Genetic Algorithm Framework for Stochastic Open Pit Production Scheduling in the Presence of Grade Uncertainty

Shadrach Yaw Amponsah, Pawoumodom Matthias Takouda and Eugene Ben-Awuah
Mining Optimization Laboratory (MOL)
Laurentian University, Sudbury, Canada

ABSTRACT

In the optimization of open pit production scheduling, the major challenge found in literature is finding the balance between optimality and computational time. Mathematical programming models such as Mixed Integer Linear Programming (MILP) are capable of attaining optimal solutions. However, this comes at the expense of computational time for tractable optimization problems. Evolutionary algorithms such as Genetic Algorithm (GA) are able to generate good solutions at shorter computational time. In this research, we present an evolutionary algorithm framework based on GA to solve the stochastic open pit production scheduling problem in the presence of grade uncertainty. For implementation, a set of equally probable simulated orebodies generated through Sequential Gaussian Simulation are used as input to the stochastic optimization model. Two case studies are presented which compares results from a stochastic GA with a stochastic MILP model. For Case study 2, while the SMILP model was at a gap of 101% after 28 days, the SGA model generated NPV of $10,045M at 10.6% gap after 1.5 hours

1. Introduction

Genetic algorithm is a metaheuristic evolutionary algorithm that has widely been studied and applied in combinatorial optimization problems in the areas of Finance, Supply Chain Management, and Information Technology. Some researchers have investigated its application to mining-related optimization spanning across different aspects of mine planning optimization. Denby and Schofield [26]; Myburgh and Deb [77]; Alipour et al. [4]; Paithankar and Chatterjee [81]; Alipour et al. [5] used genetic algorithm for open pit production scheduling optimization. Ahmadi and Shahabi [2] also used genetic algorithm for cut-off grade optimization while Ruiseco et al. [83] used genetic algorithm in ore-waste pit limit optimization. Similarly, Franco-Sepúlveda et al. [37] used genetic algorithm for the optimization of open-pit mining operations with geological and market uncertainty.

Mine planning optimization is a complex yet necessary combinatorial optimization problem. Combinatorial optimization problems are problems whose optimal solution must be obtained from a finite number of possibilities [7]. Mine planning optimization specifies the source, destination, time, and extraction sequence of mineral resources that maximizes the net present value (NPV) of a mining operation. A resultant activity of mine planning is the production scheduling of the mineral resource from the mine. Production scheduling can be carried out on either an open pit or an underground mine with various physical and technical constraints. Production scheduling can also be classified into short term, medium term or long term. Caccetta and Hill [18] described the time and sequence of extracting blocks from an open pit mine in order to maximize the NPV of the

A modified version of this paper has been submitted to the International Journal of Mining, Reclamation and Environment

ISBN: 978-1-55195-486-8

mining project as open pit production scheduling (OPPS) optimization. Alipour et al. [4] referred to the OPPS problem as a combinatorial optimization problem in the class of Non-deterministic polynomial-time (NP) hardness. An optimization problem is said to be NP-hard if the algorithm for solving it can be converted to one for solving any NP problem. NP-hard therefore means "at least as hard as any NP-problem," although it might, in fact, be harder [97]. As the OPPS optimization problem gets larger and the number of integer variables increase, finding an optimal solution to the problem in some instances becomes intractable or uses too much computing resources when an exact solution methodology is implemented. An optimization problem is tractable if a solution is obtained in polynomial time. This solution may or may not be optimal. Setting an optimality gap for the optimization problem can ensure tractability for exact algorithms. An optimality gap, therefore, refers to the difference between the best known solution (best integer) and the value that bounds the best possible solution [55]. An optimal solution in the case of exact algorithms is a solution with a 0% optimality gap. This demonstrates that the solution is the best that exists because the difference between the best integer and the best bound is 0%.

This research focuses on using a metaheuristic optimization approach in the field of evolutionary algorithms to tackle the NP-hard large scale OPPS problem. Bianchi et al. [16] define metaheuristics as algorithms that incorporate or develop heuristics (heuristics are simple approximate algorithms that look for good solutions in a solution space) to solve an optimization problem in a generic framework. Metaheuristics are thus higher level than heuristics, as the term "meta" in metaheuristics implies. The concept of metaheuristics is mostly inspired by natural biology.

The conventional approach to orebody modelling based on Ordinary Kriging [62] generates a single interpolated block model for pit limit and production scheduling optimization. In using this single interpolated block model for production scheduling, geological uncertainties which are inevitable in a typical mining project are not taken into consideration. This may result in schedules that either 1) overestimate or 2) underestimate the true representation of the optimal solution. Researchers such as Dimitrakopoulos and Ramazan [30]; Sabour and Dimitrakopoulos [84] and more recently Mbadozie [70] have investigated the incorporation of grade uncertainty in the OPPS problem formulation using simulation and mathematical programming. Multiple realizations of the block model are generated with Sequential Gaussian Simulation (SGS) and used as input to the mathematical programming model. The primary limitation of their implementation relate to generating tractable solutions for large scale optimization problems in a reasonable run time. The optimization solution run time is directly related to the problem size which is also a function of the size of the deposit, the number of simulation realizations, and the life-of-mine.

In this research, a metaheuristic optimization framework based on genetic algorithm has been designed and implemented for a large scale OPPS problem. A real number chromosome encoding technique is used in the genetic algorithm initial population to make possible partial block processing. Two variations of the GA framework referred to as Deterministic Genetic Algorithm (DGA) and Stochastic Genetic Algorithm (SGA) were implemented. DGA basically refers to the application of the GA framework in a conventional approach to production scheduling using an OK block model, while SGA refers to the application of the GA framework in a stochastic approach to production scheduling using SGS block model realizations. The conventional approach to the OPPS problem which does not consider grade uncertainty is investigated with a Mixed Integer Linear Programming (MILP) model with CPLEX and the DGA framework. The stochastic formulation of the OPPS problem that incorporates grade uncertainty is also considered and the resulting problem is optimized with a Stochastic Mixed Integer Linear Programming (SMILP) model with CPLEX and the SGA framework. The NPV and solution time for the conventional and stochastic frameworks are compared.

2. Summary of Literature Review

Open pit production scheduling (OPPS) problems can be defined as specifying the time and sequence in which blocks should be extracted from the mine in order to maximize the NPV subject to a variety of physical, environmental, operational and economic constraints [18]. Production scheduling of an open pit mine is a major concern in mine planning and a complex optimization problem. Usually, the planning of an open pit mine starts with finding the ultimate digging or pit limit. This pit limit provides the list of blocks to be considered for production scheduling. The main algorithm used in the literature to find the ultimate pit limit is the Lerch Grossman (LG) algorithm [66]. Once the final pit is determined, the production scheduling process can commence. Researchers in their bid to solve the OPPS problem have formulated mathematical models with different optimization techniques [9; 18; 61]. These models take the form of an objective function for maximizing the NPV subject to the set constraints. There are two major research areas in the development of production scheduling algorithms: (1) Deterministic algorithms and (2) Heuristics and Metaheuristic optimization algorithms.

Johnson [57] introduced Linear Programming (LP) for OPPS problems. The author did not obtain an optimal schedule for the problem due to the computational intractability. LP however proved to be a capable option for modelling the OPPS problem. Gershon [43]; Dagdelen [21] presented a MILP model which was an improvement of the LP model by Johnson. Formulating the model with MILP allowed for some of the decision variables to be presented as continuous variables to prevent infeasibility and allow for partial block processing. The model however could not obtain an optimum schedule for a real-size large scale OPPS problem. Caccetta and Hill [18] proposed a MILP model solved with branch and cut algorithm for the OPPS problem. The authors used a cutting plane algorithm and a search strategy involving best first and depth first search to achieve a ‘‘good spread’’ of possible pit schedules. Their approach was capable of solving the OPPS problem on a small and medium scale optimization problem with 6,720 to 209,664 blocks. However, the optimization was computationally expensive. Due to commercialization and confidentiality agreements, the authors did not provide detailed information about their work. Dimitrakopoulos and Ramazan [29] also proposed a MILP model for solving the OPPS problem. In their approach, they presented waste blocks as continuous variables in order to reduce the number of integer variables and improve the solution time.

Integer programming has also been studied and applied to the OPPS problem. Dagdelen and Johnson [22] formulated the OPPS problem with integer programming and solved it using the Lagrangian relaxation method. Lagrangian relaxation is a method used to reduce the complexity of the optimization problem by relaxing one or more constraints. A penalty term and a multiplier known as a Lagrangian multiplier used in the relaxed constraint is then added to the objective function. This is done to avoid violations of the relaxed constraint [11]. In Dagdelen and Johnson [22] formulation, the mining and processing constraints were relaxed and adjusted with Lagrangian multipliers to find the optimal solution. The authors decomposed the problem into smaller problems to allow for tractability of the solution. Ben-Awuah et al. [15] implemented a MILP model that incorporates goal programming; a reward and penalty based approach to maximize the NPV. The authors used the clustering algorithm developed by Tabesh and Askari-Nasab [92] to reduce the size of the optimization problem to ensure computational tractability. Their case study involved 16,985 blocks, and their model was able to find the optimal solution at 0% optimality gap.

Heuristics are basic approximation algorithms that search the solution space to find a good solution and metaheuristics are algorithms that combine heuristics (that are usually very problem-specific) in a more general framework [16]. Metaheuristics are able to solve large optimization problems at a reasonable computational time. The difficulties associated with NP-hard class of problems and the general large instances of the problems make exact approaches that often generate optimal solution not ideal to solve such problems; taking into account, the resources and

computational time required. Researchers have investigated the trade-off between finding a good solution at smaller computational time and finding an optimal solution, which at times is intractable or resource intensive. The uncertainties associated with stochastic OPPS optimization problems make the application of metaheuristics more ideal in applying it to large problem instances, since it is capable of finding a good solution in a much smaller computation time. Popular metaheuristic algorithms for large scale optimization includes: tabu search, genetic algorithm, simulated annealing, ant colony optimization, and particle swarm optimization.

Tabu search (TS) is a metaheuristic algorithm used to solve large combinatorial optimization problems. The process of TS was designed by Glover [44]. It optimizes or improves a solution by searching through a neighbourhood of solutions and selecting the best ones. TS classifies certain solutions as forbidden (taboo; where the name 'tabu' is derived from) to prevent the algorithm from selecting those solutions which is a strategy to avoid cycling of the algorithm. There are three TS specific concepts that improves it solution approach over other combinatorial optimization algorithms according to Bianchi et al.[16]. These concepts are: best improvement, tabu lists, and aspiration criteria. Best improvement is an approach in TS algorithm in which each existing solution is replaced with its best neighbouring solution. This is a method for avoiding local optima, however it may result in cycling when each current solution is replaced. TS counteracts this problem by creating a tabu list. Tabu list is a list that stores attributes of recently visited solutions. The type of attribute saved is the 'move' made by the algorithm in arriving at a result. The algorithm is then restricted from selecting from this set of solutions with attributes on the tabu list. The aspiration criterion is a criterion check in the TS algorithm that, if met, permits a 'move' to a banned solution to be chosen. Such criterion, according to Glover [45], can be set as follows: if the existing solution is worse than the newer one, then the tabu can be overridden. TS has been explored by researchers to solve the OPPS problem. Lamghari and Dimitrakopoulos [65] used TS and Variable Neighborhood Descent (VND) algorithms for solving the OPPS problem. In their implementation, a definite number of neighborhood was set and the algorithm was made to search the neighborhood until the optimal solution was found. The authors implemented their model on a copper and gold dataset to validate the effectiveness of the proposed model. In their conclusion, the authors stated that a near optimal solution was found at a reasonable computational time. Senécal and Dimitrakopoulos [86] presented a TS that uses multi-neighborhood to solve the long term OPPS problem. In their approach, the authors considered multiple processing streams under mineral uncertainty. The objective function from their model maximizes the discounted cash flow and penalizes deviations from production targets.

Alipour et al. [4] presented a Genetic Algorithm (GA) approach for the OPPS problem. The researchers used an initial population of 20 with each population consisting of a 10 x 20 matrix in the GA model which represents the total blocks in the copper orebody. The initial population was then normalized to cater for the various constraints associated with the OPPS problem, namely; the mining capacity, processing capacity and block precedence constraints. A fitness function to evaluate the population was also formulated. The OPPS problem was solved with the GA and the results were compared to solution from IBM CPLEX solver. The authors indicated that, the computation time needed to solve the optimization problem with GA was significantly lower than that of the IBM CPLEX solver. However, the optimal solution from the GA was 5% lesser than that from the IBM CPLEX solver. The researchers further emphasized GA as computationally efficient but does not always give the optimal solution compared to LP and MILP with IBM CPLEX solver.

Another application of GA to the OPPS problem was presented by Alipour et al. [5]. In this application, the authors built on their earlier research in 2017. A 3D orebody model consisting of 10,529 blocks was used for a case study. The authors compared the GA solution to the solution from SimSched DBS software [80] developed based on surface constrained stochastic life-of-mine production scheduling by Marinho de Almeida [68]. From the analysis by the authors, GA proved capable of solving not just a 2D OPPS problem but also a 3D OPPS problem. SimSched DBS

obtained its solution in a shorter computation time than the GA. This was because SimSched DBS software mined blocks accumulated as surfaces which reduces the number of decision variables and level of selectivity for processing. The optimal solution from the GA was 4% better than that from the SimSched DBS software. The authors concluded at the time that, due to the size of the problem, any comparison to an exact optimization methodology was not possible. This further emphasizes the use of metaheuristics in attempting the OPPS problem. Grade uncertainties were not considered in their research. The researchers through the case study demonstrated the viability of using a GA model to solve OPPS optimization problems.

Amponsah et al. [6] also presented a GA model to solve a small-scale 3D OPPS problem. The researchers used a literal permutation encoding scheme from Gen et al. [41] for chromosomes encoding. They compared their results to a MILP solution from CPLEX. In the authors' findings, the GA model's solution was within 10% of the MILP solution with CPLEX. This was partly because the MILP allowed for fractional block processing across multiple periods, which the GA did not. In extending Amponsah et al. [6] research, a GA framework that allows for fractional block processing across multiple periods is presented in this research. The extended GA model, also considers grade uncertainties in its formulation and optimization.

Kirkpatrick et al. [60] proposed simulated annealing (SA) as a combinatorial optimization algorithm. SA optimization algorithm in principle is based on local search heuristics, and uses a pre-defined neighborhood structure on the search space. In the OPPS problem, Kumral and Dowd [63] proposed a SA algorithm approach to solve the problem. The authors simulated the mineral deposit with Sequential Gaussian Simulation and created the block model. The ultimate pit limit was then determined from the block model by pit-blend using LP and the LG algorithm. Their model contained a total of 2,773 blocks, SA was able to provide a suitable and uniform mine schedule in a relatively short computational time. The authors however; used lagrangian parameterization to incorporate the constraints of the optimization into the objective function which created sub-pits within the ultimate pit to allow for the satisfaction of the tonnage capacity constraints. Goodfellow and Dimitrakopoulos [50] also presented a SA optimization approach to the OPPS problem. Their model used a stochastic push-back design to adjust and minimize the deviation of materials sent to the waste and processing destinations. A copper deposit was used as the case study by the researchers and found that SA to be capable of handling real world application since the algorithm efficiently handled the multiple metals, slope zones and the multiple destinations.

Ant colony optimization (ACO) is a population based metaheuristic algorithm. The concept that inspired ACO is based on the behavior ants display when plying a route in search of food [32]. Ants on their quest for food scan their nest in a random manner, and when a food source is found, the ant releases a chemical called pheromone on its way to the nest. This chemical serves as a way of communicating to other ants to ply the same route in search of food [27]. The route with the highest pheromone concentration tends to be the preferred route or the shortest. Subsequently, the pheromone evaporates as time goes on. In combinatorial optimization problems, Dorigo et al. [32] presented the ACO as an optimization algorithm for the Travelling Salesman Problem (TSP). An application of the ACO algorithm was proposed by Shishvan and Sattarvand [88] for the OPPS problem. In their implementation, the authors used the Max-Min ant colony system and tested the model on a copper-gold deposit. The deposit consisted of 350,000 blocks. The ACO algorithm generated a 12% improvement in the initial mining schedule. The authors carried out a sensitivity analysis on the ACO parameters consisting of initial pheromone values, pheromone evaporation rate, and perturbation distance. In the author's findings, they stated that a higher initial pheromone value reduced the algorithms chances of exploring better solutions thereby generating poor results. In the analysis of the evaporation rate, the authors concluded that lower evaporation rate increases the time spent by the algorithm on poor solutions.

