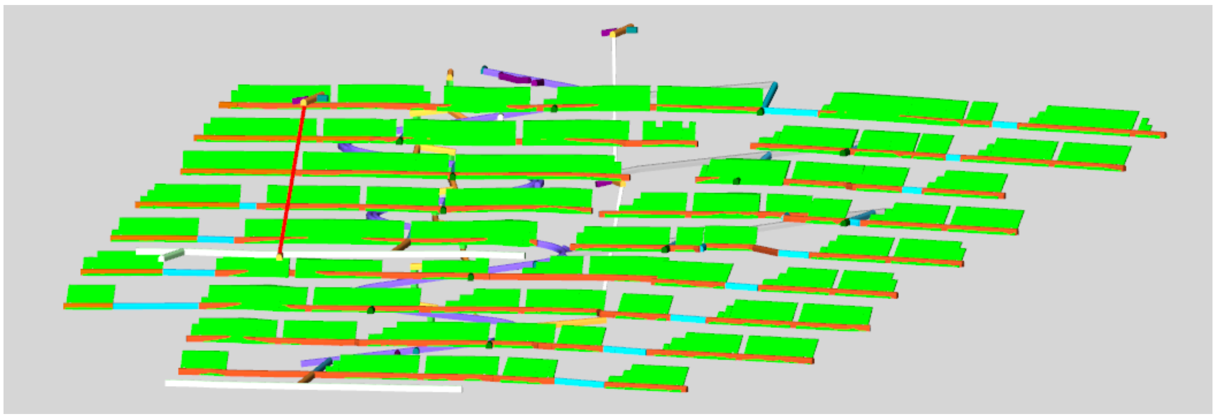


Figure 26 – Problem instance with 2044 tasks.

Since maximizing NPV is the objective, it is expected that solutions try, as much as possible, to postpone (or even exclude from schedule) lower profit tasks (higher cost and lower revenue such as development of lower grade stopes) and to anticipate high profit tasks (such as high grade stopes). In addition, as makespan (period costs) is a negative parcel in the NPV, that may lead to a maximum utilization of the input set of machines (production capability associated to the period cost that accounts for the fixed production structure).

Therefore, a solution will aim an optimized result for cut-off grades (selection of blocks) and machine allocation (sequencing). Each solution will represent an strategy for fixed scale of operation, mining method and processing route (section 1.3). Scale of operation could be evaluated by rerunning the algorithm with different inputs for number of machines and rates. Processing routes could be assessed changing plant recoveries (metal content, costs, and thus, profit values of each task). Mining method would require more work: redesign and new logical sequence (precedencies) for each different proposed method.

In order to asses the proposed algorithms and solvers, besides the full problem, other 6 smaller instances of the same problem (10, 73, 145, 489, 716, and 1279 tasks) are also considered. Physically, these smaller instances represents smaller mine areas (a panel, levels and part of a level) of the whole designed orebody presented in image 26. These smaller instances are shown in figure 27.

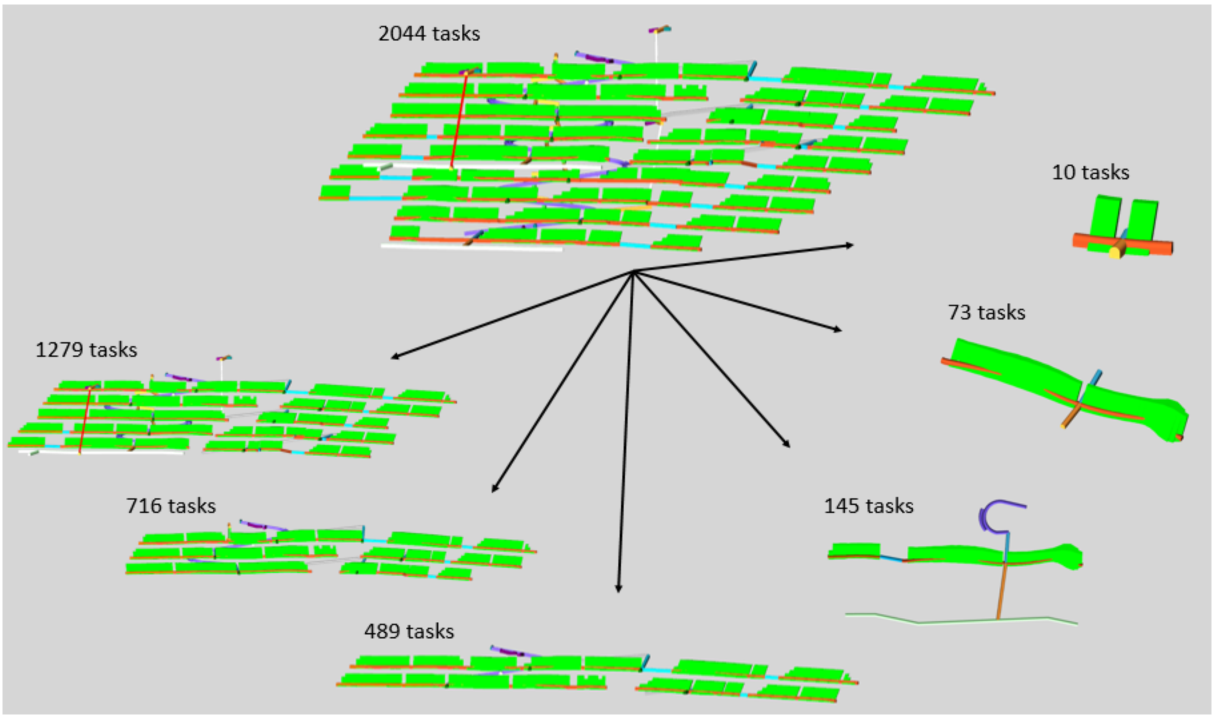
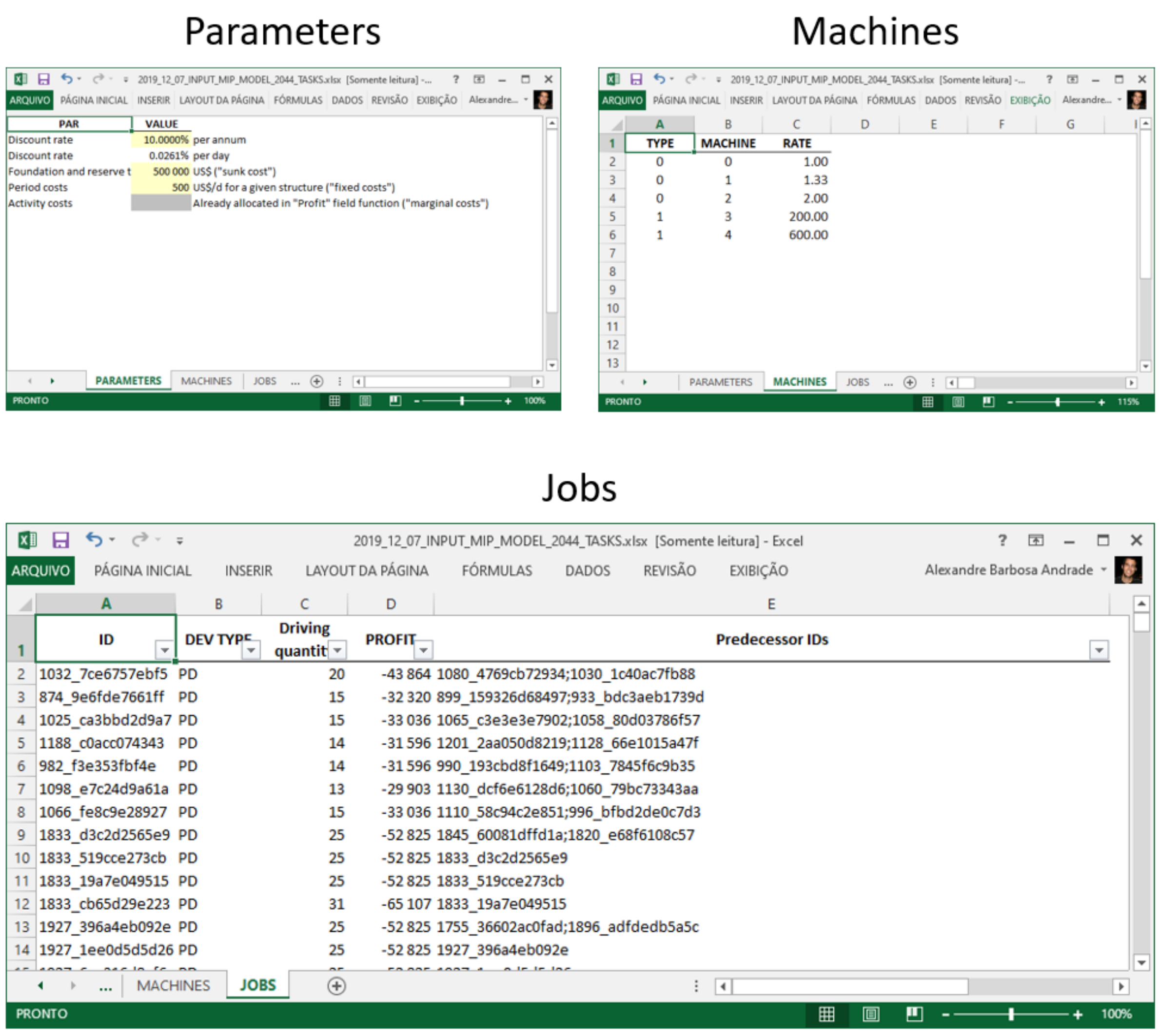


