Development of New Metaheuristic Tools for Long Term Production Scheduling of Open Pit Mines

Von der Fakultät für Georessourcen und Materialtechnik
der Rheinisch -Westfälischen Technischen Hochschule Aachen

zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften

genehmigte Dissertation
vorgelegt von **M.Sc.**

Asif Khan

aus Swabi, Pakistan

Berichter: Univ.-Prof. Dr.-Ing. Christian Niemann-Delius
Univ.-Prof. Dr.rer.nat.habil. Marco Lübbecke

Tag der mündlichen Prüfung: 31. August 2015

Acknowledgements

I would like to express my sincere gratitude to my supervisor Prof. Dr.-Ing.Christian Niemann-Delius for his guidance, support and for providing me with all the necessary facilities to carry out this research work.

I would like to express my special appreciation and thanks to my co-supervisor Prof. Dr. Marco Lübbecke for his time to evaluate the final manuscript of my thesis and for his valuable comments and suggestions. I would also like to thank Dr. Javad Sattarvand for his constructive suggestions during the planning process of this research work.

I am also grateful to all my colleagues at the Institute of Surface Mining and Drilling (BBK3), RWTH Aachen University for helping me in various ways to the successful completion of this Thesis.

I would like to thank all of my family for their support and encouragement.

Finally, I would like to thank Higher Education Commission of Pakistan and DAAD for funding this research.

ZUSAMMENFASSUNG

Langfristige Produktionsplanung in Tagebauen ist eine großes und komplexes Optimierungsproblem, das seit 1960 ausgiebig in der Fachliteraturdiskutiert wird. Es ist darauf gerichtet, eine zeitliche Abfolge der Abbaublöcke bei der Gewinnung von mineralischen Rohstoffen im Tagebau zu ermitteln, die den Nettobarwert (Net Present Value, NPV) optimiert und gleichzeitig eine Reihe von physischen und betrieblichen Einschränkungen (constrains) erfüllen kann. Dabei wird im Allgemeinen die Modellierung des Erzkörpers in einem s.g.n. Blockmodell als ein Basis-Eingabe verwendet. Ein solches Blockmodell unterteilt eine Lagerstätte von gleichmäßig zugeschnittenen Blöcken in eine dreidimensionale Anordnung. In einem realen Großtagebau wird das Modell aus Tausende bis zu Millionen von Einzelblöcken gebildet, die in der Regel über einen Zeitraum von 5 bis 30 Jahren sequenziert werden müssen. Darauf entsteht ein großes kombinatorisches Optimierungsproblem (large combinatorial optimization problem).

Die Arbeit entwickelt einen Handlungsrahmen bzw. ein Ablaufschema mit dem das genannte rechnerisch extrem umfangreiche Problem der Abbauoptimierung mit niedrigem bis mäßig Rechenaufwand gelöst werden kann. Um das Problem der Sequenzierung effizienter zu lösen wird es in eins der optimalen Teufebestimmung umgewandelt. Diese soll die optimale Tiefe für eine bestimmte Säule des Blockmodells, die in einem bestimmten Zeitraum abzubauen ist, bestimmen. Auf diese Weise wird der auf eine gesamte Ebene bezogene rechenintensive Entscheidungsprozess vermieden. Das Ablaufschema sieht dann die Verwendung von verschiedenen metaheuristische Verfahren vor, um den Raum für optimale bzw. nahezu optimale Lösungen zu bestimmen und so das Teufenproblem und damit das Problem der Sequenzierung zu lösen. Dazu werden verschiedene spezifische Operatoren wie „solution encoding“,“ back transform“, „slope normalization“, etc. verwendet. Der vorgeschlagene Ablaufschema kann das Problem der Sequenzierung sowohl ohne als auch unter Einschluss der Unsicherheiten, die aus der geologischen und mineralogischen Qualitätsbestimmung der Blöcke entsteht, behandeln.

Zum Vergleich der Tauglichkeit der verschiedenen metaheuristischen Verfahren Particle Swarm Optimization (PSO), Bat Algorithm (BA) und Differential Evolution(DE) wurden drei Fallstudien durchgeführt. Ziel der Fallstudien war es, die Fähigkeiten und die Effizienz des vorgeschlagenen Ablaufschemas und der jeweiligen Metaheuristiken zusammen mit ihren verschiedenen Varianten zu bestimmen. Der Vergleich des entwickelten Ablaufschemas und der metaheuristischen Verfahren mit solchen Ergebnissen, die für einfache Problemstellungen unter Verwendung von CPLEX ermittelt wurden zeigt, dass erstere sowohl was Rechenzeit als auch prozentuale Abweichung und Standardabweichung betrifft Lösungen deutlich schneller und in akzeptabler Qualität liefert.

ABSTRACT

Long term production scheduling of open pit mines is a large scale and complex optimization problem that has been extensively discussed in the technical literature since 1960s. It seeks to specify such an extraction sequence of ore and waste materials from the ground that maximizes the Net Present Value (NPV) of the operation while satisfying a set of physical and operational constraints. Block model representation of the orebody is commonly used as a basic input for this purpose. The block model discretize the ore deposit into a three dimensional array of regular sized blocks. A real sized open pit mine may contain thousands to millions of blocks of these blocks that may be needed to be scheduled over a time horizon typically ranging from 5 to 30 years which makes it a large combinatorial optimization problem.

This thesis presents a framework that aims to handle the above mentioned computationally expensive problem of the open pit mines with low to moderate computational cost. To handle the scheduling problem more efficiently the proposed framework converts it into optimum depth determination problem. This so called optimum depth determination problem aims to find the optimum depth to be mined along a particular column of the block model in a certain period. In this way this framework helps to avoid computationally expensive scheduling decisions making process on the block level. The framework then uses a real valued / continuous population based metaheuristic technique to search the solution space for finding optimum or near to optimum solution of this so called optimum depth determination problem and consequently of the production scheduling problem. Different framework specific operators such as solution encoding, back transform, slope normalization etc. are also used during this process. The proposed framework can handle the production scheduling problem with or without the condition of grade uncertainty.

Three different case studies have been carried out using Particle Swarm Optimization (PSO), Bat Algorithm (BA) and Differential Evolution (DE). The aim of these case studies was to determine the capabilities and efficiency of the proposed framework and of the respective metaheuristic technique along with of their different variants. By making comparison with the

results obtained using CPLEX in terms of computational time and solution quality it was learnt that the proposed procedure can produce results of reasonable quality in relatively shorter period of time with smaller % gap and standard deviation.

The following research papers have already been published or in the review process using some parts of this work:

- Asif Khan and Christian Niemann-Delius, "*Production Scheduling of Open Pit Mines Using Particle Swarm Optimization Algorithm*" , Advances in Operations Research, vol. 2014, Article ID 208502, 2014. DOI:10.1155/2014/208502

- Asif Khan and Christian Niemann-Delius, *"Long Term Production Scheduling of Open Pit Mines using Particle Swarm and Bat Algorithms under grade uncertainty"*, The Southern African institute of Mining and Metallurgy (In review).

TABLE OF CONTENTS

LIST OF FIGURS

LIST OF TABLES

LIST OF ABBREVIATIONS

NPV	Net Present Value
GA	Genetic Algorithm
SA	Simulated Annealing
ACO	Ant Colony Optimization
PSO	Particle Swarm optimization
BA	Bat Algorithm
DE	Differential Evolution
UPL	Ultimate Pit Limit
LP	Linear Programming
IP	Integer Programming
MILP	Mixed Integer Linear Programming
LG	Lerchs & Grossmann's Algorithm
SGS	Sequential Gaussian Simulations
FIPS	Fully Informed Particle Swarm
GCPSO	Guaranteed Convergence Particle Swarm Optimization
QPSO	Quantum Particle Swarm Algorithm
AMPSO	Angle Modulated Particle Swarm Optimization
MBA	Modified Bat Algorithm
BBA	Binary Bat Algorithm
CP	Constraint Programming
LTPS	Long Term Production Scheduling
CR	Cross Over

Chapter 1. Introduction

Today's high tech society requires a consistent and sustainable supply of different raw materials to meet its current and projected future needs. Different surface and underground mining techniques are commonly used to mine these materials. Surface mining, being the major contributor of currently produced minerals have many advantages over underground mining in terms of productivity, use of large equipment size, recovery, crew safety, labor requirements etc. Surface mining methods can be categorized into strip, alluvial, In-situ and open pit mining methods [1]. In open pit mining the mining process starts by a digging a pit in the ground surface to remove the overburden and to access the underlying mineralized material. The process starts with a small pit that grows into larger and larger pit till the mining operation ends. The designing and production scheduling of these open pit mines is a complex and important problem that has been extensively discussed in the literature since 1960s. It commonly aims to define such an extraction sequence of the mineralized material from the ground that produces maximum possible discounted profit (NPV) while satisfying a set of physical and operational constraints.

1.1. Problem Definition

Long term production scheduling of the open pit mines is a well known and well discussed problem in the mining industry. The approaches that are being used to deal with this problem can be broadly divided into conventional / deterministic and uncertainty based approaches. In the conventional approaches the planning process usually starts with the construction of a geologic block model which divides the ore body and the surrounding rock into three dimensional arrays of regular usually identical sized blocks. A set of attributes such as grade, specific gravity etc. are then assigned to each one of these blocks estimated using some form of spatial interpolation technique e.g. krigging, inverse distance method etc. and the drillhole sample data. These estimated block attributes are then used to define the economic values of these blocks and their processing destination once mined. These economic and geologic block models become the basic input for the subsequent production scheduling problem. One key and important drawback of this approach is its assumption that all the input parameters are known with certainty while on the

contrary a certain degree of uncertainty is almost always associated with these parameters, ignoring them may result into unrealistic and false scheduling decisions.

The uncertainty in the input parameters may be caused by different technical (geological, mining), financial or environmental factors involved in the planning process of the open pit mines. The uncertainty caused by geological factors which is commonly termed as geological or grade uncertainty is considered to be the most important source of uncertainty for the planning and scheduling process of open pit mines. This grade / geological uncertainty is caused by the fact that the grade values of the individual blocks are estimated using very sparse drillhole sample data, and usually a significant and variable level of uncertainty is associated with each on one of these estimated values. Geostatistical conditional simulation techniques provides a framework to quantify this grade related uncertainty by generating multiple equiprobable simulated realizations of the orebody [2, 3]. The availability of these techniques provides the opportunity to integrate this grade related uncertainty into the scheduling process for generating better production schedule in terms of maximum achievable NPV and meeting the yearly production targets. Over the recent years different stochastic programming models for integrating grade uncertainty into the scheduling process have been proposed in the technical literature to produce better and more realistic production schedules [4-6].

Irrespective of the approach that is being used to deal with this problem the main difficulty lies in solving it for a real sized open pit mine in reasonable amount of time. An ore deposit may contain thousands to millions of blocks that may have to be scheduled over a time horizon typically ranging from 5 to 30 years while satisfying different physical and operational constraints and with or without considering the potential uncertainty in the input data. This makes it a large combinatorial optimization problem that may be extremely difficult and computationally expensive to solve using the currently available optimization techniques on the currently available hardware.

In recent years a new class of computationally less expensive algorithms i.e. metaheuristic techniques have attracted the attention of several researchers to solve the mine design and

production scheduling problem such as Genetic Algorithms[7, 8] Simulated Annealing [9, 10], Ant Colony optimization [11, 12] etc. Though these techniques do not guarantee the optimality of the final solution that they produce but they can produce sufficiently good solutions with relatively less computational cost. This research focuses on the application of similar metaheuristic techniques to the open pit mine scheduling problem to explore their efficiency and capabilities.

1.2. Thesis aim and objectives

This thesis mainly focuses on the development of a framework that aims to handle the long term production scheduling problem of the open pit mines with low to moderate computational cost. To explore the solution space more efficiently the proposed framework turns the block scheduling problem into optimum depth determination problem. This so called optimum depth determination problem aims to find the optimum depth to be mined along a certain column of the block model in a certain period. In this way the proposed approach converts a discrete optimization problem into a real valued / continuous optimization problem and then employs a real valued / continuous population based metaheuristic technique to efficiently explore the search space for finding optimum or near to optimum solution of the problem. By using this approach this framework helps to avoid computationally expensive scheduling decision making process on block level and also makes the implementation process much simple. The main steps of the proposed framework can be summarized as:

i. Read input data: Block model with simulated or estimated grade values, economical and technical parameters

ii. Determine the Ultimate Pit Limits (UPL) of the open pit mine

iii. Initialize population of random feasible solutions

iv. Apply a specific real valued / continuous population based metaheuristic technique to explore the search space for finding better solutions along with framework specific operators such as: solution encoding, slope normalization, back transform etc.

v. Stop once the iterative process ends : Export the best solution

To explore the solution space for finding optimum or near to optimum solution of the scheduling problem, real valued / continuous variants of the following three different populations based metaheuristic techniques have been used:

- Particle Swarm Optimization (PSO)
- Bat Algorithm (BA)
- Differential Evaluation (DE)

The proposed framework can handle the long term production scheduling problem of the open pit mines with or without the condition of grade uncertainty. To quantify and integrate the condition of grade uncertainty into the optimization process multiple equally probable simulated realizations of the orebody have been used.

As this framework was not developed by keeping in mind a specific real valued population based metaheuristic technique therefore it provides a general platform that can be used to explore and compare the capabilities of different real valued / continuous population based metaheuristic technique when applied to open pit mine scheduling problem with or without the condition of grade uncertainty.

1.3. Outline of the thesis

The remainder of the thesis is organized as follows:

Chapter 2 presents a literature review and survey of the previous work including various techniques proposed so far for dealing with open pit mine design and production scheduling problem. A brief review of both deterministic and stochastic approaches for handling this problem are described in this chapter.

Chapter 3 provides a detailed description of the following three different population based metaheuristic techniques and of their different variants:

- Particle Swarm Optimization (PSO)
- Bat Algorithm (BA)
- Differential Evaluation (DE)

Chapter 4 presents a detailed description of the proposed framework for applying the above mentioned population based metaheuristic techniques to the long term production scheduling problem. This chapter also presents three different case studies to explore the efficiency and capability of the above mentioned metaheuristic techniques along with the proposed framework. Chapter 5 presents a summary of the work and a discussion on future research directions.

Chapter 2. Open Pit Mine Planning and Production Scheduling

Open pit mining is a surface mining technique, whereby the extraction of ore or waste starts downwards from the earth's surface by digging a pit. The process progresses with deeper and deeper pit until the mining operation ends. Traditionally the planning process of open pit mines starts with the construction of a geological block model. The geological block model discretizes the orebody and the surrounding rock into three dimensional arrays of regular size adjacent non-overlapping blocks. The size of these individual blocks is influenced by several important factors such as exploration drilling pattern, orebody geology, and mine equipment size etc.[13]. A set of attributes such as metal grades, specific gravity etc. are then assigned to each one of these blocks using some form of spatial interpolation technique such as krigging, Inverse distance weighting etc. and the drillhole sample data.

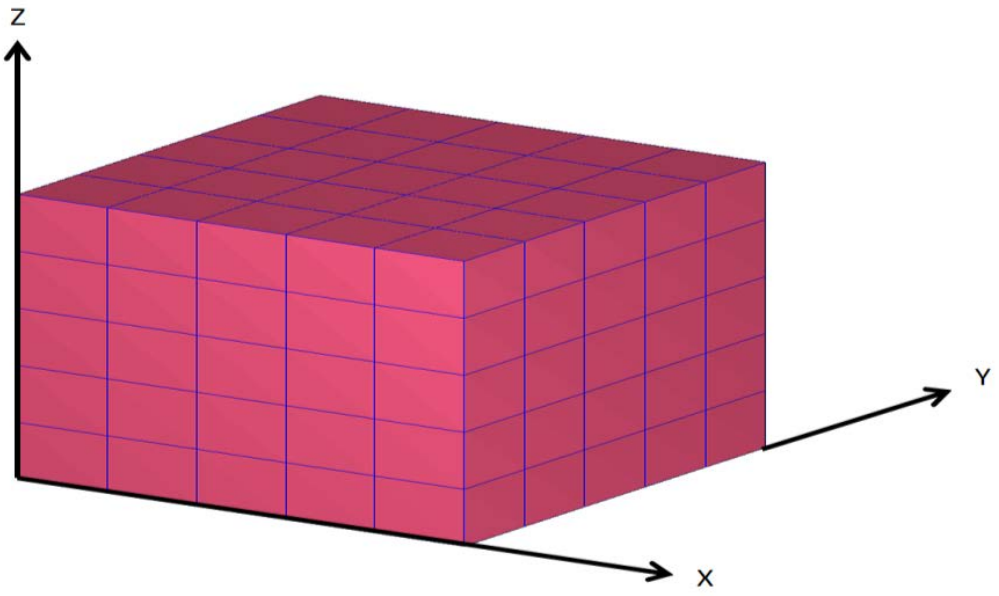

Figure 2.1: Block model representation of the ore deposit

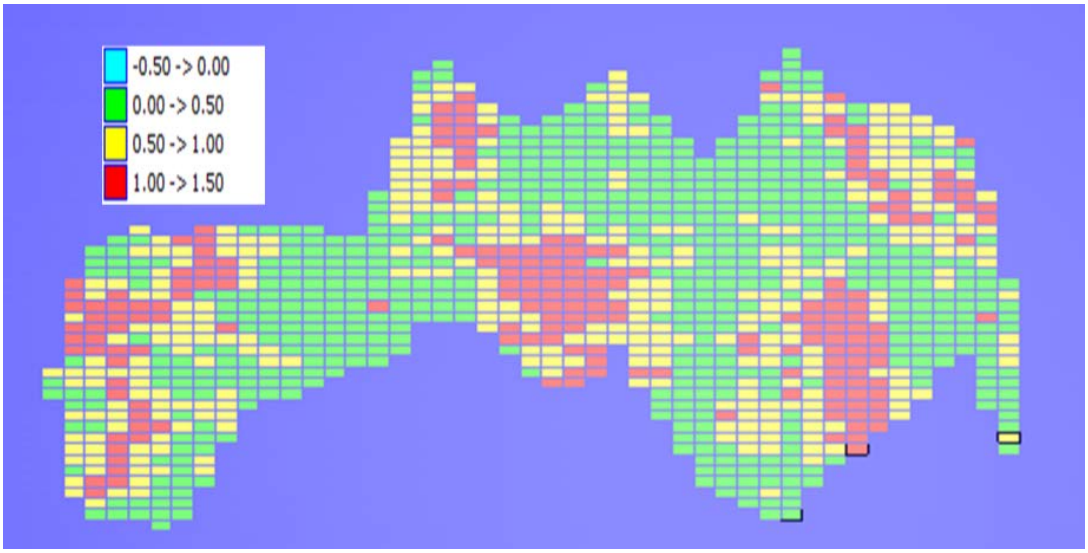

Figure 2.2: Cross sectional view of a copper (CU) deposit with estimated grade values in the North-South direction

The blocks are then divided into two groups i.e. waste and ore blocks. The blocks whose prospective profit exceeds their processing cost are categorized as an ore block to be sent for processing once mined while the rest are the waste blocks. An economic value is assigned to each one of these individual blocks by taking into account their respective estimated grade or metal content, prospective destination once mined (Mill, leach pad, waste dump), process recovery and the economic parameters such as metal price, mining cost and processing costs using by equation 2.1.

$$V_b = \begin{cases} NR_b - MC_b - PC_b & if \quad NR_b > PC_b \\ -MC_b & if \quad NR_b \leq PC_b \end{cases} \qquad 2.1$$

$$NR_b = T_b * g_b * R * (P - S)$$

Where V_b represents the economic value of block b, NR_b is the net revenue of block b, MC_b and PC_b represents the mining and processing costs of block b respectively and are the function of block tonnage, g_b represents estimated block grade, T_b represents block tonnage, R , P and S represents process recovery in percentage , metal price and selling or refining costs respectively.

This block model with predefined geological and economic parameters becomes basic input for the subsequent long term production planning and optimization process. The main objective of the which is to define such an extraction sequence of the blocks that maximize the net present value (NPV) of the mining operation while satisfying different physical and operational constraints such as precedence constraints, mining and processing capacity constraints, grade blending constraints etc. Achieving this goal may be quit challenging and extremely difficult as a real sized block model representing an ore deposit may contain thousand to millions of blocks that may have to be scheduled over a time horizon typically ranging from 5 to 30 years making it a large combinatorial optimization problem.

2.1. Mathematical Programming Formulation of the open pit mine production scheduling problem

Different mathematical programming formulations of the open pit mine production scheduling problem using different modeling approaches such as linear, Integer and Mixed Integer programming has been discussed in the literature. Johnson [14] presented a linear programming formulation of the open pit mine production problem and used Dantzig wolf decomposition principles to solve it. This model can simultaneously consider the time value of money, multiple processing streams and dynamic cutoff grade strategy but due to the linear nature of the decision variables may result into fractional block extraction causing the production schedule to be suboptimal or even infeasible when the blocks are mined as a whole [15]. Gershon [16] presented a mixed integer programming formulation of the sequencing problem. This model allows the partial extraction of the blocks if all the predecessor blocks are already being mined but has too many binary decision variables, making it computationally expensive to solve. In the recent years integer programming has quite frequently been used to model the production scheduling problem of the open pit with the objective to maximize the net present value of the operation while satisfying different physical and operational constraints, one such formulation similar to the one presented in [17] can be defined as:

Objective Function:

$$Maximize \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{d=1}^{D} v_{it}^{d}\, x_{it}^{d} \qquad 2.2$$

Where

$$v_{it}^{d} = \frac{V_i^d}{(1+dis)^t}$$

$\boldsymbol{v_{it}^{d}}$ Represents the discounted economic value of block i if mined in period t and sent to destination d

$$x_{it}^{d} = \begin{cases} 1 & if\ block\ i\ is\ mined\ in\ period\ t\ and\ sent\ to\ destination\ d \\ 0 & otherwise \end{cases}$$

dis = annual discount rate

N = Total number of blocks

i = Block index ($i = 1,2,3 \ldots \ldots N$)

T = Total Number of scheduling periods

t = scheduling period index ($t = 1,2,3 \ldots \ldots T$)

D = Total Number of destinations (e.g. waste dump, Mill, leach pad etc.)

d = Represents destination index (e.g. $d = 1$ if block is mined as waste, $d = 2$ if block is mined and sent to mill etc. ($d = 1,2,3 \ldots \ldots D$)

w_i = Block Tonnage

Subject to:

Reserve Constraint: These constraints are used to ensure that a block is mined at most once during the time horizon.

$$\sum_{d=1}^{D}\sum_{t=1}^{T} x_{it}^{d} \leq 1 \qquad \forall \quad i \tag{2.3}$$

Slope Constraints: These constraints are used to ensure that a block can only be mined if all its predecessors blocks or overlying blocks have already been mined in or before period ***t,*** to ensure safe slopes in all direction. The slope constraints are governed by the geotechnical conditions of a particular mine and must be obeyed to ensure that the resulted pit does not collapse in on itself.

$$\sum_{d=1}^{D} x_{it}^{d} - \sum_{d=1}^{D}\sum_{\tau=1}^{t} x_{j\tau}^{d} \leq 0 \quad \forall\ i\ ;\ \forall\ t \tag{2.4}$$

Where ***j*** ϵ (set of predecessors blocks of block ***i***)

Mining Capacity Constraints: The total material (ore and waste) mined during each period should be within the predefined upper and lower limits.

$$\sum_{d=1}^{D}\sum_{i=1}^{N} w_i * x_{it}^{d} \leq \overline{W}_t \quad \forall\ t \tag{2.5}$$

$$\sum_{d=1}^{D}\sum_{i=1}^{N} w_i * x_{it}^{d} \geq \underline{W_t} \quad \forall\ t \tag{2.6}$$

Where w_i represents the tonnage of block i, $\underline{W_t}$ and $\overline{W}_t$ are the upper and lower limits of the available mining capacity in period t respectively.

Processing Capacity Constraints: The ore material sent to each processing destination e.g. mill, leach pad etc. should be within predefined processing limits.

$$\sum_{i=1}^{N} w_i * x_{it}^{d} \geq \underline{O}^{d} \quad \forall\, t; \quad \forall\, d \tag{2.7}$$

$$\sum_{i=1}^{N} w_i * x_{it}^{d} \leq \overline{O}^{d} \quad \forall\, t; \quad \forall\, d \tag{2.8}$$

Where $\underline{O_t}$ and $\overline{O}_t$ are the upper and lower limits of the available processing capacity in period t at processing destination d respectively.