Particle swam optimization (PSO) is a nature inspired metaheuristic optimization which was first proposed by Kennedy and Eberhart [58]. PSO is based on the social interaction of individuals living together in groups. PSO algorithm performs the search process by using a population of individuals living in groups [59]. Khan and Niemann-Delius [59] implemented the PSO on a OPPS problem, the authors used a continuous version of the PSO and a guaranteed convergence PSO algorithm. The authors' inspiration for this approach was the number of blocks available in the ultimate pit limit of an open pit mine which may contain hundreds of thousands of blocks therefore according to the authors using a continuous PSO reduced the computational time. The precedence constraints in the OPPS problem was handled by normalization in the model, the model checks and repair each solution to ensure solution feasibility at all times. The other constraints were handled by the application of a penalty method. The proposed model was implemented on two copper orebodies with 10,120 and 7,863 blocks respectively. The model was successful in solving the OPPS problem with an optimality gap of 12.61% for the first case study and 4.80% for the second case study.

2.1. Stochastic open pit optimization in the presence of grade uncertainty

The conventional OPPS problem is optimized with a single interpolated orebody block model which does not account for grade uncertainties. As the conventional OPPS approach does not consider grade uncertainties, a true representation of the optimal NPV is rarely achieved. As reported by Sabour and Dimitrakopoulos [84], due to uncertainties associated with mining projects, the mining industry in Canada lost in excess of 1.4 billion dollars in 1991. In the incorporation of uncertainties in the OPPS problem, the stochastic model takes several simulated orebody realizations as input with each orebody model having varying grades. The stochastic model then seeks to optimize for the maximum NPV and minimum waste management cost, while providing risk-based solution that tends to minimize deviations from operational targets.

Dimitrakopoulos and Ramazan [30] introduced a stochastic integer programming (SIP) formulation that considered grade uncertainty. The authors elaborated that the SIP model considers multiple scenarios and generate a desirable outcome for a set of specified objectives which made its application to the OPPS problem preferable. The authors implemented their SIP model on two case studies: a gold deposit and a copper deposit. The case study with the gold deposit had 22,296 blocks. In the analysis of the results by the authors, the gold deposit case study was optimized in two stages with both optimizations totaling 42 hours in computational time with 14 simulated orebody realizations. The authors indicated that the SIP model with the simulated orebody realizations had a 9.7% increase in NPV over the traditional (conventional) mixed integer programming (MIP) model with a single interpolated orebody block model. There was a similar outcome from the copper case study. The number of simulated orebody realizations for this case study was 20 and the authors recorded an increase in NPV of ~ 25% over the traditional MIP model's NPV. Mbadozie et al. [71] {Mbadozie, 2020 #214}also presented a stochastic mixed integer linear programming (SMILP) formulation for oil sands production scheduling and waste management that considers grade uncertainty. The author used 20 orebody realizations to represent grade variability in the deposit. The results from the oil sands case study demonstrated that the SMILP schedule generated 16.85% improvements in NPV over the conventional schedule. These promising gains in NPV from stochastic production schedules form the basis of this research.

3. Genetic Algorithm

Genetic algorithm (GA) is an evolutionary algorithm that follows biological processes as proposed by Darwin [48; 52]. GA, therefore, generates its solution to the optimization problem by strictly following the biological evolution process; such as inheritance, crossover, mutation, and selection. The inherent theory in this process is the survival of the fittest where organisms with good or fitter genes survive and transfer their genes to the next generation. Consequently, only organisms with

the best gene will exist over time. GA follows the same approach when formulating problems. In summary, the GA workflow includes the following: 1) An initial population of individuals is created; the fitness functions of the created individuals are evaluated. 2) A set of genes and chromosomes are selected based on the fittest individuals; the selected genes will then crossover and mutate. 3) Elitism is then applied on the best individuals in the population to keep them for the next generation. 4) This process is repeated until a population of the best genes are obtained or a set of maximum generations are reached. Fig. 1 shows a flow chart of the genetic algorithm process.

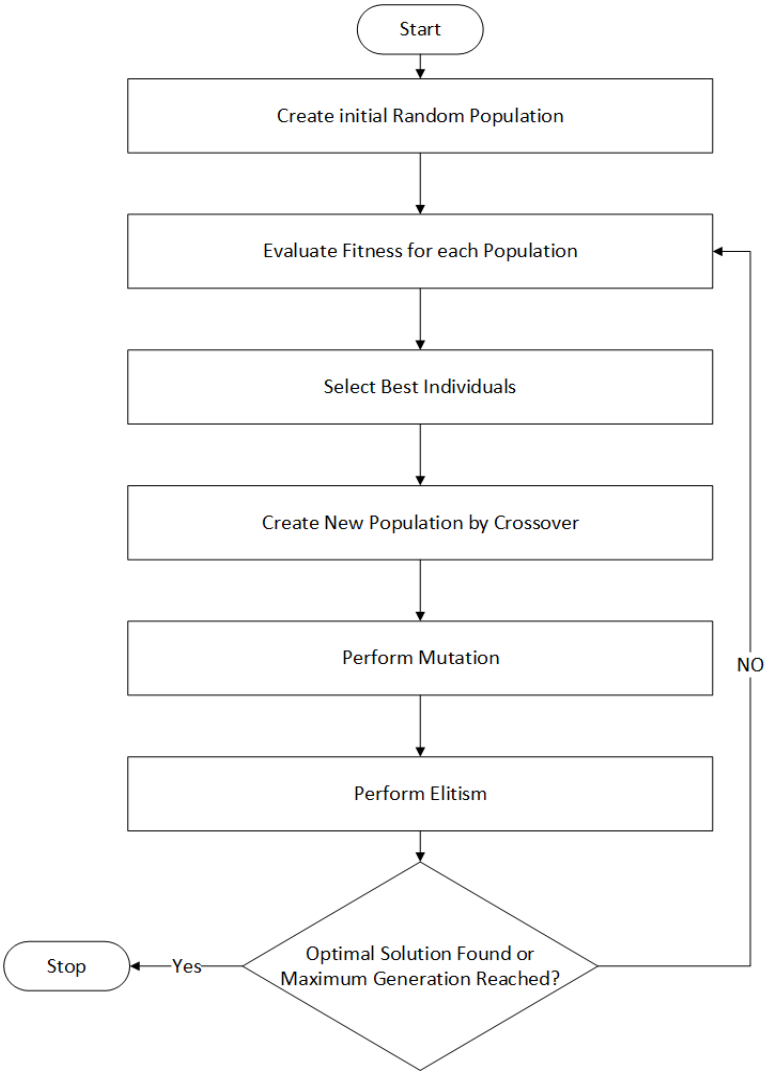

Figure 1. Flow chart of genetic algorithm process.

3.1. Initial population

This is the starting point of the GA where the population is initialized. According to Gen et al. [40], there are two methods for generating initial population; the heuristic method and the random number generation method. In the random number generation method, GA randomly generates a solution space based on the problem size and this is referred to as the initial population. This initial population can be generated from a Gaussian distribution to increase diversity. The population includes multiple solutions, which represents chromosomes and genes of individuals. Each chromosome has a set of variables which simulates the genes [75]. In the heuristic approach, a problem specific encoding algorithm is used to generate the initial population. Gen et al. [40] however illustrated that this approach sometimes explore only a smaller portion of the solution

space in a large scale combinatorial optimization problem. This then leads to a local optimum in the GA. In this research, the random number generation method was used. Fig. 2 shows a sample initial population (Chromosomes 1, 2 and 3) obtained by the random number generation method. This figure shows the flexibility in setting up GA optimization problems as the initial population can be represented in different ways and with different characters.

Chromosome ***1***	0	0	1	0	1	0	1	0	0	1	1
Chromosome ***2***	3	5	7	4	8	1	8	12	56	4	7
Chromosome ***3***	0.3	0.2	0.6	0.05	0.6	0.52	0.41	0.3	0.3	0.54	0.8
~											
Chromosome ***n***	A	B	H	J	I	L	K	M	G	H	V

Chromosome / Solution

Genes

Figure 2. Sample initial population showing genes and chromosomes.

3.2. Chromosome encoding

Chromosome encoding is an essential process in GA. It specifies the nature of the genotype in a population. The encoding scheme is mostly problem dependent and thus relies on the structure of the problem being optimized. Binary encoding, real number encoding, literal permutation encoding (LPE), and general data structure encoding are the various classifications of chromosome encoding according to Gen et al. [40]. In binary encoding, the genes in the population are represented by either 0 or 1. The encoded genes are then decoded to decimals when evaluating for their fitness. This process is done for every gene in the chromosome and may pose performance issues for large number of genes. This encoding forms the genotype of a feasible solution to the problem. An example of a problem that benefits from the binary encoding scheme is the general knapsack problem. In the knapsack problem, the objective is to find the sum of weights producing the maximum profit or minimum cost to a problem while respecting the stipulated capacity of the knapsack. Given a set of n $\forall_n \in \{1,...n\}$ items each having a weight of w_i and a value of v_i with a maximum capacity of C, the knapsack problem can be modelled as in Eqs. and [36].

$$Max \sum_{i=1}^{n} v_i x_i$$

Subject to

$$\sum_{i=1}^{n} w_i x_i \leq C$$

Where x_i is the decision variable to this problem which can be encoded as 1 (if selected) or 0 (otherwise). When binary encoding is employed, it indicates that the problem assumes only discrete values, which is not always the case for many optimization problems.

Real number encoding or continuous variable encoding is the representation of the decision variables in the genotype with real numbers as opposed to binary encoding. In this process, there is no binary to decimal decoding and this improves the efficiency of the approach. According to Gen et al. [40], real number encoding is suitable for functional and constrained optimization problems. The genotypic representation in real number encoding is close to the phenotypic space of the problem since there is no conversion between both spaces. To represent genes with this encoding

scheme, genes have to be between a lower and upper bound of the decision variable. This is represented in Eq. .

$$x_n = (x_{ub} - x_{lb}) \times r + x_{lb}$$

Where x_n is the *n-th* gene; x_{ub} is the upper bound; x_{lb} is the lower bound; and *r* is a random number between [0, 1].

LPE or order encoding is the representation of the gene by the permutation of the decision variables. Since LPE is represented as the permutation of the decision variables, it is mostly suitable for optimization problems that involve permutation. An example is the travelling salesman problem. In the travelling salesman problem, a salesman has to visit *n* number of cities and every city can be visited only once. In such a problem, the genotype can be encoded as the order or sequence in which the cities are visited which is the permutation of the *n* cities. Fig. 3 shows a sample LPE. Chromosome A and B in Fig. 3 shows the sequence in which the cities can be visited by the travelling salesman as represented by the GA.

Chromosome A	1	4	2	3	6	7	8	10	12	14	15	17	9	11	13	16	5

Chromosome B	2	4	3	5	7	6	8	10	15	14	16	12	11	13	17	1	9

Figure 3. Literal permutation encoding representation.

Every combinatorial optimization problem that is optimized with GA requires its own chromosome encoding technique that represents the problem in great detail. There are no one size fits all chromosome encoding although certain type of combinatorial optimization problems do benefit from specific encoding schemes. Once a suitable encoding scheme is modeled for the problem, the various genetic operators can then be formulated around the specific chromosome encoded. The phenotype of the population is then derived from the genotype representation in the encoded chromosomes. In this research, multiple chromosome encoding was used to represent the genes in the population. A real number encoding or continuous encoding technique was used to represent the genotype of each block in the population. This allowed for fractions of blocks in the population to be processed. In addition, the LPE technique was also used as part of the encoding scheme to provide the order or sequence in which blocks could be mined. More on the problem specific chromosome encoding techniques used in this research are discussed in Section 4.

3.3. Fitness function

In GA, the fitness function is the function used to determine the viability of a gene in a population. The objective function in an optimization problem in GA is referred to as the fitness function. This function tests the population at every generation and becomes the means of determining fitter genes that survives to the next generation. The fitness function also aides in the selection of parents from the population as selection algorithms in GA ranks population based on their fitness value. When evaluating the fitness function, the genotype from the population is converted or decoded into phenotype. This phenotype is evaluated and assigned a value which becomes the fitness of that solution. The decoding of the gene is dependent on the chromosome encoding scheme used during the initialization of the problem. At the end of every generation, the fitness value of each member of the population is assessed and ranked based on the objective of the optimization problem. If the objective of the optimization is to minimize cost, then the member solution with the least minimum fitness value is chosen as the best solution from that generation. If the objective of the optimization is to maximize profit, then the member solution with the maximum fitness value is selected. The

fitness function and its evaluation ensures that the GA does not violate the general objective of the problem.

3.4. Selection

GA uses different selection algorithms with the inspiration of attaining the fittest individual from the initial population. Sharma and Gargi [87]; Sivanandam and Deepa [88] defined selection in GA as a method that randomly picks chromosomes out of the population according to their fitness function value. The higher the fitness function value, the better chance that an individual will be selected. There are several popular selection algorithms but there is no one preferred selection algorithm for GA. Each selection algorithm may have advantage over the other based on the specific problem being optimized. Various selection algorithms found in the literature are: Boltzmann selection [46], Roulette wheel selection also known as the Fitness proportionate selection algorithm [19; 47], Tournament selection [17; 47], Random selection, Rank selection, and Stochastic universal sampling [89]. Roulette wheel selection and Tournament selection are explored for this research as they are the two well studied selection algorithms in GA [64].

3.5. Crossover

Crossover is the method of selecting two parents at random and recombining their chromosomes at a point with the intent of making offspring with better genes [90]. Single point crossover and double point crossover are the two main approaches to crossover. Fig. 4 shows a single and Fig. 5 shows a double point crossover. However there are several other crossover approaches used in the literature: uniform crossover [85], three parents' crossover [94], half uniform crossover [54], partially matched crossover [49], position-based crossover [91], order crossover [24], cycle crossover [79], multi-point crossover [34], masked crossover [67], and heuristic crossover [51]. In this research, the double point crossover was implemented. Double point crossover was used since it has a high capacity to transmit useful genetic information from parent to offspring based on the study by [35].

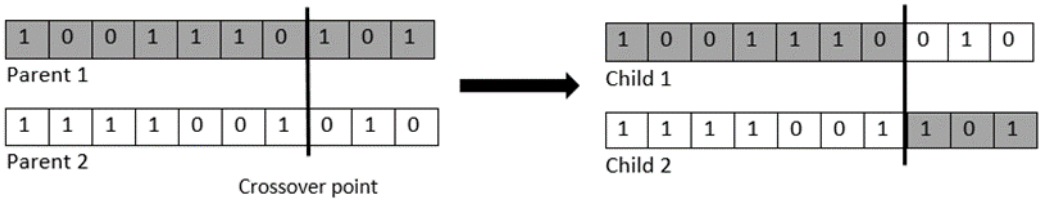

Figure 4. Single point crossover.

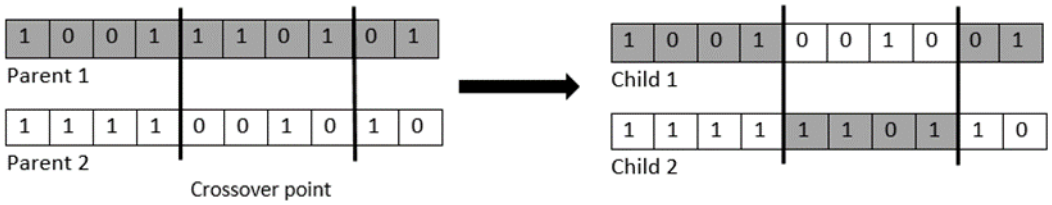

Figure . Double point crossover.

3.6. Mutation

Mutation is the next stage after crossover in the GA algorithm process. When offspring from two parents are generated through crossover, it may occur that these offspring do not possess good enough genes to generate a good solution. Therefore, the GA process introduces mutation to alter or change the genes. Mutation is the other way to get new genomes. Mutation results in changing the value of genes [90]. These changes occur randomly with a probability of mutation parameter set between [0, 1]. A random number in the same interval is generated for each gene in the new child.

If this random number is less than the probability of mutation, the gene is assigned with a random number within the lower and upper bounds of the decision variable [76]. In Fig. 6 is an illustration of before and after a mutation process where a new gene is introduced.

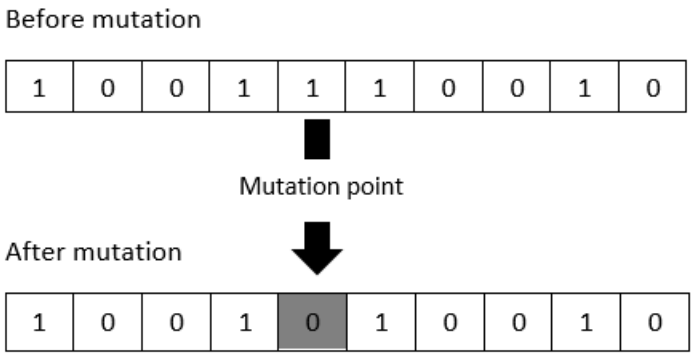

Figure 6. Sample mutation.