Figure 27 – Smaller instances of the case study set.

Each instance is a smaller area of the same data set.

Regardless of the instance (and task number), each designed solid has physical properties attached to it (ROM mass, grades, recovered metal, activity cost, revenue, profit, etc.) and also a list of predecessors of the logical sequence (mining method). This data comes as an output of the design step (section 1.4). This project was designed and scheduled (logical sequence) with use of Deswik (2020), a mining planning software. In Deswik.CAD, solids are designed and geological models interrogated for physical attributes. Deswik.IS is used mostly to create dependencies between tasks (logical sequence). In Deswik.SCHED, revenue, costs, and profits are calculated.

Ultimately, all algorithms developed herein will use as input a spreadsheet with 3 sheets: parameters (discount rates and period costs), machines (number, type and rate of each machine), and jobs (tasks with type, driving quantities, profit, and predecessors). Figure 28 depicts an input file. Tasks inputs are copied from Deswik.SCHED and results can also be pasted back to it.



PAR	VALUE	
Discount rate	10.0000%	per annum
Discount rate	0.0261%	per day
Foundation and reserve t	500 000	US$ ("sunk cost")
Period costs	500	US$/d for a given structure ("fixed costs")
Activity costs		Already allocated in "Profit" field function ("marginal costs")

TYPE	MACHINE	RATE
0	0	1.00
0	1	1.33
0	2	2.00
1	3	200.00
1	4	600.00

ID	DEV TYPE	Driving quantit	PROFIT	Predecessor IDs
1032_7ce6757ebf5	PD	20	-43 864	1080_4769cb72934;1030_1c40ac7fb88
874_9e6fde7661ff	PD	15	-32 320	899_159326d68497;933_bdc3aeb1739d
1025_ca3bbd2d9a7	PD	15	-33 036	1065_c3e3e3e7902;1058_80d03786f57
1188_c0acc074343	PD	14	-31 596	1201_2aa050d8219;1128_66e1015a47f
982_f3e353fbf4e	PD	14	-31 596	990_193cbd8f1649;1103_7845f6c9b35
1098_e7c24d9a61a	PD	13	-29 903	1130_dcf6e6128d6;1060_79bc73343aa
1066_fe8c9e28927	PD	15	-33 036	1110_58c94c2e851;996_bfbd2de0c7d3
1833_d3c2d2565e9	PD	25	-52 825	1845_60081dffd1a;1820_e68f6108c57
1833_519cce273cb	PD	25	-52 825	1833_d3c2d2565e9
1833_19a7e049515	PD	25	-52 825	1833_519cce273cb
1833_cb65d29e223	PD	31	-65 107	1833_19a7e049515
1927_396a4eb092e	PD	25	-52 825	1755_36602ac0fad;1896_adfdedb5a5c
1927_1ee0d5d5d26	PD	25	-52 825	1927_396a4eb092e

Figure 28 – Spreadsheet with inputs of the problem set with 2044 tasks.

6.2 Implementation - Solving tools

All algorithms are development in Python (2020), initially in Jupyter Notebook Jupyter (2020) and later translated to .py scripts.

Pandas (2020) and NumPy (2020) are the main libraries used in this project and play a core role in data manipulation, modelling, and input/output interface through tables, matrices and arrays.

Swifter, (CARPENTER, 2020), is a key component that adds multiprocessing (multi-core) capabilities to the genetic algorithm. Solvers packages, (IBM, 2019) and

(OR-TOOLS, 2019), are also used but genetic algorithm is developed without any specific GA package. Plotly (2020) is used to generate Gantt graphs images for rapid results analysis.

To solve the different algorithms in a uniform platform, Google Cloud Computing, Google (2020), is used with "n1-standard-16" (16 vCPUs, 60 GB memory) virtual machines on Linux, (DEBIAN, 2020).

Each algorithm was executed constrained to a time limit of 24 hours, since that is more than a reasonable time for mine planning scheduling.

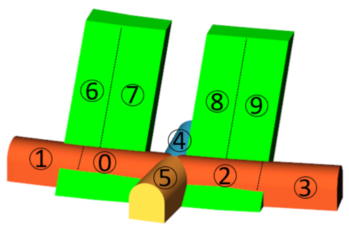
All the implemented code is available at:

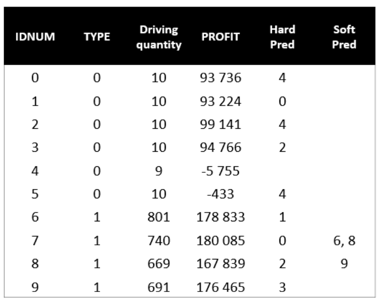
<https://github.com/alexandre-b-andrade/ug-cog-sched-opt>.

6.3 Methodology analysis - Algorithm decisions

Before comparing results for the different solvers and algorithms, the analysis is detailed on smaller instances to explain the algorithms actions on them, i.e., how each step of the methodology works on real data.

Figure 29 shows the ten-task problem instance and its result after CPLEX solve model 2, optimizing NPV, (subsection 4.3.2). This schedule has only one starting point, task 4. Even having negative profit, task 4 is scheduled since the depending chain of tasks will offset its negative profit.





IDNUM	TYPE	Driving quantity	PROFIT	Hard Pred	Soft Pred
0	0	10	93 736	4	
1	0	10	93 224	0	
2	0	10	99 141	4	
3	0	10	94 766	2	
4	0	9	-5 755		
5	0	10	-433	4	
6	1	801	178 833	1	
7	1	740	180 085	0	6, 8
8	1	669	167 839	2	9
9	1	691	176 465	3	

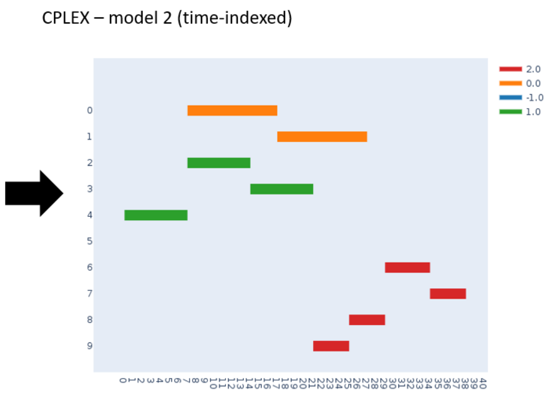


Figure 29 – 10-task instance input and output (model 2).