Metal Production Constraints: These constraints are used to ensure that the metal produced in each period should be within pre-defined limits.

$$\sum_{i=1}^{N} m_i * x_{it}^{d} \leq \overline{M}^{d} \quad \forall\, t = 1,2 \ldots . T; \tag{2.9}$$

$$\sum_{i=1}^{N} m_i * x_{it}^{d} * \geq \underline{M}^{d} \quad \forall\, t = 1,2 \ldots . T; \tag{2.10}$$

Where $\boldsymbol{m_i}$ represent the metal content of the block $\boldsymbol{i}$.

Though this formulation can holistically consider almost all the important aspects of the planning process of the open pit mines but solving it using the exact optimization algorithms such as branch and bound algorithm etc. may be extremely expensive. For instance defining production schedule for a moderate open pit mine containing 1, 000, 00 blocks needed to be scheduled over 10 periods with three possible destination e.g. waste dump, leach pad and mill may require 3, 000, 000 binary decision variables which is beyond the capabilities of currently available commercial softwares on the current hardware.

To make the sequencing problem computationally more tractable normally prior to define the optimum extraction sequence the ultimate pit limits are determined which substantially reduces the overall size of the main production scheduling problem by excluding the blocks that will never be the part of the optimum final schedule [18]. The solution to Ultimate Pit Limit (UPL) problem returns a collection of blocks that will results into maximum possible undiscounted profit while respecting the required pit slope constraints. The mathematical formulation of the ultimate pit limit problem can be defined as [19]:

$$Maximize \sum_{i=0}^{N} v_i x_i \qquad 2.11$$

Subject To:

$$x_i \leq x_j \quad \forall\, i \quad j \in P_i \qquad 2.12$$

$$x_i \in [0,1]$$

Where v_i represents the economic value of block i, N represents the total number of blocks in the block model, x_i represents a binary variable corresponding to block i which take the value 1 if the block i is inside the UPL and 0 otherwise and P_i represents the predecessor group of block i to ensure stable pit slopes. An important property of this formulation is the unimodular nature of the constraint matrix which ensure that the decision variable will take on integer value even if integrality condition on the decision variables is relaxed [16]. Graph theory based Lerchs and Grossmann algorithm [20] or Max- flow algorithms [14, 21] are commonly used for defining the ultimate pit limits. The block values and the pit slope greatly affect the size and shape of the optimal ultimate pit. Generally the size of optimal ultimate pit increases with increase in the block values and the pit get deeper if the slopes increases [22]. The solution to ultimate pit limit problem defines the amount of ore to be mined, waste to be removed, the metal content to be recovered and is usually used to justify the project economically. It is also used as a guide to locate mine site facilities. The UPL may still contain thousand to millions of blocks making the task of defining the optimum production schedule computationally expensive and challenging

problem. To make this problem computationally more tractable different techniques have been discussed in the literature, some of which will be briefly discussed in the following section.

2.1.1. *Improving the tractability of the problem*

Ramazan and Dimitrakopoulos [23] discussed different strategies for formulating mixed integer programming model of open pit mine production scheduling problem to expedite the optimization process. To reduce the number of required binary decision variables it was proposed to define the decision variables representing the ore blocks as binary, leaving the decision variables for waste blocks as linear if the production capacity constraints are tight. To avoid or minimize partial block mining it was suggested that the values of the same value blocks in the objective function should be changed by a small amount.

Guapp [17] proposed different strategies to speed up the solution times of the production scheduling problem by reducing the number of required decision variables, adding cuts to strengthen the formulation and using Lagrangian relaxation techniques. Though these techniques can improve the efficiency of optimization process but may not be sufficient for solving the production scheduling problem of very large open pit mines.

Ramazan [24] proposed *Fundamental Tree Algorithm* for block clustering to reduce the number of required decision variables. The fundamental tree is defined to be any combination of blocks within ultimate pit limits or a pushback that can be mined while obeying the following conditions:

i. Do not violate the slope constraints
ii. Has positive combined economic value
iii. No subset of the combined blocks can be found without violating the first two conditions

An LP formulation was proposed for the clustering purpose. This algorithm is able to reduce the number of required decision variable, eliminate the gap problem but its implementation may be quit complex and in the case of large deposit may result into large number of fundamental trees

which will still need large number of binary variable in the IP model that can be extremely difficult to solve [15].

Tabesh and Askari-Nasab [25] proposed a two stage clustering algorithm for block aggregation/clustering to reduce the size of the open pit mine scheduling problem. This approach uses a two stage procedure. In the first stage the block on a given single bench are clustered based on the so called similarity index using an agglomerative hierarchical algorithm. In the second stage a Tabu search procedure is used to improve the solution by reducing the number of precedence arcs between the generated clusters on the current bench and clusters on the bench below resulting in the reduction of the required binary decision variables in the scheduling problem formulation.

Goodwin et al. [26] proposed an alternative formulation for the open pit mine scheduling problem. In this formulation the precedence / slope constraints were defined by using so called state constraints. They proposed the receding horizon strategy to solve the multi period production scheduling problem. The proposed formulation and the solution procedure performed better in terms of the computational times and the final solution produced when applied to a real life case study.

Cacceta and Hill [18] presented a new formulation by defining variables representing whether a block is mined *by* period t. The formulation maximize the net present value of the operation while considering different physical and operational constraints such as block precedence, blending, Mining, Milling, stockpiles, minimum bottom width and maximum vertical depth. To efficiently solve this problem the authors proposed a branch and cut strategy though they did not provide full information about it due to the commercialization of their software. This approach is able to produce good solution for medium size production planning problem but finding optimal solution for large problem might be difficult. Secondly it cannot optimize the cutoff grade policy during the optimization process [15]. To reduce the computational times required to solve the problem formulation proposed by Cacceta and Hill [18], Bley et al. [27] suggested decision variable reduction techniques and cuts by exploiting the special structure of the precedence constraint knapsack problem formed by combining the precedence and production capacity constraints.

2.2. Ultimate Pit Limits (UPL) based approaches

This approach represents the traditional way of dealing with the mine sequencing problem by dividing it into a series much simpler sub problems [28]. In this approach the solution process starts with the determination of optimal ultimate pit limits (UPL) followed by the determination of optimal phase design. To achieve this goal a series of nested pits are generated inside the ultimate pit limits by gradually increasing the commodity price from a small value and solving the resulting sequence of ultimate pit limit problems. This process is usually termed as parameterization. The block values during this process may be modified using the following equation [29]:

$$v_i(\lambda) = \alpha\,\lambda - \beta \qquad 2.13$$

Where v_i represent modified block value, α represents the revenue parameter such as commodity price, β represents different cost parameters such as mining or processing costs, λ represents revenue factor. A particular value of λ may increase or decrease commodity price and modify the block values likewise. The nested pits are then grouped together into required number of pushbacks by considering production rates and different operating constraints followed by the determination optimum cutoff grade policy and sequence of extraction of the blocks in each push back separately [28]. Though this approach makes the block sequencing problem computationally more tractable but making such inter dependent decisions separately may compromise the optimality of the final solution.

2.2.1. *Lagrangian relaxation Approach*

In order to solve the multi period mine sequencing problem Dagdelen and Johnson [30] proposed the Lagrangian Parameterization technique by relaxing the complicated production capacity constraint into the objective function and by decomposing the resulting multi period problem into much simpler single period problem. A formulation similar to the one defined by Dagdelen and Johnson in [30] can be represented as:

$$\text{Maximize } Z(\lambda) = \sum_{i=1}^{N}\left(v_i - \sum_{m \in M} \lambda_m \, a_{mi}\right) x_i \; + \sum_{m \in M} \lambda_m \, b_m \qquad 2.14$$

$$x_i \leq x_j \quad i = 1, 2, \ldots\ldots\ldots\ldots\ldots N; \quad j \in P_i \qquad 2.15$$

$$x_i \in [0,1]$$

Where

M = The total number of production capacity constraints

m = Production capacity constraint index

a_{mi} = Coefficient of block $\boldsymbol{i}$ in the production capacity constraint index at $\boldsymbol{m}$

λ_m = Lagrangian multiplier of the production capacity index at $\boldsymbol{m}$

v_i represents the economic value of block i, N represents the total number of blocks in the block model, x_i represents a binary variable corresponding to each block i which take the value 1 if the block i is inside the UPL and 0 otherwise and P_i represents the predecessor group of block i to ensure stable pit slopes. The resulting problem has structure similar to the ultimate pit limit (UPL) problem which can be efficiently solved using different UPL algorithms such as Max flow algorithms [21] etc. To define the optimum production schedule a series of ultimate pit problems are solved. To adjust the values of the Lagrange multipliers for defining optimum production sequence the authors proposed to use subgradient optimization technique. The approach is not able to consider the dynamic cutoff grade in the optimization process and may not converge in certain situations where appropriate Lagrange multiplier values cannot be found out [15]. Later on Akaika and Dagdelen extended this work and proposed a 4D-Network relaxation method with the ability of considering dynamic cutoff grade and stock pile option during the scheduling process [31]. The method may not always find the optimum solution due to the possibility of gap

problem occurring. The gap problem refers to large differences in quantity and quality of material to be mined in the successive phases [29].

2.3. Dynamic Programming

This approach attempts to solve complex optimization problems by dividing them into simpler sub problems. The optimal solutions for these sub problems can be found comparatively easily which can be combined together to find the optimal solution for the main problem. The first attempt to apply dynamic programming to simultaneously solve the pit optimization and block sequencing problem was made by Roman [32]. The main advantage of this algorithm was its ability to consider the time value of money and block sequencing in the determination of the final pit limits but also had the following disadvantages [15]:

- Due to the complexity of the proposed algorithm could not be applied to large deposits
- No guarantee of meeting mining and milling constraints in each period

Later on Dowd and Onur proposed a dynamic programming formulation of the long term production planning problem [33, 34]. The proposed approach was able to consider all kinds of necessary constraints and removal of unattractive sequences whenever detected but needed long computational times to produce reliable solutions and was not able to integrate the dynamic cutoff grade concept [15] in the optimization process. Tolwinski and Underwood [35] proposed a sequential optimization model to solve the long term production scheduling problem. The model maximize the net present value of the operation and is able to consider different constraints such as maximum allowable slope angles, minimum working space and the constraints needed to ensure the consistent flow of ore to the processing plant. The authors combined dynamic programming, stochastic optimization, artificial intelligence and heuristic rules to deal with the proposed model. The main draw backs of the proposed procedure are its inability to provide a guaranteed optimal solution in the mathematical sense secondly for large deposits there is no guaranty that a feasible solution can be found [15]. Later on Tolwinski [36] and Tolwinski Golosinski [37] proposed a procedure based on depth first search technique of the dynamic

programming to solve long term production scheduling problem. This approach is able to consider all types of necessary constraints and could be applied to large open pit mines but getting an optimum result in terms of the maximum possible NPV is not guaranteed. Erarslan and Celebi [38] proposed a simulative optimization model that can simultaneously solve ultimate pit limit problem and production planning problem. They used dynamic programming techniques to solve this model. The model is able to incorporate all type of operating constraints such as grade blending, stockpiling, plant facilities but like many other dynamic programming based approaches the proposed technique cannot handle medium or large deposits and the optimality of the final solution is not guaranteed when applied to small deposits.

2.4. Metaheuristic techniques

The main limitations of the techniques discussed in the previous sections are either their high computational cost when applied to real sized open pit mine scheduling problem or they can only solve a simplified version of the original problem. In the recent years a new class of computationally less expensive algorithms i.e. metaheuristic techniques such as Genetic Algorithms, Simulated Annealing, Ant Colony optimization etc. have attracted the attention of several researchers to solve the mine design and production scheduling problem. Though these techniques do not guarantee the optimality of the final solution that they produce but they can generate sufficiently good solutions with relatively less computational cost and can handle nonlinear constraints and objective function which makes them attractive alternative choice over the computationally expensive exact optimization techniques. In the following sections some of the metaheuristic techniques that have been applied so far to the production scheduling problem of the open pit mines will be briefly discussed.

2.4.1. *Genetic Algorithm (GA)*

Genetic algorithms are population based stochastic optimization techniques originally proposed by Holland [39, 40] in the late 1960s. Genetic algorithms are inspired by the principles of natural selection and evolution to produce better quality solution for a problem at hand. The search process in the case of genetic algorithms starts with a population of solutions randomly generated

in the search space that iteratively evolve over a series of generations by applying genetic operators and probability techniques such as reproduction, crossover and mutation to each member of the population. The process continues until a sufficiently good solution has been found or the maximum number of generations has passed depending on the stopping criteria that have been defined.

The first attempt to apply genetic algorithms to the open pit mine design and production scheduling problem was made by Danby and Schofield [41]. The main advantage of their approach was its ability to simultaneously solve the ultimate pit limit and production scheduling problems. The main steps of this approach can be summarized as:

1: Generate a population of random feasible solutions of size ***n***

2: Evaluate the fitness values of each member of the population using a fitness function

3: Reproduction of pit population using probabilistic techniques, during the reproduction step depending on the fitness of the individual solution they either survive to the next generation or die out.

4: Crossover of pits /solutions to produce a new population

5: Mutation of pits/solutions to maintain diversity and prevent the premature convergence of the algorithm

6: Application of pit normalization procedure to ensure that extraction constraints are satisfied

7: Local optimization to enhance the fitness of individual solutions

8: ***If*** stopping criteria is met then ***exit, else*** go to step 2

The main disadvantage of this procedure was its high computational cost as the size of the problem was increased. The approach was initially implemented for 2D and was later on extended to 3D [42].

2.4.2. *Ant Colony optimization (ACO)*

Ant Colony optimization (ACO) algorithms are nature inspired stochastic population based metaheuristic techniques initially proposed by Dorigo, Maniezzo and Colorni in 1991 [11, 43]. ACO algorithms are based on the foraging behavior of the real ants [43]. During the search for food the ants start exploring the surrounding area by wandering randomly unless they find the food source. While carrying the food back to the nest the ants deposits a chemical pheromone trail on the ground, the quantity of the deposited pheromone depends on the quality and quantity of the discovered food. This pheromone trail acts as a guide for the other ants to reach the food source instead of wandering randomly. With the passage of time the pheromone trail starts to evaporate making the longer paths unattractive than the shorter paths due to the high evaporation rate and low deposition rate of the pheromone on the longer path. These characteristic of the real ant colonies are used to develop the Ant Colony Optimization algorithms [44].

Sattarvand and Niemann-Delius [11, 12] proposed a metaheuristic technique based on Ant Colony Optimization for long term open pit mine planning. The main steps of their proposed procedure can be summarized as [11]:

i. Define the economic and technical parameters and the block model

ii. Calculate the ultimate pit limits and the initial push back design i.e. generate an initial sub optimal solution

iii. Initialize the pheromone values for the blocks based on the sub optimal solution defined in the previous step, these values represents the desirability of a block to be the deepest point of the mine being mined in a certain period / pushback

iv. Generate random solutions using the pheromone values

v. Update the pheromone values by using evaporation and deposition operators

vi. If the stopping criterion is met exit and export the optimized UPL and production schedule else continue the iterative process.

This approach was initially implemented for 2D case but later on extended to more general three 3D case by Soleymani and Sattarvand [45]. The proposed algorithm is quit flexible and can handle simple as well as complex objective functions.

2.4.3. *Simulated Annealing (SA)*

Simulated Annealing is a stochastic optimization technique proposed independently by Kirkpatrick, Gelatt and Vecchi in 1983 [46] and Cerny in 1985 [47] for finding the optimum or near to optimal solution of multi model complex functions. Simulated annealing is inspired by the annealing process in metallurgy. In the annealing process a solid is heated until the thermal stresses are released followed by a slow cooling process to get the optimum configuration of crystals. In simulated annealing algorithm the search process starts by defining an initial random solution followed by the perturbation of that solution to create new solution. The new solution is accepted if its fitness value is better than the previous one otherwise the decision about accepting it is dependent on the current temperature of the system which enables the algorithm to escape the local optimal positions. The temperature of the system is gradually decreased during the iterative process until the process ends. The initial temperature and cooling rate has profound effect on the overall performance of the simulated annealing algorithm. A very low initial temperature may result in premature convergence on the other hand a very high initial temperature cause the algorithm spending more time on poor initial solution. Similarly a high cooling rate may cause the algorithm to get stuck into local optimum and a slow rate may increase the computation time [11, 48].

A simulated annealing based solution approach for mine production scheduling problem was first proposed by Kumral and Dowd [10]. The main steps of this approach can be defined as:

i. Define the pit blend limits to identify the set of blocks that satisfy the content requirements of the final blend using a linear programming approach

ii. Determine the ultimate pit limits by using the Lerch Grossmann algorithm and the blocks identified in the previous step

iii. Generate an initial suboptimal solution using Lagrangian Parameterization

iv. Improve this suboptimal solution produced in the previous step using multi objective simulated annealing based approach

The main advantage of this approach was its ability to consider multiple objectives during the optimization process. The independent determination of the ultimate pit limits and production schedule may be counted as the disadvantage of the this approach[11].

2.5. Uncertainties in the input parameters

Traditional approaches of open pit mine planning and production scheduling (as discussed before in the previous sections) are based on the assumption that all the input parameters are known with certainty and represent the reality while on the contrary a certain degree of uncertainty is almost always associated with these parameters, ignoring them may result into unrealistic and false planning and scheduling decisions. There may be several different sources involved causing these uncertainties in the input parameters, the most important one can be broadly classified as [49]:

- Technical (geological and mining)
- Financial
- Environmental

The geological uncertainty is considered to be one of the most important sources of uncertainty that is needed to be managed in the planning process to meet the project expectations. Geological uncertainty is caused by the fact that different attributes of the individual blocks are estimated using very sparse drillhole data and usually a significant and variable level of uncertainty is almost always associated with each on one of these estimated values. Due the presence of this uncertainty it may happen that a block that has been identified as an ore block during the planning stage may turn out to be a waste block once mined causing failure in meeting the production targets. Geostatistical conditional simulation techniques provides a framework to

quantify this geological uncertainty by generating multiple equiprobable simulated realizations of the ore body having the same statistical characteristics as the original data [2, 3]. Conditional simulation is class of Mote Carlo simulation technique. It was introduced 1970s and since then several different simulations methods have been proposed in the literature such as Sequential Gaussian Simulations (SGS), LU simulations, direct block simulations etc. [3]. In the sequential methods the conditioning data is extended to include all the original drill hole sample data and the previously simulated values. These sequential simulation methods are based on the application of Bayes Theorem:

$$P(A_1, A_2 \dots \dots \dots A_n) = P(A_n | A_1, A_2 \dots A_{n-1}) \cdot P(A_{n-1} | A_1, A_2 \dots A_{n-2}) \dots P(A_2 | A_1) \cdot P(A_1) \quad 2.16$$

Given the joint simulation of z-values at k locations conditioned to n surrounding data the realizations can be generated by drawing from a condition distribution function:

$$F(u_1, u_2 \dots u_k ; z_1, z_2 \dots z_k | n) = F(u_k; z_k | n + 1 - k) \dots F(u_2; z_2 | n + 1) \cdot F(u_1; z_1 | n) \quad 2.17$$

Where u represents the location and z represents the attribute of interest. The sequential simulation techniques can be used to generate conditional realizations of Multi-Gaussian or non-Gaussian random functions as long as the conditional distribution can be defined. In this study sequential Gaussian simulation technique has been used to generate multiple equiprobable simulated realizations of an ore deposit. A detailed description of these methods can be found in [2, 3]. To help the final decision making process for producing more reliable results different techniques for the use of these simulations in the planning and scheduling process of the open pit mines have been proposed in the literature, some of these will be briefly discussed here.

Ravenscroft [50] discussed the limitation of the conventional methods of sensitivity analysis in reserve estimation and mine planning and outlined a procedure for risk analysis in mine scheduling using equiprobable conditionally simulated ore body models. The author concluded that the traditional approaches are not able to accommodate the quantified risk and recommended the development of new approaches with the ability of directly incorporating the grade uncertainty into the production planning process. Dowd [51] proposed a risk assessment procedure for open pit mine planning. The approach combines multiple simulated realizations of the ore body and different financial variables drawn at random from their predefined distributions to define the economic block model, ultimate pit limits and production schedule. The distributions of the resulting output parameters e.g. Net Present value (NPV), Internal Rate of return (IRR) etc. is then used to assess their associated risk. The main advantage of this approach is its ability in combining several different financial e.g. commodity price, mining costs, processing costs and grade uncertainty in defining the risk profile of a mining project to aid the decision making process. Dimitrakopoulos et al. [52] showed the effects of geological uncertainty on the key performance indicator of a project (Net Present Value (NPV), mill feed grade, ore tonnage, production costs) by comparing the optimization results obtained using a single estimated orebody model to the one obtained using a set of conditionally simulated realizations of the orebody. The authors stressed on the need of more direct integration of the uncertainty in the optimization process to produce more reliable results. Dimitrakopoulos et al. [53] proposed a maximum upside / minimum downside risk approach for finding "*best*" schedule under the condition of grade uncertainty. The processes for finding the best schedule starts with optimizing schedule for each simulated ore body independently followed by comparing these schedules to see how they perform when applied to other simulations in terms of key performance indicators such as maximum annual return, minimum ore tonnage, and minimum metal production. The main goal this approach was to select a schedule among the set of schedules that perform better than expected (upside potential) while minimizing the risk of performing below expectation (down side risk). Though this approach gives some insight in the decision making process but is unable to directly and explicitly integrate the geological uncertainty in the optimization process to generate a single optimal schedule.

2.5.1. *Mathematical Programming Approaches for stochastic production scheduling*

Different mathematical programming models for incorporating grade uncertainty into the optimization process have been proposed in the literature for determining the optimum long term production schedule of the open pit mines in the recent years.

Dimitrakopoulos and Ramazan [54] presented a probabilistic formulation for production scheduling of multi element deposits under the condition of geologic uncertainty. In this approach probabilities are assigned to individual blocks that represent the desirability of the blocks being mined in a particular period. The proposed mathematical formulation try to generate a solution that simultaneously minimize the deviations from quality, equipment mobility and space requirements while respecting all the required operational constraints. This formulation does not directly attempt to maximize the net present value of the operation and does not make full use of joint local uncertainty in the simulations; these points can be counted as the limitations of this approach.

Ramazan and Dimitrakopoulos [6] proposed a two stage stochastic integer programming formulation with fixed recourse to address the production scheduling problem of the open pit mines under the condition of grade uncertainty. The general idea of a two stage stochastic model with fixed recourse as defined by Birge and Louveaux [55] can be stated as*:*

"In a situation where a set of decisions has to be made on some random events make the first stage decisions without full information about random events and then take the corrective actions (second stage decisions) to compensate for the negative effects of uncertainty when full information on the random events are revealed [55]*."*

The proposed formulation attempts to maximize the expected net present value of the production schedule (first stage decisions) while simultaneously minimizing the deviations from ore, grade and metal production targets (second stage decisions) under the condition of grade uncertainty quantified through multiple simulated realizations of the orebody.

- **Two-stage stochastic formulation of the open pit mine production scheduling problem under grade uncertainty**

The two-stage stochastic (mixed-integer) model with recourse formulation of the open pit mine scheduling problem similar to the one proposed in [6] can be represented as follows:

$$\text{Max} \sum_{t=1}^{T}\left[\sum_{i=1}^{N} E\{(NPV)_i^t\}\, x_i^t \; - \frac{1}{s}\sum_{s=1}^{S}(p_t^{o-}\, y_{ts}^{o-} \; + \; p_t^{o+}\, y_{ts}^{o+} + p_t^{m-}\, y_{ts}^{m-} + \; p_t^{m+}\, y_{ts}^{m+})\right] \quad \textbf{2.18}$$

The objective function is composed of two parts, the first part is for the maximization of the expected NPV of the production schedule and second part represents the minimization of the expected recourse costs whenever the stochastic constraint i.e. 2.19, 2.20, 2.21 and 2.22 are violated.