3.7. Elitism

Elitism is a genetic operator applied to the chromosomes obtained after selection, crossover and mutation in some cases to preserve or copy the traits of the best chromosomes to the next generation [3; 33]. This helps to keep a fit chromosome in each generation at all times by ensuring that these fit chromosomes are not lost during the iteration process. Elitism as a genetic operator was not part of the initial operators during the theoretical formulation of GA but has through the years proven to be efficient when applied as a genetic operator [76; 3].

3.8. Constraint handling

Evolution algorithms have different methods for handling constraints. These methods are generally problem dependent although some may be applied across different optimization problems. In the literature, the methods for handling constraints can be grouped into four different categories; the repair method, rejection method, penalty method and modification of genetic parameters [40; 72]. Each method has merits that may suit a particular combinatorial optimization problem over the other.

In the repair method, population with infeasible chromosome is neither discarded nor penalized but rather a deterministic method for normalizing the infeasible chromosome is applied. This converts the infeasible chromosome to a feasible one [73]. The method for normalizing or repairing the chromosome must consider the bounds of the constraints and create a chromosome that lies in the feasible region. This method is problem dependent and cannot be applied to any problem without first re-writing the repairing algorithm to suit the said problem. Two approaches of this method exists: (1) Always replacing the infeasible chromosome in the population with the repaired chromosome, and (2) using the repaired chromosome only for evaluation purposes without feeding it into the evolution. Both approaches are used in the literature. Nakano and Yamada [78] used the always replacing approach and termed it as "forcing" where a feasible chromosome *g'* repaired from an infeasible chromosome *g* is forced to replace the chromosome *g* in the population.

The rejection method also termed as "death penalty" by Michalewicz [73] works by completely removing any infeasible chromosome from the population. In this approach, any infeasible chromosome in the population is discarded as opposed to being repaired. This approach has limitations. The initial population generated for a problem may have several infeasible solutions and per this method all these infeasible solutions need to be discarded which may lead the GA into premature convergence. Michalewicz [72] tested this method on five different cases and stated that it performed worse than the other constraints handling approaches. Outright rejection of infeasible solutions go against the nature of evolution algorithms [82].

The penalty method is widely used and by far the easiest to implement. Constrained optimization problems are converted to unconstrained problems by applying a penalty function to the objective function of the problem. According to Dasgupta and Michalewicz [23], the basic approach is to extend the objective function which in GA is represented as the fitness function of a Chromosome *i* in the following Equation;

$$fitness\ function = f(i) \pm Q(i)$$

Where $Q(i)$ represents the penalty for an infeasible Chromosome *i*, and $f(i)$ represents the objective function of the problem. For a maximization problem, the penalty function is expressed as $Q(i) < 0$ where *i* is infeasible and $Q(i) = 0$ where *i* is feasible; whereas $Q(i) > 0$ where *i* is infeasible and $Q(i) = 0$ where *i* is feasible for a minimization problem. The general challenge with penalty functions as stated by Michalewicz [72]; Richardson et al. [82] is knowing exactly what degree of penalty to apply to an infeasible chromosome or solution since all infeasible solutions are not alike. Fig. 7 illustrates feasible and infeasible solution regions in a solution space. Assuming Solution *x* is the optimal solution without any prior knowledge, Solution *c* is closer to the optimal solution than Solution *b* and Solution *a,* although these solutions are in the infeasible region. Solution *c* may contain certain genes that may be relevant to attaining the optimal solution as opposed to Solution *b* and Solution *a*. Therefore, applying the same penalty value in this instance may not be ideal. Secondly, Solution *y* is farther than Solution *c* relative to the optimal Solution *x* although Solution *y* is in the feasible region. These complexities make finding the appropriate penalty value challenging.

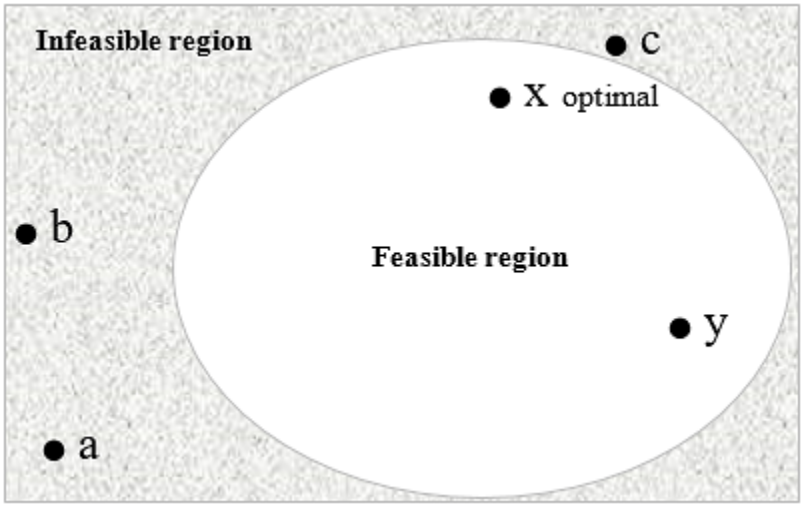

Figure 7. Feasible and infeasible solution space modified after Michalewicz [73].

The modification of genetic parameters approach works by creating problem specific methods that (1) represent the problem, and (2) modifies the conventional GA parameters such as crossover and mutation to keep the optimization problem in the feasible domain [40]. The approach always ensure the GA is kept only in the feasible search space at all times. Although this is desirable, it also limits the search space for the GA. In this research, both the repair and penalty methods were implemented in handling constraints.

3.9. Termination

Generally, GA terminates when the maximum number of generations are reached. GA can also be terminated when a desired fitness value is met or when subsequent iterations does not improve the solution quality.

4. Open Pit Production Scheduling (OPPS) Optimization Framework

4.1. Model formulation for open pit production scheduling

The NPV of OPPS is based on the economic block value (EBV) of individual blocks in the orebody block model. The EBV of a block depends on its value and the costs incurred in mining and processing the block. The cost of mining a block is a function of the block's location in relation to how deep the block is from the surface and how far it is to its final destination. To calculate the NPV, the EBV is discounted since OPPS is undertaken over multiple periods. The discounted profit from block *n* is therefore given as the discounted revenue generated from mining block *n* minus the discounted cost for extracting and processing block *n*. This is presented in Eq..

$$discounted\ \ profit_n^t = discounted\ revenue_n^t - discounted\ \cos t_n^t \quad \forall_n \in \{1,...N\}; \forall_t \in \{1,...T\};$$

4.1.1. Indices and set

$s \in \{1,....,S\}$	index for realizations
$n \in \{1,....,N\}$	index for blocks
$t \in \{1,....,T\}$	index for scheduling periods
$N = \{1,...,N\}$	set of all blocks in the model
$S = \{1,...,S\}$	set of all equally probable orebody realizations
$H_n(D)$	For each block, there is a set $H_n(D)$ defining the immediate predecessor blocks that must be extracted prior to extracting block *n* with safe slopes; where *D* is the total number of blocks in $H_n(D)$

4.1.2. Parameters

r	discount rate
o_n	ore tonnage in block *n*
$o_{n,s}$	ore tonnage in block *n* of realization *s*
w_n	waste tonnage in block *n*

$w_{n,s}$	waste tonnage in block n of realization s
dr	geological discount rate
v_n^t	revenue obtained by selling the final product within block n in period t, minus the extra discounted cost of mining all the material in block n as ore
$v_{n,s}^t$	revenue obtained by selling the final product within block n of realization s in period t, minus the extra discounted cost of mining all the material in block n as ore
q_n^t	cost of mining all the materials in block n in period t as waste
$q_{n,s}^t$	cost of mining all the materials in block n of realization s in period t as waste
Cl^t	lower bound of the mining capacity in period t
Cu^t	upper bound of the mining capacity in period t
Ql^t	lower bound of the processing capacity in period t
Qu^t	upper bound of the processing capacity in period t
g_n	average grade in ore portion of block n
$g_{n,s}$	average grade in ore portion of block n for realization s
$\underline{g}^t$	lower bound of the required average head grade in period t
$\overline{g}^t$	upper bound of the required average head grade in period t
pc_{o-}^t	penalty cost for lower ore tonnage target deviation in period t
pc_{o+}^t	penalty cost for upper ore tonnage target deviation in period t
pc_{g-}^t	penalty cost for lower grade target deviation in period t

pc^t_{g+} penalty cost for upper grade target deviation in period *t*

4.1.3. Decision variables

$x^t_n \in [0,1]$ continuous variable representing the portion of block *n* to be extracted as ore and processed in period *t*

$y^t_n \in [0,1]$ continuous variable representing the portion of the block *n* to be mined in period *t*; fraction of *y* characterizes both ore and waste in the block

$b^t_n \in \{0,1\}$ b^t_n binary integer variable controlling the precedence of extraction of mining block *n;* b^t_n equal to one if extraction has started in period *t,* otherwise it is zero

$od^t_{s,+} \in [0,1]$ continuous variable representing the excess from the ore tonnage upper bound in period *t* for realization *s*

$od^t_{s,-} \in [0,1]$ continuous variable representing the shortage to the ore tonnage lower bound in period *t* for realization *s*

$gd^t_{s,+} \in [0,1]$ continuous variable representing the excess from the grade upper bound in period *t* for realization *s*

$gd^t_{s,-} \in [0,1]$ continuous variable representing the shortage to the grade lower bound in period *t* for realization *s*

4.2. Deterministic MILP formulation

In the conventional formulation of the OPPS, grade uncertainties are not considered and the main objective is to maximize the NPV of the mining operation subject to a set of constraints. The objective function and constraints are outlined in Eqs. to . This MILP formulation is consistent with the research undertaken by Askari-Nasab et al. [10].

4.2.1. Objective function

The objective function of the MILP model (Eq.) is formulated to maximize the NPV of the mining operation. The objective function consists of two continuous decision variables for block *n*. The first decision variable x^t_n represents the portion of block *n* to be processed in period *t* if it is ore. The decision variable y^t_n represents the portion of block *n* to be extracted in period *t*; fraction of *y* characterizes both ore and waste in the block. Using continuous decision variables allows for the fractional extraction of blocks in different periods.

$$Max \sum_{t=1}^{T} \sum_{n=1}^{N} \left(\frac{v^t_n \times x^t_n - q^t_n \times y^t_n}{(1+r)^t} \right)$$

Subject to:

4.2.2. Mining capacity constraints

Eqs. and define the mining capacity constraints for each period. Eq. defines maximum capacity for mining. This ensures that the total amount of material mined is less or equal to the stipulated capacity of mining equipment while Eq. defines the minimum capacity and controls the minimum amount of materials mined. These constraints are controlled by the continuous decision variable y_n^t and allows the mine planner to use different mining capacities in each period throughout the life-of-mine.

$$\sum_{n=1}^{N}(o_n + w_n) \times y_n^t \leq Cu^t \qquad \forall_t \in \{1,...T\};$$

$$\sum_{n=1}^{N}(o_n + w_n) \times y_n^t \geq Cl^t \qquad \forall_t \in \{1,...T\};$$

4.2.3. Processing capacity constraints

The processing capacity constraints aids the mine planner in ensuring a consistent feed throughout the mine life, resulting in a mine-to-mill operation that is well integrated. This is a soft constraint and depends on the availability of ore blocks. The processing objective may not be met in some periods depending on the orebody's ore grade distribution. In such circumstances, pre-stripping might be considered to ensure a consistent mill feed. This effectively forces the optimizer to mine waste in the early stages so that when ore production begins, the plant feed supply will be consistent and uniform. Eqs. and define the processing capacity of the mining operation. Eq. sets the upper bound and Eq. sets the lower bound for the amount of ore processed. These constraints are controlled by the continuous decision variable x_n^t and allows the mine planner to provide a uniform mill feed throughout the life-of-mine. In practice, the processing targets must be set with minimal periodic deviations to ensure maximum utilization of the mill.

$$\sum_{n=1}^{N}(o_n \times x_n^t) \leq Qu^t \qquad \forall_t \in \{1,...T\};$$

$$\sum_{n=1}^{N}(o_n \times x_n^t) \geq Ql^t \qquad \forall_t \in \{1,...T\};$$

4.2.4. Grade blending constraints

The goal of blending in production scheduling is to mine in such a way that the ore materials fulfil the processing plant's quality and quantity specifications. The grade blending constraints are essential constraints during production scheduling. These constraints ensure that an acceptable range of ore is sent to the mill at all times. Therefore, this grade range should be set between a lower and upper limit to facilitate blending of mill feed material. Eq. defines the upper limit of the ore grade and Eq. defines the lower limit of the ore grade to be sent to the mill. These constraints are controlled by the continuous decision variable x_n^t.

$$\sum_{n=1}^{N}(g_n - \overline{g}^t) \times (o_n \times x_n^t) \leq 0 \qquad \forall_t \in \{1,...T\};$$

$$\sum_{n=1}^{N}(g_n - \underline{g}^t)\times(o_n \times x_n^t) \geq 0 \qquad \forall_t \in \{1,...T\};$$

4.2.5. Block precedence constraints

Eqs. to enforce the block extraction precedence constraints. Binary integer decision variable, b_n^t, is used to control the precedence of block extraction. b_n^t is equal to one if the extraction of mining blocks has started by or in period *t*; otherwise, it is zero. For each mining block *n,* Eq. checks the set of immediate predecessor blocks in $H_n(D)$ that must be mined prior to mining block *n.* Eq. checks that extraction of mining block *n* can start only when the mining block has not been previously extracted. Eq. ensures that once extraction of block *n* starts, this block is available for extraction in subsequent periods.

$$b_n^t - \sum_{i=1}^{t} y_d^i \leq 0 \quad d \in H_n(D) \qquad \forall_n \in \{1,...N\};\ \forall_t \in \{1,...T\};$$

$$\sum_{i=1}^{t} y_n^i - b_n^t \leq 0 \qquad \forall_n \in \{1,...N\};\ \forall_t \in \{1,...T\};$$

$$b_n^t - b_n^{t+1} \leq 0 \qquad \forall_n \in \{1,...N\};\ \forall_t \in \{1,...T-1\};$$

4.2.6. Variable control constraints

Eq. ensures that the total ore material mined in any given scheduling period is less or equal to the sum of the ore, and waste materials mined in that period. Eqs. and ensures that the sum of the partials of block *n* extracted is at most one over all periods at the end of the mine life.

$$\sum_{n=1}^{N}(o_n \times x_n^t) \leq \sum_{n=1}^{N}((o_n + w_n)y_n^t) \qquad \forall_t \in \{1,...T\};$$

$$\sum_{t=1}^{T} y_n^t \leq 1 \qquad \forall_n \in \{1,...N\};$$

$$\sum_{t=1}^{T} x_n^t \leq 1 \qquad \forall_n \in \{1,...N\};$$

4.2.7. Non-negativity constraints

Non-negativity constraints monitor the decision variables to ensure they do not take negative values. Eq. defines the non-negativity of decision variables.

$$x_n^t, y_n^t, b_n^t \geq 0 \qquad \forall_n \in \{1,...N\};\ \forall_t \in \{1,...T\};$$

4.3. Stochastic MILP formulation in the presence of grade uncertainty

The stochastic formulation for the OPPS problem considered in this research is modified after the formulation by Vallejo and Dimitrakopoulos [96]; Mbadozie [71]. The approach for including grade uncertainty in the mining project stems from having multiple simulated orebody realizations generated through SGS which are equally probable and serve as input to the stochastic model. Equally probable orebody realizations mean each simulated realization can be a valid representation of the actual orebody. The simulated orebody realizations capture the varying grade distribution that would not have been realized with a single interpolated block model based on a method like Kriging. Previous research from Albor and Dimitrakopoulos [1]; Vallejo and Dimitrakopoulos [96] have identified that, 20 simulated orebody realizations are adequate to capture the uncertainty in grade distributions.

4.3.1. Multi-objective function

The objective function for the stochastic model is derived from the average of all the simulated orebody realizations. Since these realizations are equally probable, each realization depicts varying grades for the orebody model. This can be assumed as having *S* number of schedules at the end of the optimization with each *s* schedule representing a probable solution. To simultaneously optimize with all the equally probable orebody realizations, an average of the revenue and cost from the realizations are taken into account in the objective function (Eq.).