Legend shows machines where 0 and 1 are for development (tasks of type 0) and 2 for stoping (tasks of type 1). Vertical axis represents the task ID and horizontal axis the time in days. Machine 1 has a greater rate than machine 0.

After task 4 allocation, there are 3 branches: task 5 has negative profit and no positive successors, thus it is not allocated and is represented by machine -1 (can be seen in Gantt legend, figure 29); task 0 branch has tasks 0, 1 and 6 giving a smaller cumulative profit than task 2 branch (which has tasks 2, 3, 8 and 9). The latter is prioritized and allocated to the best machine, anticipating its side (anticipating higher profit and improving NPV). Task 7 depends of both sides and is the last one scheduled. Besides tasks 4 and 5, profit values are much greater than the period costs (fixed cost), consequently all tasks are scheduled. It could be said that eliminating tasks (e.g. 1 and 6) to anticipate others (e.g. 7) and reduce the period cost parcel would not be as profitable as scheduling the whole design.

For the same 10-task input, showed in image 29, model 1 would result in scheduling all tasks and model 3 would eliminate task 5 but not try to anticipate task 1 (since task 1 profit adds the same parcel to the objective function regardless of the starting time). This is depicted in figure 30.

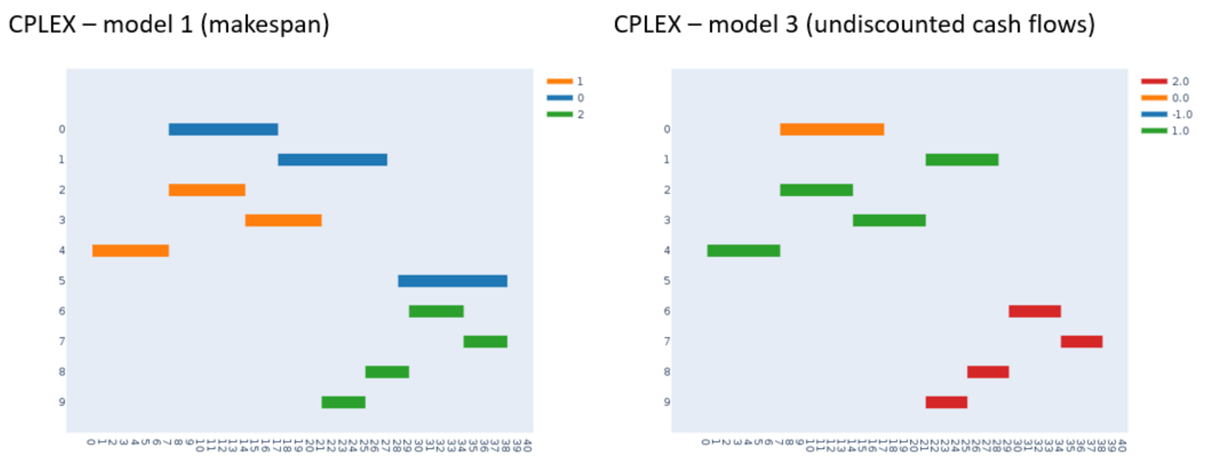


Figure 30 – 10-task instance solutions for models 1 (makespan) and 3 (undiscounted).

A second example, solved by the genetic algorithm, considers the 73-task instance optimized for different period costs. In this example, period costs vary from 500 US$/day to 50 000 US$/day, each of them could represent a different mining strategy. As seen on image 31, a greater period cost pressures the selection for high margin areas. In some cases, excluding an positive-profit area to reduce makespan (and its relative period cost parcel) will improve NPV.

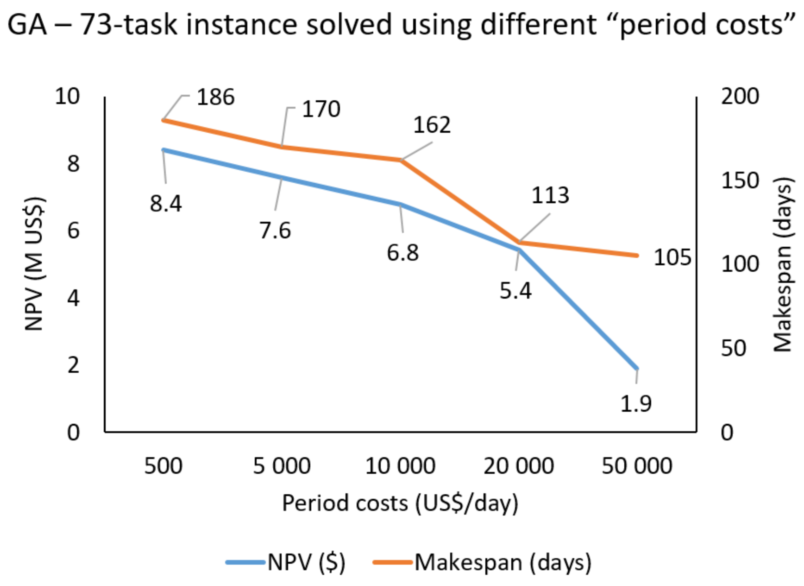


Figure 31 – Solutions of the 73-task instance to different period costs input.

A third example, solved by CPLEX, shows results for the 145-task instance. For testing purposes, a geology drilling development (only costs involved) is included on design without any dependencies to production areas (positive values that could offset the investment on the geology drilling development).

Both models 2 and 3 understand that extra development as avoidable cost and remove it from the scheduling. This is highlighted in image 32*b* with red circles. Low grade (low value) portion of production areas is also excluded by the algorithm among other tasks that are excluded due to scheduling capacities.

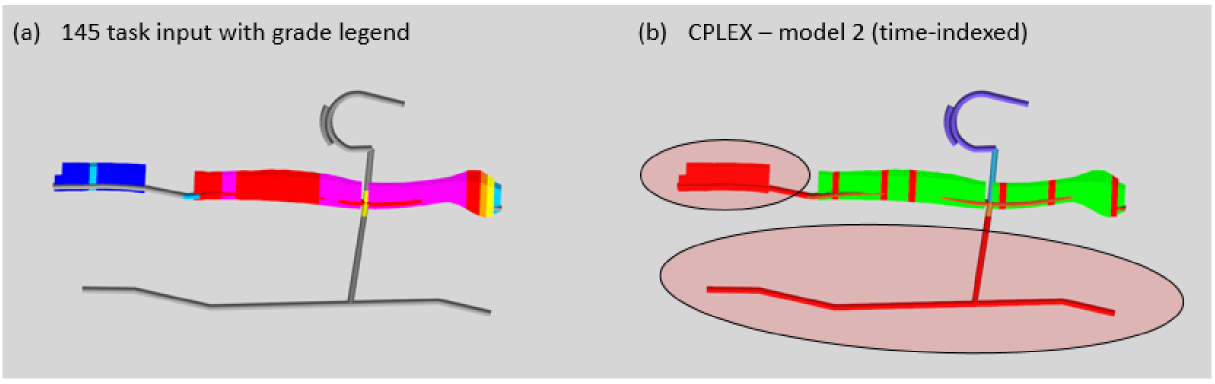

(a) 145 task input with grade legend
(b) CPLEX – model 2 (time-indexed)

Figure 32 – 145-task instance.

(a) shows the input design with grade legend (grey has 0 grade; blue, low grade; red and pink, high grades). (b) shows the areas that were removed by the CPLEX to optimize NPV (model 2).