Stochastic constraints:

The upper and lower processing capacity constraints i.e. equation 2.19 and 2.20 and the upper and lower metal production constraints i.e. equation 2.21 and 2.22 represents the scenario dependent stochastic constraints. These constraints are modeled as soft constraints where the violation is penalized in the objective function.

$$\sum_{i=1}^{N} o_{is} * x_{it} + \; y_{ts}^{o-} \; \geq \; \underline{O_t} \qquad \text{t} = \; 1{,}2 \; \ldots . \text{T}\,, \qquad \text{s} = \; 1{,}2 \ldots \text{S} \quad \textbf{2.19}$$

$$\sum_{i=1}^{N} o_{is} * x_{it} - \; y_{ts}^{o+} \; \leq \; \overline{O}_t \qquad \text{t} = \; 1{,}2 \; \ldots . \text{T}\,, \qquad \text{s} = \; 1{,}2 \ldots \text{S} \quad \textbf{2.20}$$

$$\sum_{i=1}^{N} m_{is} * x_{it} + y_{ts}^{m-} \geq \underline{M_t} \qquad \text{t} = 1,2 \ldots . \text{T}, \qquad \text{s} = 1,2 \ldots \text{S} \qquad \textbf{2.21}$$

$$\sum_{i=1}^{N} m_{is} * x_{it} - y_{ts}^{m+} \leq \overline{M}_t \qquad \text{t} = 1,2 \ldots . \text{T}, \qquad \text{s} = 1,2 \ldots \text{S} \qquad \textbf{2.22}$$

$x_{it} \in (0\ ,1)$

$$y_{ts}^{o-}\ , y_{ts}^{o+}, y_{ts}^{m-}, y_{ts}^{m+} \geq 0$$

Where

T: Total Number of periods

t: Time period index, t = 1, 2 …. T

N: Total number of blocks

i: Block index, $i = 1, 2 \ldots N$

S: total number of simulations of the ore body

s: Simulation index s = 1, 2 … S

$E\{(NPV)_i^0\}$: is the undiscounted expected economic value of block i and is calculated as follows

$$E\{(NPV)_i^0\} = \sum_{s=1}^{S} BV_{is}^0 \Big/ S$$

Where BV_{is}^0 is the undiscounted economic value of block i according to simulation s.

$E\{(NPV)_i^t\}$: is the expected discounted economic value of block i if mined in period t and is calculated as follows:

$$E\{(NPV)_i^t\} = E\{(NPV)_i^0\}/(1+dis1)^t$$

$\boldsymbol{x_{it}}$: are the first stage scenario independent binary decision variables which take the value 1 if block i is mined in period t and 0 otherwise.

p_t^{o-}, p_t^{o+}: are the discounted unit costs for shortage or surplus ore produced in period t respectively and are calculated as:

$$p_t^{o-} = \frac{p_0^{o-}}{(1+dis2)^t}, \quad p_t^{o+} = \frac{p_0^{o+}}{(1+dis2)^t}$$

p_t^{m-}, p_t^{m+}: are the discounted unit costs for shortage or surplus metal produced in period t respectively and are calculated as:

$$p_t^{m-} = \frac{p_0^{m-}}{(1+dis2)^t} \quad , \quad p_t^{m+} = \frac{p_0^{m+}}{(1+dis2)^t}$$

y_{ts}^{o-} , y_{ts}^{o+}: are the second stage scenario dependent continuous variables represent the shortage or surplus amount of ore produced in period t if scenario s occurs respectively.

y_{ts}^{m-}, y_{ts}^{m+} : are the second stage scenario dependent continuous variable represents the shortage or surplus amount of metal produced in period t if scenario s occurs respectively. The second stage (recourse) decision variables are dependent on the outcome of the ore body realizations and of the first stage decision variables i.e. $\boldsymbol{x_{it}}$.

w_i : Tonnage of block i

o_{is} : ore content of block i according to simulation s

m_{is}: is the metal content of block i under scnario s.

$\underline{W_t}$ and $\overline{W_t}$ are the Lower and upper limits of the available mining capacity in period t respectively.

$\underline{O_t} and\ \overline{O_t}$ are the Lower and upper limits of the available processing capacity in period t respectively.

$\underline{M_t}$ and $\overline{M_t}$ are lower and upper limits of the required metal production in period t respectively.

The formulations of the scenario independent constraints i.e. reserve constraints, slope constraints, mining capacity constraints remain similar to the one described in the deterministic model of the problem in section 2.1. Though this formulation allows more direct integration of the grade uncertainty in the optimization process though the use of multiple equiprobable realizations of the orebody but solving this formulation for real sized ore deposit may be extremely difficult and computationally expensive.

Menabde et al.[4] proposed a mixed integer programming formulation to optimize production schedule and cutoff grade policy simultaneously under grade uncertainty. To find the optimum cutoff grade policy the possible cutoff grades have been divided into discrete sets in this approach. The authors used a block aggregation scheme to reduce the number of required decision variables. By comparing the results of three different tests it was shown that the proposed approach can substantially improve the NPV of the schedule under grade uncertainty. The results also highlighted the importance of the simultaneous optimization of production schedule and cutoff grade policy.

Golamnejad et al. [5] proposed a chance constrained based integer programming formulation of the open pit mine scheduling problem. In this method the grade distribution function of each block is used to account for grade uncertainty of that particular block independently from other blocks which is far from reality and could be regarded as an unrealistic assumption[56]. The objective function was defined to maximize the net present value of the operation while simultaneously minimize its standard deviation.

2.5.2. *Stochastic optimization of Ultimate Pit Limits and Pushback design*

Meagher et al. [57] proposed a parametric minimum cut algorithm in a stochastic framework to address the ultimate pit limit problem and pushback design under the condition of geologic and market uncertainties. The authors extended the conventional parametric maximum flow / minimum cut approach to account for geologic and market uncertainties by using multiple equiprobable realizations of the orebody and metal price simulations[29]. Asad and Dimitrakopoulos [29] extended the approach proposed by Meagher et al. [57] and introduced a concept that incorporates time dependent discounted block values for defining ultimate pit limits and push back design under market and grade uncertainties. To exploit the classical structure of maximum flow algorithm the production capacity constraints were put in the objective function using Lagrangian relaxation technique in this approach. The resulting optimization problem has totally unimodular structure that can easily be solved using any linear programming or minimum cut / Max flow algorithm. They proposed a modified subgradient method to select the values of the Lagrangian parameter for designing phases with uniform amount of materials to avoid the gap problem instead of choosing them through trial and error procedure.

2.5.3. *Stochastic Mine Production Scheduling with Metaheuristic techniques*

Considering the computationally intensive nature of the open pit mine production scheduling problem with uncertainty in the input parameters, the main focus of research in the recent years have turned towards finding ways to determine approximate solutions of this problem by using different heuristic and metaheuristic techniques which will also remain the main focus of this study.

Godoy and Dimitrakopoulos [58] proposed a simulated annealing based framework for integrating grade uncertainty in to long term production scheduling of open pit mines. The main aim of this framework was to generate a single schedule that reduces deviations from ore and waste production targets across all scenarios. The process starts with generating optimal schedules for each scenario independently from each other followed by the application of the simulated annealing technique to produce a single solution that minimizes the deviations from

ore and waste production targets. This framework integrates the joint local uncertainty into the optimization process by minimizing deviations from production targets for each realization of the orebody. Due to the multistage nature of this approach its implementation may be quit complicated and cumbersome, secondly the optimality of the final solution is not guaranteed these could be counted as the disadvantages of this framework[15].

Lamghari and Dimitrakopoulos[56] presented a Tabu search based approach for long term production scheduling of open pit mines with grade uncertainty. Tabu search is a metaheuristic optimization technique presented independently by Glover and Hansen[56]. The proposed approach starts with generating an initial feasible solution of the two stage stochastic optimization model of the open pit mine scheduling problem with fixed recourse using a heuristic procedure. Tabu search procedure is then applied to improve the quality of this initial solution by maximizing its expected NPV while simultaneously minimizing the production capacity constraints violations. To search the feasible solution space more extensively the authors proposed two different diversification strategies based on long term memory of the search and variable neighborhood search method for reinitializing the initial solution whenever the procedure terminates. The numerical results indicated that the proposed approach is able to generate sufficiently good solutions for small to medium sized problems in reasonably good computational times.

Chapter 3. Metaheuristic Techniques

Optimization plays an important role in different fields of science, engineering, economics, computer science etc. to facilitate the decision making process. Optimization process consists of modeling a problem in terms of some evaluation function and related constraints (if any) and then employing an optimization algorithm to find the best solution among the feasible candidate solutions that optimize the objective function. Depending on the problem at hand the objective of the optimization can be anything from maximizing the profit, efficiency and output to minimizing the costs, energy consumption etc. of a process[59]. Over the years numerous optimization techniques have been proposed in the technical literature. Different approaches are commonly used to classify these techniques. These classification approaches are commonly based on the algorithmic structure, types of variables, smoothness of the objective function etc. One possible classification of the currently available optimization techniques is given in Figure 3.1.

For most of the optimization problems the number of candidate solutions is usually very large, this makes the quest of finding the best solution among them very hard and computationally expensive. In such cases heuristic and metaheuristic techniques are commonly used to find sufficiently good solutions of these problems in reasonable amount of time. The term heuristic refers to experience-based techniques for finding optimal or near to optimal solution of a problem and the term “meta” means high level, so metaheuristic algorithms solves optimization problems using high level techniques / heuristics [60]. Though these techniques do not guaranty the optimality of the solution that produce but they can generate good solutions in reasonable amount of time. In the following sections the following three different metaheuristic techniques will be briefly discussed:

- Particle Swarm Optimization
- Bat Algorithm
- Differential Evolution

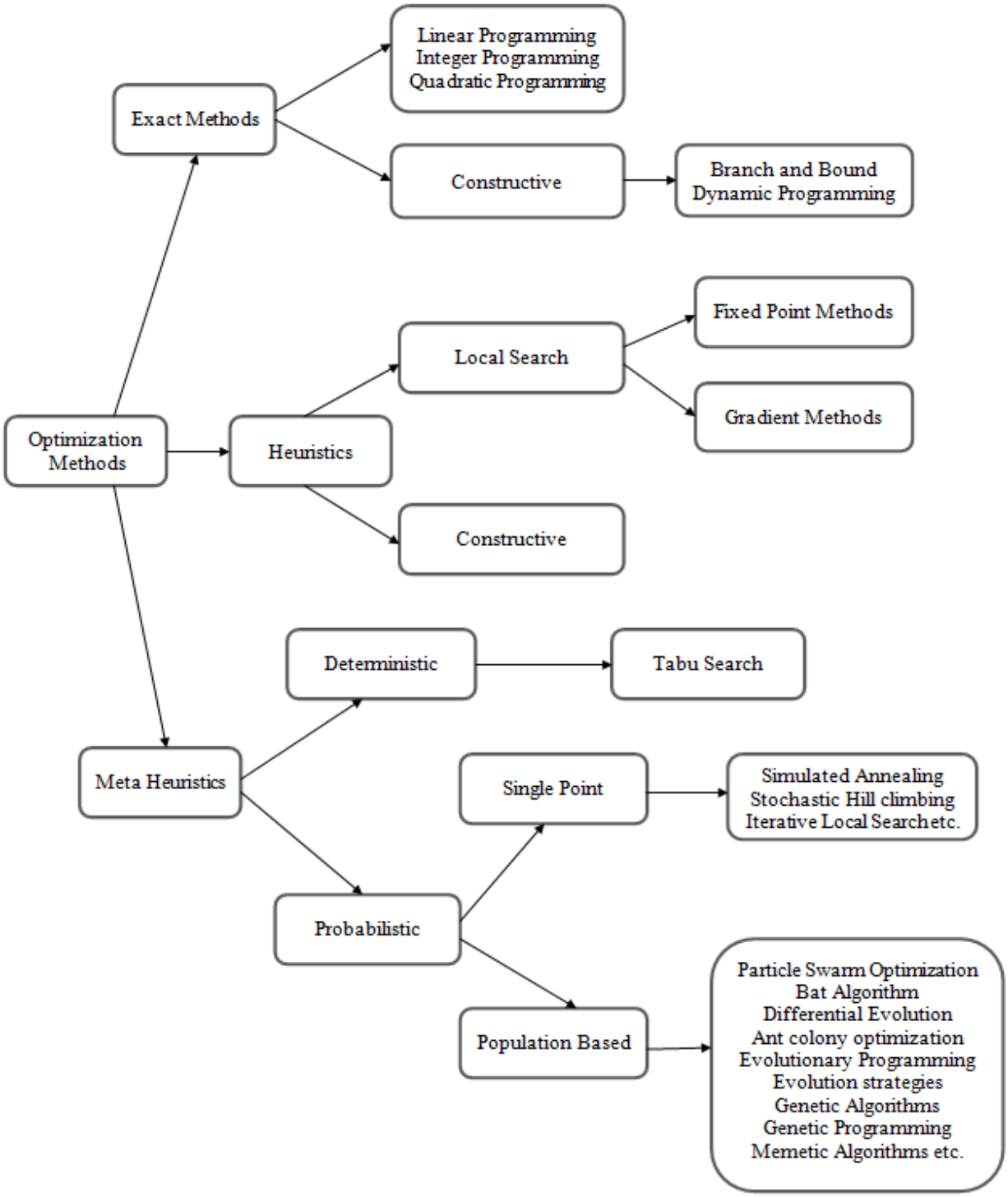

Figure 3.1: Classification of optimization Methods [61]

3.1. Particle Swarm Algorithm

Particle swarm optimization (PSO) is a population based stochastic optimization technique first presented by Kennedy and Eberhart in 1995 [62, 63]**.** PSO is a nature inspired algorithm based on the social interaction of individuals living together in groups e.g. bird flock, fish schools, animal herds etc. Unlike deterministic gradient based algorithms, PSO does not need the gradient information of the objective or fitness function which makes it a better choice in the situations where getting gradient information is either difficult or computationally expensive. The underlying concept of the PSO algorithm is very simple and it need less number of parameters to be adjusted by the user which makes it easy to implement in comparison to other heuristic / Meta-heuristic techniques [64].

3.1.1. *Origin and Background*

In the 1980s both computer scientist and zoologist were interested in understanding the dynamics behind the synchronous movement of organism living together in groups [65, 66]**.** Computer scientist were mainly interested in creating realistic visualization of bird flocks, fish schools etc., zoologist on the other hand were interested in discovering the underlying rules which enabled theses large number of individuals to move synchronously. In 1983, William T. Reeves a well-known expert in computer graphics presented a method for modeling fuzzy objects such as fire, clouds, and water and named this method as "*Particle System*"[67]. Reeves claimed that the motion of the fuzzy objects like fire, clouds, and water cannot be modeled by using simple affine transformation as these objects do not possess well-defined, smooth and shiny surfaces. Instead of using surface elements such as polygon etc. to represent these fuzzy objects, a cloud of particles was suggested to be used to define their volume. This proved to be much promising alternative for modeling fuzzy objects having irregular, complex and ill- defined shapes. In his approach a particle was defined to be a point in three dimensional space, with a set of attributes attached to it such as: initial position, initial velocity, initial color, Initial transparency, shape and lifetime. With the passage of time new particles are generated into the system, moved to new positions by updating their positions and attached attributes, and the particles die from the

system as their predefined life time reaches to zero[67]. In 1987, Reynolds [66] presented an elaboration of particle system approach for simulating flocking behavior of birds. In his approach, particles were replaced by birds, possessing a full local coordinate system, geometrical shape and the ability to interact with its neighbors in the flock. Instead of using a strong centralized control, each individual's perception of the dynamic environment is restricted to its local neighborhood. In order to simulate the flocking behavior, following rules with decreasing order of importance were assigned to each individual:

- Collision avoidance: Avoid collision with neighboring flock members; it serves to maintain a minimum required separation distance among the individuals of the flock.
- Velocity Matching: Try to match the velocity with other flock members.
- Flock centering: This helps to keep the simulated bird stick together.

In 1990, Heppner and Grenander [65] proposed a Stochastic Nonlinear Model to simulate flocking behavior of birds. Similar to Reynolds's approach they assigned a set of rule to each bird, to determine the movement of individual birds in the flock. But their model differs from the Reynolds's method in a couple of ways. Unlike Reynolds's deterministic approach their model has a stochastic part and their birds were additionally attracted to a central roost. The basic principle behind the synchrony of the flocking behavior in both the models presented by Reynolds [66] and Heppner and Grenander [65] was thought to be the result of the birds' effort to maintain an optimum distance with its neighbors [63]. Kennedy and Eberhart [63] extended these models which finally lead to the development particle swarm optimizer. They created an initial simulation with two basic components i.e. nearest-neighbor velocity matching to create synchrony of the movement and craziness to give the simulation a life like appearance. In the second version of the simulation the roost previously used by Heppner and Grenander [65] was replaced by a so called "cornfield vector", so now the bird instead of knowing the exact position of the roost are able to evaluate distance to it. In each iteration each agent now adjusts its trajectory to minimize its cornfield vector using two kinds of information:

- Using its own memory about the best value and the corresponding best position it has experienced so far.

- Using the information about the global best position discovered by any of its flock mates.

After excluding the unnecessary parameters e.g. craziness, nearest neighbor velocity matching Kennedy and Eberhart came up with a strong optimization tool known as Particle Swarm optimization that will be discussed in detail in the following sections.

3.1.2. *The Standard Particle Swarm Algorithm*

PSO algorithm mimics the swarming behavior of individuals living together in groups e.g. fish schools, bird flock, animal herds etc. Particle swarm optimization performs the search process by using a population (Swarm) of individuals (Particles) [62-64]. Each individual (Particle) is a potential solution to the optimization problem. At first a random starting position and random velocity is assigned to each particle of the swarm. The velocity represents the speed of "*flying*" of the particles through the search space [64]. To explore the search space for finding better solutions the velocity and position of each particle is then updated using equation 3.1 and 3.2 respectively during the iterative process.

$$\vec{v}_{i,t} = \vec{v}_{i,t-1} + \underbrace{c_1 \vec{r}_1 . (\vec{p}_{i,t-1} - \vec{x}_{i,t-1})}_{\text{Cognitive component}} + \underbrace{c_2 \vec{r}_2 . (\vec{g}_{t-1} - \vec{x}_{i,t-1})}_{\text{Social component}} \quad 3.1$$

$$\vec{x}_{i,t} = \vec{x}_{i,t-1} + \vec{v}_{i,t} \quad 3.2$$

Where

$\vec{x}_{i,t}, \vec{x}_{i,t-1}$: represents the current and previous position of the particle i

$\vec{v}_{i,t}, \vec{v}_{i,t-1}$: represent the current and previous velocity of the particle i

$\vec{p}_{i,t}$: is the personal best position experienced by particle i until iteration t

$\vec{g}_{i,t}$: is the global best position experienced by the population of particles until iteration t

$\vec{r}_1, \vec{r}_1$: represents uniform random real numbers in the range of (0, 1)

c_1, c_2 : are called the acceleration coefficients which are used to control the influence of cognitive and social term on the particle's velocity

The particles update their velocity by using three main components. They take advantage of their own previous successful experience (cognitive component), they imitate better swarm members (social component) and they use their velocity in the previous iteration (velocity memory), without the last two components the particles will keep travelling at the current speed in the same direction until they hit the boundary of the solution space. The algorithm may find an optimal solution if it is lying on its flying trajectory which is very rare to happen [68]. The objective of this iterative process is to stochastically traverse the solutions space and keep on finding better and better position till the termination of the iterative process. The Pseudo code of the PSO algorithm is given in Algorithm 1.

3.1.3. *Parameter Settings*

The basic PSO as described in the previous section needs only a small number of parameters to be adjusted by the user [69]. These parameters comprise the population size and the acceleration coefficient. In order to make the algorithm more stable and to improve its convergence properties other parameters like Maximum velocity, inertia weight and constriction coefficients were also introduced later on, which will be discussed in detail in the coming sections. These parameters have significant effect on the overall performance of the algorithm [70].

3.1.4. *Population Size*

In comparison to other population based algorithms e.g. Genetic Algorithms, PSO needs smaller population [64]. Population size is usually empirically selected by taking into account the

dimensionality and perceived difficulty of the problem. Usually a larger population is needed for multimodel problems; smaller population in these cases will end up in a local optimum position. On the hand a larger population for simple problems will result in an increased computational cost. Population sizes in the range of 20-50 are usually used [69]. Consideration the dimensionality (D) of the search space Clerc suggested the following equation to determine the swarm size (S) [71]

$$S = 10 + 2\sqrt{2D} \qquad 3.3$$

3.1.5. *Acceleration Coefficients*

The acceleration coefficients $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$ **,** determine the influence of random forces on the relative velocity of the particle towards its personal best $\vec{\boldsymbol{p}}_{\boldsymbol{i}}$ and neighbourhood best $\vec{\boldsymbol{g}}_{\boldsymbol{t}}$ [72]. Their values greatly influence the overall movement of the particles. A higher value of the acceleration coefficients will result in abrupt movement of the particles and they can easily get trapped into false optimum position, on other hand a low value will slow down the particles' movement and increase the computational cost [70]. In the original version of PSO Kennedy and Eberhart recommended the value for $\boldsymbol{c_1}$ *and* $\boldsymbol{c_2}$ to be equal to 2 which on average makes the weight for social and cognition part to be 1 [62].

Algorithm 1 : Pseudo code of the PSO Algorithm

1. Initialize a population (swarm) of particles with random initial positions $\vec{x}_{i,0}$ and random velocities $\vec{v}_{i,0}$ in the search space.
2. Initialize each particle's personal best position $\vec{p}_{i,0}$ to its initial position $\vec{x}_{i,0}$
3. Calculate the fitness value of each particle at its initial position $\vec{x}_{i,0}$ and determine the initial global best

   Position $\vec{g}_0$
4. **While (** t < maximum number of generations)
5. $\boldsymbol{for\ all}$ particles $\boldsymbol{do}$
6. Update the particle's velocity and position using equation **3.1** and **3.2** respectively.
7. Calculate the fitness value of each particle at its current position $\vec{x}_{i,t}$
8. $\boldsymbol{If}\ fitness(\vec{x}_{i,t})$ is better than the $fitness(\vec{p}_{i,t-1})\boldsymbol{then}$
9. $\vec{p}_{i,t} = \vec{x}_{i,t}$
10. $\boldsymbol{end\ if}$
11. $\boldsymbol{If}\ fitness(\vec{p}_{i,t})$ is better than the $fitness(\vec{g}_{t-1})\boldsymbol{then}$
12. $\vec{g}_t = \vec{p}_{i,t}$
13. $\boldsymbol{end\ if}$
14. $\boldsymbol{end\ for}$
15. If the stopping criterion is met: $\boldsymbol{end\ while}$

3.1.6. *Maximum velocity*

In the early implementation of PSO it was observed that the system may become unstable with particles' speed increasing without control [70, 72]. To prevent this explosion velocity

confinement mechanism was thought to be necessary. One such mechanism can be defined as [63]

$$\boldsymbol{if}\ \ \vec{v}_{i,d} > \vec{v}_{i,d,max}\boldsymbol{then}$$

$$\vec{v}_{i,d} = \vec{v}_{i,d,max}$$

$$\boldsymbol{else\ if}\ \ \vec{v}_{i,d} < -\ \vec{v}_{i,d,max}\ \boldsymbol{then}$$

$$\vec{v}_{i,d} = -\ \vec{v}_{i,d,max}$$

$$\boldsymbol{end\ if}$$

The velocity limits along each coordinate of the solution space is usually different from the other coordinate of the search space as each coordinate has its own dynamic range which can be different from other coordinates' dynamic ranges. This parameter is problem dependent and is usually chosen empirically. If the search space is defined by the bounds [-$\vec{x}_{max}, \vec{x}_{max}$], the value of $\vec{v}_{max}$ is usually defined as:

$$\vec{v}_{max} = k * \vec{x}_{max}\ \ where\ 0.1 \leq k \leq 1.0 \qquad 3.4$$

This parameter does not restrict the values of $\vec{\boldsymbol{x}}_{\boldsymbol{i,t}}$ to the range $[-\vec{v}_{max}, \vec{v}_{max}]$ it merely limits the step size of the particle during each iteration. A large value of the parameter may result in increased exploration and the system may not find a feasible solution. On the other hand the particles can get trapped in a local optimum if a smaller value is used.