The multi-objective function has two components: 1) Maximize the NPV of the mining operation (Eq.); and 2) Minimize the cost of uncertainty associated with deviating from the operating targets, including ore tonnage and ore grade (Eq.). This is achieved by applying penalty costs and a geological risk discount rate to the ore tonnage and ore grade targets. Continuous deviation decision variables $od^t_{s,+}$, $od^t_{s,-}$, $gd^t_{s,+}$ and $gd^t_{s,-}$ as well as their respective penalty parameters pc^t_{o+}, pc^t_{o-}, pc^t_{g+} and pc^t_{g-} are used for minimizing deviations from ore tonnage and ore grade production targets defined by Eqs. to . These are introduced in the second component of the objective function (Eq. to enable the optimizer to select realization blocks with ore tonnage and ore grade that minimizes deviations from the corresponding production targets simultaneously through a balancing act. For example, if the optimizer selects realization blocks with high grade, it will lead to a large ore tonnage deviation resulting from reduced ore reserve which is undesirable and vice versa.

Additionally, a geological risk discount rate (*dr*) is applied to the cost of deviation to defer the risk of not meeting production targets to later periods. From Eq. , by applying the *dr* parameter as a denominator tied to periods, early periods have larger impact on the minimization objective function value than later periods. This means the overall penalty value is higher in the earlier periods than in latter periods ensuring that early-year deviations from stated targets are lower than later-year deviations. Conceptually, the higher penalty in earlier periods drive the optimizer to limit deviations from the ore tonnage and ore grade targets early in the mine life and postpone extraction of areas with larger deviations until later periods when more geological understanding of the deposit becomes available.

$$Max \frac{1}{s} \sum_{s=1}^{S} \sum_{t=1}^{T} \sum_{n=1}^{N} \left(\frac{v^t_{n,s} \times x^t_n - q^t_{n,s} \times y^t_n}{(1+r)^t} \right)$$

Where *S* is set of all equally probable realizations.

$$Min\frac{1}{S}\sum_{s=1}^{S}\sum_{t=1}^{T}\left(\frac{pc_{o+}^{t}\times od_{s,+}^{t}+pc_{o-}^{t}\times od_{s,-}^{t}}{(1+dr)^{t}}\right)+Min\frac{1}{S}\sum_{s=1}^{S}\sum_{t=1}^{T}\left(\frac{pc_{g+}^{t}\times gd_{s,+}^{t}+pc_{g-}^{t}\times gd_{s,-}^{t}}{(1+dr)^{t}}\right)$$

Eqs. and can be combined together as a single objective function as shown in Eq. .

$$Max\frac{1}{s}\sum_{s=1}^{S}\sum_{t=1}^{T}\sum_{n=1}^{N}\left(\begin{array}{l}\left(\frac{v_{n,s}^{t}\times x_{n}^{t}-q_{n,s}^{t}\times y_{n}^{t}}{(1+r)^{t}}\right)-\\ \frac{1}{N}\left(\frac{pc_{o+}^{t}\times od_{s,+}^{t}+pc_{o-}^{t}\times od_{s,-}^{t}}{(1+dr)^{t}}\right)-\\ \frac{1}{N}\left(\frac{pc_{g+}^{t}\times gd_{s,+}^{t}+pc_{g-}^{t}\times gd_{s,-}^{t}}{(1+dr)^{t}}\right)\end{array}\right)$$

Subject to:

4.3.2. Mining capacity constraints

Eqs. and defines the mining capacity constraint for each period. Eq. ensures that the total blocks tonnage mined is equal to or less than the stipulated capacity of mining equipment while Eq. controls the minimum amount of materials mined. The tonnage of materials mined is the sum of the ore tonnage and waste tonnage represented as $o_{n,s}$ and $w_{n,s}$ respectively. The continuous decision variable y_n^t controls this extraction process in each period.

$$\sum_{n=1}^{N}(o_{n,s}+w_{n,s})y_n^t\le Cu^t \qquad \forall_t\in\{1,...T\};\ \forall_s\in\{1,...S\};$$

$$\sum_{n=1}^{N}(o_{n,s}+w_{n,s})y_n^t\ge Cl^t \qquad \forall_t\in\{1,...T\};\ \forall_s\in\{1,...S\};$$

4.3.3. Processing capacity constraints

Eqs. and define the processing capacity of the mining operation for each period. Eq. sets the upper bound and Eq. sets the lower bound for the amount of ore processed. The deviation decision variables $od_{s,+}^t$ and $od_{s,-}^t$ are introduced to serve as buffers to the ore tonnage targets. These decision variables are penalized in the objective function (Eq.) to ensure that the ore tonnage targets are achieved with minimum deviation. These constraints are controlled by the continuous decision variables x_n^t, $od_{s,+}^t$ and $od_{s,-}^t$ in each period.

$$\sum_{n=1}^{N}(o_{n,s}\times x_n^t)-od_{s,+}^t\le Qu^t \qquad \forall_t\in\{1,...T\};\ \forall_s\in\{1,...S\};$$

$$\sum_{n=1}^{N}(o_{n,s}\times x_n^t)+od_{s,-}^t\ge Ql^t \qquad \forall_t\in\{1,...T\};\ \forall_s\in\{1,...S\};$$

4.3.4. Grade blending constraints

Eq. defines the upper limit of the ore grade and Eq. defines the lower limit of the ore grade to be sent to the mill in each period. The deviation decision variables $gd^t_{s,+}$ and $gd^t_{s,-}$ are introduced to serve as buffers to the ore grade targets. These decision variables are penalized in the objective function (Eq.) to ensure that the ore grade targets are achieved with minimum deviation. These constraints are controlled by the continuous decision variables x^t_n, $gd^t_{s,+}$ and $gd^t_{s,-}$ in each period.

$$\sum_{n=1}^{N} g_{n,s} \times (o_{n,s} \times x^t_n) - \sum_{n=1}^{N} \overline{g}^t \times (o_{n,s} \times x^t_n) - gd^t_{s,+} \le 0 \quad \forall_t \in \{1,...T\};\ \forall_s \in \{1,...S\};$$

$$\sum_{n=1}^{N} g_{n,s} \times (o_{n,s} \times x^t_n) - \sum_{n=1}^{N} \underline{g}^t \times (o_{n,s} \times x^t_n) + gd^t_{s,-} \ge 0 \quad \forall_t \in \{1,...T\};\ \forall_s \in \{1,...S\};$$

4.3.5. Block precedence constraints

Eqs. to enforce the block extraction precedence constraints. Binary integer decision variable, b^t_n, is used to control the precedence of block extraction. b^t_n is equal to one if the extraction of mining blocks has started by or in period *t*; otherwise, it is zero. For each mining block *n,* Eq. check the set of immediate predecessor blocks in $H_n(D)$ that must be mined prior to mining block *n.* Eq. checks that extraction of mining block *n* can start only when the mining block has not been previously extracted. Eq. ensures that once extraction of block *n* starts, this block is available for extraction in subsequent periods.

$$b^t_n - \sum_{i=1}^{t} y^i_d \le 0 \quad d \in H_n(D) \qquad \forall_n \in \{1,...N\};\ \forall_t \in \{1,...T\};$$

$$\sum_{i=1}^{t} y^i_n - b^t_n \le 0 \qquad \forall_n \in \{1,...N\};\ \forall_t \in \{1,...T\};$$

$$b^t_n - b^{t+1}_n \le 0 \qquad \forall_n \in \{1,...N\};\ \forall_t \in \{1,...T-1\};$$

4.3.6. Variable control constraints

Eq. ensures that the total ore material mined in any given scheduling period is less or equal to the sum of the ore, and waste materials mined for all realizations in that period. Eqs. and ensure that the sum of the partials of block *n* extracted is at most one over all periods at the end of the mine life.

$$\sum_{n=1}^{N} (o_{n,s} \times x^t_n) \le \sum_{n=1}^{N} ((o_{n,s} + w_{n,s}) y^t_n) \qquad \forall_t \in \{1,...T\};\ \forall_s \in \{1,...S\};$$

$$\sum_{t=1}^{T} y^t_n \le 1 \qquad \forall_n \in \{1,...N\};$$

$$\sum_{t=1}^{T} x_n^t \le 1 \qquad \forall_n \in \{1,...N\};$$

4.3.7. Non-negativity constraints

Eq. defines the non-negativity constraints for the decision variables for mining, processing, extraction precedence, and ore tonnage and ore grade target deviations. These constraints enforce that none of these variables can take on negative values during the optimization process.

$$x_n^t,\ y_n^t,\ b_n^t,\ od_{s+}^t,\ od_{s-}^t,\ gd_{s+}^t,\ gd_{s-}^t \ge 0 \qquad \forall_n \in \{1,...N\};\ \forall_t \in \{1,...T\};\ \forall_s \in \{1,...S\};$$

4.4. Genetic algorithm problem representation

The starting point of the optimization problem in GA is the problem initialization which consists of the chromosome encoding phase. A multi-layer chromosome encoding technique was used in this research: (1) a literal permutation encoding scheme, and (2) a real number or continuous variable encoding scheme. Fig. 8 shows a sample of the chromosome encoding represented in the GA. The literal permutation encoding was employed for the genes representing the period and realization where real number encoding was used for the genes representing the fractions blocks extracted.

Block index	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*~*	*n*
Period	5	6	5	4	3	1	1	2	2	6	7	8	7	8	8	~	8
% Block	0.3	0.2	0.5	0.2	0.1	0.3	0.6	0.9	0.4	0.22	0.23	0.41	0.74	0.25	0.33	~	0.2

Figure 8. Sample chromosome encoding.

In this research, every block is assumed to be mined over at most two periods. Therefore, the chromosomes were encoded in two halves as shown in Fig. 9. The first and second halves represent the fractions of each block mined at different periods respectively. The constraints in Eqs. and are therefore satisfied in the chromosome encoding when these two halves are reconciled at the end of the optimization. Every block in the model is mined at most once during the mine life.

Chromosome B

Block index	*1*	*2*	*3*	*4*	*~*	*n*
Period	5	6	5	4	~	1
% Block	0.3	0.2	0.5	0.2	~	0.3

Chromosome A

Block index	*1*	*2*	*3*	*4*	*~*	*n*
Period	3	9	1	5	~	2
% Block	0.7	0.8	0.5	0.8	~	0.7

Figure 9. Sample chromosome encoded in two halves.

Eq. was used in generating the genes in the chromosome representing the block fractions since this requires a continuous variable encoding. The other genes were generated using a Gaussian random distribution. The GA was implemented to optimize either a deterministic or stochastic production schedule based on input from the user. The objective function of the optimization was represented as a fitness function to test each solution in the population. The fitter population survives the current generation and proceeds in the iteration process.

4.4.1. Constraints handling and representation

The major constraint in the OPPS problem that presents great levels of complexities is the precedence constraint. The precedence constraint determines the sequence of block extraction and ensures that blocks on the surface are extracted in the same or an earlier period to blocks directly beneath them. As shown in Fig. 10, Blocks 1, 2 and 3 must be extracted prior to extraction of Block 10 or in the same period as Block 10. In three-dimensional (3D) block representation, every block has at least nine different blocks forming its precedence.

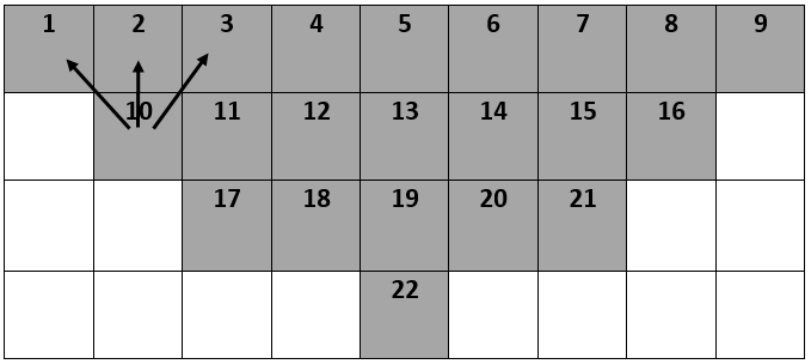

Figure 10. Block extraction precedence modified after Ben-Awuah and Askari-Nasab [12].

In order to ensure that the precedence constraints are enforced as defined, a check-and-repair method is implemented in the GA. In each population, if a block cannot be mined in period *t* due to precedence constraints, it is moved to the next period or a period where the requirements of immediate predecessor blocks in $H_n(D)$ are not in violation. The entire population is then normalized to accept the current gene as a feasible solution for evaluation. For every violation of the normalized precedence constraints, the fitness function is penalized to ensure that the optimizer finds a feasible solution in each generation.

Capacity constraints are treated as knapsack problems. Knapsack problems are primarily resource allocation problems. The maximum allowable capacity is derived from the optimization problem since the scheduling is performed annually (periods). All the extracted blocks for that period should be less or equal to the maximum capacity for that period. Using sliding window technique [8; 20], an array containing the tonnage of every block scheduled in each period is created. The total tonnage of every period is checked against the maximum capacity from the optimization problem. When the maximum capacity for a period is reached, all subsequent blocks or block fractions that were originally in that period are moved to the next period. The population is then normalized afterwards so that the current population contains the right blocks that satisfy the capacity constraint for that period. The window is then slid to the next period and the steps above are repeated until the last period is reached.

4.4.2. Normalization

Due to the randomness associated with GA at every stage of the optimization process, the genes in each population tend to violate constraints when mutation or crossover occurs. There is therefore the need to normalize the population after each mutation and crossover to ensure that the constraints are satisfied [26]. This process is termed as normalization or regularization. Fig. 11 shows an example of a double point crossover that violates the precedence constraints. As illustrated in Fig. 11(A), before crossover occurs, both Parents 1 and 2 are feasible solutions satisfying the precedence constraints. That is, all blocks on the lower level are extracted in periods later than or equal to extraction periods of blocks above them. During crossover, the genes in Parents 1 and 2 representing Blocks 10, 11, 12 and 13 within the crossover point are swapped. Child 1 receives genes from Parent 2 whiles Child 2 receives genes from Parent 1. As highlighted

in Fig. 11(B), after the crossover, both Child 1 and Child 2 violate the precedence constraint because Block 10 in Child 1 and Block 18 in Child 2 are extracted in periods earlier than the blocks above them and therefore require normalization. This process ensures that a population with a feasible solution is kept at all times throughout the GA optimization process.

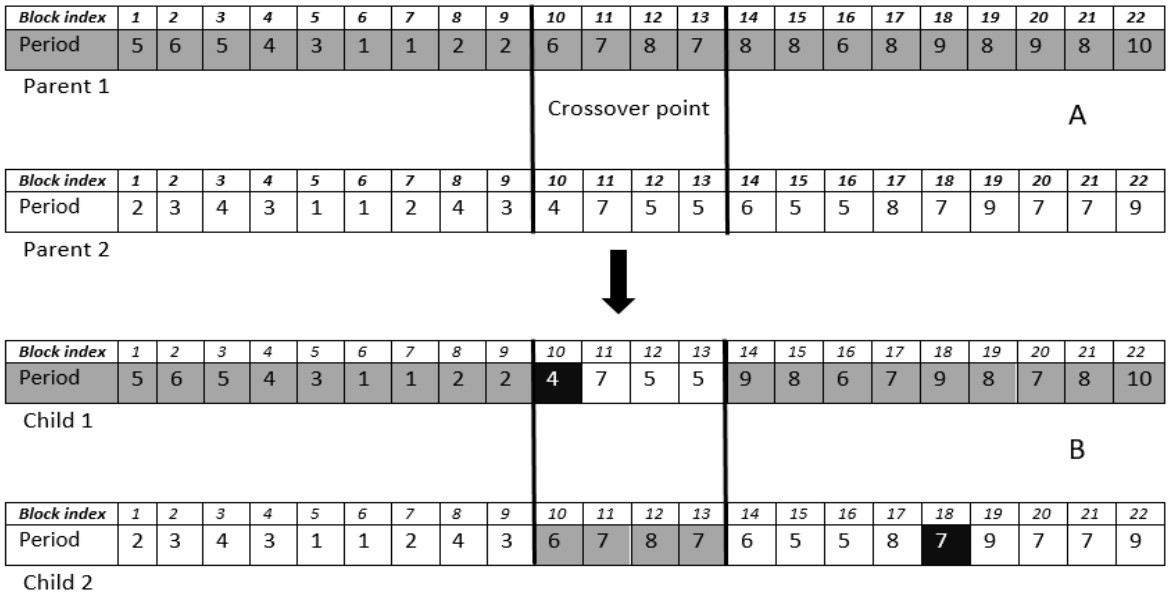

Figure 11. Sample crossover showing precedence constraint violation. (A) Illustrates a feasible solution before crossover. (B) Demonstrates a solution that violates the precedence constraint after crossover.