6.4 Results and findings - Outputs

The three models (section 4.3) are coded in 3 solvers (section 5.1). Therefore, 9 different solution methods for mathematical programming should be available for each instance. Besides them, a tenth method is the genetic algorithm developed.

The instances are the same for all models. Besides the 10-task instance, which uses 3 machines, other instances consider 5 machines among development and production.

6.4.1 Exact solvers

In general, exact solvers linear models increase their size substantially with input tasks and machine numbers. This can be seen in graph 33 below. This graph also highlights a key difference between models: Model 1 (makespan) and model 3 (undiscounted value) have less variables than model 2 (that optimizes NPV with binary time-indexed variables). On the other hand, model 2 has much less constraints on it when compared to models 1 and 3.

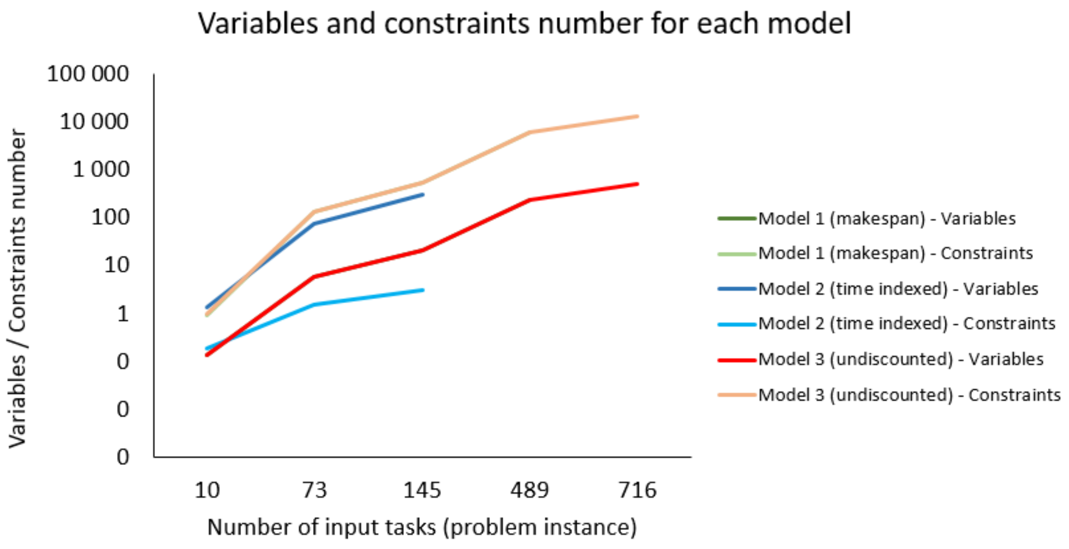
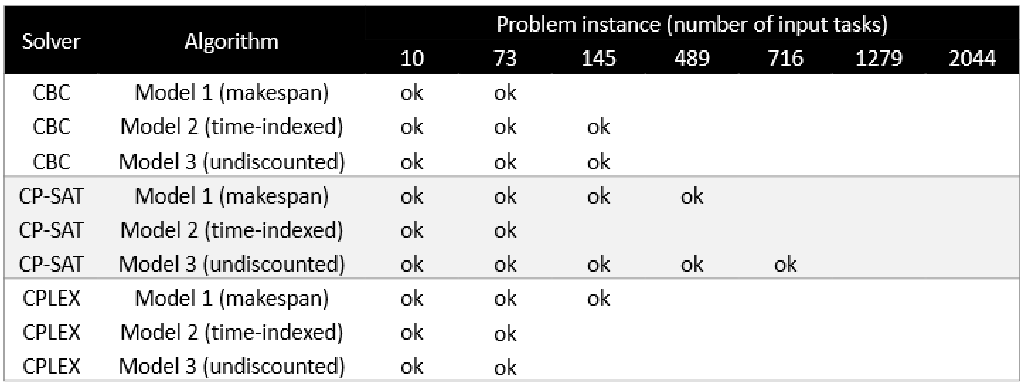


Figure 33 – Number of variables and constraints per model.

Model 1 and model 3 are so similar that they basically overlap. Vertical axis is on a logarithmic scale.

Consequently, only smaller instances could be solved by CBC (CBC, 2019), CPLEX (IBM, 2019) and CP-SAT (CP-SAT, 2019). In more complex cases (greater number of variables and constraints), exact solvers scripts crashed mostly due to RAM memory constraints (some of the instances had problems even earlier, at the model creation and allocation to memory). This is shown on table 34.

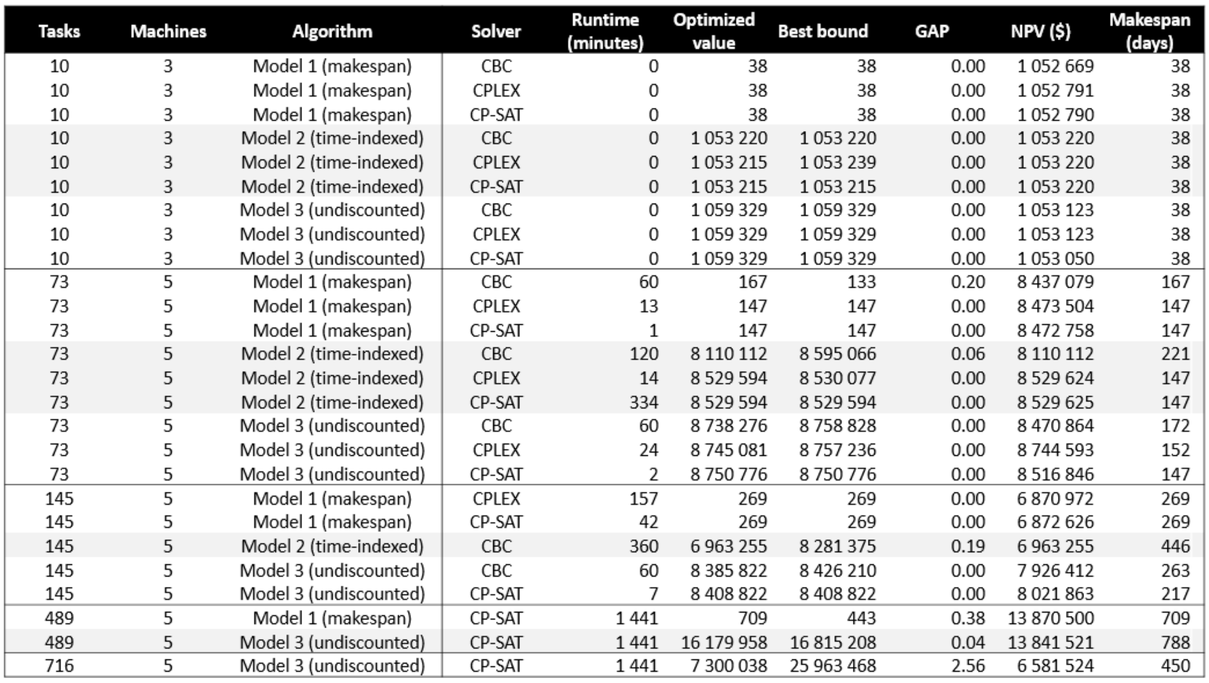


Solver	Algorithm	Problem instance (number of input tasks)						
		10	73	145	489	716	1279	2044
CBC	Model 1 (makespan)	ok	ok					
CBC	Model 2 (time-indexed)	ok	ok	ok				
CBC	Model 3 (undiscounted)	ok	ok	ok				
CP-SAT	Model 1 (makespan)	ok	ok	ok	ok			
CP-SAT	Model 2 (time-indexed)	ok	ok					
CP-SAT	Model 3 (undiscounted)	ok	ok	ok	ok	ok		
CPLEX	Model 1 (makespan)	ok	ok	ok				
CPLEX	Model 2 (time-indexed)	ok	ok					
CPLEX	Model 3 (undiscounted)	ok	ok					

Figure 34 – Exact solvers runs.