3.1.7. *Inertia Weight*

To reduce or eliminate the need of the problem dependent parameter V_{max} and to enhance the searching capabilities of the PSO algorithm Shi and Eberhart proposed the following modified velocity update equation:

$$\vec{v}_{i,t} = \omega * \vec{v}_{i,t-1} + c_1 \vec{r}_1 . (\vec{p}_{i,t-1} - \vec{x}_{i,t-1}) + c_2 \vec{r}_2 . (\vec{g}_{t-1} - \vec{x}_{i,t-1}) \quad 3.5$$

Where ω is termed as the inertia weight which is used to control the influence of the particle's previous velocity on its current velocity [68]. The parameter was introduced in an effort to balance global and local search ability of the PSO algorithm. In the early experiments by Shi and Eberhart it was learned that a higher value of the inertia weight (>1.2) enables the algorithm to perform extensive exploration while a smaller value (<0.8) makes the algorithm more exploitative enabling it to perform local search. Initially Shi and Eberhart [68] proposed that the inertia weight can be positive constant or linear or nonlinear decreasing function of time such as:

$$\omega = (\omega_{max} - \omega_{min}) \frac{(iter_{max} - iter)}{iter_{max}} + \omega_{min} \quad 3.6$$

Inertia weight decreasing with time was found to be better than fixed inertia weight, because larger inertia weight in the early iterations helps to find the promising region of the search space and the smaller inertia weight at the end facilitates the local search. It was also found out that while using inertia weight based approach the value of V_{max} can be simply fixed to the value of the dynamic range of each variable along each dimension. Eberhart and Shi also proposed a fuzzy system to adapt ω and reported significant improvements [73].

3.1.8. *Constriction Coefficients*

When the Particle swarm algorithm is run without any control mechanism for clamping the particles' velocities, the velocities can rapidly increase and system will explodes after a few iterations. Researchers have identified this phenomenon in the early experiments with the algorithm, they used V_{max} for damping the dynamics of the particles, but the reason behind this phenomenon was not fully understood [72]. Clerc and Kennedy proposed a constriction coefficients based PSO algorithm after analyzing the trajectory of a single particle using some simplifying assumptions e.g. single particle, absence of randomness and search stagnation i.e. no

better solution are found during later iterations [72, 74]. These coefficients were introduced to prevent velocity explosion, ensure convergence and eliminate the need to set problem based maximum velocity parameter. They noted that constriction coefficients can be implemented using different ways. The most simple and frequently used method to incorporate constriction coefficients into the velocity update equation also known as Constriction type1" can be defined as:

$$\vec{v}_{i,t} = \chi(\vec{v}_{i,t-1} + c_1 \vec{r}_1 . (\vec{p}_{i,t-1} - \vec{x}_{i,t-1}) + c_2 \vec{r}_2 . (\vec{g}_{t-1} - \vec{x}_{i,t-1})) \quad 3.7$$

Where $\varphi = c_1 + c_2 > 4 \; and$

$$\chi = \frac{2}{\varphi - 2 + \sqrt{\varphi^2 - 4\varphi}} \quad 3.8$$

φ is commonly fixed to 4.1, $c_1 = c_2 = 2.05$, which results into constant multiplier $\boldsymbol{\chi}'$ s value to be 0.7298 and each of the two terms i.e. $(\vec{p}_{i,t-1} - \vec{x}_{i,t-1})$ and $(\vec{g}_{t-1} - \vec{x}_{i,t-1})$ multiplied by 1.49445 times uniform random number between 0 and 1. Eberhart and Kennedy [73] compared the performance of PSO using inertia weight based approach versus using constriction coefficients based approach. To get better performance they suggested to use constriction coefficient based approach while limiting the maximum velocity to the dynamic rang of the variable along each dimension.

3.1.9. *Population Topology*

Particle swarm optimization algorithm is based on the social learning process of individuals living together in groups. During the search process the individual particles not only take advantage of their own past experience but also use those of the neighboring particles. This relationship among the particles is commonly defined by population topology. The initial particle swarm optimization algorithm was inspired and based on the bird flocking simulations[65, 66]

where the positions of the birds were iteratively modified using several rules including some that take into account the physical position of the neighboring birds in the search space. So the initial population topologies were based on the physical proximity in the search space. These kinds of topologies were soon abandoned as they were computationally expensive and were showing undesirable convergence properties [72]. The focus of research soon shifted towards index based population topologies that will be discussed briefly in the following sections.

3.1.9.1. *Static Topologies*

In this kind of topologies the neighbors and the neighborhood do not change during the run time [72]. Examples of the static topologies include the following:

- ***gbest topology:*** is the one where each particle is influenced by the best particle of the entire population[62]. The flow of information in this kind of topology is usually very fast that may lead to premature convergence.
- ***lbest topology:*** is the one where each particle is connected to its K immediate neighbours, where K represents the indexed based neighbours of each particle of the population. This topology allows more through search of the search space as subpopulations may converge in different regions of the search apace. This topology is less susceptible to get stuck in a local optimum position but usually show much slower rate of convergence in comparison to gbest topology [62].

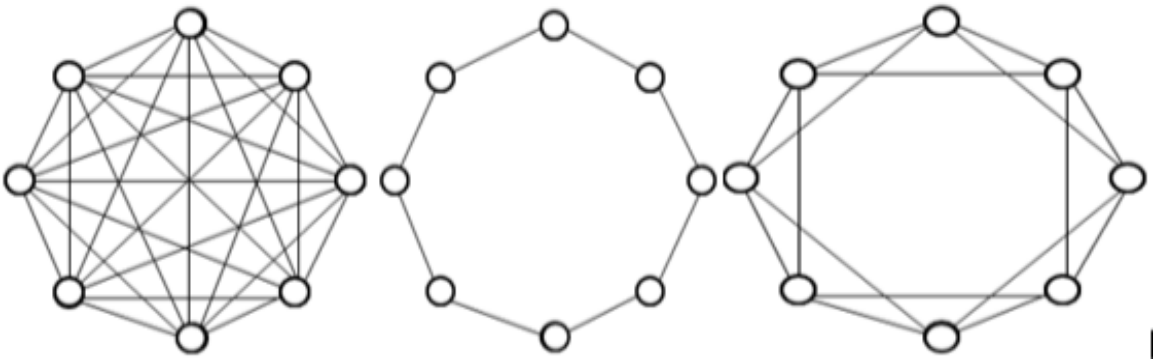

Figure 3.2: Population Topologies gbest (left), lbest with k = 2 (middle) and k=4 (right)

- *von Neumann topology:* In this kind of topology the particles are arranged like nodes of a grid, where each particle is connected to four other particles i.e. one above, one below, one on the right and on the left. Kennedy and Mendes[75] compared the performance of different population topologies on a wide variety of bench mark problems, they recommended von Neumann topology due to its consistent performance in comparison to other topologies.

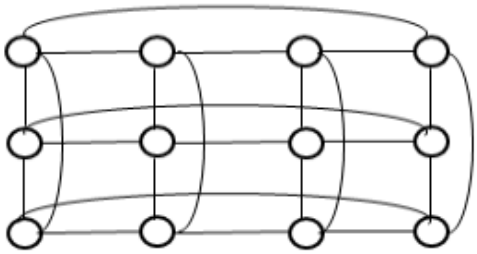

Figure 3.3: von Neumann topology

3.1.9.2. ***Dynamic Topology***

In this kind of topologies the neighbors and the neighborhood are subject to change during the iterative process. In order to improve the exploration and exploitation properties of the PSO algorithm Suganthan [76] proposed a dynamic strategy for defining the population topology. The author suggested to start the iterative process with smaller neighborhoods i.e. lbest topology for exploring the search space more thoroughly and slowly increase the size of neighborhood till the population is fully connected towards the end of iterative process.

Peram et al. [77] proposed a modified variant of the PSO algorithm called FDR-PSO. The authors introduced a new term in the velocity update equation which force a particle to move towards a nearby particle with better prior position along a certain direction. There is no guaranty that the same neighbor will be selected for update along each dimension. The FDR-PSO showed better performance than the standard PSO when tested on a set of bench mark problems.

Janson and Middendorf [78] proposed a hierarchical version of the PSO algorithm (H- PSO) where the particles are arranged in a dynamic hierarchy and during each iteration a particle is

influenced by its own best position and by the best position of the particle directly above it in the hierarchy. The particles move up in the hierarchy if they found better positions during the iterative process. The proposed approach produced better results on most of the bench mark problems.

Liang and Suganthan [79] proposed a dynamic multi swarm particle swarm optimizer (DMS - PSO). In the proposed method the authors divided the swarm into many small subpopulations and then randomly regrouped these subpopulations during the iterative process to exchange information among them. This modified variant of the PSO algorithm produced better results when compared to the other PSO variants.

3.1.10. *Different Variants of the PSO Algorithm*

To improve the performance of the basic PSO algorithm generally or for some specific applications over the years numerous variants of the PSO algorithms have been proposed in the technical literature. Some of these approaches will be briefly discussed here.

3.1.10.1. *Fully Informed Particle Swarm (FIPS)*

In the traditional PSO algorithm the velocity of an individual particle is iteratively adjusted using its current position, its previous velocity and the best positions experienced so far by the particle and by any of its neighbours. Mendes et al. [80] proposed an alternative i.e. Fully Informed Particle Swarm (FIPS) algorithm, where a particles uses all its neighbors to adjust its velocity instead of considering only the best one. The authors extended the constriction coefficients based velocity update equation [74] (as discussed in section 3.1.8) to accommodate the influence of all the neighboring particles on an individual particles movement that can be mathematically represented as:

$$\vec{v}_{i,t} = \chi\left(\vec{v}_{i,t-1} + \sum_{k=1}^{N_i} \varphi_k * \vec{r} * (\vec{p}_k - \vec{x}_{i,t-1})\right) \quad 3.9$$

Where $\varphi_k = \frac{\varphi_{max}}{N_i} \quad \forall\ k\ \in\ N,$

φ_{max} is commonly fixed to 4.1 which results into constant multiplier χ' s value to be 0.7298, N_i represents the total number of neighbors of particle i, $\vec{r}$ is a vector of random real number between 0 and 1, $\vec{p}_k$ is the best position experienced so far by particle k lying in the neighborhood of particle i, $\vec{x}_{i,t-1}$ and $\vec{v}_{i,t-1}$ represents the position and velocity of particle i until iteration $t-1$.

3.1.10.2. *Modified Particle Swarm Optimizer (GCPSO)*

Van den Bergh [81] proposed a modified variant of the PSO algorithm to counter the problem of premature convergence to solutions that are not guaranteed to be the local optimum. The following modified equation was proposed to update the velocity of the best particle of the population or a certain neighborhood:

$$\vec{v}_{b,t+1} = -\vec{x}_{b,t} + \vec{g}_t + \omega\vec{v}_{b,t} + \rho_t\,(1 - 2\,\vec{r}_t) \qquad 3.10$$

Combining this modified velocity update equation of global / local best particle with the position update equation 3.2 results into the following new position update equation:

$$\vec{v}_{b,t+1} = \vec{g}_t + \omega\vec{v}_{b,t} + \rho_t\,(1 - 2\,\vec{r}_t) \qquad 3.11$$

Where b represent the index of the best particle, $\vec{r}_t \in [0,1]$ is a vector of uniform random numbers, $\vec{x}_{b,t}$ term resets the particle's position to the best position, ρ is a scaling factor used to perform a random search around the global best position, its value is modified during the iterative process using an adaptive strategy [82].

3.1.10.3. *Particle Swarm Optimization for Binary Problems*

The original PSO algorithm was proposed to solve continuous optimization problems however the concept was latter on expanded to solve the binary optimization problems as well. The first binary version of PSO algorithm was proposed by Kennedy and Eberhart in 1997 [69]. According to this approach the velocity v_{id}represents the probability of a bit x_{id}taking the value 1 or 0. The velocity update equation remains the same except that personal best ($\vec{\boldsymbol{p}}_{\boldsymbol{i,t-1}}$) and local/global best ($\vec{\boldsymbol{g}}_{\boldsymbol{t-1}}$) are now vectors of binary bits. As v_{id} represents probability of flipping a bit it must be mapped to the interval [0, 1]. A logistic function (or sigmoid function) $S(v_{id})$can be used to do this mapping. The authors used the logistic function $s(v_{id}) = \frac{1}{1+e^{-v_{id}}}$ to map $v_{id}$to generate the probability thresholds. Particles' positions are then updated using the following procedure:

$$\boldsymbol{if}\ rand()\ < S(v_{id})\boldsymbol{then}$$

$$x_{id} = 1$$

$$\boldsymbol{else}\ x_{id} = 0$$

$$\boldsymbol{end\ if}$$

Where $rand$ ( ) is drawn at random from a uniform distribution in the interval [0.0, 1.0]. The maximum allowable velocity V_{max} is used to control the mutation rate of the binary bits. Unlike the real- valued version of PSO where a higher value of V_{max}results in increased exploration, in the binary version of PSO a smaller value of V_{max} increases the mutation rate and so the exploration.

Shuyuan et al. [83] proposed a Quantum PSO algorithm (QPSO) to solve binary optimization problems. According to this approach each particles i possess two vectors i.e. one position vector x_i and a quantum vector Q_i, where the dth component of Q_i vector represents the probability of x_{id} to take the value 1. During the iterative process the Q_i vector is updated as follows:

$$Q_{groupbest}(t) = \alpha * x_{groupbest}(t) + \beta(1 - x_{groupbest}(t)) \quad 3.12$$

$$Q_{selfbest}(t) = \alpha * x_{selfbest}(t) + \beta(1 - x_{selfbest}(t)) \quad 3.13$$

$$Q_{groupbest}(t+1) = c_1 * Q(t) + c_2 * Q_{selfbest}(t) + c_3 * Q_{groupbest}(t) \quad 3.14$$

Where $c_1 + c_2 + c_3 = 1 (0 < c_1, c_2, c_3 < 1)$ and $\alpha + \beta = 1$ $(0 < \alpha, \beta < 1)$. After updating Q_i vector, x_i is updated by generating a random number between 0 and 1 and if $rand() < Q_{id}$ then x_{id} takes the value 1 and else 0. This algorithm needs extra computational effort to tune the additional parameters (α, β) which could be counted as drawback of this approach [84]. In the recent different variant of the PSO algorithm have been proposed in the technical literature to tackle binary optimization problem like Angle Modulated Particle Swarm Algorithm presented by Pampara et al. [85], particle swarm optimization method based on the theory of immunity in biology presented by Afshinmanesh et al.[86], etc. each one with their own advantages and disadvantages, discussing each one of these approaches is beyond the scope of this thesis. A detailed review of different PSO algorithms for solving binary optimization problems can be found in [84] presented by Jordehi and Jasni.

3.1.10.4. *Particle Swarm Optimization for Discrete Problems*

To deal with the problems that involve discrete / integer variables rounding off strategy is commonly used. In this approach the discrete / integer variable are treated as normal continuous variables during the iterative process but at the end these variables are rounded off to the nearest discrete / integer value. The rounding off procedure can be implemented in each iteration or at the end of the optimization process [84, 87-89]. Though this approach makes the implementation process simple and computationally efficient but has the following disadvantages[84]:

- The rounding off process may cause infeasibility

- The fitness value of the rounded off solution could be different from original solution with the possibility that a discrete solution with more distance from the original continuous optimum solution has a better fitness value in comparison to the rounded off solution.

3.1.10.5. *PSO Algorithm for constrained Optimization*

In constraint optimization while optimizing the objective function the solution must satisfy a certain number of constraints as well. Mathematically a constrained optimization problem can be defined as [90]

$$Optimize \quad f(X) \tag{3.15}$$

Subject To: $$g_j(X) \leq 0 \qquad j = 1,2, \ldots\ldots\ldots\ldots\ldots J \tag{3.16}$$

$$h_k(X) = 0 \qquad k = 1,2, \ldots\ldots\ldots\ldots\ldots K \tag{3.17}$$

$$X = (x_1, x_2, x_3 \ldots\ldots\ldots\ldots x_n)^T, \quad X \in [x_{min}, x_{max}]$$

Where X is a vector of search variables in the search space with predefined domain, $f(x_i)$ represents the objective function that has to be optimized, equation 3.16 and 3.17 represents inequality and equality constraints respectively that has to be satisfied during the optimization process. Like other metaheuristic optimization techniques PSO algorithm does not have an explicit mechanism for constraint handling and usually following strategies are used to handle constrained optimization problem [90, 91].

1. Penalty Methods

According to this approach to solve the constrained optimization problem it is turned into an unconstrained optimization problem by incorporating the constraints into the objective function and penalizing their violation during the iterative process. Though these methods are easy to

implement but they do not guarantee a solution where no constraints are violated and the success mainly depends on the penalty method being used.

2. Repair Methods

This approach uses different repair operators to move an infeasible solution near to the feasible solution space. These methods are usually more difficult to implement and might be computationally expensive as well.

3. Constraint Preserving Methods

According to this approach only feasible solutions are generated during the iterative process. The solutions are initialized within the feasible domain and during the iterative process only feasible solutions are generated to maintain the feasibility.

3.2. Bat Algorithm

Bat Algorithm (BA) is a stochastic population based optimization technique first proposed by Yang in 2010 [60] . It is based on the echolocation characteristics of bats. The bats use their echolocation capabilities to find prey and avoid obstacles even in complete darkness. The bats emits short duration (typically in the range of 8 to 10 ms) loud sound pulses with constant frequency in the region of 25 kHz to 150 kHz and listen for the echo that bounces back from the surrounding objects to find the food or avoid the obstacles. They usually emits 10 to 20 such sound pulses per second and can increase the pulse emission rate to about 200 pulses per second as they get closed to their prey. To transform these unique properties of bats into an optimization algorithm Yang idealized the following rules [60, 92] :

1 All bats use their echolocation characteristics to find out its distance from a certain object and can differentiate between food/prey and background barrier in some magical way.

2 Bats fly randomly with velocity v_i at position x_i with a frequency f_{min}, and can vary the wavelength λ and loudness A_0 of their emitted sound pulse to find the food. Depending on

their distance from the prey they can adjust the rate and wavelength or frequency of their emitted pulse.

3 The loudness varies from a large positive value A_0 to a minimum constant value A_{min}

Each bats current position is a potential solution to the optimization problem. The bats use the following rules to update their positions:

$$f_i = f_{min} + (f_{max} - f_{min})\,\beta \quad 3.18$$

$$v_i^{t+1} = v_i^t + (x_i^t - x_*)\, f_i \quad 3.19$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad 3.20$$

Where f_i represents the current frequency of particle i, f_{min}, f_{max} are the minimum and maximum allowable frequencies respectively, initially a random frequency drawn from the uniform distribution in the interval $[f_{min}\,,\, f_{\max}]$ is assigned to each bat.

β is uniform random number between 0 and 1

x_* represents the global best position / solution experienced by the bats (population) so far

x_i^{t+1} *and* x_i^t represents the current and previous positions of particle i

v_i^{t+1} *and* v_i^t represents the current and previous velocities of particle i

In the local search part (step 6 to 9) a new solution is generated using random walk around a solution selected among the currently known best solutions.

$$x_{new} = x_{old} + \varepsilon\, \overline{A^t} \quad 3.21$$

Where $\varepsilon \in [-1, 1]$ is a uniform random number and $\overline{A^t}$ is the average loudness of all the bats in the current time step. The pulse emission rate and the loudness of the bats are updated as the iteration proceed using the following equations:

$$A_i^{t+1} = \alpha A_i^t \quad 3.22$$

$$r_i^{t+1} = r_i^o [1 - e^{(-\gamma t)}] \quad 3.23$$

Where A_i^{t+1} *and* A_i^t are the current and previous loudness of bat i, α and γ are constants and r_i^{t+1} *and* r_i^o are the current and initial pulse rates. The values of A_i^t and r_i^o can typically be [1,2] and [0,1] respectively.

3.2.1. *Different Variants of the BAT Algorithm*

The standard bat algorithm usually shows quick convergence behavior by switching from exploration to exploitation in a very initial stage of the optimization process that may lead to premature convergence causing poor performance [59]. To improve the performance of the standard Bat algorithm and to make it applicable to different optimization problems in the recent years different variants of the Bat algorithms have been proposed in the technical literature. Some of these approaches will be briefly discussed here.

3.2.1.1. *Modified Bat Algorithm (MBA)*

Yilmaz et al. [92] proposed a modified bat algorithm (MBA) to improve the poor exploration capabilities of the of the bat algorithm (BA). They proposed to assign pulse emission rate r and loudness A to each dimension of the solution separately instead of assigning a single value to all the dimensions of a solution and proposed to use the following procedure for updating the position along a certain dimension j of a solution:

Algorithm 2 : Pseudo code of the Bat Algorithm

1. Initialize bat population with random initial positions $\vec{x}_{i,0}$ and random velocities $\vec{v}_{i,0}$ in the search space.
2. Initialize each bat's frequency f_i , pulse rate r_i , and the loudness A_i
3. Calculate the fitness value of each particle at its initial position $\vec{x}_{i,0}$and determine the initial global best position x_*
4. **While** (t < maximum number of generations)
5. Generate a new solution by updating the frequency, velocity and position using equation (3.18), (3.19) and (3.20) respectively.
6. ***If*** (rand > r_i)
7. Select a solution among the best solutions
8. Generate a local solution around the selected best solution
9. ***end if***
10. ***If*** $(rand < A_i \; and \; f(x_i) \; is \; better \; than \; f(x_*))$
11. Accept new solution
12. Increase r_i , reduce A_i
13. ***end if***
14. Ranks the bats and find the current best x_*
15. If the stopping criterion is met:***end while***

$$x_{ij}^{t+1} = \begin{cases} x_{uj}^{t} + \varepsilon \bar{A}_j^t & rand_j > r_{ij} \\ x_{ij}^{t} & Otherwise \end{cases} \quad 3.24$$

Where $\bar{A}_j^t$ represents average loudness of dimension j of all solutions at time t, u indicates a solution selected among the best solution. The loudness and pulse rate are updated as:

$$\mathrm{A}_{ij}^{t+1} = \begin{cases} \alpha\ \mathrm{A}_{ij}^{t} & rand_j > \mathrm{r}_{ij} \\ \mathrm{A}_{ij}^{t} & Otherwise \end{cases} \quad 3.25$$

$$\mathrm{r}_{ij}^{t+1} = \begin{cases} \mathrm{r}_i^o \left[1 - e^{(-\gamma\, t)}\right] & rand_j > \mathrm{r}_{ij} \\ \mathrm{r}_{ij}^{t} & Otherwise \end{cases} \quad 3.26$$

In the MBA a candidate solution is included in the population if $rand < \bar{A}_i$ where $\bar{A}_i$ is the average loudness of a solution i.

3.2.1.2. *Binary Bat Algorithm (BBA)*

Nakamura et al. [93] proposed a binary version of the Bat algorithm i.e. binary bat algorithm (BBA) for feature selection and image processing. The authors modeled the search space as a *D* dimensional Boolean lattice in which the bats moves across the corners of the hypercube to find an optimum position. In this approach the bat's positions are represented by binary vectors. The authors used the following sigmoid function to restrict the position of the individual bats to binary values.

$$S\left(v_i^{\,j}\right) = \frac{1}{1 + e^{-v_i^{\,j}}} \quad \textbf{3.27}$$

$$x_i^{\,j} = \begin{cases} 1 & if \quad S\left(v_i^{\,j}\right) > \rho \\ 0 & Otherwise \end{cases} \quad \textbf{3.28}$$

Where $v_i^{\,j}$ represent the velocity component of the i^{th} bat in the j^{th} direction, ρ represents a uniform random number between 0 and 1.

3.3. Differential Evolution Algorithm

Differential Evolution (DE) is a population based stochastic optimization technique for continuous optimization problems first presented by Storn and Price in 1995 [94-96]. DE algorithm is one of the direct competitors of the PSO algorithm. Like PSO algorithm the DE algorithm is conceptually simple, easy to implement and need only a few user defined control parameters during the optimization process. The DE algorithm uses a population of vectors to search the solution space. Let P be the size of population. Each vector $x_i \in P$ is a potential solution of the optimization problem. The search process starts with the population of vectors randomly initialized in the solution space followed by the application of mutation, cross over and selection operators to transform these vectors over several generations to find the optimum or near to optimum solution of the problem at hand. These operators will be briefly discussed in the following sections.