4.4.3. Mutation and crossover strategy

Genetic operators such as crossover and mutation are key to the success of any genetic algorithm optimization process and as such finding the best strategy for them is always paramount. In this research, a 'smart' mutation was implemented to curtail the complexities in handling the partial extraction of blocks in the population. Fig. 12 shows a sample chromosome with twenty genes. To represent a chromosome with n number of blocks or genes; the corresponding length of that chromosome is $2n$. This method is used to implement the assumption that each block can be extracted in at most two periods. The first part of the chromosome represented as Chromosome A in Fig. 12 shows portions of the blocks and corresponding periods the extraction occurs in; same for Chromosome B.

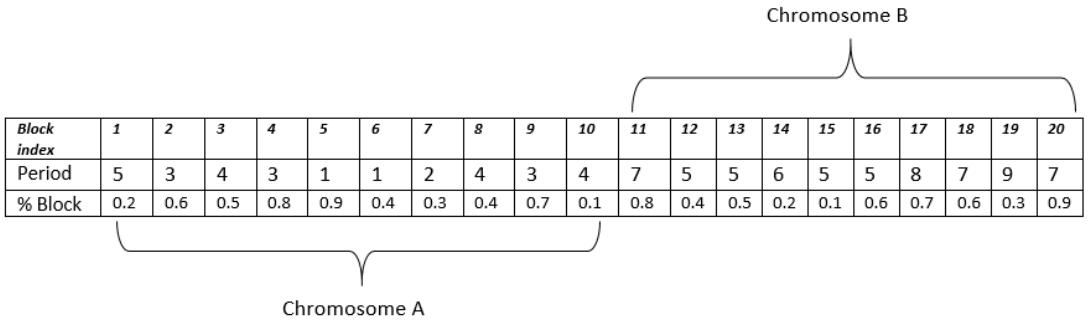

Figure 12. Sample chromosome representing the multi-part chromosome encoding.

During the mutation process, a probability of mutation is applied to determine the gene that must mutate. In Fig. 13, genes at block index 6, 7, 8 and 9 of Chromosome A will mutate per the probability of mutation applied. This however needs to occur in tandem with the corresponding genes in Chromosome B. The mutation algorithm keeps the index of the genes in Chromosome A

and determines the corresponding position of the other genes in Chromosome B. When the mutation occurs in Chromosome A, a subsequent mutation and normalization takes place in Chromosome B. The genes representing the periods are mutated at random from a feasible set of periods that the block can be extracted in. Based on the precedence constraints. Given a feasible set of period (3, 2, and 4) for Block 6, a period is chosen at random and assigned to the block during the mutation for the period of that block. The mutation for the fractions of blocks that should be extracted is determined by Eq. . The mutation algorithm again keeps the index of the gene representing the fraction of the block to be extracted in Chromosome A and determines the corresponding position of the other gene in Chromosome B. The double point crossover used in this research also employs the same chromosome representation and index retention approach. Fig. 14 shows a sample chromosome after mutation showing the result of mutation for Chromosome A and Chromosome B. Table 1 shows the pseudo code for the proposed mutation strategy used to handle the block extraction by the GA.

Fig. 15 shows the flow chart for the GA optimization process and sub processes. A two-step GA framework was implemented; where based on the data provided and the input from the user, the framework will decide whether the problem is stochastic or conventional before proceeding to evaluate the fitness function for each population.

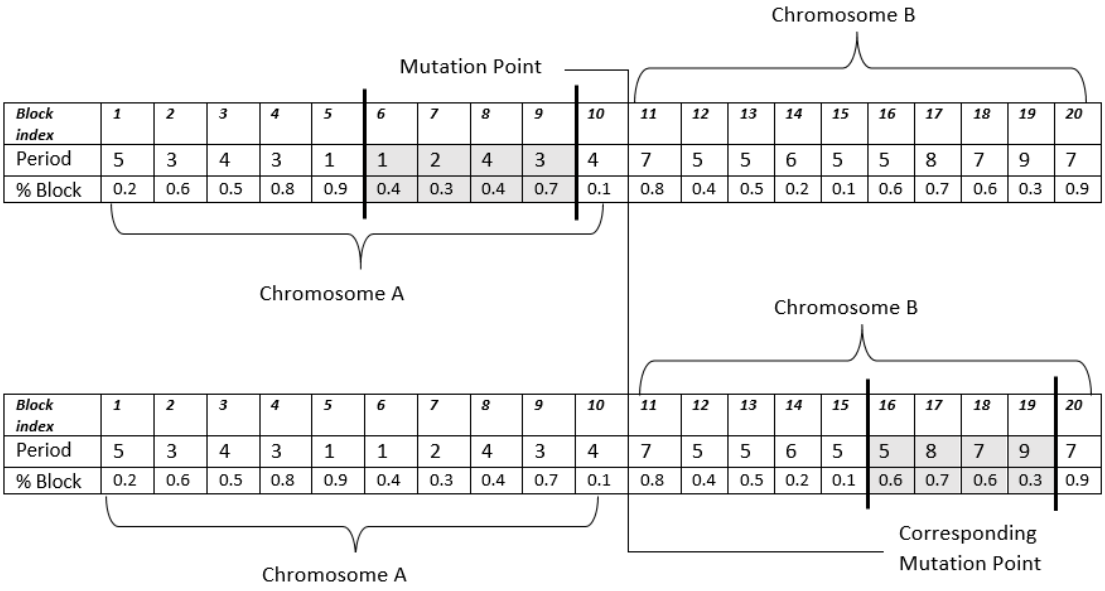

Figure 13. Sample chromosome before mutation showing the mutation point of Chromosome A and Chromosome B.

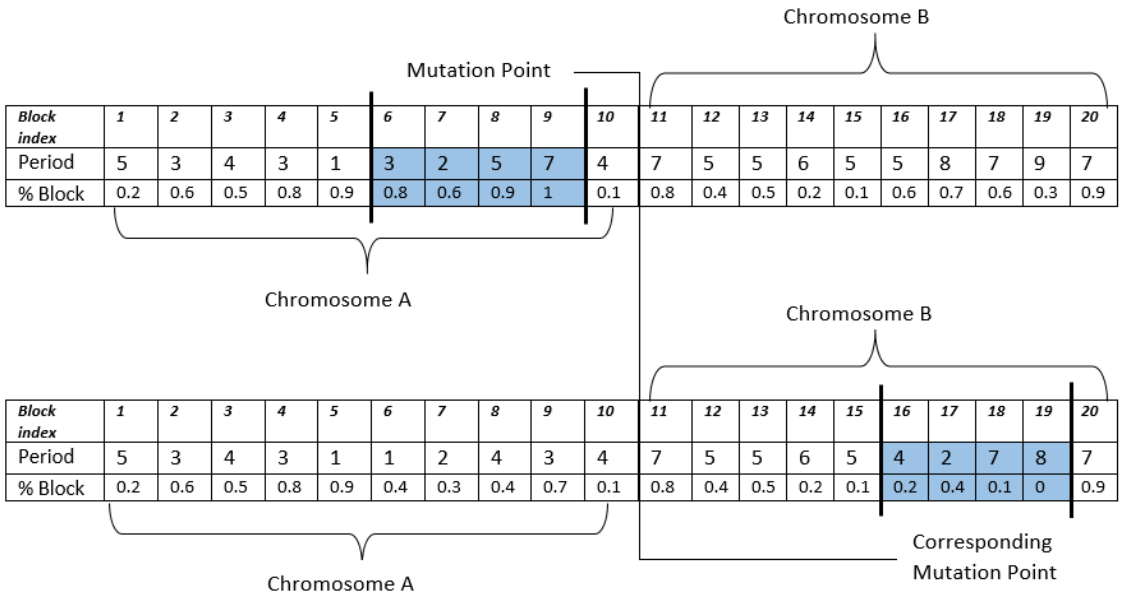

Figure 14. Sample chromosome after mutation showing the result of mutation for Chromosome A and Chromosome B.

Table 1. Pseudo code for the proposed mutation strategy.

Pseudo Code for proposed mutation strategy
Start
get *chromosomeLength;*
get *popabilityOfMutation;*
get *numberOfBlocks*
Select number of individual to be mutated based on the propablilityOfMutation.
split the chromosome into two halves A and B based on numberOfBlocks
While *n = Number of individuals to be mutated*
Get the index a of the individual in chromosome A and corresponding index b in chromosome B
Perform mutation on gene n at a in chromosome A and gene n at b in chromosome B
EndWhile
Combine chromosome A and B after mutation and return to the main generation
End

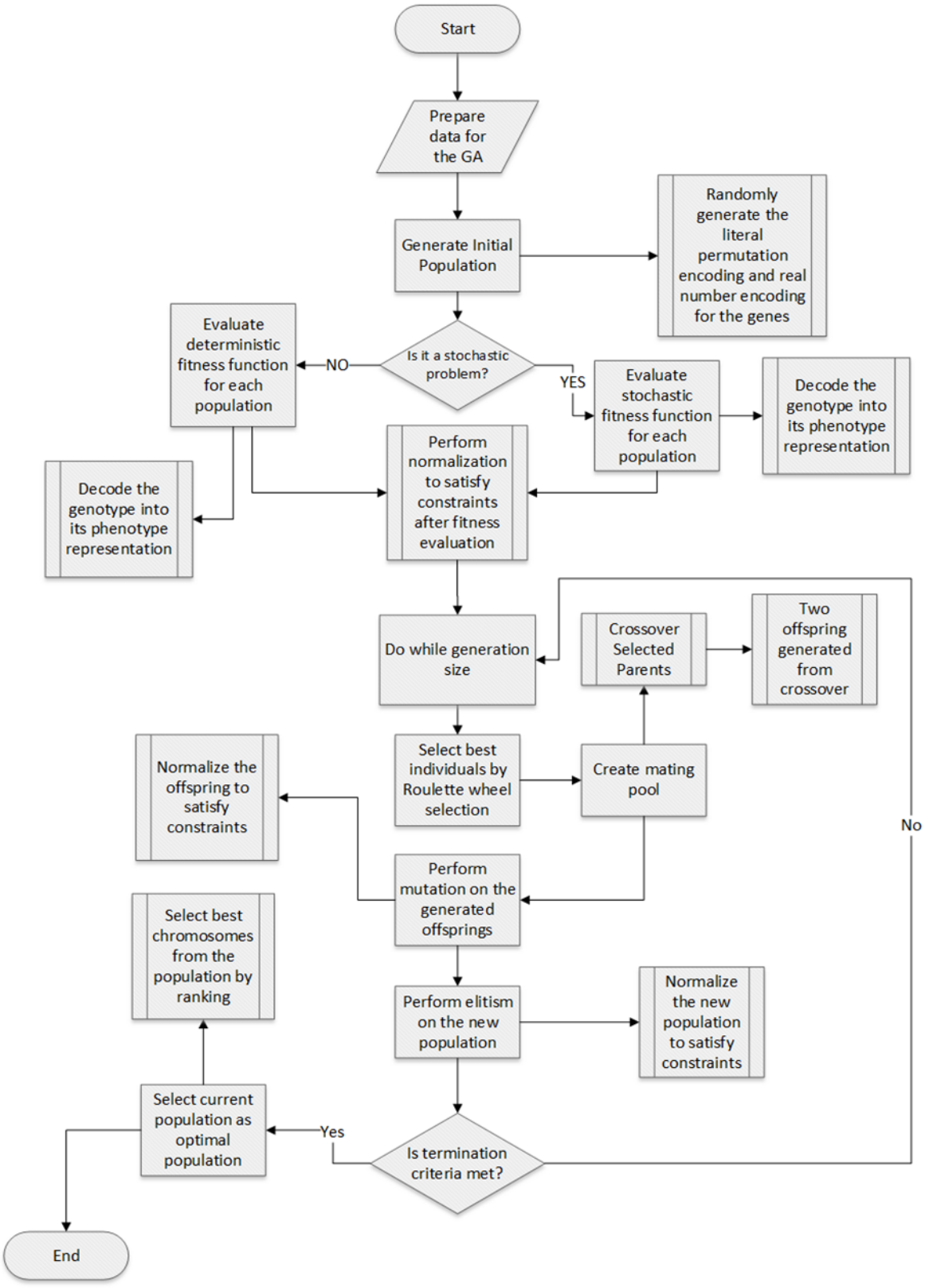

Figure 15. Proposed GA optimizations and sub process.

5. Computational Experiments

The GA model was implemented for two different oil sands datasets obtained from Ben-Awuah and Askari-Nasab [12] and Mbadozie [70]; the first case study with 4476 blocks and the second with 1569 blocks. Two scenarios were implemented for the first case study as proof of concept: (1) A deterministic model with GA (DGA); and (2) A stochastic model with GA (SGA). Subsequently, the SGA model was implemented for the second case study. The orebody model for the DGA was based on Ordinary Kriging which did not consider grade uncertainty. The SGA scenario considered grade uncertainty through equally probable orebody realizations generated using sequential Gaussian simulation. Table 2 shows the block model data and Table 3 outlines the economic parameters for both case studies. Table 4 highlights the mining and processing requirements for both case studies. In Table 5, the risk parameters for the stochastic scenario are outlined. The DGA results were compared with a similar implementation using MILP model with CPLEX formulated by Mbadozie (2022). The SGA results were also compared with a similar implementation using a Stochastic MILP (SMILP) model with CPLEX formulated by Mbadozie (2022). These comparisons were done to assess the practicality of the generated schedules as well as the NPV and computational efficiency. The production schedule for Case study 1 was optimized over ten periods whereas Case study 2 was scheduled over twenty periods. The primary focus for the GA framework was to generate a uniform and practical schedule while respecting the constraints for all periods in the schedule. The DGA and SGA were implemented in a MATLAB environment (MathWorks Inc., 2020) on a Lenovo ThinkPad computer with Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz and 16 GB of RAM. Table 6 outlines the GA parameters used in both case studies.

Table 2. Oil sands block model data for case studies.

Block model data (Units)	Case study 1	Case study 2
Total block tonnage (Mt)	318	3539
Total ore tonnages (Mt)	145	1141
Block dimensions (m x m x m)	50 x 50 x 15	300 x 300 x 15
Mine life (Years)	10	20
Number of blocks	4476	1569

Table 3. Economic parameters for case studies.

Parameter (Units)	Value
Mining cost ($/tonne)	4.60
Processing cost ($/tonne)	5.03
Selling price ($/bitumen %mass)	4.50
Economic discount rate (%)	10

Table 4. Mining and processing requirements for case studies.

	Case study 1		Case study 2	
Parameters (Units)	**Min value**	**Max value**	**Min value**	**Max value**
Mining capacity (Mt/year)	25	32	100	150

Processing capacity (Mt/year)	10	14	25	50
Ore bitumen grade (%m)	7	16	7	16

Table 5. Risk parameters for stochastic scenario.

Parameters (Units)	value
Number of realizations	20
Cost of shortage in ore production ($/tonne)	5
Cost of excess in ore production ($/tonne)	10
Cost of shortage in ore bitumen grade ($/%m)	2.5
Cost of excess in ore bitumen grade ($/%m)	1.5

Table 6. GA parameters used for both case studies.

GA parameter	Description
Population size	**20**
Selection type	**Roulette wheel**
Crossover type	**Double point**
Probability of mutation	**0.2**
Probability of crossover	**0.85**
Probability of elitism	**0.2**
Maximum generations	**1000**

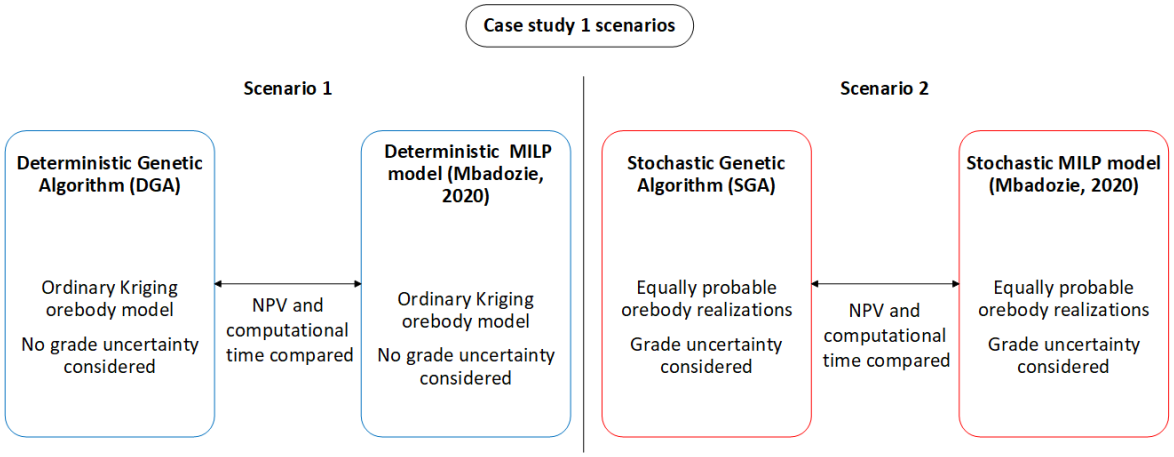

Figure 16. Case study 1 scenario comparisons.