Completed ones are marked with "ok". Blank ones have failed.

The obtained results for the instances that could be run are shown on table 35. It is worth noting that the instances with 489 tasks or more that could be solve reached the time limit of 24 hours. CP-SAT results of 716-task instance is only feasible and presents a huge GAP.



Tasks	Machines	Algorithm	Solver	Runtime (minutes)	Optimized value	Best bound	GAP	NPV ($)	Makespan (days)
10	3	Model 1 (makespan)	CBC	0	38	38	0.00	1 052 669	38
10	3	Model 1 (makespan)	CPLEX	0	38	38	0.00	1 052 791	38
10	3	Model 1 (makespan)	CP-SAT	0	38	38	0.00	1 052 790	38
10	3	Model 2 (time-indexed)	CBC	0	1 053 220	1 053 220	0.00	1 053 220	38
10	3	Model 2 (time-indexed)	CPLEX	0	1 053 215	1 053 239	0.00	1 053 220	38
10	3	Model 2 (time-indexed)	CP-SAT	0	1 053 215	1 053 215	0.00	1 053 220	38
10	3	Model 3 (undiscounted)	CBC	0	1 059 329	1 059 329	0.00	1 053 123	38
10	3	Model 3 (undiscounted)	CPLEX	0	1 059 329	1 059 329	0.00	1 053 123	38
10	3	Model 3 (undiscounted)	CP-SAT	0	1 059 329	1 059 329	0.00	1 053 050	38
73	5	Model 1 (makespan)	CBC	60	167	133	0.20	8 437 079	167
73	5	Model 1 (makespan)	CPLEX	13	147	147	0.00	8 473 504	147
73	5	Model 1 (makespan)	CP-SAT	1	147	147	0.00	8 472 758	147
73	5	Model 2 (time-indexed)	CBC	120	8 110 112	8 595 066	0.06	8 110 112	221
73	5	Model 2 (time-indexed)	CPLEX	14	8 529 594	8 530 077	0.00	8 529 624	147
73	5	Model 2 (time-indexed)	CP-SAT	334	8 529 594	8 529 594	0.00	8 529 625	147
73	5	Model 3 (undiscounted)	CBC	60	8 738 276	8 758 828	0.00	8 470 864	172
73	5	Model 3 (undiscounted)	CPLEX	24	8 745 081	8 757 236	0.00	8 744 593	152
73	5	Model 3 (undiscounted)	CP-SAT	2	8 750 776	8 750 776	0.00	8 516 846	147
145	5	Model 1 (makespan)	CPLEX	157	269	269	0.00	6 870 972	269
145	5	Model 1 (makespan)	CP-SAT	42	269	269	0.00	6 872 626	269
145	5	Model 2 (time-indexed)	CBC	360	6 963 255	8 281 375	0.19	6 963 255	446
145	5	Model 3 (undiscounted)	CBC	60	8 385 822	8 426 210	0.00	7 926 412	263
145	5	Model 3 (undiscounted)	CP-SAT	7	8 408 822	8 408 822	0.00	8 021 863	217
489	5	Model 1 (makespan)	CP-SAT	1 441	709	443	0.38	13 870 500	709
489	5	Model 3 (undiscounted)	CP-SAT	1 441	16 179 958	16 815 208	0.04	13 841 521	788
716	5	Model 3 (undiscounted)	CP-SAT	1 441	7 300 038	25 963 468	2.56	6 581 524	450

Figure 35 – Exact models results for the different instances.

The first intention in proposing model 1 was to reduce processing time of subsequent models 2 or 3, working similarly to a preprocessing step for them. However, since model 1 takes about the runtime of model 2 and 3, its use would not make sense in most of the cases. In addition, model 3, could not be used to find an upper bound input to model 2, since it could limit the makespan more than necessary and limit model

2 results. Thus, the same allocation heuristic used in the Genetic Algorithm is used to calculate a first upper-bound limit (makespan) for all three models, limiting all starting time variables.

In general, model 2 and model 3 results were similar, but it has to be considered that this is affected by the size (number of tasks) of the inputs: the smaller the size, the smaller the discount rate impact (main difference between model 2 and 3).

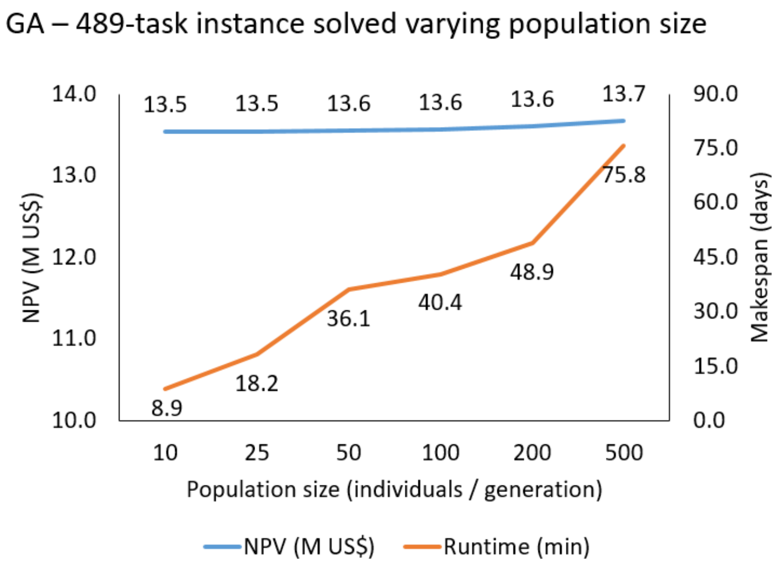
It is not the main intent of this study to compare solvers in their architecture details and for general cases. Thus, this is done only based on the runs undertaken with these models and instances. We see that, for model 2, which has more variables, usually CPLEX runtime is smaller than the others. CP-SAT worked better with models that have more constraints ("or constraints"): models 1 and 3, resulting in faster solving time in these when compared to CPLEX and CBC. CBC solver seemed to not fully enjoy the multiprocessing CPUs as CPLEX and CP-SAT did, and had greater runtime in most of the cases. For the same models, results varied less than 1% in all runs, not justifying an specific solver due to solution improvement, as expected.

For the instances considered and solved herein, runtime could not be directly related to input size (number of tasks and machines) whether linearly, exponentially, etc.

6.4.2 Genetic algorithm

Genetic algorithm requires extra runs to set up the main parameters (described in subsection 5.2.8). Since to evaluate all parameters combinations would be tremendously time consuming, tests are undertaken for two main parameters: population size and selective pressure. 489-task instance is considered for this evaluations as it is deemed to have adequate size for that. Main quality driver is the NPV evolution, runtime and number of different NPV evaluated (i.e. minimum number of individuals considered).

Graph 36 compares the evolution of NPV (objective value) according to different population sizes. NPV increases until penultimate generation (12^{th} in total) for 10, 25 and 50 individuals per generation. Beyond that NPV plateaus earlier, in 10^{th}, 6^{th} and 8^{th} for 100, 200 and 500 individuals per generation, respectively. And yet, improvements on this generations compared to the 4^{th} earliest ones were less than 1%.



6

Figure 36 – GA population size definition.

Graph 37 shows how selective pressure affects NPV. As explained in algorithm 5 (subsection 5.2.4), this parameter controls the percentage of next generation individuals that are selected based on their fitness value rank only (higher NPV). For this runs, 12 generations with 50 individuals per generation were used. Thus, 600 different individuals evaluated. Altough keeping best individuals in population is desired, having different individuals is also required. Unique NPV is presented to measure this balance, and it can be seen that it reduces as the selective pressure increases. Based on this, 0.25 was chosen as the selective pressure parameter. Runtime was about the same in all runs (35 to 40 min). NPV slightly greater when selective pressure was 0.25.