3.3.1. ***Mutation***

In the mutation process a so called mutant vector is generated for each member (target vector) of the population using a mutation operator. In its simplest form a mutant vector is generated by adding the weighted difference of two randomly chosen vectors to a third randomly chosen vector using equation 3.29:

DE / rand / 1 $$\vec{m}_{i,t} = \vec{x}_{r1,t-1} + F \,.\left(\vec{x}_{r2,t-1} - \vec{x}_{r3,t-1}\right) \quad 3.29$$

The other most commonly used strategies for generating a mutant vector can be presented as [97, 98]:

DE / best / 1 $$\vec{m}_{i,t} = \vec{x}_{best,t-1} + F \,.\left(\vec{x}_{r1,t-1} - \vec{x}_{r2,t-1}\right) \quad \textbf{3.30}$$

DE / current to best / 1

$$\vec{m}_{i,t} = \vec{x}_{i,t-1} + F \cdot \left(\vec{x}_{best,t-1} - \vec{x}_{i,t-1}\right) + F \cdot \left(\vec{x}_{r1,t-1} - \vec{x}_{r2,t-1}\right) \quad \textbf{3.31}$$

DE / best / 2

$$\vec{m}_{i,t} = \vec{x}_{best,t-1} + F \cdot \left(\vec{x}_{r1,t-1} - \vec{x}_{r2,t-1}\right) + F \cdot \left(\vec{x}_{r3,t-1} - \vec{x}_{r4,t-1}\right) \quad \textbf{3.32}$$

DE / rand / 2

$$\vec{m}_{i,t} = \vec{x}_{r1,t-1} + F \cdot \left(\vec{x}_{r2,t-1} - \vec{x}_{r3,t-1}\right) + F \cdot \left(\vec{x}_{r4,t-1} - \vec{x}_{r5,t-1}\right) \quad \textbf{3.33}$$

DE / rand to best / 2

$$\vec{m}_{i,t} = \vec{x}_{r1,t-1} + F \cdot \left(\vec{x}_{best,t-1} - \vec{x}_{i,t-1}\right) + F \cdot \left(\vec{x}_{r2,t-1} - \vec{x}_{r3,t-1}\right) + F \cdot \left(\vec{x}_{r4,t-1} - \vec{x}_{r5,t-1}\right) \quad \textbf{3.34}$$

Where DE / x / y notation is commonly used to identify a particular mutation scheme with DE stands for differential evolution, x specifies the vector to be mutated and y represents the number of difference vectors used [94] $\vec{m}_{i,t}$ represents the mutant vector**,** r_1, r_2, r_3, r_4, r_5 represents randomly chosen and mutually exclusive vector indices these indices must be different from the running index i as well, F represent a constant real factor with value typically chosen between 0 and 1 and is used to control the influence of differential variation [95], $\vec{\boldsymbol{x}}_{\boldsymbol{best,t-1}}$ represents the individual vector with best fitness value experienced by the population so far.

3.3.2. *Cross Over*

During this process a trail vector ($\vec{\boldsymbol{u}}_{\boldsymbol{i,t}}$) is generated by mixing the components of the mutant vector ($\vec{m}_{i,t}$) and the target vector ($\vec{\boldsymbol{x}}_{\boldsymbol{i,t-1}}$). The cross over operation is used to enhance the potential diversity of the population[95]. The following binominal cross over criteria is commonly used for this purpose:

$$\vec{u}_{ji,t} = \begin{cases} \vec{m}_{ji,t} & if\ (rand\ (j) \leq CR)\ \ or\ \ \ j = rnbr\ (k) \\ \vec{x}_{ji,t-1} & if\ (rand\ (j) > CR)\ \ or\ \ \ j \neq rnbr\ (k) \end{cases} \quad 3.35$$

Where $\vec{u}_{i,t}$ represents the trail vector, j represents the j^{th} dimension of the trail vector, CR is the user defined cross over constant, $rand\ (j)$ represents the j^{th} evaluation of a uniform random number$\in$ [0,1]. $rnbr\ (k)$ is a randomly chosen index that is used to ensure that the trail vector ($\vec{u}_{ji,t}$) gets at least one parameter from the mutant vector ($\vec{m}_{ji,t}$)

3.3.3. *Selection*

During this process the trail vector ($\vec{u}_{ji,t}$) is compared to the target vector ($\vec{x}_{ji,t}$) in terms of the fitness value to decide whether to retain it or not. The target vector ($\vec{x}_{ji,t}$) is replaced by the trail vector ($\vec{u}_{ji,t}$) if it yields better solution value otherwise it is discarded.

3.3.4. *Control Parameters*

Differential evolution (DE) algorithm has three main control parameters: Population size P, scale factor F and the crossover constant CR. These parameters have to be properly tuned to improve the overall performance of the DE algorithm for a particular problem at hand that could be a difficult task. Storn and Price [96] suggested that to get reasonable performance the population size could be chosen between 5-D and 10-D (where D represents the dimensionality of the problem), with good initial value of F to be 0.5. The effective range of F is usually between 0 and 1. The crossover parameter CR controls the probability of components exchange between the mutant and a target vector with effective values between 0 and 1. A smaller value of CR means that small numbers of parameters are changed in each iteration / generation and vice versa. For separable or decomposable functions such as $f(X) = \sum_{i=1}^{D} f_i\ x_i$ normally CR with value between 0 and 0.1 is recommended for getting acceptable results with higher probability of convergence [95, 98]. A detailed discussion over choice of these parameters can be found in [95, 98].

Algorithm 3 : Pseudo code of the Differential Evolution (DE) Algorithm

1. Initialize the control parameters: scale factor F and crossover rate CR and the population size P
2. Initialize DE population with random initial positions $\vec{x}_{i,0}$
3. Calculate the fitness value of each vector at its initial position $\vec{x}_{i,0}$and determine the initial global best position x_{best}
4. **While (** t < maximum number of generations)
5. ***for each solution do***
6. Generate a mutant vector ($\vec{m}_{i,t}$) using a particular mutation scheme as explained in section 3.3.1
7. Generate a trail vector ($\vec{U}_{i,t}$) using Cross over operator as explained in section 3.3.2
8. Calculate the fitness value of the trail vector $\vec{U}_{i,t}$
9. ***If*** $fitness(\vec{U}_{i,t})$ is better than the $fitness(\vec{x}_{i,t-1})$***then***
10. $\vec{x}_{i,t} = \vec{U}_{i,t}$
11. ***end if***
12. ***end for***
13. Ranks the population to find the global best position x_{best}
14. If the stopping criterion is met: ***end while***

Chapter 4. Application of the Metaheuristic Techniques to the Long Term Production Scheduling Problem of the Open Pit Mines

In this chapter a framework for the application of three different populations based metaheuristic techniques to the long term production scheduling problem of the open pit mines will be introduced. In the proposed framework the production scheduling problem has been turned into optimum depth determination problem which seeks to find optimum depth to be mined along a certain column of the block model in a certain period. The main reasons for using the proposed approach can be described as:

- A real sized open pit mine may contain thousands to millions of blocks inside the ultimate pit limits that has to be scheduled over a time horizon typically ranging from 5 to 30 years, making these scheduling decisions on the block level may be computationally very expensive.

- The blocks in th**e** same column of a block model are stacked on top of each other that defines a precedence relationship among these blocks combining this feature of block model with one dimensional nature of metaheuristic techniques (being used in this study) dictates that the block scheduling problem can be turned into optimum depth determination problem which makes the underlying calculations much faster and the implementation quit simple.

During the iterative process these depths are manipulated using the governing rules and equations of a particular metaheuristic technique to find the optimum or near to optimum solution of the problem. The proposed framework is quit flexible and can easily accommodate different real valued population based metaheuristic techniques. The main components of the proposed framework will be briefly discussed in the following sections.

4.1. Initial Solutions

To generate the initial population of random feasible solutions the following two different approaches have been used:

4.1.1. *Constraint Programming Approach*

In this approach the open pit production scheduling problem was turned into a constraint satisfaction problem which was then solved using the CP optimizer of the of the commercially available optimization software CPLEX to get multiple feasible solutions. A Constraint satisfaction problem needs the following three pieces of information to define a problem:

- A set of decision variables
- A finite set of possible values for each variable
- Constraint defining the inter variable relationship

The solution to this problem is to determine the value of every variable from its domain such that all the constraints are satisfied. This is achieved by using different search algorithms such as arc and path consistency, simple backtracking and forward checking and different heuristics to guide the search [99]. Though this technique can generate quit diverse initial population (as shown in Figure 4.1) but becomes computationally expensive when applied to larger problems so its use was stopped later on.

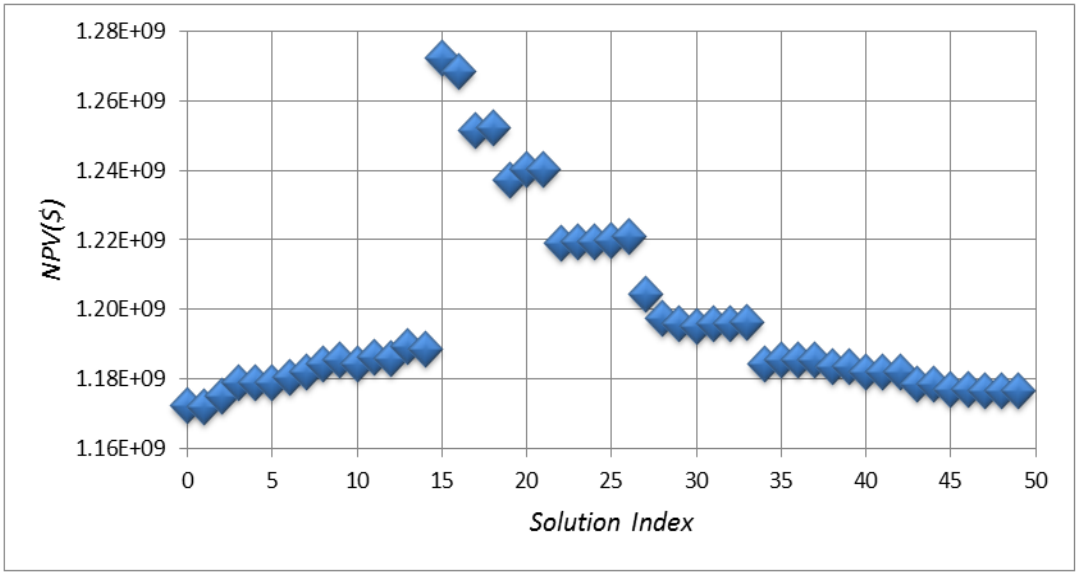

Figure 4.1: objective function values (NPV) of the initial population generated using constraint programming technique

4.1.2. *Heuristic Approach*

To efficiently generate an initial population of random feasible solutions a fast sequential heuristic procedure was developed latter on. In this approach the solution generation process starts by generating a list of free blocks that are available to be mined in the current period because of one of the following reasons:

- Their predecessors are either already being scheduled to be mined in the current or prior period

Or

- Due to their position in block model they do not have one.

A block is selected at random from the list of these free blocks and assigned to the current period and the list is updated again. The process continues for the current period until the mining and processing capacities for it are satisfied in the average sense before moving on to the next period. Mining and processing capacity constraints are handled as hard and soft constraints respectively during this process. The heuristic approach gives more preference to satisfy the processing

capacity constraints in each period during this process. In the case of production scheduling under grade uncertainty more preference was given to the blocks with highest probability of being ore by considering the set of simulated values for that particular block and the predefined cutoff grade. The process stops when either the free blocks list is empty or the number of periods is finished for the current solution. To further diversify the generated solutions the annual mining capacity per period for each solution is chosen at random from the interval between average and maximum allowed mining capacity for that period.

4.2. Solution Encoding

As mentioned before that in the proposed framework the production scheduling problem of the open pit mines has been turned into optimum depth determination problem. This so called optimum depth determination problem aims to find the optimum depth to be mined along a certain column of the block model in a particular period. In order to determine these mining depths in each column for each period a so called solution encoding scheme has been used. The proposed solution encoding scheme determines the deepest block along a certain column to be mined in a certain period. This value (depth) is assigned to a variable corresponding to that column and period. The values of these "depths variables" are then updated / manipulated using different metaheuristic techniques in an effort to find better solutions.

4.3. Back Transform

Once the values of the depth variables have been updated during iterative process a so called back transform scheme is used to determine the period to which a block has been assigned. The procedure takes the depth variable for a certain column and period and determines all the blocks that are lying above that depth and below the depth being defined for the same column for the prior period. Obviously there is no upper limit for the first period. This process progresses from the first to the last period. During the optimization process it is ensured that depth for the successive periods along a certain column do not cross each other.

4.4. Constraint Handling

Like other metaheuristic algorithms the one applied in the current study does not have an explicit mechanism for constraint handling. Considering the special structure of the constraints involved in production scheduling problem of the open pit mines the following two different constraint handling techniques have been used.

4.4.1. *Solution Normalization*

In the proposed framework the mining depths along each column for each period are determined independently from each other which may result in an infeasible solution in terms of the required slope angles as shown in Figure 4.2. After a solution has been back transformed using the back transform procedure (as explained in section 4.3) a so called solution normalization procedure is used to turn it into a feasible solution in term of the required slope angle. The said procedure progresses from the first period to last period while moving up from the deepest bench to top most bench. During this procedure the following two conditions for each block are checked to assign it to a certain period

- If a block has already been assigned to the current period by the back transform procedure assign it to the current period

else

- If not check whether any of its immediate successor blocks are assigned to the current or later periods if yes assign it to the current period.

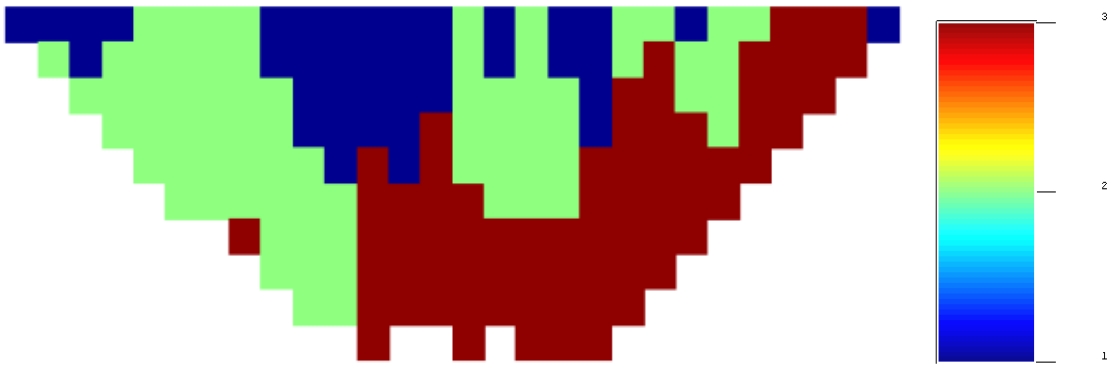

Figure 4.2: A three period infeasible back transformed solution in terms of the required slope angles (45^0 in this case)

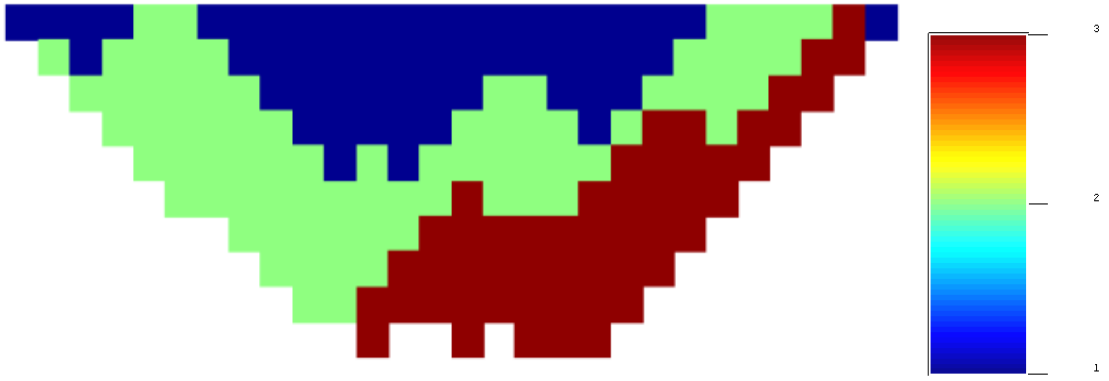

Figure 4.3: Feasible solution in terms of the required slope angles (45^0in this case) after applying the normalization procedure.

4.4.2. *Penalty method for capacity constraint violation*

To deal with the violations of the capacity constraints a constant penalty method has been used. According to this approach a penalty is added to the objective for per ton violation of the capacity constraints to decrease the quality of the infeasible solutions. To implement this technique the following modified objective functions have been used during the iterative process for finding better production schedule with or without the condition of grade uncertainty.

- Modified objective function for the Deterministic variant of LTPS

$$\text{Max} \sum_{t=1}^{T}\sum_{i=1}^{N} v_i^t\, x_i^t \;-\; p_t^{o-}\, y_t^{o-} - p_t^{o+}\, y_t^{o+} \;-\; p_t^{M-}\, R_t^{M-} - p_t^{M+}\, R_t^{M+} \qquad \textbf{4.1}$$

Where

v_i^t: represent the discounted block value (*constant*).

$$x_{it} = \begin{cases} 1 & if\ block\ i\ is\ mined\ in\ period\ t \\ 0 & otherwise \end{cases}$$

p_t^{M-}, p_t^{M+}: Represent the discounted unit costs (Penalty) for shortage or surplus Rock (ore + waste) produced in period t respectively (*constant*).

p_t^{o-}, p_t^{o+}: Represent the discounted unit costs (Penalty) for shortage or surplus ore produced in period t respectively (*constant*).

$R_t^{M-}\ and\ R_t^{M+}$: Represents the shortage or excess amount of rock (ore + waste) produced in period t (*Variable*).

$y_t^{M-}\ and\ y_t^{M+}$: Represents the shortage or excess amount of ore produced in period t respectively (*Variable*).

The value of the penalty is problem dependent and is obtained for each problem using trial and error procedure.

- Modified objective function for the Stochastic variant of LTPS

$$Max \sum_{t=1}^{T}\left[\sum_{i=1}^{N} E\{(NPV)_i^t\}\, x_i^t \; - \frac{1}{s}\sum_{s=1}^{S}(p_t^{o-}\, y_{ts}^{o-} \; + \; p_t^{o+}\, y_{ts}^{o+} + p_t^{m-}\, y_{ts}^{m-} + \; p_t^{m+}\, y_{ts}^{m+})\right] \qquad 4.2$$
$$- \; p_t^{M-}\, R_t^{M-} \; + \; p_t^{M+}\, R_t^{M+}$$

Where

T: Total Number of periods

t: Time period index, t = 1, 2 …. T

N: Total number of blocks

i: Block index, $i = 1, 2 \ldots N$

S: total number of simulations of the ore body

s: Simulation index s = 1, 2 … S

$E\{(NPV)_i^0\}$: is the undiscounted expected economic value of block i and is calculated as follows

$$E\{(NPV)_i^0\} = \; \sum_{s=1}^{S} BV_{is}^0 \Big/ S$$

Where BV_{is}^0 is the undiscounted economic value of block i according to simulation s.

$E\{(NPV)_i^t\}$: is the expected discounted economic value of block i if mined in period t and is calculated as follows:

$$E\{(NPV)_i^t\} = \; E\{(NPV)_i^0\}/(1 + dis1)^t$$

$\boldsymbol{x_{it}}$: are the first stage scenario independent binary decision variables which take the value 1 if block i is mined in period t and 0 otherwise.

p_t^{o-} , p_t^{o+}: are the discounted unit costs for shortage or surplus ore produced in period t respectively and are calculated as:

$$p_t^{o-} = \frac{p_0^{o-}}{(1+dis2)^t}, \quad p_t^{o+} = \frac{p_0^{o+}}{(1+dis2)^t}$$

p_t^{m-} , p_t^{m+}: are the discounted unit costs for shortage or surplus metal produced in period t respectively and are calculated as:

$$p_t^{m-} = \frac{p_0^{m-}}{(1+dis2)^t} \quad , \quad p_t^{m+} = \frac{p_0^{m+}}{(1+dis2)^t}$$

y_{ts}^{o-} , y_{ts}^{o+}: are the second stage scenario dependent continuous variables represent the shortage or surplus amount of ore produced in period t if scenario s occurs respectively.

y_{ts}^{m-} , y_{ts}^{m+} : are the second stage scenario dependent continuous variable represents the shortage or surplus amount of metal produced in period t if scenario s occurs respectively. The second stage (recourse) decision variables are dependent on the outcome of the ore body realizations and of the first stage decision variables i.e. $\boldsymbol{x_{it}}$.

w_i : Tonnage of block i

o_{is} : ore content of block i according to simulation s

p_t^{M-} , p_t^{M+}: Represent the discounted unit costs (Penalty) for shortage or surplus Rock (ore + waste) produced in period t respectively (*constant*).

R_t^{M-} *and* R_t^{M+} : Represents the shortage or excess amount of rock (ore + waste) produced in period t (*Variable*).

4.5. Numerical Experiments

The proposed framework for generating long term production schedule of the open pit mines has been implemented using Microsoft Visual studio 2010 (C++) programming environment. The capabilities and the efficiency of different variants of the PSO, Bat and DE algorithms will be checked using this program. All the numerical experiments have been completed on AMD Phenom IIX 4 945 (3 GHz) and 4 GB Ram running under windows 7.

4.5.1. *Long Term Production Scheduling (LTPS) under Input Data Certainty*

In this section the results of some numerical experiments will be discussed where it is assumed that all the input data is known with certainty and represent the reality. This approach (as explained in Chapter 2) is commonly categorized as the conventional / traditional way of dealing with the production scheduling problem and will be referred to as the "deterministic variant" of LTPS throughout this study.

Two different data sets representing two different hypothetical copper deposits have been used to conduct the numerical experiments in this part of the study. The process started by first dividing the blocks into two categories i.e. ore and waste blocks using a predefined fixed cutoff grade strategy followed by the definition of the economic block model. The ultimate pit limits for both the deposits were determined by solving the following ultimate pit limit problems using commercial solver CPLEX. The required slope angles were assumed to be 45^0 in all directions.

$$Maximize \sum_{i=0}^{N} v_i x_i \qquad 4.3$$

Subject To:

$$x_i \leq x_j \quad \forall\, i \quad j \in P_i \qquad 4.4$$

Where v_i represents the economic value of block i, N represents the total number of blocks in the block model, x_i represents a binary variable corresponding to block i which take the value 1 if

the block i is inside the UPL and 0 otherwise and P_i represents the predecessor group of block i to ensure stable pit slopes.

The following assumptions were made to carry out the numerical experiment for determining the long term production schedule of the blocks lying inside the predefined ultimate pit limits.

- The length of each scheduling period was assumed to be 1 year.
- The discount rate was assumed to be 8% per year.
- The upper and lower limits for Processing and Mining Capacity were set to be within ± 15% of the average available quantity of ore and rock within the UPL for each period of the scheduling horizon respectively.

The integer programming formulation of the production scheduling problem (as described in Chapter 2 section 2.1) using previously mentioned data sets and parameters was solved using commercial solver CPLEX, this solution will be used as a benchmark to assess the performance of the different variants of the Meta heuristic algorithms in terms of computational time and solution quality. Metal production constraints were not considered in this process. General information of the blocks lying inside the ultimate pit limits and about the solutions found by CPLEX are given in Table 4.1

Problem No	Number of Blocks	Block weight (tons)	Block size (m)	Number of Periods	Optimality Gap (%)	CPU Time (hours)
1	10120	5715	10x10x10	5	12.61	215.33
2	7836	5715	10x10x10	4	4.80	105.25

Table 4.1: General information about the blocks within UPL and optimization results

The performance of all the variants of the Meta heuristic algorithms were found out to be greatly affected by the numerical values of the penalties used during the iterative process. Using relatively higher values were causing the algorithm to get trapped in a local optimum without

exploring better solution regions while on the other hand using very low penalties were causing excessive violation of the production capacity constraints. To deal with this problem the following two different sets of penalties have been used during the optimization process.

- **Algorithmic Penalties:** These penalties were used during the iterative process for getting acceptable results in terms of the quality of the final solution produced. Their values are data dependent and are needed to be determined using trial and error procedure.

- **Actual penalties:** These are data independent penalties and were used to calculate fitness value of the final solution produced at the end of the iterative process. For the current study their values were assumed to 4 $ and 7 $ for per ton violation of mining and processing capacity constraints respectively.

Due to the stochastic nature of the metaheuristic algorithms each problem was solved 10 times using different variants of the Metaheuristics techniques under study. The average relative % Gap between best solutions generated by the different variants of the metaheuristic algorithms i.e. Z_{Meta} and the optimal solution found by CPLEX i.e. Z_{CPLEX} are calculated using the following equation:

$$\%\ Gap = \frac{Z_{CPLEX} - Z_{Meta}}{Z_{Meta}} * 100$$

In the following sections the results of the numerical experiments with PSO, Bat and Differential evolution algorithms when applied to the deterministic version of LTPS of the open pit mines will be discussed.