Fig. 16 presents a summary of the experimental methodology and comparisons made between the DGA and MILP model, and SGA and SMILP model for Case study 1. For Case study 2, the SMILP integer solution was terminated after 28 days. Therefore, CPLEX was used to solve the relaxed LP problem to estimate the optimality gap for the GA results. Eq. by IBM ILOG CPLEX Inc [55] was used to ascertain the optimal difference between the GA solution and the relaxed LP solution by CPLEX.

$$\frac{|bestbound - bestinteger|}{(1e-10+|bestinteger|)}$$

The *bestbound* in Eq. for an optimization problem refers to the objective function value at which a feasible optimal solution could potentially exist [55]. In the case of an intractable integer problem, the *bestbound* is the only solution. This is the case because the relaxed problem does not have a

bestinteger solution. Therefore, in determining the gap for the GA solution using Eq. , the relaxed objective function value is represented as the *bestbound* and the solution for the GA as the *bestinteger* to compute the optimality gap.

5.1. Results and discussion

5.1.1. Case study 1 comparative analysis: MILP model with CPLEX and DGA results

The results from the DGA for Scenario 1 was compared with a similar implementation from the MILP model with CPLEX at 0% optimality gap. The NPV generated from the proposed DGA and MILP model with CPLEX were $1,830 M and $1,929 M respectively. The total time taken for the MILP to generate its results was 4.1 hours whereas the DGA generated its results in 1.9 hours. The NPV of the DGA solution was 5.1% less than that of the MILP solution. The DGA was able to generate a uniform schedule over the mine life. The production schedule results generated by the DGA are shown in Table 7. Fig. 17 and Fig. 18 shows the cross-sectional view and the plan view of the extraction sequence generated by the DGA for the production schedule respectively. It can be seen from Fig. 17 that, the DGA model enforced the precedence constraints set in the optimization problem; blocks were mined according to their precedence and scheduled appropriately. Blocks on the lower levels were mined in later periods as opposed to blocks on the surface The total tonnage and ore tonnage generated by the DGA are shown in Table 8. Table 8 also shows the duration and NPV comparison of the MILP with the DGA results. It can be seen from Fig. 19A that the DGA respected the maximum annual mining capacity constraint which was set at 32 Mt across all the scheduling periods. Although less material was extracted in the first period, the extraction gradually ramped up and was uniform for the subsequent periods until declining in the last period. The maximum annual processing capacity of 14 Mt was respected by the DGA as seen in Fig. 19A. Fig. 20 shows the graph of the average ore bitumen grade for the DGA and the MILP with CPLEX production schedules. It can be seen from Fig. 20 that there is a gradual decline of the ore bitumen grade as the mine life progresses, which ultimately influences the NPV.

Table 7. Scheduling results for the DGA.

Period	Total tonnage (Mt)	Ore tonnage (Mt)	Average ore bitumen grade (%m)
1	24.41	8.51	12.87
2	31.93	13.32	12.32
3	31.94	13.54	12.00
4	31.90	13.96	11.31
5	31.93	13.91	11.33
6	30.91	13.91	10.54
7	31.93	13.90	10.81
8	31.90	13.94	9.92
9	31.27	13.93	8.76
10	9.13	6.70	7.92

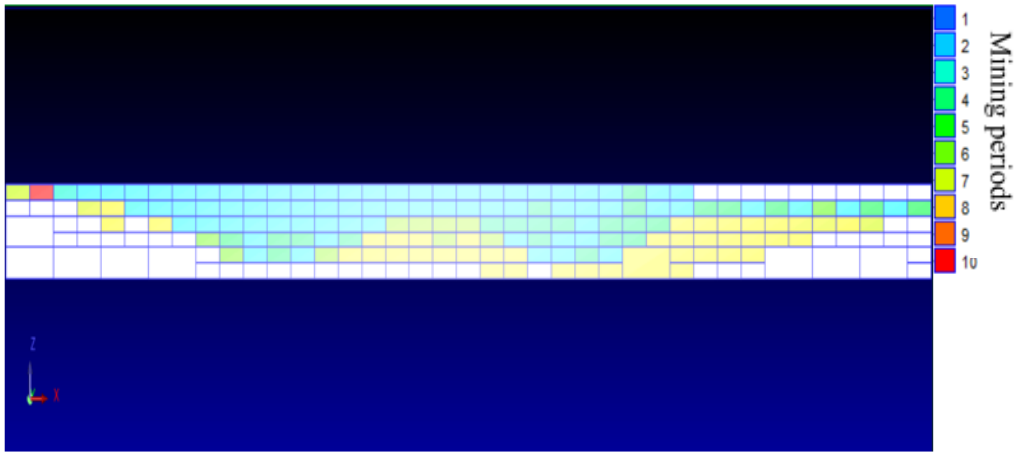

Figure 17. Cross sectional view of the block extraction sequence by the DGA.

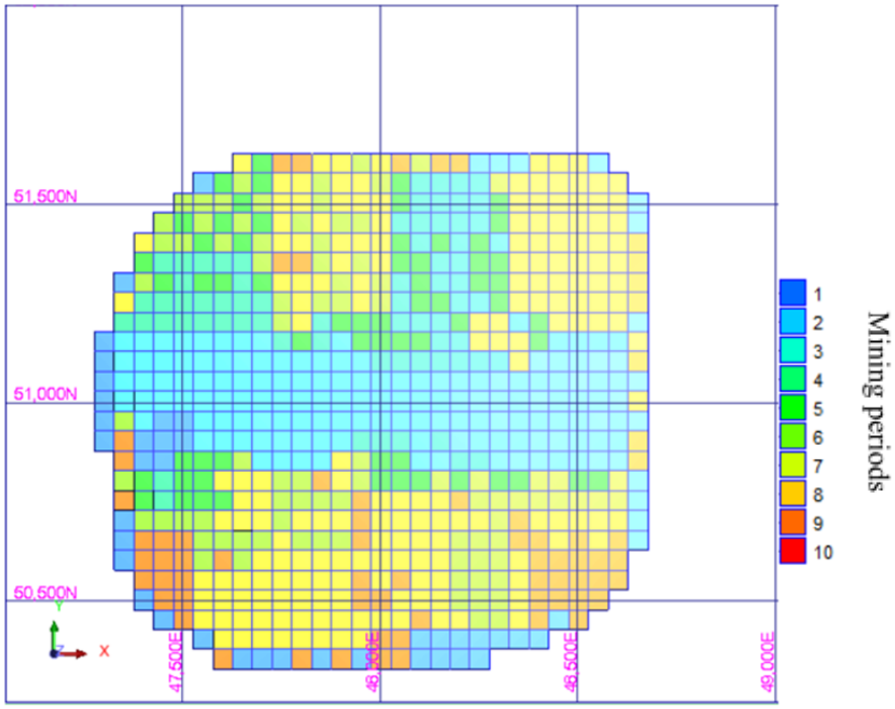

Figure 18. Plan view of the block extraction sequence by the DGA on Bench 3.

Table 8. Solution comparison between the MILP model with CPLEX and the DGA.

Parameter (Units)	MILP model with CPLEX	DGA
Number of blocks	4476	4476
Tonnage mined (Mt)	287	285
Ore processed (Mt)	121	124
NPV ($M)	1929	1830
Time (hours)	4.1	1.9
Optimality gap (%)	0	-

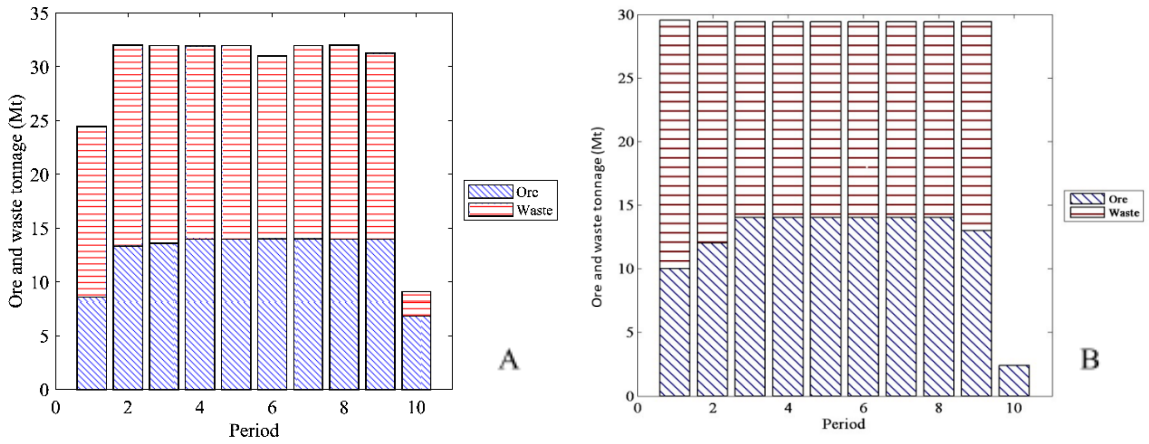

Figure 19. Total tonnages mined. (A) Illustrates the total tonnage mined by the DGA and (B) illustrates the total tonnage mined from the MILP with CPLEX [71].

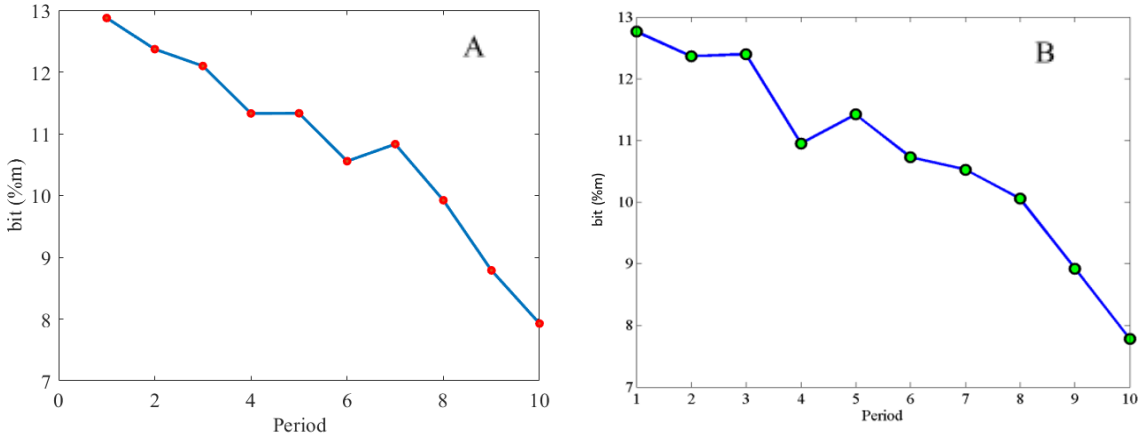

Figure 20. Average ore bitumen grade from the DGA illustrated in (A) and average ore bitumen grade from the MILP model with CPLEX illustrated in (B).

For Scenario 2, the motivation for the SGA was to ascertain the impact of grade uncertainty on the production schedule. To achieve this, multiple simulated orebody realizations generated through SGS were used as input to the optimization problem. The NPV generated from the SGA and SMILP model with CPLEX were $2,128 M and $2,248 M respectively. The NPV for the SGA was 5.3% less than that for the SMILP model with CPLEX. The optimality gap for the SMILP model with CPLEX was set at 5%. The total time taken for the SMILP to generate its results was 11.70 hours whereas the SGA generated its results in 2.9 hours. Table 9 shows the scheduling results. Fig. 21 and Fig. 22 show the extraction sequence of the SGA. Table 10 shows the solution comparison between the SMILP model with CPLEX and the SGA. The impact of grade uncertainty is evident in the NPV generated by the stochastic schedule. The NPV generated by the SGA schedule was 16.3% better than the NPV from the DGA schedule. In Fig. 23, a comparison between the SGA and the SMILP model with CPLEX is shown. The capacity constraints were respected by the SGA as seen in Fig. 23A. Fig. 24 shows the average ore bitumen grade comparison between the SGA and SMILP model as well as comparison with individual orebody realizations. From Fig. 25, it can be observed that, the stochastic model maintained a balanced average grade throughout the mine life, which accounted for the improvement in NPV compared to the DGA's average grade, which declined gradually as the mine life progressed.

Table 9. Scheduling results for the SGA.

Period	Total tonnage (Mt)	Ore tonnage (Mt)	Average ore bitumen grade (%m)
1	24.44	9.71	11.67
2	31.92	13.30	11.80
3	31.91	13.51	11.61
4	31.93	13.53	11.72
5	31.91	13.84	11.67
6	30.90	13.90	11.70
7	31.93	13.93	11.52
8	31.94	13.98	11.65
9	31.22	12.9	11.41
10	9.10	5.70	11.73

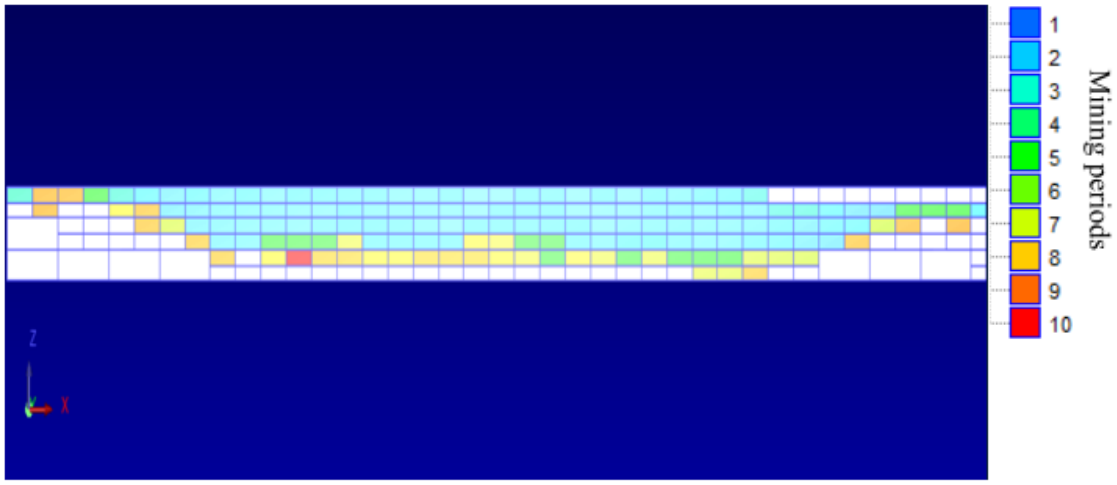

Figure 21. Cross sectional view of the block extraction sequence by the SGA.

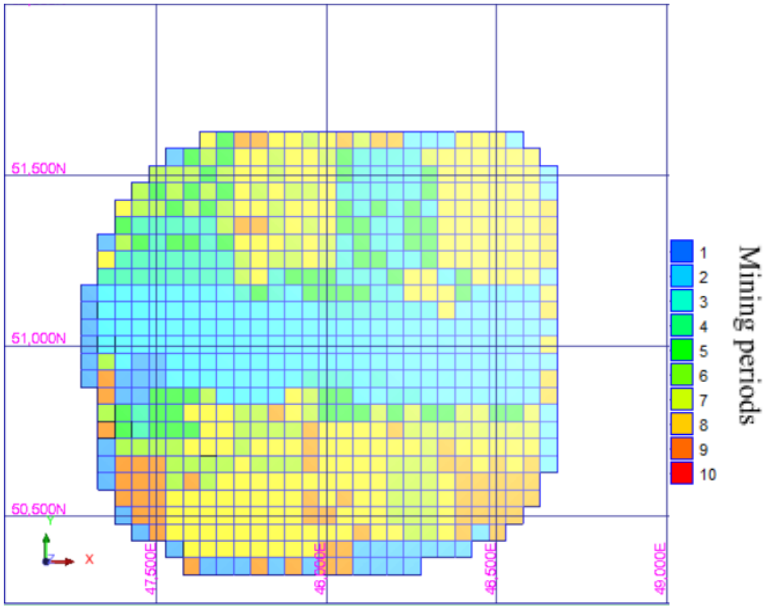

Figure 22. Plan view of the block extraction sequence by the SGA on Bench 3.