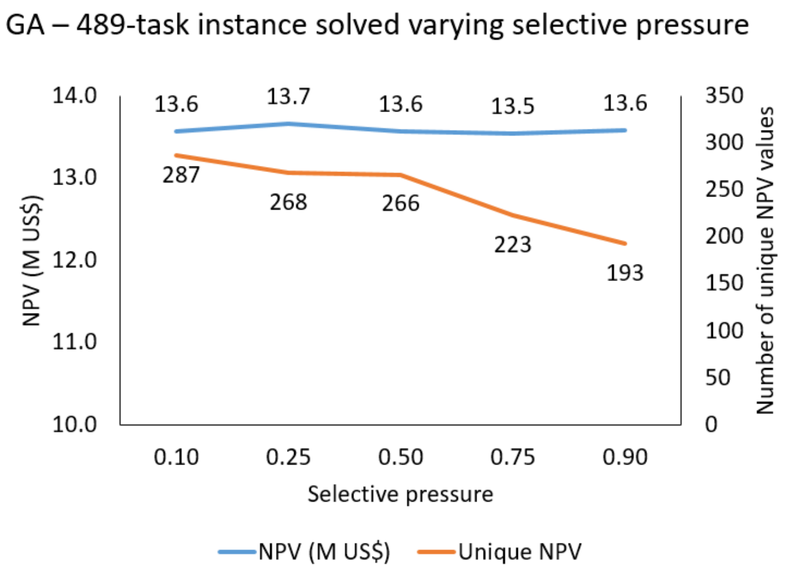
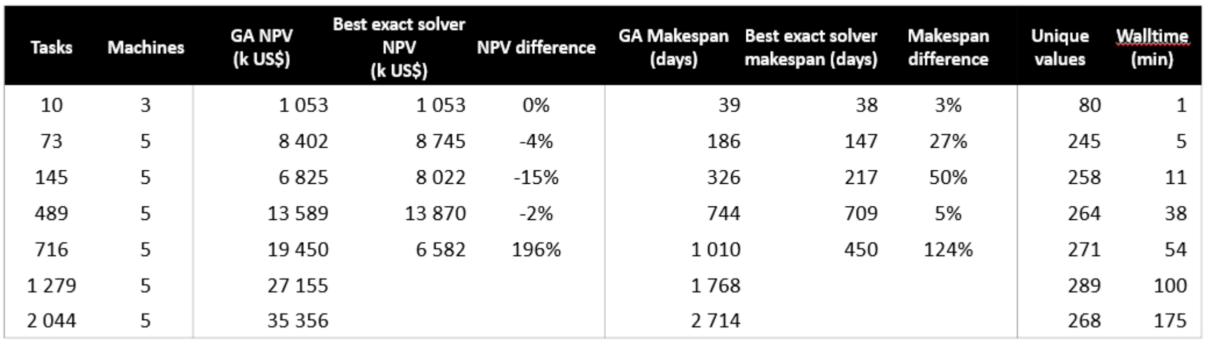


Figure 37 – GA selective pressure definition.

Having chosen the parameters, all instances are run and compared to exact solvers (best of all solvers and models) solutions, when they are available. Table 38 compare the results. It is important to emphasize that, oppositely to the exact solvers, GA algorithm takes linear time to solve a problem. Thus, solving a 2000-task problem would take approximately 200 times the time required to solve a 10-task problem. This predictability of the number of calculations (and thus, of runtime) is another important feature of this algorithm. Exact solvers could optimize the results only for the instances up to 489 tasks. Thus, exact solver results for 716-task instance are not optimum and were limited by the runtime of 24 hours.



Tasks	Machines	GA NPV (k US$)	Best exact solver NPV (k US$)	NPV difference	GA Makespan (days)	Best exact solver makespan (days)	Makespan difference	Unique values	Walltime (min)
10	3	1 053	1 053	0%	39	38	3%	80	1
73	5	8 402	8 745	-4%	186	147	27%	245	5
145	5	6 825	8 022	-15%	326	217	50%	258	11
489	5	13 589	13 870	-2%	744	709	5%	264	38
716	5	19 450	6 582	196%	1 010	450	124%	271	54
1 279	5	27 155			1 768			289	100
2 044	5	35 356			2 714			268	175

Figure 38 – GA results for the different instances.

Based on these runs, it was observed that although different individuals are indeed created, NPV improvements after local search were small (less than 1%).

Ultimately, considering comparisons presented in this chapter, it can be said that GA seemed the best alternative to solve for large problem instances.

7 Conclusions

The selection (definition of cut-off grades) and scheduling of tasks in underground mines is a complex and highly relevant problem, both from logistical and financial point of views. To tackle this problem, three different models are proposed herein and implemented with different solvers.

Linear programming models developed herein have shown notable results with regards to tasks selection and scheduling for an underground project. Genetic algorithm implemented for model 2, section 4.3.2, has proven to better balance solution quality and runtime, delivering reliable solutions that are practical and applicable to underground mine planning.

The implementation of the proposed methodology and algorithm can lead to an efficient planning and evaluation of multiple strategies through optimization and automation of the process.

All the implemented code is available at:

<https://github.com/alexandre-b-andrade/ug-cog-sched-opt>.

In addition, the theoretical framework, presented in the first part of this dissertation, brings a starting guide to underground mining strategic planning with references that are currently part of the mine industry and, therefore, this may be basis for other studies alike.

Future work can be developed, at least, through three approaches: current algorithm performance improvement, algorithm adaption to a wider range of methods and inputs, and study of alternative techniques.

Performance improvements, towards reduced runtime, could be achieved with the addition of preprocessing routines, such as the accumulation of similar tasks (type and profit). Solving the problem in different precision rounds, accepting higher GAP at the beginning and then reducing GAP and makespan iteratively, could be beneficial specially to model 2 algorithms. For the genetic algorithm, the allocation routine (decodification step) concentrates most of the computational requirements. Thus, any improvement on its performance (or solution quality) would directly improve the whole algorithm. Other subject for improvement of the GA is the auto-tuning of parameters.

The proposed algorithms can solve many problem situations already. However, more tests and case studies with different methods (logical sequence) and inputs could be evaluated. An interesting addition to the models would be to consider economic parameters (that determine profit values) varying with time. A second one would be to model the GA for a multi-objective optimization (e.g. makespan and NPV) generating alternative solutions to the current model (scenarios). Also, current user interface consists of basic spreadsheets and could be improved to a more user-friendly interface.

Other solving methods could be subject of investigation and comparison. For example, pseudoflow algorithms have proven to be a good tool for open pit mining planning and should be more investigated for underground mining problems.

7.1 Participation in conferences

- "Otimização do lucro da extração de minério de uma mina subterrânea", 3º ERPO SE - Encontro Regional de Pesquisa Operacional, Limeira, São Paulo (2018).
- "Understanding plan's priorities: Short term scheduling optimization", 39th International Symposium 'Application of Computers and Operations Research in the Mineral Industry' (APCOM 2019), Wroclaw, Poland (2019).
- "Economic optimization of rib pillars placement in underground mines", 39th International Symposium 'Application of Computers and Operations Research in the Mineral Industry' (APCOM 2019), Wroclaw, Poland (2019).

Glossary

B | C | D | F | H | M | O | R | S | T

B

blasts

Blast, with use of explosives, of holes in rocks as a method to excavate development tunnels and production blocks (stopes).

Desmontes de rocha com explosivos em túneis de desenvolvimento ou realces de produção.

C

crosscuts tunnels

Type of development tunnels (galleries) that connect main permanent accesses (declines) to the tunnels that exposes the orebody (oredrives).

Travessas são túneis (galerias) de acesso que conectam os acesso principais e permanentes aos túneis localizados no corpo de minério.