4.5.1.1. *Application of PSO Algorithm to the deterministic version of LTPS*

Certain additional steps required by the proposed framework have been added to the standard PSO algorithm to make it applicable to the production scheduling problem of the open pit mines. The pseudo code of the adapted PSO algorithm is given in Algorithm 4. During all the

experiments the values of the inertia weight (ω) was set to 0.7298 and of the acceleration coefficients (c_1 & c_2) were fixed to 1.49445 as proposed by Clerc and Kennedy in [74].

Algorithm 4 : Pseudo code of the PSO Algorithm for production scheduling problem of the open pit mines

---

1. Input: block model, economic and technical parameters
2. Initialize a population (swarm) of particles with random initial positions $\vec{x}_{i,0}$ and random velocities $\vec{v}_{i,0}$ in the search space (i = 1, 2 …….. N).
3. Initialize each particle's personal best position $\vec{p}_{i,0}$ to its initial position $\vec{x}_{i,0}$
4. Calculate the fitness value of each particle at its initial position $\vec{x}_{i,0}$and determine the initial global best position $\vec{g}_0$
5. ***While*** **(** t < maximum number of generations)
6. Encode each particle's current position ($\vec{x}_{i,t}$), its personnel best position ($\vec{p}_{i,t}$), and the populations' global best position ($\vec{g}_t$)
7. ***for all*** particles ***do***
8. Update the particle's velocity and position using equation of the PSO Algorithm
9. Back transform
10. Normalize the solution
11. Calculate the fitness value of the particle at its current position $\vec{x}_{i,t}$
12. ***If*** $fitness(\vec{x}_{i,t})$ is better than the $fitness(\vec{p}_{i,t-1})$***then***
13. $\vec{p}_{i,t} = \vec{x}_{i,t}$
14. ***end if***
15. ***If*** $fitness(\vec{p}_{i,t})$ is better than the $fitness(\vec{g}_{t-1})$***then***
16. $\vec{g}_t = \vec{p}_{i,t}$
17. ***end if***
18. ***end for***
19. If the stopping criterion is met:***end while***

Population size: In comparison to other evolutionary algorithms e.g. Genetic Algorithms, PSO needs smaller population [64]. After conducting a series of experiments a population size containing 50 particles was found out to be performing well for the problems under consideration. A population of 50 feasible solutions (particles) was generated using heuristic procedure described previously. This same population with same random initial solution positions will be used in all the subsequent experiments to make a fair comparison in terms of the quality of the final solution produced by the different variants of PSO algorithm.

The efficiency of the following different variants of the PSO and GCPSO algorithm has been checked during the numerical experiments.

- **Global (Gbest):** it is the one where each particle is influenced by the best particle of the entire population
- **Local (Lbest):** where each particle is only influenced by the best particle of its K immediate neighbours, where K represents here the total number of indexed based neighbours of each particle of the population.
- **Multi start:** Considering the fast convergence behaviour of the Global variant of the PSO/GCPSO algorithm a Multi start strategy has been applied as a diversification strategy, where the particles are reinitialized to their initial position whenever the algorithm does not shows any improvement in Gbest value for a certain number of iteration.

The maximum number of generations (iterations) has been used as termination criteria in this study and its appropriate value was determined after observing the convergence behavior of the PSO algorithm and its different variants when applied to a particular data set (as shown Figure 4.4). The chances of getting better positions in the consecutive iteration were found out to be quit low so instead of using the adaptive strategy for updating the ρ value during the iterative process (as proposed in [82]) a constant value has been used in this study. A value equal to 5 was found out to be working well for the problems under consideration. While applying the local variant of GCPSO algorithm the positions of the local and global best particle are updated using the GCPSO update equation once separated from the rest of the particles to avoid updating their position multiple times in the same iteration.

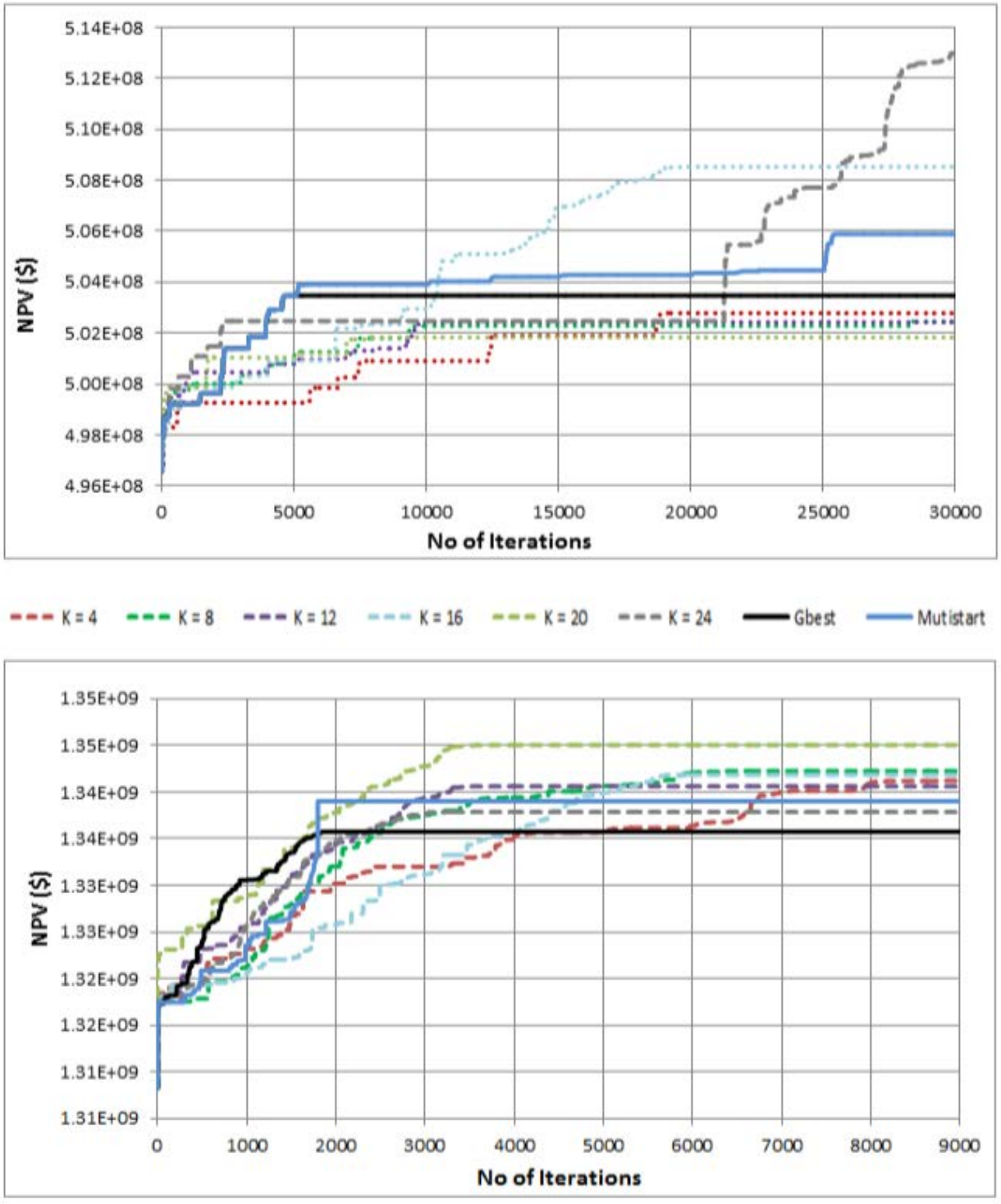

Figure 4.4: Convergence behaviour of the different variants of the PSO algorithm during a single run when applied to Problem 1(Top) and Problem 2 (Bottom) [100]

LOCAL PSO	% GAP		CPU TIME (MINUTES)
	Mean	Standard Deviation	
K = 4	5.46	0.36	132.36
K = 8	5.01	0.51	129.96
K = 12	4.84	0.50	131.77
K = 16	4.67	0.67	132.31
K = 20	5.03	0.63	134.83
K = 24	4.67	0.61	134.25
Global PSO	5.16	0.51	124.34
Multistart PSO	4.43	0.59	124.09

Table 4.2: Numerical results of different variants of the PSO algorithm in the case of Problem 1 [100]

LOCAL GCPSO	% GAP		CPU TIME (MINUTES)
	Mean	Standard Deviation	
K = 4	5.84	0.17	138.83
K = 8	5.36	0.27	143.09
K = 12	5.53	0.31	145.75
K = 16	5.07	0.41	145.34
K = 20	5.38	0.44	145.55
K = 24	5.44	0.31	145.79
Global GCPSO	5.15	0.51	122.40
Multistart GCPSO	4.34	0.50	118.19

Table 4.3: Numerical results of different variants of the GCPSO algorithm in the case of Problem 1[100]

LOCAL PSO	% GAP		CPU TIME (MINUTES)
	Mean	Standard Deviation	
K = 4	2.00	0.24	22.55
K = 8	1.95	0.30	21.25
K = 12	1.82	0.13	21.31
K = 16	1.92	0.16	21.60
K = 20	1.90	0.20	21.76
K = 24	1.99	0.25	21.86
Global PSO	2.23	0.22	22.46
Multistart PSO	2.06	0.27	24.26

Table 4.4: Numerical results of different variants of the PSO algorithm in the case of Problem 2 [100]

LOCAL GCPSO	% GAP		CPU TIME (MINUTES)
	Mean	Standard Deviation	
K = 4	3.28	0.26	22.29
K = 8	2.96	0.39	22.72
K = 12	3.09	0.27	23.17
K = 16	2.70	0.50	23.50
K = 20	2.63	0.63	23.53
K = 24	2.28	0.36	23.94
Global GCPSO	2.67	0.47	20.32
Multistart GCPSO	2.70	0.62	20.12

Table 4.5: Numerical results of different variants of the GCPSO algorithm in the case of Problem 2 [100]

The following observations can be drawn from the results presented in Figure 4.4, Table 4.2, Table 4.3, Table 4.4 and Table 4.5:

- The Global variant of the PSO algorithm is the one where each particle is influenced by the best particle of the entire population and is considered to be more susceptible to get trapped in a local optimum position and has shown similar kind of convergence behavior which can be observed in Figure 4.4

- The local variants shows relatively slower rate of convergence with different neighborhood sizes.

- The Multistart strategy proved to be a better option to overcome the premature convergence problem of the global variants of both the PSO and GCPSO algorithm and on average have produced better solution with smaller relative gap.

- The standard deviation of % Gap in almost all the cases has very small values showing the robustness of the procedure. Generally the average relative gap tends to increase with problem size.

- The computation time required by the algorithm depends on the factors defining and increasing the size of the required computations to be performed before the termination criteria is met such as population's size, number of blocks and number of periods. The computation time increases linearly with the number of iterations for a specific problem with specific algorithmic settings.

- It was observed that the required number of generation's generally increases with the problem size i.e. for larger problem the algorithm needs more generations (iterations) to converge and vice versa.

- The proposed procedure in all the cases can produce sufficiently better quality results in relatively shorter period of time in comparison to the exact optimization algorithms implied by CPLEX.

4.5.1.2. *Application of Bat Algorithm to the deterministic version of LTPS*

The pseudo code of the adapted Bat algorithm is given in Algorithm 5, where some additional steps have been added to make it applicable to the production scheduling problem of the open pit mines. The following parameters have been used during all the numerical experiments.

Parameter Name	Population size	Initial Loudness of the Bats (A_i^o)	Initial Pulse Emission Rate (r_i^o)	f_{min}	f_{max}	α	γ
Value	50	2	1	0.0	0.6	0.9	0.1

Table 4.6: Parameters used during numerical experiments

The values of the so called actual penalties will remain fixed and similar to the one used in the case of PSO algorithm whiles the data and algorithm dependent algorithmic penalties have been adjusted to the appropriate values for getting acceptable results in terms of capacity constraints violations. When the standard procedure for updating the positions of the individual bats was used it was found out that the algorithm converges after a few iterations and the violations of

production capacity constraint increases as the iterative process progresses resulting in poor performance. To tackle these problems the following additional modifications and adjustments have been made:

- **Modified Velocity update equation**

Both variants of the Bat algorithm showed poor performance when the standard velocity update equation (equation 3.19) was used, therefore the following modified equation was used.

$$v_i^{t+1} = \boldsymbol{\omega} v_i^t + (x_i^t - x_*) f_i \qquad 4.5$$

Where $\omega \in [0.4, 0.6]$ was introduced to control the influence of a bat's previous velocity on its current velocity.

- **Maximum Mining Depth**

This parameter was introduced to constrain the maximum allowable depth that could be mined in a certain period along a certain column in an effort to minimize the violations of the production capacity constraints. Appropriate values have to be determined using trial and error procedure for a particular data set.

- **Velocity Clamping**

This parameter was introduced to slow down the convergence rate of the algorithm by controlling the magnitude of the velocity (step size) during the iterative process.

- **Random Walk Factor**

The random walk factor $\delta \in [0.02, 0.09]$ was introduced to control the movement of the bats during the random step part.

$$x_{new} = x_{old} + \delta \varepsilon \overline{A^t} \quad 4.6$$

Algorithm 5 : Pseudo code of the Bat Algorithm for production scheduling problem of the open pit mines

1. Input: block model, economic and technical parameters
2. Initialize bat population with random initial positions $\vec{x}_{i,0}$ and random velocities $\vec{v}_{i,0}$ in the search space (i = 1, 2 …….. N).
3. Initialize each bat's frequency f_i , pulse rate r_i , and the loudness A_i
4. Calculate the fitness value of each particle at its initial position $\vec{x}_{i,0}$and determine the initial global best position x_*
5. **While** (t < maximum number of generations)
6. Encode each particle's current position ($\vec{x}_{i,t}$) and the populations' global best position (x_*)
7. Generate a new solution by updating the frequency, velocity and position using equation (3.18), (3.19) and (3.20) respectively.
8. ***If*** (rand > r_i)
9. Select a solution among the best solutions
10. Generate a local solution around the selected best solution
11. ***end if***
12. Back transform
13. Normalize
14. Calculate the fitness value of each particle at its current position $\vec{x}_{i,t}$
15. ***If*** $(rand < A_i \; and \; f(x_i) \; is \; better \; than \; f(x_*))$
16. Accept new solution
17. Increase r_i , reduce A_i
18. ***end if***
19. Ranks the bats and find the current best x_*

If the stopping criterion is met:***end while***

	% GAP		CPU TIME (SECONDS)	NO OF GENERATIONS
	Mean	Standard Deviation		
Bat Algorithm				
$\omega = 0.6$	5.63	0.80	316	1500
$\omega = 0.4$	5.91	0.68	342	1500
Modified Bat Algorithm				
$\omega = 0.6$	5.65	0.78	433	2000
$\omega = 0.4$	5.79	0.73	495	2000

Table 4.7: Numerical results of different variants of the BAT algorithm in the case of Problem 1

	% GAP		CPU TIME (SECONDS)	NO OF GENERATIONS
	Mean	Standard Deviation		
Bat Algorithm				
$\omega = 0.6$	3.17	0.58	68	600
$\omega = 0.4$	3.04	0.43	68	600
Modified Bat Algorithm				
$\omega = 0.6$	4.95	0.64	278	2000
$\omega = 0.4$	4.72	0.77	215	2000

Table 4.8: Numerical results of different variants of the BAT algorithm in the case of Problem 2

Both variants of the Bat Algorithm show much faster convergence rate and needs relatively less number of iteration and computational time to converge despite the fact that several additional parameters were included to slow down the convergence rate as shown in Figure 4.5. The proper tuning of these parameters for a particular data set might be time consuming as they are needed to be determined through trial and error procedure. Unlike PSO algorithm (described in the previous section) where global best position is updated at the end of each generation in the case

of Bat algorithm it is updated after each function evaluation and so is reported in Figure 4.5. Both variants of the Bat algorithms showed much erratic convergence behavior than PSO algorithm as described in the previous section.

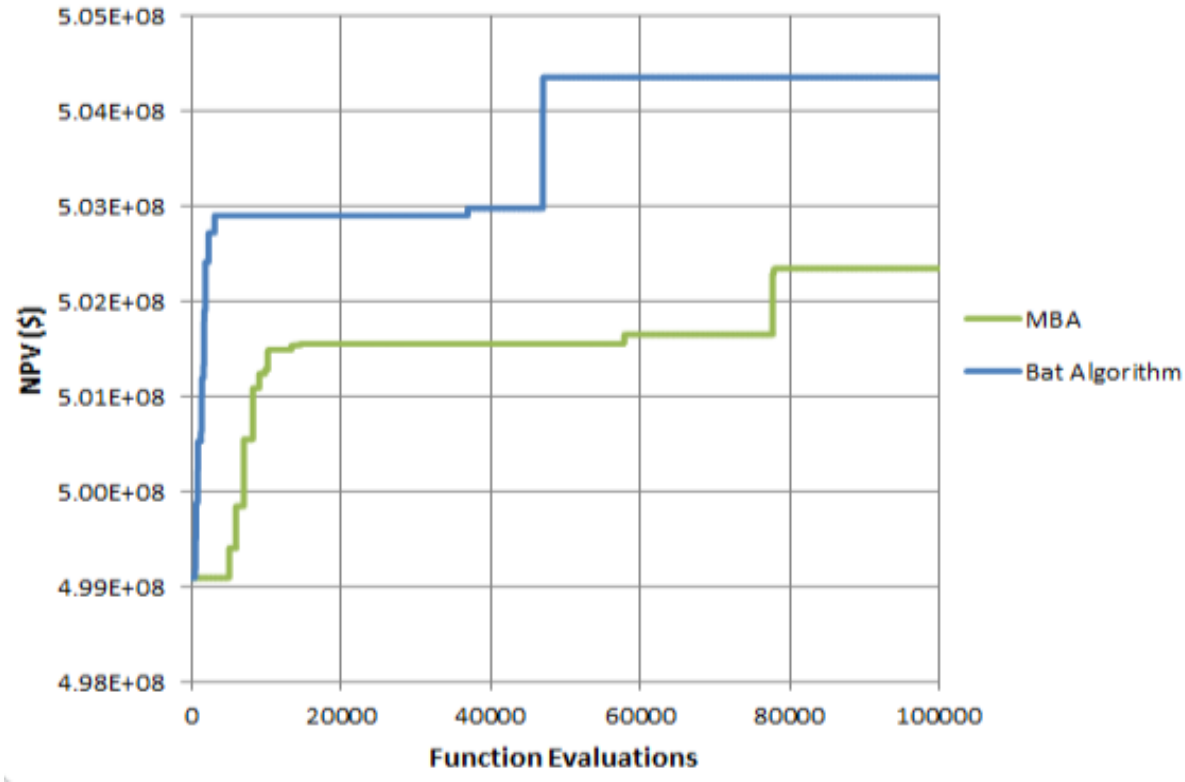

Figure 4.5: Convergence behaviour of Bat and Modified Bat algorithm during a single run when applied to Problem 1

4.5.1.3. *Application of DE Algorithm to the deterministic version of LTPS*

The pseudo code of the adapted Differential Evolution (DE) algorithm is given in Algorithm 6, where some necessary additional steps have been added for applying it to the production scheduling problem of the open pit mines. The values of the so called actual penalties will remain fixed and similar to the one used in previous numerical experiments whiles the data and algorithm dependent algorithmic penalties have been adjusted to the appropriate values for

getting acceptable results in terms of capacity constraints violations. DE algorithm has the following three main control parameters that are needed to be adjusted by the user to get acceptable results for a specific problem at hand.

- Population size P
- Scale factor $F \epsilon$ [0,1]
- Crossover constant $CR \in [0,1]$

Like the numerical experiments described in the previous sections with PSO and BAT algorithms the population size has been fixed to 50 solutions here as well. A series of numerical experiments were carried out to study the effects of scale factor F and crossover constant CR on the overall performance of the DE algorithm and to find out a proper combination of these parameters for the problems under study. The mutation scheme mentioned in equation 3.29 was used during these experiments. The results of the numerical experiments revealed that the performance of the DE algorithm for the LTPS was greatly affected by the value of the crossover constant (CR) being used. After experimenting with different values of CR and F, 0.1 was found out to be the proper value for CR, which almost always produced better quality results when used in different combinations with scale factor F. The numerical experiments further revealed that with CR being fixed to 0.1 the scale factor was generally reducing the convergence rate as it increases as shown in Figure 4.6 and Figure 4.7 for problem 1 and 2 respectively .

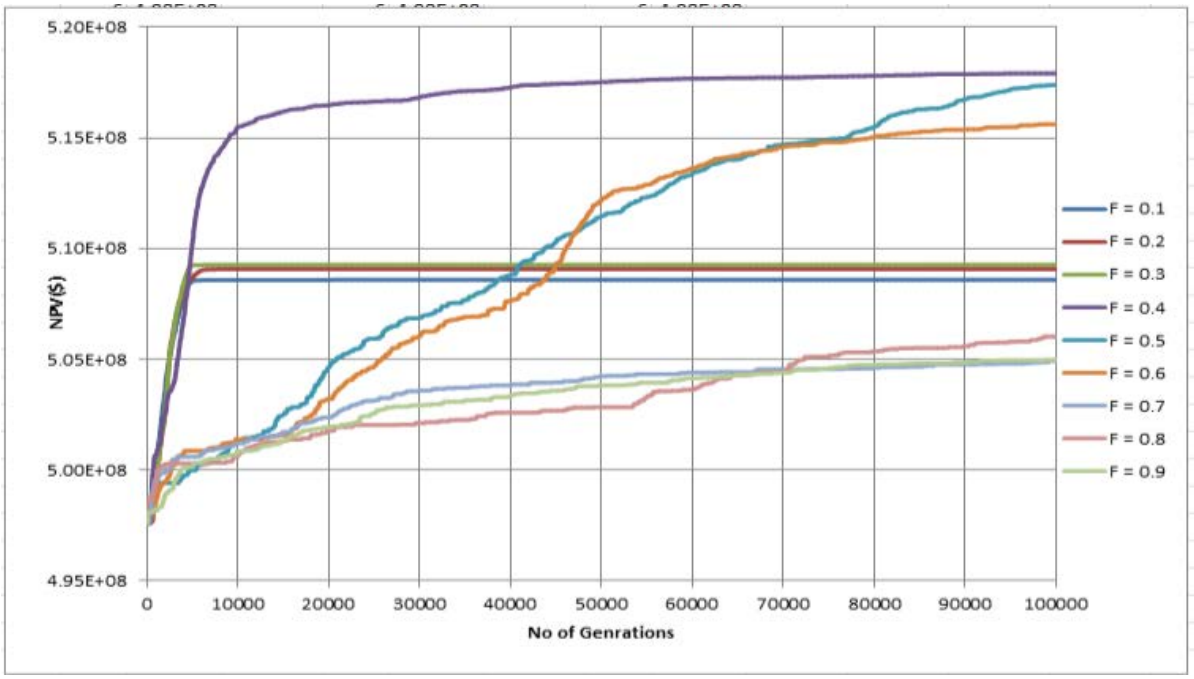

Figure 4.6: Convergence behaviour of DE algorithm during a single run when applied to Problem 1 with CR = 0.1

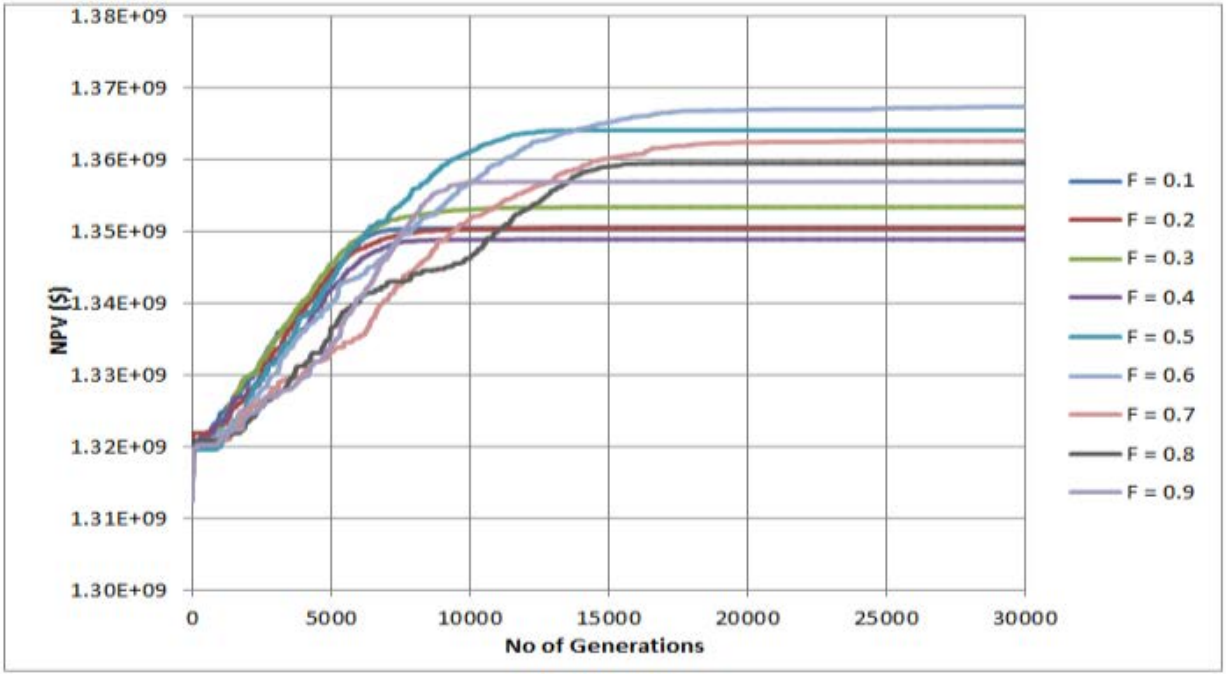

Figure 4.7: Convergence behaviour of DE algorithm during a single run when applied to Problem 2 with CR = 0.1

Algorithm 6 : Pseudo code of the Differential Evolution Algorithm for production scheduling problem of the open pit mines

1. Input: block model, economic and technical parameters
2. Initialize the control parameters: scale factor F and crossover rate CR and the population size P
3. Initialize DE population with random initial positions $\vec{x}_{i,0}$
4. Calculate the fitness value of each vector at its initial position $\vec{x}_{i,0}$and determine the initial global best position x_{best}
5. **While (** t < maximum number of generations)
6. ***for each solution do***
7. Encode each particle's current position ($\vec{x}_{i,t}$)
8. Generate a mutant vector ($\vec{m}_{i,t}$) using a particular mutation scheme as explained in section 3.3.1
9. Generate a trail vector ($\vec{U}_{i,t}$) using Cross over operator as explained in section 3.3.2
10. Back transform
11. Normalize
12. Calculate the fitness value of the trail vector $\vec{U}_{i,t}$
13. ***If*** $fitness(\vec{U}_{i,t})$ is better than the $fitness(\vec{x}_{i,t-1})$***then***
14. $\vec{x}_{i,t} = \vec{U}_{i,t}$
15. ***end if***
16. ***end for***
17. Ranks the population to find the global best position x_{best}
18. If the stopping criterion is met: ***end while***

Considering the stochastic nature of the DE algorithm the optimization process was run 10 times to get some reliable statistics on its performance when applied to the LTPS problem with CR being fixed to 0.1 while varying the value of F from 0.1 to 0.9. Maximum number of iteration was used as the termination criteria and the appropriate values for it were selected after observing the convergence behavior of the DE algorithm for a particular problem and for a particular combination of CR and F values. The DE algorithm was showing much slower convergence rate for $F > 0.4$ for problem 1, therefore the numerical experiments were restricted to $F \leq 0.4$ for problem 1. The results are reported in Table 4.9 and Table 4.10 for problem 1 and 2 respectively.