Table 10. Solution comparison between the SMILP model with CPLEX and the SGA.

Parameter (Units)	SMILP model with CPLEX	SGA
Number of blocks	4476	4476
Tonnage mined (Mt)	290	287
Ore processed (Mt)	124	125
NPV ($M)	2248	2128
Time (hours)	11.70	2.9
Optimality gap (%)	5	-

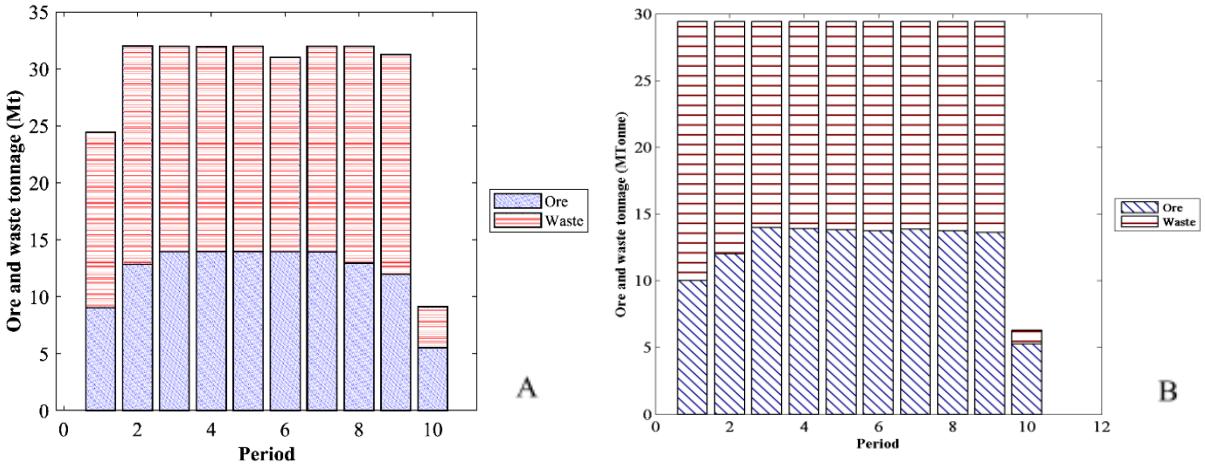

Figure 23. Total tonnages mined. (A) Illustrates the total tonnage mined by the SGA and (B) illustrates the total tonnage mined from the SMILP with CPLEX [71].

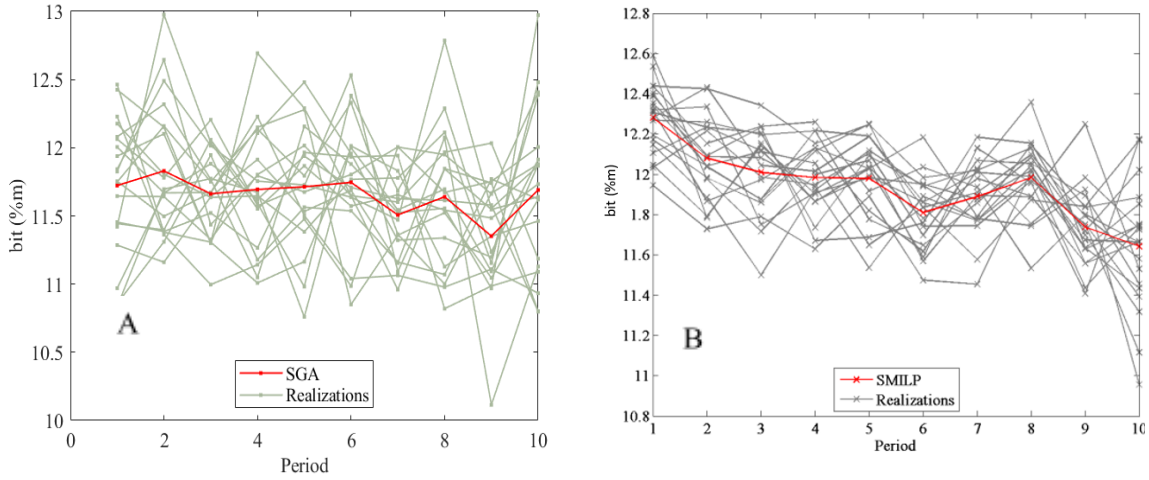

Figure 24. Average ore bitumen grade from the SGA illustrated in (A) with 20 realizations and the average ore bitumen grade from the SMILP model with CPLEX illustrated in (B) with 20 realizations.

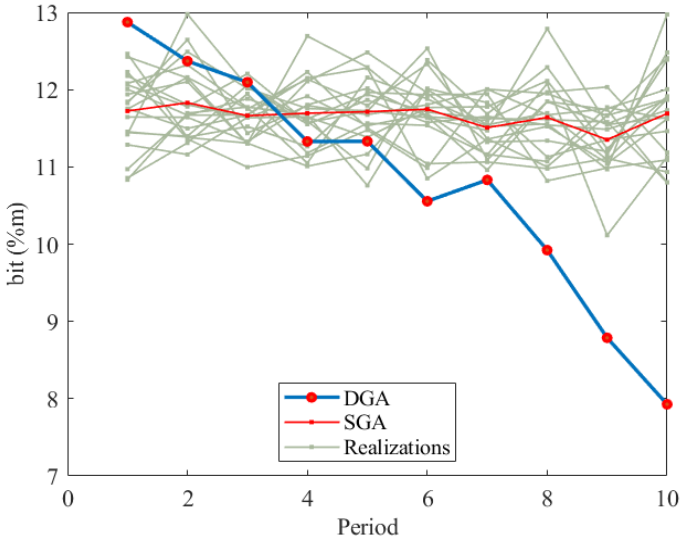

Figure 25. Average ore bitumen grade comparison for the DGA, SGA and 20 realizations.

5.1.2. Case study 2 comparative analysis: SMILP model with CPLEX and SGA results

The solution for Case study 2 while the SMILP model was at a gap of 101% after 28 days, the SGA generated a solution in 1.5 hours. This further emphasizes the application of metaheuristics to NP-hard combinatorial optimization problems. In the bid to verify and compare the results generated by the SGA, a relaxed LP form of the problem was solved and Eq. was used to compute the optimality gap. Based on Eq. , the objective function value generated by the relaxed LP was 12,810 and that for the SGA was 11,629. Using Eq. gives us a gap of 10.16%. This therefore means the SGA solution is in the worst case scenario at 10.16% of the optimal solution to the SMILP model if it exists since the relaxed LP solution is the upper bound to it. The NPV generated from the SGA was $10,045 M. Table 11 shows the total tonnage, ore tonnage and average ore bitumen grade for the GA scheduling results. The solution comparison for NPV, runtime, and optimality gap are summarized in Table 12. The SGA results respected the maximum annual mining capacity constraint set at 150 Mt and maximum annual processing capacity constraint set at 50 Mt as seen in Fig. 26. Fig. 27 shows the average ore bitumen grade per period for the SGA schedule.

Table 11. Scheduling results for the SGA.

Period	Total tonnage (Mt)	Ore tonnage (Mt)	Average ore bitumen grade (%m)
1	149.91	29.98	9.80
2	149.84	33.96	10.14
3	149.94	37.92	9.79
4	149.62	39.98	9.53
5	149.72	49.95	9.94
6	149.87	49.93	10.04
7	149.82	49.82	9.56
8	149.11	49.91	10.16
9	149.32	49.92	9.93
10	149.33	49.90	9.82

11	149.28	49.96	9.79
12	149.33	49.89	9.39
13	149.36	49.90	9.81
14	149.41	49.93	10.22
15	149.28	47.92	10.05
16	149.46	44.86	9.78
17	149.64	44.94	10.30
18	149.25	44.86	9.84
19	149.55	44.92	9.62
20	149.30	28.64	9.17

Table 12. Solution comparison between the SMILP model with CPLEX and the SGA.

Parameter	SMILP model with CPLEX	SGA
Number of blocks	1569	1569
Tonnage mined (Mt)	-	2990
Ore processed (Mt)	-	897
NPV ($M)	-	10054
Runtime (hours)	*Terminated after 28 days	1.5
Optimality gap from relaxed LP (%)	101	10.6

* Job scheduling policies - Compute Canada Cedar Cluster

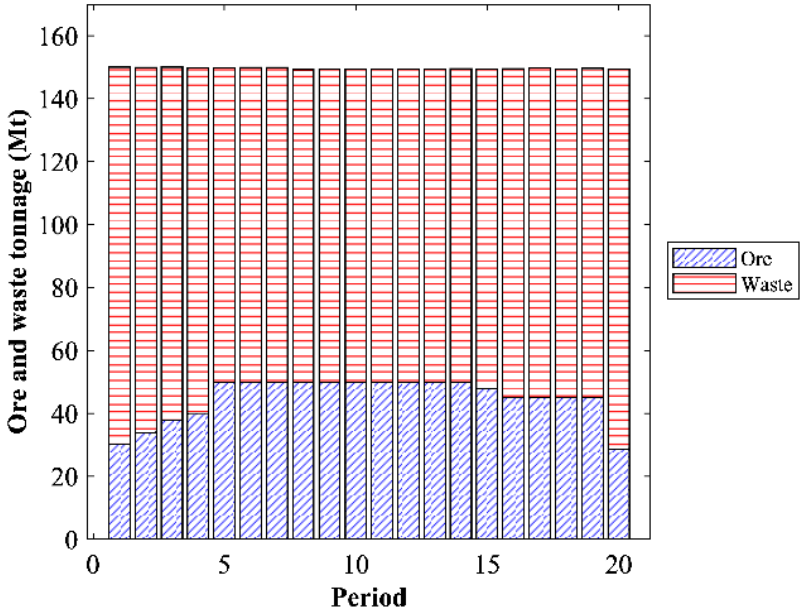

Figure 26. Total tonnage mined for the SGA.

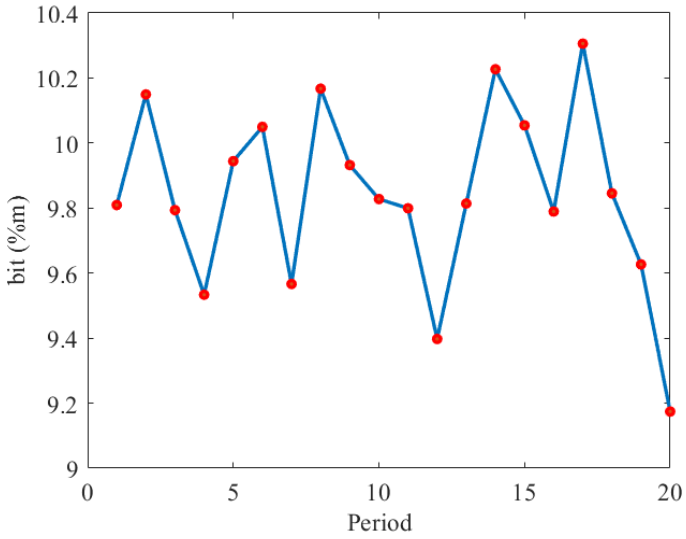

Figure 27. Average ore bitumen grade for the SGA.

6. Conclusions and Future Work

In this research, the authors presented a GA framework for solving the OPPS problem and evaluated it with two case studies. A multiple chromosome-encoding scheme was proposed and implemented to represent the blocks and periods of extractions. Deterministic and stochastic production scheduling scenarios were investigated in this research; the deterministic production schedule with a block model based on ordinary kriging, and the stochastic production schedule based on SGS orebody realizations. The equally probable simulated orebody realizations capture the varying grade distribution in the mineralization of the deposit to allow for consideration of grade uncertainty. Due to the multiple chromosome encoding scheme, the GA was capable of fractional block processing. In the implementation of the GA, the relaxed LP solution was used as the upper bound to estimate the optimality gap for the GA solution. The solutions from the GA were compared with that from mathematical programming models with CPLEX solver to assess the practicality of the generated schedules as well as the NPV and computational efficiency.

For deterministic production scheduling in Case study 1 Scenario 1, the NPV of the DGA schedule was 5.1% less than that of the MILP model with CPLEX schedule while there was a 53.7% improvement in computational time comparing the DGA solution runtime to that of the MILP model with CPLEX solution runtime. For the second scenario based on stochastic production scheduling, while the NPV of the SGA schedule was 5.3% less than that of the SMILP model with CPLEX schedule, there was 75.2% improvement in computational efficiency comparing the SGA solution runtime to that of the SMILP model with CPLEX solution runtime. It is also important to note that the NPV generated by the SGA schedule was 16.3% better than the NPV from the DGA schedule. For Case Study 2, the solution from the SMILP model with CPLEX was terminated after 28 days at 101% gap while the SGA generated solution in 1.5 hours at 10.6% optimality gap. In both case studies, the GA models produced uniform schedules over the life of mine, although the NPVs were lower than that from the MILP and SMILP models with CPLEX solver.

In summary, the results generated by the GA were encouraging in the area of computational efficiency. In cases where the mathematical programming model solution runtime is lengthy or intractable, GA proves to be capable of generating a ‘good’ solution at a reasonable runtime. The authors’ ongoing research aims at extending the GA model to include stockpiling and investigating the best combination of genetic parameters to improve the GA computational time and solution quality.

7. References

[1] Albor, F. and Dimitrakopoulos, R. (2009). Stochastic mine design optimisation based on simulated annealing: pit limits, production schedules, multiple orebody scenarios and sensitivity analysis. *Mining Technology*, *118* (2), 79-90.

[2] Ahmadi, M. R. and Shahabi, R. S. (2018). Cutoff grade optimization in open pit mines using genetic algorithm. *Resources Policy,* 55(March 2018), 184-191.

[3] Ahn, C. W. and Ramakrishna, R. S. (2003). Elitism-based compact genetic algorithms. *IEEE Transactions on Evolutionary Computation,* 7(4), 367-385.

[4] Alipour, A., Asghar, A., Jafari, A. and Tavakkoli-Moghaddam, R. (2017). A genetic algorithm approach for open-pit mine production scheduling. *International Journal of Mining and Geo-Engineering,* 51(1), 47-52.

[5] Alipour, A., Khodaiari, A. A., Jafari, A. and Tavakkoli-Moghaddam, R. (2020). Production scheduling of open-pit mines using genetic algorithm: a case study. *International Journal of Management Science and Engineering Management,* 15(3), pp. 176-183.

[6] Amponsah, S., Eme, P., Takouda, P.M and Ben-Awuah, E. (2021). Genetic algorithm for open pit mine production scheduling optimisation problems. In Proceedings of the 40th Conference on Application of Computers and Operations Research in the Minerals Industry (APCOM 2021). Johannesburg, South Africa, pp. 335-346.

[7] Anani, A.K., (2016). Applications of simulation and optimization techniques in optimizing room and pillar mining systems. Missouri University of Science and Technology. Pages 241

[8] Anirudh, S. (2020). Sliding window algorithm Available at https://redquark.org/cotd/sliding_window/. [accessed 1 Dec. 2020].

[9] Askari-Nasab, H., Awuah-Offei, K. and Eivazy, H. (2010). Large-scale open pit production scheduling using mixed integer linear programming. International Journal of Mining and Mineral Engineering, 2(3), 185-214.

[10] Askari-Nasab, H., Pourrahimian, Y., Ben-Awuah, E. and Kalantari, S., (2011). Mixed integer linear programming formulations for open pit production scheduling. Journal of Mining Science, 47(3), 338-359.

[11] Badiozamani, M.M. and Askari-Nasab, H., (2010). Lagrangian relaxation of the MILP open pit production scheduling formulation. *Mining Optimization Laboratory*, p.157.

[12] Ben-Awuah, E. and Askari-Nasab, H. (2011). Oil sands mine planning and waste management using mixed integer goal programming. *International Journal of Mining, Reclamation and Environment,* 25(3), 226-247.

[13] Ben-Awuah, E., Askari-Nasab, H., Awuah-Offei, K. and Awuah-Offei, K. (2012). Production scheduling and waste disposal planning for oil sands mining using goal programming. *Journal of Environmental Informatics,* 20(1), 20-33.

[14] Ben-Awuah, E., Richter, O., Elkington, T. and Pourrahimian, Y., (2016). Strategic mining options optimization: Open pit mining, underground mining or both. *International Journal of Mining Science and Technology*, *26*(6), pp.1065-1071.

[15] Ben-Awuah, E., Askari-Nasab, H., Maremi, A. and Seyed Hosseini, N. (2018). Implementation of a goal programming framework for production and dyke material planning. *International Journal of Mining, Reclamation and Environment,* 32(8), 536-563.