D

downstream costs

Costs from a production stage onwards given that the decision up to this stage is already made due to other reasons. For instance, if a development tunnel is to be excavated to access an ore block, its downstream costs will be only the haulage and processing costs (if considered ore). In this example, drill and blast costs were already considered at the time of the economic analysis of the stope block.

Custos que ocorrem a partir da etapa atual do processo produtivo. Assim, uma vez que determinado volume de rocha já está desmontado e transportado para uma pilha, seus possíveis downstream costs são os custos de destiná-lo como estéril ou beneficiá-lo numa planta de tratamento. Ou seja, os custos de desmonte e transporte de rocha não são considerados downstream costs.

drawpoint

Type of development tunnel used for haulage ore from production blocks (stopes).

Tipo especício de galeria (túnel) destinado ao transporte de minério.

F

flat-lying orebodies

Mineral deposits (orebodies) that have low inclination (dip).

Depósitos minerais (corpos mineralizados) de baixa inclinação.

H

hanging wall

The upper wall of an inclined vein, fault, or other geologic structure. Opposed to footwall.

Parte de rocha não mineralizada (estéril) que está acima do corpo mineralizado. Footwall é o análogo para a parte abaixo do corpo mineralizado.

M

main decline

Type of development tunnel (gallery) that is used along the whole life of an orebody to access its different levels. Thus it is considered a permanent access.

Rampa principal é um tipo específico de desenvolvimento (galeria, túnel) que é utilizado ao longo de toda produção de um corpo mineralizado para acesso aos diversos níveis. Deste modo, é considerado um acesso permanente.

mineral

A solid inorganic substance of natural occurrence. One of more minerals form rocks.

Um mineral é um composto químico sólido que ocorre naturalmente em sua forma pura. Minerais, quando em conjunto, formam rochas.

O

open cast

Surface mining techniques of extracting rock or minerals from the earth by their removal in hillsides or stripes (stripping mining). This form of mining differs from extractive methods that require tunnelling into the earth (underground mining).

Forma de lavra de um corpo mineralizado a céu aberto porém sem a formação de uma cava (característica de um open pit). Desta forma, são lavras em encontas ou em tiras.

open pits

Surface mining method of extracting rock or minerals from the earth by their removal from an open pit (inverted cone shape excavation).

Método de lavra em cavas a céu aberto (formato de cone invertido).

ore

Mineral or rock from which it can be extracted one of more substances that are economically useful. The economic definition is the factor that differentiate ore from an mere mineral. Thus, on the same orebody, we can have ore in some zones and on mineralization in others (economically infeasible areas).

Minério é todo mineral ou rocha do qual se pode extrair uma ou mais substâncias economicamente úteis. A questão econômica é o fator que diferencia minério de um mineral. Então, num mesmo corpo mineralizado, podemos ter porções de minério e outras de apenas mineralização (partes inviáveis de um corpo mineralizado).

ore stopes

Openings of large underground rooms by the excavation of ore usually using long-hole drilling in vertical or sub-vertical shapes.

Realces de minério são grandes câmaras de produção em que a rocha é desmontada para posterior transporte, beneficiamento e venda da substância de interesse ali contida.

orebody

Mineralized zone defined three-dimensionally by specific rocks or part of them that have different characteristics (metal grades, composition, etc.) from their surroundings and that has a prospect to be considered ore.

Apesar da tradução literal: corpo de minério, muitas vezes opta-se por se utilizar a definição "corpo mineralizado", já que a definição de minério depende da viabilidade econômica de um corpo mineralizado. Desta forma, orebody é uma delimitação tridimensional de uma determinada rocha ou de parte dela que contém características diferentes (teor, composição, etc.) das presentes ao seu redor. Também denominada depósito mineral.

oredrives

Type of development tunnel (gallery) that exposes the orebody for following production activities (drill, blast and haul of stopes).

Subníveis, ou galerias de minério, é um tipo específico de desenvolvimento (galeria, túnel) que corta o corpo mineralizado de modo a expô-lo para subsequente produção.

outcropping

Mineral deposit (orebody) that are near to surface or on the surface of the earth.

Afloramento é quando ocorre de um corpo mineralizado ter continuidade até a superfície, o que facilita sua economicidade para lavra a partir de método a céu aberto.

R

rules of thumb

Empirical calculation rules.

São regras empíricas.

S

stripping mining

Surface mining method of extracting rock or minerals from the earth by their removal from parallel shallow stripes. It is applied to horizontal or sub-horizontal shallow orebodies. Commonly used in bauxite mining.

Lavra em tiras é um método de lavra aplicado para corpos horizontalizados de baixa profundidade. Comumente aplicado a lavra de bauxita.

T

thin-bedded deposits

Narrow and horizontal or sub-horizontal orebodies.

Corpos mineralizados (depósitos mineralizados) horizontalizados e de baixa espessura.

Bibliography

ANDRADE, A. O. A. *Otimização do lucro da extração de minério de uma mina subterrânea*. 2018. Cited on page 56.

BÄCK, T.; FOGEL, D. B.; MICHALEWICZ, Z. *Evolutionary computation 1: Basic algorithms and operators*. [S.l.]: CRC press, 2000. v. 1. Cited on page 66.

______. *Evolutionary Computation 2: Advanced Algorithms and Operators*. [S.l.]: CRC press, 2000. v. 1. Cited on page 66.

BARBOSA, F. *O problema de alocaçao de berços: Aspectos teóricos e computacionais*. Dissertação (Dissertação de Mestrado) — IMECC, Universidade Estadual de Campinas, 2014. Cited on page 38.

BENNETT, M. S. W. *Analysing an underground mine with open pit optimisation tools*. 2018. Cited on page 56.

BRUCKER, P. *Scheduling Algorithms*. 5. New York: Springer-Verlag B. H., 2006. Cited 2 times on pages 14 and 34.

CARPENTER, J. *swifter*. 2020. Disponível em: <https://github.com/jmcarpenter2/swifter>. Cited on page 74.

CBC. *COIN-OR Branch-and-Cut solver*. 2019. Disponível em: <https://github.com/coin-or/Cbc>. Cited 2 times on pages 62 and 78.

COELLO, C. A. C. Evolutionary multiobjective optimization: Theoretical advances and applications. In: ABRAHAM, A.; JAIN, L.; GOLDBERG, R. (Ed.). 1. Londres: Springer-Verlag, 2005. cap. Recent Trends in Evolutionary Multiobjective Optimization, p. 7–32. Cited on page 46.

CP-SAT. *CP-SAT Solver*. 2019. Disponível em: <https://github.com/google/or-tools>. Cited 2 times on pages 62 and 78.

CUMMINS, A.; GIVEN, I.; AIME., S. of Mining Engineers of. *SME Mining Engineering Handbook*. Society of Mining Engineers, American Institute of Mining, Metallurgical, and Petroleum Engineers, 1973. (Mudd series, v. 2). Disponível em: <https://books.google.com.br/books?id=B2ovAAAAIAAJ>. Cited on page 26.

DEBIAN. *Debian OS*. 2020. Disponível em: <https://www.debian.org/>. Cited on page 75.

DESWIK. *Deswik software*. 2020. Disponível em: <https://www.deswik.com/>. Cited on page 73.

DRÉO, J.; CHATTERJEE, A.; PÉTROWSKI, A.; SIARRY, P.; TAILLARD, E. *Metaheuristics for Hard Optimization: Methods and Case Studies*. Springer Berlin Heidelberg, 2006. ISBN 9783540309666. Disponível em: <https://books.google.com.br/books?id=l-acEfdFZ1MC>. Cited 2 times on pages 44 and 45.