	% GAP		NO OF GENERATIONS	CPU TIME (MINUTES)
	Mean	Standard Deviation		
F = 0.1	3.97	0.071	15000	53
F = 0.2	3.87	0.270	15000	52
F = 0.3	3.76	0.243	15000	52
F = 0.4	2.54	0.190	50000	172

Table 4.9: Numerical results of DE algorithm in the case of Problem 1 with CR = 0.1

	% GAP		NO OF GENERATIONS	CPU TIME (MINUTES)
	Mean	Standard Deviation		
F = 0.1	1.29	0.064	15000	25
F = 0.2	1.32	0.135	15000	25
F = 0.3	1.34	0.079	15000	26
F = 0.4	1.13	0.232	15000	25
F = 0.5	0.89	0.228	20000	33
F = 0.6	0.89	0.169	20000	33
F = 0.7	0.94	0.184	20000	33
F = 0.8	0.93	0.107	20000	33
F = 0.9	0.96	0.169	20000	33

Table 4.10: Numerical results of DE algorithm in the case of Problem 2 with CR = 0.1

For both the problems the DE algorithm with this particular mutation scheme i.e. DE/ rand / 1 showed much faster convergence rate for $F \leq 0.3$ at the expense of comparatively larger average

% gap. For each problem the DE algorithms shows relatively better performance in terms average % gap with reasonable convergence rate for a particular value of F with CR being fixed to 0.1. These values were identified to be 0.4 and 0.5 for problem 1 and 2 respectively.

Further numerical experiments were conducted to explore the convergence behavior and the efficiency of DE algorithm with other mutation schemes for this particular combination of F and CR value when applied to problem 1 and 2. The results are reported in Table 4.11and Table 4.10 for problem 1 and 2 respectively. For both the problems the mutation scheme no 3 i.e. DE / current to best / 1 showed comparatively faster convergence behavior and needed less number of iterations and consequently less time to produce reasonable results. While on other hand the mutation scheme no 6 i.e. DE / rand to best / 2 showed relatively slower convergence behavior and poor performance in terms of average % gap and standard deviation.

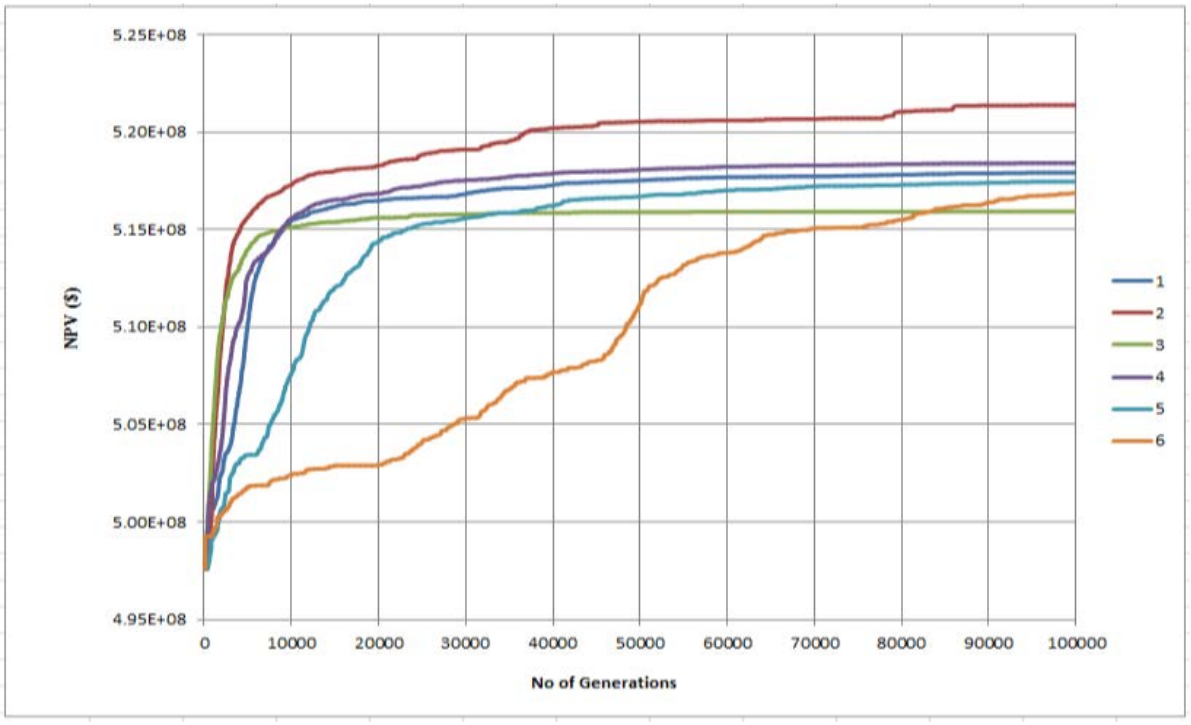

Figure 4.8: Convergence behaviour of mutation schemes with F = 0.5, CR = 0.1during a single run when applied to Problem 1

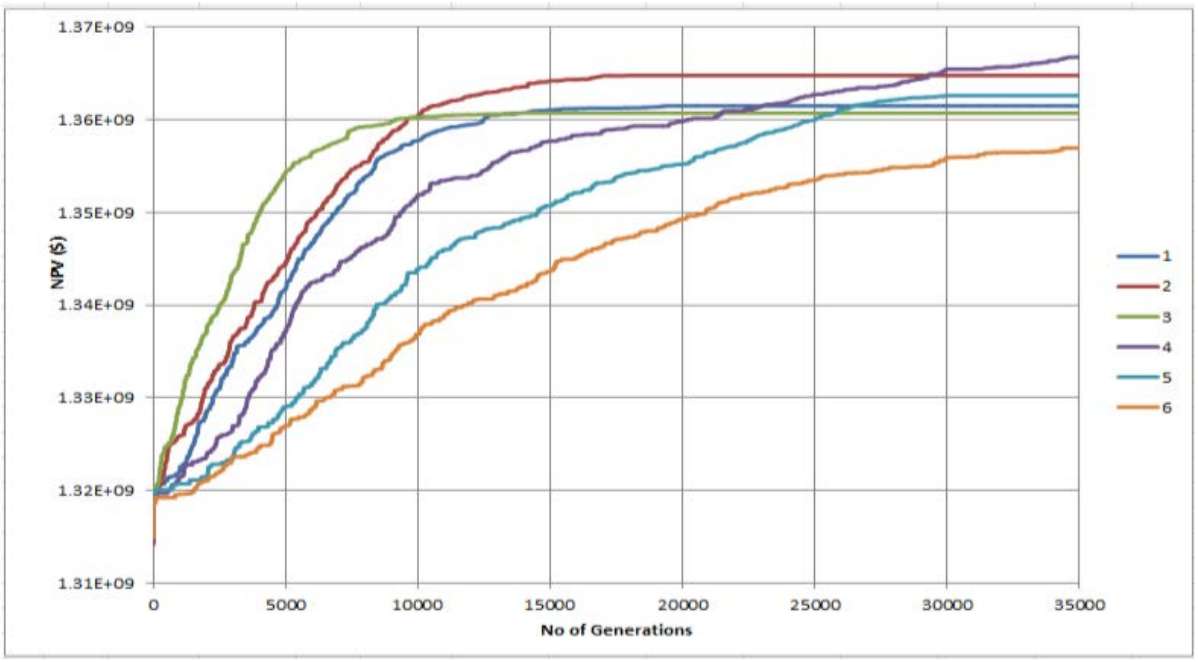

Figure 4.9: Convergence behaviour of different mutation schemes with F = 0.5, CR = 0.1during a single run when applied to Problem 2

NO	MUTATION SCHEME	% GAP		NO OF GENERATIONS	CPU TIME (MINUTES)
		Mean	Standard Deviation		
1	*DE / rand / 1*	2.54	0.190	50000	172
2	*DE / best / 1*	3.052	0.620	50000	169
3	*DE / current to best / 1*	2.87	0.345	25000	86
4	*DE / best / 2*	2.76	0.445	50000	170
5	*DE / rand / 2*	3.16	0.302	50000	169
6	*DE / rand to best / 2*	3.53	0.520	100000	342

Table 4.11:Numerical results of different mutation schemes of DE algorithm in the case of Problem 1 with CR = 0.1 and F = 0.4

NO	MUTATION SCHEME	% GAP		NO OF GENERATIONS	CPU TIME (MINUTES)
		Mean	Standard Deviation		
1	*DE / rand / 1*	0.894	0.228	20000	33
2	*DE / best / 1*	0.561	0.089	20000	33
3	*DE / current to best / 1*	0.643	0.141	15000	25
4	*DE / best / 2*	0.629	0.158	30000	50
5	*DE / rand / 2*	0.635	0.113	30000	50
6	*DE / rand to best / 2*	0.940	0.207	35000	58

Table 4.12: Numerical results of different mutation schemes of DE algorithm in the case of Problem 2 with CR = 0.1 and F = 0.5

4.5.2. *Long Term Production Scheduling (LTPS) under grade Uncertainty*

The results of some numerical experiments of the production scheduling problem under grade uncertainty (as explained in section 2.5) will be discussed in this section. This approach is commonly categorized as the uncertainty based approach for dealing with the production scheduling problem and will be referred to as the "stochastic variant" of LTPS throughout this study. A set of 15 simulated realizations of a copper deposit, generated using Sequential Gaussian simulation technique (SGS) [3] have been used to quantify the grade uncertainty of the deposit. A predefined fixed cutoff grade has been used to determine the category of a particular block according to a particular simulated realization of the orebody. As it may happen that the simulated grade of a particular block is below the predefined cutoff grade according to some simulations while above it according to others. Blocks values are determined for each simulation by taking into account the group to which they belong according to that particular simulation and then these values were used to determine the expected economic values ($E\{(NPV)_i^0\}$) of the individual blocks using equation 4.7 to construct the economic block model.

$$E\{(NPV)_i^0\} = \sum_{s=1}^{S} BV_{is}^0 \Big/ S \qquad 4.7$$

Where BV_{is}^0 is the undiscounted economic value of block i according to simulation.

This economic block model was used to define the ultimate pit limits by solving the UPL formulation (as described in Chapter 2) using commercial solver CPLEX. The required slope angles were assumed to be 45^0 in all directions. In the next step to define the optimum extraction sequence of the blocks lying inside the predetermined ultimate pit limits the technical and economical parameters given in Table 4.13 were used. The length of each scheduling period was assumed to be 1 year. The upper and lower limits for Mining, Processing and Metal capacity were set to be within ± 20% and ±10 of the average available quantity of rock, ore and metal available within the predefined UPL for each period of the scheduling horizon respectively.

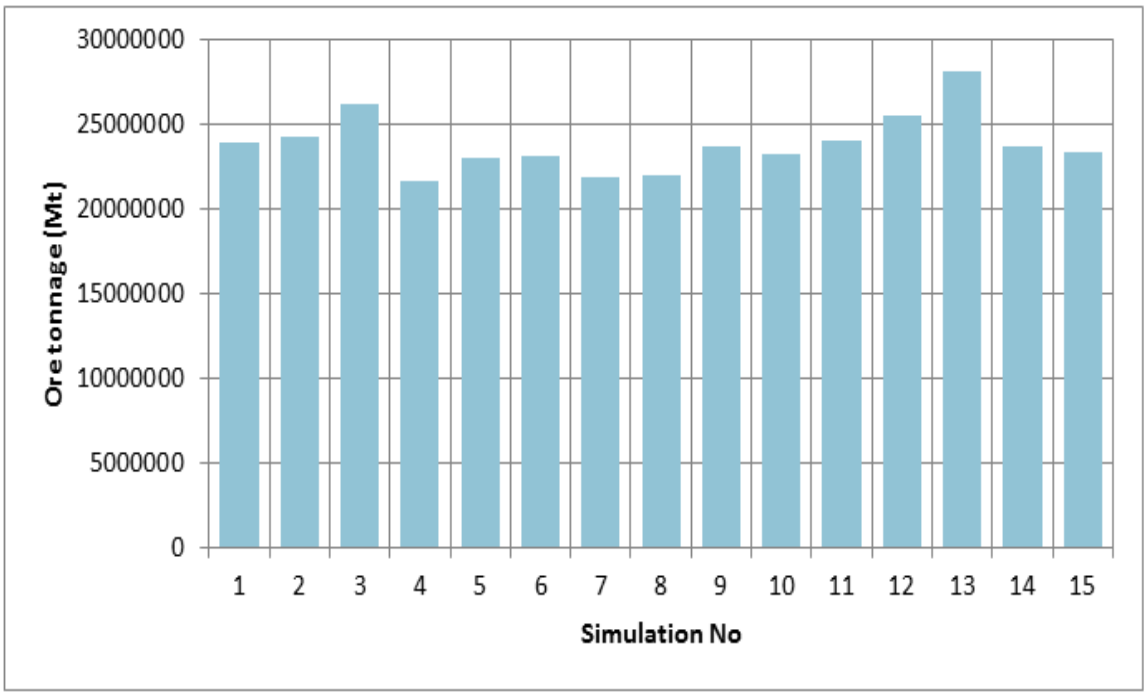

Figure 4.10: The ore content of the deposit according to different realizations of the ore body

Number of block	7107
Number of scenarios	15
Block dimensions (M)	25x25x12.5
Block Tonnage (Mt)	6500
Metal Price ($ /oz)	0.12
Mining ($ / Mt)	2
Processing Cost ($ / Mt)	10
Refining or Selling cost ($ / oz)	6.5E-3
Recovery	85 %
Discount rate (dis1,dis2)	8 %
No of periods (Years)	5

Table 4.13: Technical and economical parameters used in the case study

The two-stage stochastic programming formulation of the open pit production scheduling problem (as explained in section Chapter 2 section 2.5.1) with recourse was solved using the commercial solver CPLEX considering the blocks within the predefined UPL and the parameters mentioned in Table 4.13. This solution will be used as a benchmark to assess the performance of the metaheuristic techniques in terms of computational time and solution quality.

Objective Value (**$**)	1.75818 E 008
CPU time (Hours)	61.63
Optimality Gap (**%**)	6

Table 4.14: General information about the solution found by CPLEX

A population of 50 solutions has been used to conduct the numerical experiment for checking the efficiency and capabilities of the metaheuristic techniques under study i.e. PSO, Bat Algorithm, and DE Algorithm. This population of initial random solution was generated using the greedy heuristic procedure described in section 4.1.2. Maximum number of iterations has been used as termination criteria during these experiments; the appropriate values were determined for each algorithm independently and are reported in the respective results tables. The problem under study was solved 10 times using different variants and parameter settings of the PSO, Bat and DE algorithm to get some reliable statistics about their performance. Like previous experiments average relative % Gap (which will be used a measure of quality of the solutions produced by the meta heuristic techniques under study) between best solutions generated by the different variants of the meta heuristic algorithms i.e. Z_{approx} and the optimal solution found by CPLEX i.e. Z_{CPLEX} are calculated using the following equation:

$$\%\ Gap = \frac{Z_{CPLEX} - Z_{approx}}{Z_{approx}} * 100$$

For the problem under consideration both variants of the PSO algorithm showed better performance than Bat and DE algorithms in terms of the achieved average % Gap. DE algorithm showed relatively slower convergence rate but proved to be more robust.

	% GAP		NO OF GENERATIONS	CPU TIME (MINUTES)
	MEAN	STANDARD DEVIATION		
Global PSO	0.533	0.160	2000	8.14
Multistart PSO	0.528	0.135	2000	8.42
Bat Algorithm				
$\omega = 0.6$	0.817	0.126	2000	8.35
$\omega = 0.4$	0.790	0.094	2000	8.37
Modified Bat Algorithm				
$\omega = 0.6$	0.832	0.107	2000	8.57
$\omega = 0.4$	0.808	0.097	2000	8.43

Figure 4.11: Numerical results of different variants of the PSO and Bat algorithm when applied to the stochastic variant of LTPS

	% GAP		NO OF GENERATIONS	CPU TIME (MINUTES)
	Mean	Standard Deviation		
F = 0.1	1.109	0.028	7000	27
F = 0.2	1.102	0.029	7000	25
F = 0.3	1.121	0.032	7000	25
F = 0.4	1.114	0.022	7000	25
F = 0.5	1.085	0.035	7000	26
F = 0.6	1.075	0.200	7000	26
F = 0.7	1.092	0.055	7000	26
F = 0.8	1.085	0.024	7000	26
F = 0.9	1.086	0.021	7000	26

Figure 4.12: Numerical results of the DE algorithm when applied to the stochastic variant of LTPS with CR = 0.1

Different combinations of F and CR values for DE algorithm were tried to find out the best combination of these parameters. Through these experiments it was learned that reasonable performance of the DE algorithm for the problem under consideration can be achieved with F being fixed to 0.5, and CR to 0.1. This same combination of F and CR values was also used to

study the performance of different mutations schemes. The results of these experiments are given in Table 4.12 and Table 4.13.

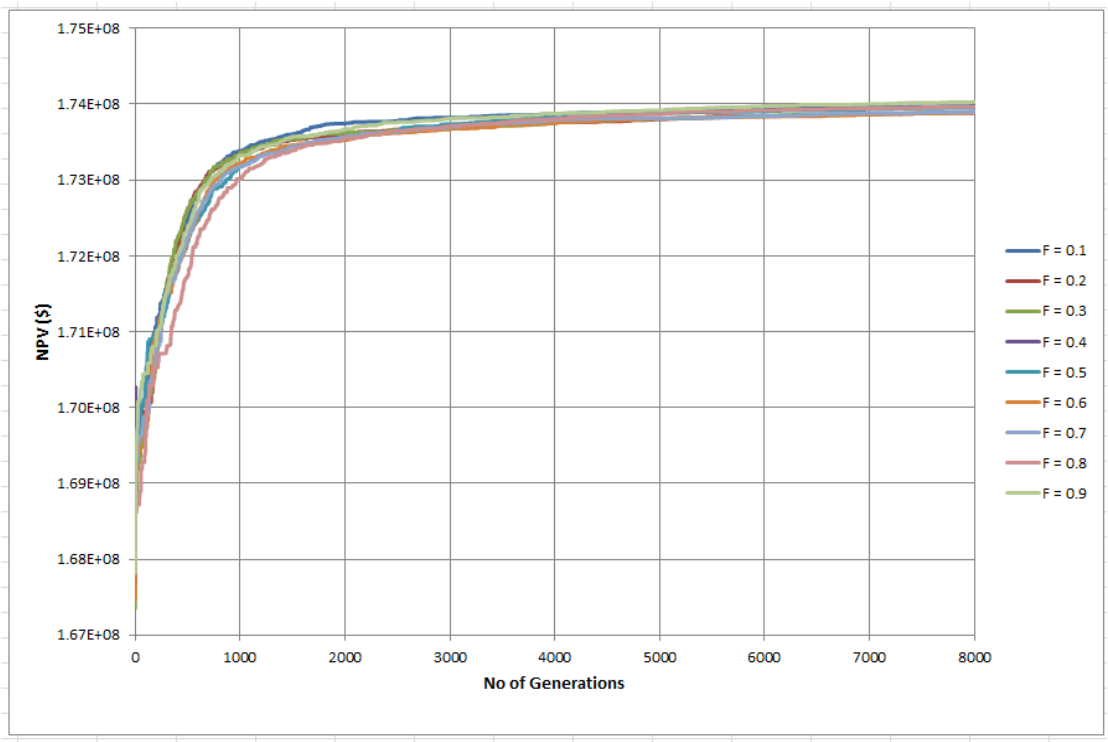

Figure 4.13: Convergence behaviour of DE algorithm during a single run when applied to the stochastic variant of the LTPS with CR = 0.1

Different mutation schemes showed more or less similar behavior and performance as they showed before in the case deterministic variant of the LTPS.