[16] Bianchi, L., Dorigo, M., Gambardella, L. M. and Gutjahr, W. J. (2009). A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing,* 8(2), 239-287.

[17] Blickle, T. and Thiele, L. (1995). A mathematical analysis of tournament selection. In Proceedings of 6th International Conference on Genetic Algorithms, San Francisco, California, 95, pp. 9-15.

[18] Caccetta, L. and Hill, S. P. (2003). An application of branch and cut to open pit mine scheduling. *Journal of Global Optimization,* 27(2-3), 349-365.

[19] Cantú-Paz, E., 1998. A survey of parallel genetic algorithms. *Calculateurs paralleles, reseaux et systems repartis*, 10(2), 141-171.

[20] Cullenbine, C., Wood, R.K. and Newman, A., (2011). A sliding time window heuristic for open pit mine block sequencing. *Optimization letters*, *5*(3), pp.365-377.

[21] Dagdelen, K. (1985). Optimum multi-period open pit mine production scheduling by Lagrangian parameterization. PhD Thesis, University of Colorado, Colorado, Pages 325.

[22] Dagdelen, K. and Johnson, T. (1986). Optimum open pit mine production scheduling by Lagrangian parameterization. In Proceedings of 19th International Symposium on the Application of Computers and Operations Research in the Mineral Industry, Society for Mining, Metallurgy & Exploration, Pennsylvania State University, USA, pp. 127-142 .

[23] Dasgupta, D. and Michalewicz, Z. (1997). Evolutionary algorithms— An Overview. In: Dasgupta, D., Michalewicz, Z. (eds) Evolutionary Algorithms in Engineering Applications. Springer, Berlin, Heidelberg, pp. 3-28. https://doi.org/10.1007/978-3-662-03423-1_1.

[24] Davis, L. (1985). Applying adaptive algorithms to epistatic domains. In Proceedings of 9th International Joint Conference on Artificial Intelligence, 85, pp. 162-164.

[25] Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation,* 6(2), 182-197.

[26] Denby, B. and Schofield, D. (1994). Open-pit design and scheduling by use of genetic algorithms. *Transactions of the Institution of Mining and Metallurgy. Section A. Mining Industry,* 103(1994), A21-A26.

[27] Deneubourg, J. L., Aron, S., Goss, S. and Pasteels, J. M. (1990). The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior,* 3(2), 159-168.

[28] Deutsch, C. and Journel, A. (1998). *GSLIB: Geostatistical Software Library.* Oxford Univ.Press, New York, 2nd ed, Pages 384.

[29] Dimitrakopoulos, R. and Ramazan, S. (2004). Uncertainty based production scheduling in open pit mining. *Transactions of the Society for Mining, Metallurgy, and Exploration*, *316* (3), 106-112.

[30] Dimitrakopoulos, R. and Ramazan, S. (2008). Stochastic integer programming for optimising long term production schedules of open pit mines: methods, application and value of stochastic solutions. *Mining Technology,* 117(4), 155-160.

[31] Dorigo, M. and Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science,* 344(2-3), 243-278.

[32] Dorigo, M., Caro, G. D. and Gambardella, L. M. (1999). Ant algorithms for discrete optimization. *Artificial life,* 5(2), 137-172.

[33] Dumitrescu, D., Lazzerini, B., Jain, L. C. and Dumitrescu, A. (2000). *Evolutionary computation*, Boca Raton, FL CRC press.

[34] Eshelman, L. J., Caruana, R. A. and Schaffer, J. D. (1989). Biases in the crossover landscape. In Proceedings of the 3rd International Conference on Genetic Algorithms. Morgan Kaufmann,San Mateo,pp. 10-19.

[35] Falkenauer, E., (1999). The worth of the uniform [uniform crossover]. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406) (Vol. 1, pp. 776-782). IEEE.

[36] Feng, Y., Wang, G.-G., Deb, S., Lu, M. and Zhao, X.-J. (2017). Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization. *Neural Computing and Applications,* 28(7), 1619-1634.

[37] Franco-Sepúlveda, G., Del Rio-Cuervo, J. C. and Pachón-Hernández, M. A. (2019). State of the art about metaheuristics and artificial neural networks applied to open pit mining. *Resources Policy,* 60, 125-133.

[38] Franco-Sepúlveda, G., Del Rio-Cuervo, J. C. and Pachón-Hernández, M. A. (2019). State of the art about metaheuristics and artificial neural networks applied to open pit mining. *Resources Policy,* 60, 125-133.

[39] Fouskakis, D. and Draper, D. (2002). Stochastic optimization: a review. International Statistical Review, 70(3), 315-349.

[40] Gen, M. and Cheng, R. (1997). Genetic Algorithms and Engineering Design. New York: John Wiley&Sons. Inc.

[41] Gen, M., Cheng, R. and Lin, L. (2008). *Network models and optimization: Multiobjective genetic algorithm approach*, Springer Science & Business Media.

[42] Geovia Dassault Systems. (2017). GEOVIA Whittle software (Version 4.7.1). Vancouver, BC, Canada: Geovia Dassault Systems.

[43] Gershon, M. E. (1983). Optimal mine production scheduling: evaluation of large scale mathematical programming approaches. *International journal of mining engineering,* 1(4), 315-329.

[44] Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & operations research,* 13(5), 533-549.

[45] Glover, F. (1990). Artificial intelligence, heuristic frameworks and tabu search. *Managerial and Decision Economics,* 11(5), 365-375.

[46] Goldberg, D. E. (1990). A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Systems,* 4(4), 445-460.

[47] Goldberg, D. E. and Deb, K. (1991). A Comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*, Morgan Kaufmann, San Mateo, California, 1, 69-93.

[48] Goldberg, D. E. and Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine Learning 3*(2/3), 95–99.

[49] Goldberg, D. E. and Lingle, R. (1985). Alleles, loci, and the traveling salesman problem. In Proceedings of International Conference on Genetic Algorithms and their Applications, Carnegie-Mellon University, Pittsburgh, PA, pp. 154-159.

[50] Goodfellow, R. and Dimitrakopoulos, R. (2013). Algorithmic integration of geological uncertainty in pushback designs for complex multiprocess open pit mines. *Mining Technology,* 122(2), 67-77.

[51] Grefenstette, J., Gopal, R., Rosmaita, B. and Van Gucht, D. (1985). Genetic algorithms for the traveling salesman problem. In Proceedings of 1st International Conference on Genetic Algorithms and their Applications, Lawrence Erlbaum, 160, pp.160-168.

[52] Holland, J. H. (1992). Genetic Algorithms. *Scientific American,* 267(1), 66-73.

[53] Horst, R. and Hoang, T. (1996). Global optimization: deterministic approaches. Springer, New York, 3rd ed, Pages 727.

[54] Hu, X.-B. and Di Paolo, E. (2009). An efficient genetic algorithm with uniform crossover for the multi-objective airport gate assignment problem. *Multi-objective memetic algorithms.* Berlin, Heidelberg: Springer, pp. 71-89.

[55] IBM, ILOG (2017). CPLEX reference manual and software. Ver. 12.8, New York, USA Available at: https://www.ibm.com/docs/en/icos/12.8.0.0?topic=parameters-relative-mip-gap-tolerance [Accessed: 1 January 2022].

[56] Job scheduling policies - Compute Canada Document [online]. (2022). CC Doc. Available from: https://docs.alliancecan.ca/wiki/Job_scheduling_policies.

[57] Johnson, T. B. (1969). Optimum open-pit mine production scheduling. In proceedings of 8th International Symposium on the Application of Computers and Operations Research in the Mineral Industry, Salt Lake City, Utah, pp. 539-562.

[58] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, 4, pp. 1942-1948.

[59] Khan, A. and Niemann-Delius, C. (2014) 'Production scheduling of open pit mines using particle swarm optimization algorithm', Advances in Operations Research. Edited by I. Kacem, 2014, p. 208502. doi:10.1155/2014/208502.

[60] Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. (1983). Optimization by simulated annealing. *science,* 220(4598), 671-680.

[61] Koushavand, B., Askari-Nasab, H. and Deutsch, C. V. (2014). A linear programming model for long-term mine planning in the presence of grade uncertainty and a stockpile. *International Journal of Mining Science and Technology,* 24(4), 451-459.

[62] Krige, D. (1951). A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Chemical, Metallurgical and Mining Society of South Africa*, *52* (6), 119-139.

[63] Kumral, M. and Dowd, P. (2005). A simulated annealing approach to mine production scheduling. *Journal of the Operational Research Society,* 56(8), 922-930.

[64] Kumar, M., Husain, M., Upreti, N. and Gupta, D., (2010). Genetic algorithm: Review and application. *International Journal of Information Technology and Knowledge Management*, 2, 451−454.

[65] Lamghari, A. and Dimitrakopoulos, R. (2012). A diversified tabu search approach for the open-pit mine production scheduling problem with metal uncertainty. *European Journal of Operational Research,* 222(3), 642-652.

[66] Lerchs, H. and Grossmann, I. (1965). Optimum design of open pit mines. *Canadian Institute of Mining*, 58 (3), 47-54

[67] Louis, S. J. and Rawlins, G. J. (1991). Designer Genetic Algorithms: Genetic Algorithms. In Proceedings of 4th International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, pp. 53-60.

[68] Marinho de Almeida, A., (2013). Surface constrained stochastic life-of-mine production scheduling. MSc Thesis, McGill University, Montreal. Pages 119.

[69] Mathworks Inc. (2020). MATLAB Software. Ver. R2020a, Massachusetts, USA.

[70] Mbadozie, O. (2020). Incorporating Grade Uncertainty in Oil Sands Mine Planning and Waste Management Using Stochastic Programming. MASc Thesis, Laurentian University, Sudbury. Pages 142.

[71] Mbadozie, O., Ben-Awuah, E., and Maremi, A. (2022). A stochastic mixed integer linear programming framework for oil sands mine planning and waste management in the presence of grade uncertainty. *CIM Journal*, 13(1), 16–37. https://doi.org/10.1080/19236026.2021.2024959.

[72] Michalewicz, Z. (1995a). Genetic algorithms, numerical optimization, and constraints. In Proceedings of 6th International Conference on Genetic Algorithms, Morgan Kauffman, San Mateo, pp. 151–158.

[73] Michalewicz, Z. (1995b). A survey of constraint handling techniques in evolutionary computation methods. In Proceedings of 4th Annual Conference on Evolutionary Programming, MIT Press, Cambridge, MA, pp. 135–155.

[74] Milani, G. (2016). A Genetic algorithm with zooming for the determination of the optimal open pit mines layout. *The Open Civil Engineering Journal,* 10(1), 301-322.

[75] Mirjalili, S. (2019). Evolutionary algorithms and neural networks theory and applications. *Studies in Computational Intelligence* 780. Springer, Berlin. doi: 10.1007/978-3-319-93025-1 Available at: http://www.springer.com/series/7092. [accessed 18 Dec. 2020]

[76] Mirjalili, S., Song Dong, J., Sadiq, A. S. and Faris, H. (2020). Genetic Algorithm: Theory, Literature Review, and Application in Image Reconstruction. In: Mirjalili, S., Song Dong, J., Lewis, A. (eds) Nature-Inspired Optimizers. Studies in Computational Intelligence, vol 811. Springer, Cham. https://doi.org/10.1007/978-3-030-12127-3_5. [Accessed 18 Dec. 2020].

[77] Myburgh, C. and Deb, K. (2010). Evolutionary algorithms in large-scale open pit mine scheduling categories and subject descriptors. In Proceedings of 12th Annual Conference on Genetic and Evolutionary Computation, Portland, Oregon, USA, pp. 1155–1162.

[78] Nakano, R. and Yamada, T. (1991). Conventional genetic algorithm for job shop problems. In Proceedings of 4th International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, pp. 474-479.

[79] Oliver, I., Smith, D. and Holland, J. R. (1987). Study of permutation crossover operators on the traveling salesman problem. Genetic algorithms and their applications. In Proceedings of 2nd International Conference on Genetic Algorithms: July 28-31, Massachusetts Institute of Technology, Cambridge, MA, Hillsdale, NJ: L. Erlhaum Associates, pp. 224–230.

[80] Ota, R., Martinez, L. and R&O Analytics, (2017). SimSched Direct Block Scheduler: A new practical algorithm for the open pit mine production scheduling problem. In: 38th APCOM, August 2017, Golden, CO, USA [online]. [Viewed 29 May 2022]. Available from: https://www.researchgate.net/publication/320179762_SimSched_Direct_Block_Scheduler_A_new_practical_algorithm_for_the_open_pit_mine_production_scheduling_problem

[81] Paithankar, A. and Chatterjee, S. (2019). Open pit mine production schedule optimization using a hybrid of maximum-flow and genetic algorithms. *Applied Soft Computing,* 81, 105507, doi: 10.1016/j.asoc.2019.105507.

[82] Richardson, J. T., Palmer, M. R., Liepins, G. E. and Hilliard, M. R. (1989). *Some guidelines for genetic algorithms with penalty functions*. In Proceedings of 3rd International Conference on Genetic Algorithms (held in Arlington), San Mateo: Morgan Kaufmann, pp. 191–197.

[83] Ruiseco, J. R., Williams, J. and Kumral, M. (2016). Optimizing ore–waste dig-limits as part of operational mine planning through genetic algorithms. *Natural Resources Research,* 25(4), 473-485.

[84] Sabour, S. A. and Dimitrakopoulos, R. (2011). Incorporating geological and market uncertainties and operational flexibility into open pit mine design. *Journal of Mining Science,* 47(2), 191-201.

[85] Semenkin, E. and Semenkina, M. (2012). Self-configuring genetic algorithm with modified uniform crossover operator. In Proceedings of International Conference in Swarm Intelligence, Berlin, Heidelberg.: Springer, pp. 414-421.

[86] Senécal, R. and Dimitrakopoulos, R. (2020). Long-term mine production scheduling with multiple processing destinations under mineral supply uncertainty, based on

multi-neighbourhood Tabu search. *International Journal of Mining, Reclamation and Environment,* 34(7), 459-475.

[87] Sharma, P. and Gargi, R. (2014). Performance analysis of different selection techniques in genetic algorithm. *International Journal of Science and Research*, 3(8), 2042-2046. Available at: www.ijsr.net [accessed 18 Dec. 2020].

[88] Shishvan, M. S. and Sattarvand, J. (2015). Long term production planning of open pit mines by ant colony optimization. *European Journal of Operational Research,* 240(3), 825-836.

[89] Sivanandam, S. N. and Deepa, S. N. (2007). *Principles of soft computing (2nd Edition)*. John Wiley & Sons.

[90] Sivanandam, S. N. and Deepa, S. N. (2008). Genetic algorithms. *Introduction to genetic algorithms.* Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 15–37. doi: 10.1007/978-3-540-73190-0_2.[accessed 18 Dec. 2020].

[91] Syswerda, G. (1991). Scheduling optimization using genetic algorithms. L Davis (Ed.), Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, pp. 332-349.

[92] Tabesh, M. and Askari-Nasab, H. (2011). Two-stage clustering algorithm for block aggregation in open pit mines. *Mining Technology,* 120(3), 158-169.

[93] Tolouei, K., Moosavi, E., Tabrizi, A. H. B., Afzal, P. and Bazzazi, A. A. (2020). An optimisation approach for uncertainty-based long-term production scheduling in open-pit mines using meta-heuristic algorithms. *International Journal of Mining, Reclamation and Environment,* 35(2), 115-140.

[94] Tsutsui, S., Yamamura, M. and Higuchi, T. (1999). Multi-parent recombination with simplex crossover in real coded genetic algorithms. In Proceedings of 1st Annual Conference on Genetic and Evolutionary Computation, 1, pp. 657-664.

[95] Vallejo, M. N. and Dimitrakopoulos, R. (2019). Stochastic orebody modelling and stochastic long-term production scheduling at the KéMag iron ore deposit, Quebec, Canada. *International Journal of Mining, Reclamation and Environment,* 33(7), 462-479.

[96] Villalba Matamoros, M. E. and Kumral, M. (2018). Calibration of Genetic Algorithm Parameters for Mining-Related Optimization Problems. *Natural Resources Research,* 28(2), 443-456.

[97] Weisstein, Eric W (2022). "NP-Hard Problem." From MathWorld--A Wolfram Web Resource. https://mathworld.wolfram.com/NP-HardProblem.html. [accessed 18 May. 2022].

[98] Zeleny, M., (1980). Multiple objectives in mathematical programming: Letting the man in. *Computers & Operations Research*, *7*(1-2), pp.1-4.

[99] Zhang, Y., Cheng, Y. and Su, J. (1993). Application of goal programming in open pit planning. *International Journal of Surface Mining and Reclamation,* 7(1), 41-45.