EMMONS, H.; VAIRAKTARAKIS, G. *Flow Shop Scheduling: Theoretical Results, Algorithms, and Applications*. Springer US, 2012. (International Series in Operations Research & Management Science). ISBN 9781461451518. Disponível em: <https://books.google.com.br/books?id=6kjjS4dfmmMC>. Cited 2 times on pages 39 and 40.

FAPESP, R. P. *A corrida da Indústria 4.0*. 2017. Disponível em: <http://revistapesquisa.fapesp.br/2017/09/22/a-corrida-da-industria-4-0>. Cited on page 14.

FONTBONA, F. T. *Optimisation methods meets the smart grid. New methods for solving location and allocation problems under the smart grid paradigm*. 2015. Cited on page 44.

GLOVER, F.; KOCHENBERGER, G. *Handbook of Metaheuristics*. Springer US, 2003. (International Series in Operations Research & Management Science). ISBN 9781402072635. Disponível em: <https://books.google.com.br/books?id=xmhoG772iuQC>. Cited on page 44.

GOOGLE. *Google Cloud Computing*. 2020. Disponível em: <https://cloud.google.com/>. Cited on page 75.

HALL, B. *Cut-off grades and optimising the strategic mine plan*. [S.l.]: The Australasian Institute of Mining and Metallurgy, 2014. ISBN 9781925100228. Cited 2 times on pages 26 and 29.

HARTMAN, H. L. *Introductory Mining Engineering*. [S.l.]: Wiley, 2002. ISBN 0471348511. Cited 2 times on pages 17 and 18.

HUSTRULID, W. A. (Ed.). *Underground Mining Methods Handbook*. [S.l.]: Society for Mining Metallurgy, 1982. ISBN 089520049X. Cited 7 times on pages 17, 18, 19, 20, 21, 22, and 23.

IBM. *IBM ILOG CPLEX Optimization Studio*. 2019. Disponível em: <https://www.ibm.com/products/ilog-cplex-optimization-studio>. Cited 4 times on pages 62, 63, 74, and 78.

JOHNSON, J. W. P. E. J. Effort and accuracy in choice. *OR Spectrum Quantitative Approaches in Management*, 1985. Disponível em: <https://doi.org/10.1287/mnsc.31.4.395>. Cited on page 44.

JUPYTER. *Project Jupyter*. 2020. Disponível em: <https://jupyter.org/>. Cited on page 74.

KENNEDY, B. A. *Surface Mining*. [S.l.]: Society for Mining, Metallurgy, and Exploration, 1990. ISBN 0873351029. Cited on page 25.

KING, B. Integrated strategy optimization for complex operations. *Orebody Modelling and Strategic Mine Planning. Perth, WA*, The Australasian Institute of Mining and Metallurgy, p. 381–398, 2004. Cited 2 times on pages 29 and 54.

KNOWLES, J.; CORNE, D.; DEB, K. *Multiobjective Problem Solving from Nature: From Concepts to Applications*. 1. Berlin: Springer-Verlag, 2008. Cited on page 46.

KRAMER, O. *Genetic Algorithm Essentials*. Springer International Publishing, 2017. (Studies in Computational Intelligence). ISBN 9783319521565. Disponível em: <https://books.google.com.br/books?id=NxLcDQAAQBAJ>. Cited 5 times on pages 14, 46, 47, 56, and 66.

LANE, K. *The Economic Definition of Ore: Cut-Off Grades in Theory and Practice*. Comet Strategy Pty Limited, 2015. ISBN 9780994185228. Disponível em: <https://books.google.com.br/books?id=kvXdswEACAAJ>. Cited 4 times on pages 26, 27, 28, and 29.

LEUNG, J. Y.-T. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. 4^aed. United States: Chapman and Hall/CRC, 2004. Cited on page 34.

LOBODA MAXIM N. ARTYOMOV, A. A. S. A. A. Solving generalized maximum-weight connected subgraph problem for network enrichment analysis. *CoRR*, abs/1605.02168, 2016. Disponível em: <http://arxiv.org/abs/1605.02168>. Cited on page 56.

MEC. *2019 minerals consumption*. 2019. Disponível em: <https://mineralseducationcoalition.org/mining-mineral-statistics>. Acesso em: 22 July 2019. Cited on page 17.

MICHALEWICZ, Z. *Genetic Algorithms + Data Structures = Evolution Programs*. 3. Berlin, Germany: Springer-Verlag, 1996. Cited on page 46.

MICHALEWICZ, Z.; FOGEL, D. *How to Solve It: Modern Heuristics*. Springer Berlin Heidelberg, 2004. ISBN 9783540224945. Disponível em: <https://books.google.com.br/books?id=RJbV_-JlIUQC>. Cited on page 43.

NEHRING, S. S. M. *Lecture notes in Mine Planning 1 - Strategy*. Edumine, 2016. Disponível em: <http://www.edumine.com/courses/online-courses/mine-planning-1-strategy/>. Acesso em: 04 August 2019. Cited on page 25.

NUMPY. *NumPy*. 2020. Disponível em: <https://numpy.org/>. Cited on page 74.

OR-TOOLS, G. *Route. Schedule. Plan. Assign. Pack. Solve. OR-Tools is fast and portable software for combinatorial optimization*. 2019. Disponível em: <https://developers.google.com/optimization/>. Cited 3 times on pages 36, 62, and 75.

OTA, L. A. M. R. R. M. Simsched direct block scheduler: A new practical algorithm for the open pit mine production scheduling problem. *APCOM 2017*, 2017. Cited on page 56.

PANDAS. *Pandas - Python Data Analysis Library*. 2020. Disponível em: <https://pandas.pydata.org/>. Cited on page 74.

PINEDO, M. *Scheduling: Theory, Algorithms, and Systems*. [S.l.]: Springer International Publishing, 2016. ISBN 9783319265803. Cited 7 times on pages 14, 34, 35, 37, 38, 40, and 42.

PLOTLY. *Plotly graphing libraries*. 2020. Disponível em: <https://plot.ly/python/>. Cited on page 75.

PYTHON. *Python*. 2020. Disponível em: <https://www.python.org/>. Cited on page 74.

SABANOV, S.; BEARE, M. Open pit scheduling features to improve project economy. *The Southern African Institute of Mining and Metallurgy*, p. 1033–1040, 2015. Disponível em: <http://www.srk.kz/files/1033-1040_MPES141_Sabanov.pdf>. Cited on page 18.

SCHULZE J. RIECK, C. S. J. Z. M. Machine scheduling in underground mining: an application in the potash industry. *OR Spectrum Quantitative Approaches in Management*, v. 38, n. 2, p. 365–403, 2016. Cited 2 times on pages 40 and 56.

SOT. *SOT - The Schedule Optimization Tool*. 2019. Disponível em: <https://www.revolutionmining.com/products/sot-v3-0>. Acesso em: 12 Oct 2019. Cited on page 56.

TAHA, H. A. *Pesquisa Operacional*. 8^aed. São Paulo: Pearson (Prentice Hall), 2007. Cited on page 13.

UNIVERSITY, Q. *Cut-off grade estimation*. 2019. Disponível em: <https://minewiki.engineering.queensu.ca/mediawiki/index.php/Cut-off_grade_estimation>. Cited on page 28.

VICKERS, E. L. Utilization of operations research in the mining industry. *Society of mining engineers of AIME*, 1962. Cited on page 13.

WANG, H. L. Underground mine planning optimization process to improve values and reduce risks. *Mining goes Digital: Proceedings of the 39th International Symposium 'Application of Computers and Operations Research in the Mineral Industry' (APCOM 2019), June 4-6, 2019, Wroclaw, Poland*, p. 335–343, 2019. Disponível em: <https://play.google.com/store/books/details?id=rqWaDwAAQBAJ&pcampaignid=books_web_aboutlink>. Cited on page 30.

WINSTON, W. L. *Operations Research*. 4^aed. United States: Brooks/Cole (Thomson), 2004. Cited on page 13.