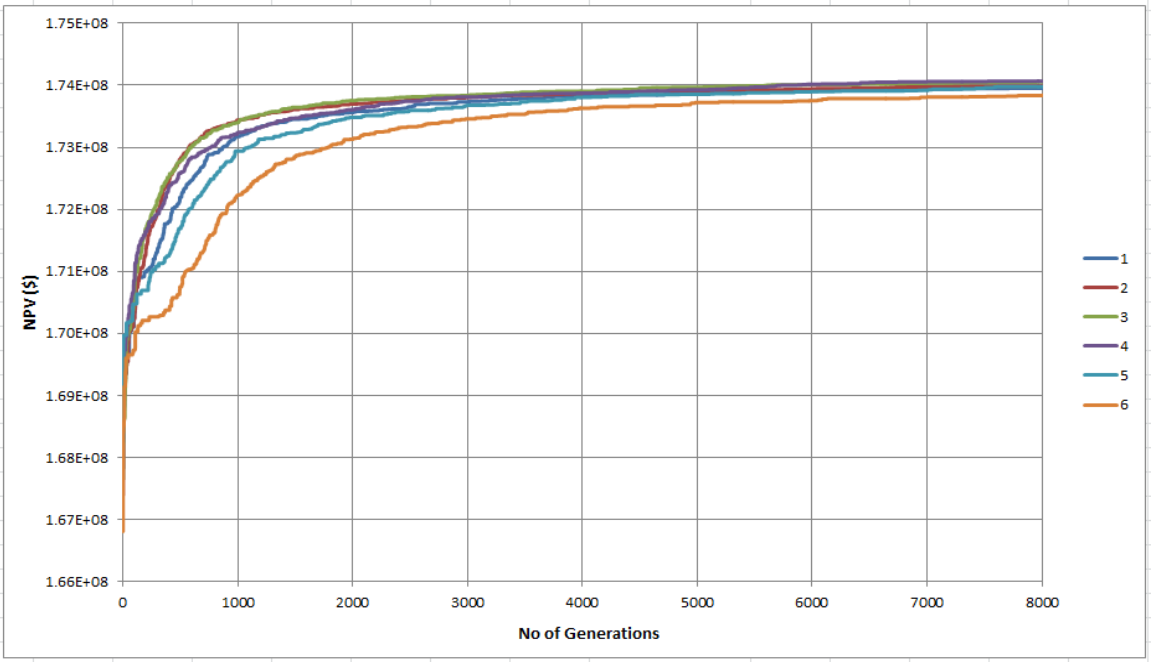

Figure 4.14: Convergence behaviour of different mutation schemes with F = 0.5, CR = 0.1during a single run when applied to the stochastic variant of the LTPS

NO	MUTATION SCHEME	% GAP		NO OF GENERATIONS	CPU TIME (MINUTES)
		Mean	Standard Deviation		
1	*DE / rand / 1*	1.085	0.035	7000	26
2	*DE / best / 1*	1.093	0.030	7000	25
3	*DE / current to best / 1*	1.048	0.026	7000	25
4	*DE / best / 2*	1.042	0.55	7000	26
5	*DE / rand / 2*	1.080	0.033	7000	26
6	*DE / rand to best / 2*	1.048	0.026	7000	25

Table 4.15: Numerical results of the different mutation schemes when applied to the stochastic variant of LTPS with CR = 0.1 and F = 0.5

Chapter 5. Conclusion and Recommendations

5.1. Conclusions

Long term production scheduling (LTPS) is an important and integral part of the planning process of any open pit mines. It aims to define such an extraction sequence of the mineralized material from the ground that produces maximum possible discounted profit i.e. NPV while satisfying a set physical and operational constraints. The following two different approaches are commonly used to handle this problem:

- *Conventional / Deterministic Approaches:* In these approaches it is assumed that all the input data is known with 100% certainty and represents the reality.

- *Stochastic / uncertainty based Approaches:* These approaches take into account the potential uncertainty in the input parameters which may be caused by different market, environmental mining and geology related factors.

Different solution approaches for solving this problem of immense importance to open pit mining operations with or without the condition of grade uncertainty has been proposed in the technical literature. The main limitations of these techniques is either their high computational cost when applied to real sized problem or they can only handle a simplified version of the original problem. This thesis mainly focuses on the development of a framework based on the following different three populations based metaheuristic techniques to handle this problem with low to moderate computational cost:

i. Particle Swarm Optimization (PSO)

ii. Bat Algorithm (BA)

iii. Differential Evolution (DE)

The proposed framework can handle both deterministic and stochastic variants of the long term production scheduling problem (LTPS) of the open pit mines. Instead of making the scheduling decision on the block level the proposed approach turns the problem into optimum depth determination problem along a certain column for a certain period; this makes the implementation much simpler and computationally efficient.

The proposed framework uses a heuristic procedure to generate the initial population of random feasible solutions. A so called encoding scheme is then used to determine the maximum depth to be mined in a certain period along a certain column. These depth variables are then manipulated using the governing rules and equations of a particular metaheuristic technique being employed in an effort to find better solutions. A Back transform scheme is then used to determine the period in which a particular block has been scheduled to be mined. Due to the one dimensional nature of these metaheuristic techniques it is possible that a back transformed solution turns out be infeasible in terms of required slope angles. A so called normalization procedure is then used to turn this infeasible solution into feasible one in terms of required slope angles. To handle the violations of the capacity constraint a penalty method have been used.

Three different case studies have been carried out to check the efficiency and capabilities of the proposed framework. By making comparison with the results obtained using CPLEX in terms of computational time and solution quality it was learned that the proposed procedure can produce better quality solutions in relatively shorter period of time with smaller % Gap and standard deviation showing the robustness of the procedure. The proposed procedure is more structured and quite flexible and can handle all different kinds of objective functions, can easily accommodate additional constraints and grade related uncertainties.

5.2. Recommendations

The thesis was mainly focused on the development of a general framework for applying three different real valued populations based metaheuristic techniques to the long term production scheduling problem of the open pit mines. The initial studies did show some promising results and these results suggests further investigation in the following fields:

- The required slope angles during all the numerical experiments were assumed to be 45^0 degrees in all the directions, however in real mining environment the condition of variable slope angles have to be met, therefore further programming is suggested to incorporate the condition of variable slope angles in the already developed routines.

- It was found out that the computational complexity of the proposed framework is influenced by several different factors such as number of blocks that are needed to be scheduled, number of periods, population size, parameter selection etc. To find out sufficiently good solution for a problem at hand the user may not always have the ability to control these parameters which may sometimes cause higher computational cost. Therefore to improve the computational performance of the proposed framework in such situations its parallel implementation using GPGPU programming technique in suggested.

- The family of population based metaheuristic techniques is not limited to the one proposed in this thesis. The applications of other population based metaheuristic techniques such as Genetic algorithms, Artificial Bee colony optimization, Firefly algorithm etc. using framework proposed in this thesis is suggested to study and compare their behaviors.

- The proposed framework is tested just for one raw material source and processing stream, further programming and testing is suggested to study its ability of dealing with the situation with multiple raw material sources and processing streams.

- Future metal prices and foreign exchange rates cannot be known with certainty at the time the scheduling decision are made which may result in suboptimal solution and can cause huge losses to the mining operation. The integration of these market related uncertainties in optimization process using real options analysis is suggested.

References

1. Hartman, H.L., *Introductory mining engineering*. 1987: Wiley.
2. Goovaerts , P., *Geostatistics for Natural Resources Evaluation* Oxford University Press, 1997.
3. Deutsch , C.V. and Journel, A.G., *GSLIB: Geostatistical Software Library and User's Guide (Applied Geostatistics Series)*. 1997.
4. Menabde, M., Froyland, G., Stone, P., and Yeates, G. *Mining schedule optimisation for conditionally simulated orebodies*. in Proceedings of the international symposium on orebody modelling and strategic mine planning: uncertainty and risk management. 2004. p. 347-352.
5. Gholamnejad, J., Osanloo, M., and Khorram, E., *A chance constrained integer programming model for open pit long termproduction planning [J]*. IJE Transactions A: Basics, 2008. **21**(4): p. 307-318.
6. Ramazan, S. and Dimitrakopoulos, R., *Stochastic optimisation of long-term production scheduling for open pit mines with a new integer programming formulation.* Orebody Modelling and Strategic Mine Planning, Spectrum Series, 2007. **14**: p. 359-365.
7. Denby, B. and Schofield, D., *Open pit design and scheduling by use of genetic algorithms.* Trans. Inst. Min. Metall. (Sec. A: Min. Industry), 103, 1994: p. A21 – A26.
8. Denby, B. and Schofield, D., *Genetic algorithms for open pit scheduling-extension into 3-dimentions*, in *Proceedings of MPES conference*. 1996: Sao Paulo, Brazil. p. 177-185.
9. Kumral , M. and Dowd, P.A., *Short-Term Mine Production Scheduling for Industrial Minerals using Multi-Objective Simulated Annealing*. 2002, Proceedings of 30th APCOM symp.: Fairbanks, Alaska. p. 731-742.
10. Kumral, M. and Dowd, P.A., *A simulated annealing approach to mine production scheduling.* Journal of the Operational Research Society, 56, 2005: p. 922–930.
11. Sattarvand, J. and Niemann-Delius, C., *Long Term Open Pit Planning by Ant Colony Optimization* PhD Dissertation. 2009: Institut für Bergbaukunde III, RWTH Aachen University
12. Sattarvand, J. and Niemann-Delius, C., *A New Metaheuristic Algorithm for Long-Term Open-Pit Production Planning/Nowy meta-heurystyczny algorytm wspomagający długoterminowe planowanie produkcji w kopalni odkrywkowej.* Archives of Mining Sciences, 2013. **58**(1): p. 107-118.
13. Hustrulid, W.A. and Kuchta, M., *Open Pit Mine Planning and Design, Two Volume Set, Second Edition*. 2006: Taylor & Francis.
14. Johnson, T.B., *Optimum open pit mine production scheduling*. Uni-versity of California, Berkeley, CA. Ph.D. Dissertation. 1968.
15. Osanloo , M., Gholamnejad, J., and Karimi, B., *Long-term open pit mine production planning: a review of models and algorithms.* International Journal of Mining, Reclamation and Environment, Vol. 22(1), pp. 3 – 35 2008.
16. Gershon, M.E., *Mine scheduling optimization with mixed integer programming.* Mining Engineering 35(4), 1983 p. 351–354.

17. Gaupp, M.P., *Methods for Improving the Tractability of the Block Sequencing Problem for Open Pit Mining*. Economics Business Division PhD Dissertation. 2008: Colorado School of Mines.
18. Caccetta, L. and Hill, S., *An Application of Branch and Cut to Open Pit Mine Scheduling.* Journal of Global Optimization, 2003. **27**(2-3): p. 349-365.
19. Tachefine, B. and Soumis, F., *Maximal closure on a graph with resource constraints.* Computers & Operations Research, 1997. **24**(10): p. 981-990.
20. Lerchs, H. and Grossmann, I.F., *Optimum design of open pit mines.* Canadian Institute of Mining Trans., 68, 1965: p. 17-24.
21. Hochbaum, D.S. and Chen, A., *Performance Analysis and Best Implementations of Old and New Algorithms for the Open-Pit Mining Problem.* Operations Research, 2000. **48**(6): p. 894-914.
22. Jeff, W., *Open Pit Optimization* Surface Mining (2nd Edition) , Society for Mining Metallurgy and Exploration Inc., 1990.
23. Ramazan , S. and Dimitrakopoulos , R., *Recent applications of operations research and efficient MIP formulations in open pit mining.* SME Transactions 316, 2004.
24. Ramazan, S., *The new Fundamental Tree Algorithm for production scheduling of open pit mines.* European Journal of Operational Research, 2007. **177**(2): p. 1153-1166.
25. Tabesh, M. and Askari-Nasab, H., *Two-stage clustering algorithm for block aggregation in open pit mines.* Mining Technology, 2011. **120**(3): p. 158-169.
26. Goodwin, G.C., Seron, M.M., Middleton, R.H., Zhang, M., Hennessy, B.F., Stone, P.M., and Menabde, M., *Receding horizon control applied to optimal mine planning.* Automatica, 2006. **42**(8): p. 1337-1342.
27. Bley, A., Boland, N., Fricke, C., and Froyland, G., *A strengthened formulation and cutting planes for the open pit mine production scheduling problem.* Computers & Operations Research, 37(9), 1641-1647. .
28. Dagdelen, K. *Open pit optimization–strategies for improving economics of mining projects through mine planning*. in Proceedings 17th International Mining Congress and Exhibition of Turkey. 2001. p. 117-121.
29. Asad, M.W.A. and Dimitrakopoulos, R., *Implementing a parametric maximum flow algorithm for optimal open pit mine design under uncertain supply and demand.* J Oper Res Soc, 2013. **64**(2): p. 185-197.
30. Dagdelen, K. and Johnson, T.B. *Optimum open pit mine production scheduling by Lagrangian parameterization*. in 19th APCOM Symposium of the society of mining engineers (AIME). 1986.
31. Akaike, A. and Dagdelen, K. *A strategic production scheduling method for an open pit mine.* in Proceedings of the 28th Application of Computers and Operations Research in the Mineral Industry. 1999. p. 729 – 738.
32. Roman, R.J. *The role of time value of money in determining an open pit mining sequence and pit limits*. in 12th Symposium on the Application of Computers and Operation Research in the Mineral Industries (APCOM). 1974. p. 72 – 85.
33. Onur, A.H. and Dowd, P.A., *Open pit optimization-part 2: production scheduling and inclusion of roadways.* Trans. Inst. Min. Metall. (Sec. A: Min. Industry), 1993: p. 102, A105 – A113.

34. Dowd, P.A. and Onur, A.H. *Optimizing open pit design and sequencing*. in Proceedings of the 23rd International Symposium on the Application of Computers and Operations Research in The Mineral Industries. 1992. p. 411 – 422.
35. Tolwinski, B. and Underwood, R. *An algorithm to estimate the optimal evolution of an open pit mine*. in Proceedings of the 23rd International Symposium on the Application of Computers and Operations Research in The Mineral Industries. 1992. p. 399 – 409.
36. Tolwinski, B. *Scheduling production for open pit mines*. in Proceedings of APCOM'98. 1998. p. 19 – 23.
37. Tolwinski, B. and Golosinski, T.S. *Long term open pit scheduler*. in Proceedings of the International Symposium on Mine Planning and Equipment Selection. 1995. p. 256 – 270.
38. Erarslan, K. and Celebi, N., *A simulative model for optimum open pit design.* CIM Bull 2001. **94**: p. 59-68.
39. Holland, J.H., *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. 1975: University of Michigan Press.
40. Holland, J.H., *Outline for a Logical Theory of Adaptive Systems.* J. ACM, 1962. **9**(3): p. 297-314.
41. Denby, B. and Schofield, D., *Open-pit design and scheduling by use of genetic algorithms.* Transactions of the Institution of Mining and Metallurgy. (Section A: Mining Industry). 1994. **103**: p. A.21-A26.
42. Denby, B. and schofield, D. *Genetic algorithms for open pit scheduling-extension into 3-dimensions*. in MPES. 1996. Sao Paulo, Brazil p. 177–185.
43. Dorigo, M. and Stützle, T., *Ant Colony Optimization*. 2004: MIT Press.
44. Dorigo, M. and Blum, C., *Ant colony optimization theory: A survey.* Theoretical Computer Science, 2005. **344**(2–3): p. 243-278.
45. Shishvan, M.S. and Sattarvand, J., *Long term production planning of open pit mines by ant colony optimization.* European Journal of Operational Research, 2015. **240**(3): p. 825-836.
46. Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P., *Optimization by Simulated Annealing.* Science, 1983. **220**(4598): p. 671-680.
47. Černý, V., *Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm.* Journal of Optimization Theory and Applications, 1985. **45**(1): p. 41-51.
48. Thomas, G.S. *Optimisation and Scheduling of Open Pits via Genetic Algorithms and Simulated Annealing*. in First International Symposium on Mine simulation via the Internet 1996.
49. Dimitrakopoulos, R., *Conditional simulation algorithms for modelling orebody uncertainty in open pit optimisation.* International Journal of Surface Mining, Reclamation and Environment, 1998. **12**(4): p. 173-179.
50. Ravenscroft, P.J., *Risk analysis for mine schedulling by conditional simulation* Trans. Inst. Min. Metall. A, 1992. **101A,** : p. 104–108.

51. Dowd, P.A., *Risk assessment in reserve estimation and open pit planning* in *Risk assessment in extractive industries* 1994, Trans. Instn Min. Metall.(Sec. A: Min. Industry): University of Exceter p. A148-A154.
52. Dimitrakopoulos, R., Farrelly, C.T., and Godoy, M., *Moving forward from traditional optimization: grade uncertainty and risk effects in open-pit design.* Mining Technology, 2002. **111**(1): p. 82-88.
53. Dimitrakopoulos, R., Martinez, L., and Ramazan, S., *A maximum upside/minimum downside approach to the traditional optimization of open pit mine design.* Journal of Mining Science, 2007. **43(1)**: p. 73-82.
54. Dimitrakopoulos, R. and Ramazan, S., *Uncertainty based production scheduling in open pit mining.* . SME Trans., 2004. **316**: p. 106-112.
55. Birge , J.R. and Louveaux , F., *Introduction to Stochastic Programming*. Springer Series in Operations Research and Financial Engineering 2011: Springer Science & Business Media.
56. Lamghari, A. and Dimitrakopoulos, R., *A diversified Tabu search approach for the open-pit mine production scheduling problem with metal uncertainty.* European Journal of Operational Research, 2012. **222**(3): p. 642-652.
57. Meagher, C., Sabour, S.A., and Dimitrakopoulos, R. *Pushback design of open pit mines under geological and market uncertainties*. in Proceedings of the International Symposium on Orebody Modeling and Strategic Mine Planning: Old and New Dimensions in Changing World. 2009. p. 297–304.
58. Godoy, M. and Dimitrakopoulos, R., *Managing risk and waste mining in long-term production scheduling.* SME Trans., 2004. **316**: p. 43-50.
59. Yang, X.-S. and He, X., *Bat algorithm: literature review and applications.* Int. J. Bio-Inspired Comput., 2013. **5**(3): p. 141-149.
60. Yang, X.-S., *A New Metaheuristic Bat-Inspired Algorithm,* in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010).* 2010, Springer Berlin Heidelberg. p. 65-74.
61. Mendes, R., . *Population Topologies and Their Influence in Particle Swarm Performance*. e. PhD Thesis. 2004: Universidade do Minho, Portugal.
62. Eberhart, R. and Kennedy, J. *A new optimizer using particle swarm theory*. in Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on. 1995. p. 39 - 43.
63. Kennedy, J. and Eberhart, R. *Particle swarm optimization*. in Proceedings of IEEE International Conference on Neural Networks. 1995. p. 1942-1948
64. Kennedy, J.F., Kennedy, J., Eberhart, R.C., and Shi, Y., *Swarm Intelligence*. 2001: Morgan Kaufmann Publishers.
65. Heppner, F. and Grenander, U., *A stochastic nonlinear model for coordinated bird flocks*, in *The ubiquity of chaos*, Krasner, E., Editor. 1990, AAAS Publications. p. 233-238.
66. Reynolds, C.W. *Flocks, herds and schools: A distributed behavioral model*. in Proceedings of the 14th annual conference on Computer graphics and interactive techniques. 1987. ACM. p. 25-34.
67. Reeves, W.T., *Particle Systems—a Technique for Modeling a Class of Fuzzy Objects.* ACM Trans. Graph., 1983. **2**(2): p. 91-108.

68. Yuhui, S. and Eberhart, R. *A modified particle swarm optimizer*. in Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence. 1998. p. 69-73.
69. Kennedy, J. and Eberhart, R.C. *A discrete binary version of the particle swarm algorithm*. in IEEE International Conference on Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation. 1997. p. 4104-4108.
70. Rezaee Jordehi, A. and Jasni, J., *Parameter selection in particle swarm optimisation: a survey.* Journal of Experimental & Theoretical Artificial Intelligence, 2013: p. 1-16.
71. Maurice Clerc, *Particle Swarm Optimization*. 2006: Wiley-ISTE.
72. Poli, R., Kennedy, J., and Blackwell, T., *Particle swarm optimization.* Swarm Intelligence, 2007. **1**(1): p. 33-57.
73. Eberhart, R.C. and Shi, Y. *Comparing inertia weights and constriction factors in particle swarm optimization*. in Proceedings of the Congress on Evolutionary Computation. 2000. p. 84-88
74. Clerc, M. and Kennedy, J., *The particle swarm - explosion, stability, and convergence in a multidimensional complex space.* Evolutionary Computation, IEEE Transactions on, 2002. **6**(1): p. 58-73.
75. Kennedy, J. and Mendes, R. *Population structure and particle swarm performance*. in Proceedings of the 2002 Congress on Evolutionary Computation CEC '02. . 2002. p. 1671-1676.
76. Suganthan, P.N. *Particle swarm optimiser with neighbourhood operator*. in Proceedings of the 1999 Congress on Evolutionary Computation CEC 99. . 1999. p. 1958-1962.
77. Peram, T., Veeramachaneni, K., and Mohan, C.K. *Fitness-distance-ratio based particle swarm optimization*. in Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE. 2003. p. 174-181.
78. Janson, S. and Middendorf, M., *A hierarchical particle swarm optimizer and its adaptive variant.* Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 2005. **35**(6): p. 1272-1282.
79. Liang, J.J. and Suganthan, P.N. *Dynamic multi-swarm particle swarm optimizer*. in Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE. 2005. p. 124-129.
80. Mendes, R., Kennedy, J., and Neves, J., *The fully informed particle swarm: simpler, maybe better.* Evolutionary Computation, IEEE Transactions on, 2004. **8**(3): p. 204-210.
81. van den Bergh , F., *An analysis of particle swarm optimizers*. PhD Thesis. 2002: Department of Computer Science, University of Pretoria.
82. van den Bergh , F., *An Analysis of Particle Swarm Optimizers* Faculty of Natural and Agricultural Science , University of Pretoria PhD 2001.
83. Shuyuan, Y., Min, W., and Licheng, J. *A quantum particle swarm optimization*. in Evolutionary Computation, 2004. CEC2004. Congress on. 2004. p. 320-324 Vol.321.
84. Rezaee Jordehi, A. and Jasni, J., *Particle swarm optimisation for discrete optimisation problems: a review.* Artificial Intelligence Review, 2012: p. 1-16.
85. Pampara, G., Franken, N., and Engelbrecht, A.P. *Combining particle swarm optimisation with angle modulation to solve binary problems*. in Evolutionary Computation, 2005. The 2005 IEEE Congress on. 2005. p. 89-96 Vol.81.

86. Afshinmanesh, F., Marandi, A., and Rahimi-kian, A. *A Novel Binary Particle Swarm Optimization Method Using Artificial Immune System*. in Computer as a Tool, 2005. EUROCON 2005.The International Conference on. 2005. p. 217-220.
87. Laskari, E.C., Parsopoulos, K.E., and Vrahatis, M.N. *Particle swarm optimization for integer programming*. in Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on. 2002. p. 1582-1587.
88. Qin, J., Li, X., and Yin, Y., *An algorithmic framework of discrete particle swarm optimization.* Applied Soft Computing, 2012. **12**(3): p. 1125-1130.
89. Fukuyama, Y. and Yoshida, H. *A particle swarm optimization for reactive power and voltage control in electric power systems*. in Evolutionary Computation, 2001. Proceedings of the 2001 Congress on. 2001. p. 87-93 vol. 81.
90. Aziz, N.A.A., Mohemmed, A.W., Alias, M.Y., and Aziz, K.A., *Particle Swarm Optimization for Constrained and Multiobjective Problems: A Brief Review* in *International Conference on Management and Artificial Intelligence IPEDR vol.6 IACSIT Press, Bali, Indonesia* 2011.
91. Paquet, U. and Engelbrecht, A.P. *A new particle swarm optimiser for linearly constrained optimisation*. in Evolutionary Computation, 2003. CEC '03. The 2003 Congress on. 2003. p. 227-233 Vol.221.
92. S.Yilmaz, E.U.K., Y.Cengiz *Modified Bat Algorithm* ELEKTRONIKA IR ELEKTROTECHNIKA 2014. **Vol 20, No 2,**: p. 71-78.
93. Nakamura, R.Y.M., Pereira, L.A.M., Costa, K.A., Rodrigues, D., Papa, J.P., and Yang, X.S. *BBA: A Binary Bat Algorithm for Feature Selection*. in Graphics, Patterns and Images (SIBGRAPI), 2012 25th SIBGRAPI Conference on. 2012. p. 291-297.
94. Storn, R. and Price, K., *Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces.* Journal of Global Optimization, 1997. **11**(4): p. 341-359.
95. Das, S. and Suganthan, P.N., *Differential Evolution: A Survey of the State-of-the-Art.* Evolutionary Computation, IEEE Transactions on, 2011. **15**(1): p. 4-31.
96. Storn, R. and Price, K., *Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces.* Journal of Global Optimization, 1995.
97. Qin, A.K. and Suganthan, P.N. *Self-adaptive differential evolution algorithm for numerical optimization*. in Evolutionary Computation, 2005. The 2005 IEEE Congress on. 2005. p. 1785-1791 Vol. 1782.
98. Neri, F. and Tirronen, V., *Recent advances in differential evolution: a survey and experimental analysis.* Artificial Intelligence Review, 2010. **33**(1-2): p. 61-106.
99. Barbara, M.S., *A Tutorial on Constraint Programming*. 1995: Division of Articial Intelligence University of Leeds SCHOOL OF COMPUTER STUDIES, RESEARCH REPORT SERIES Report 95.14.
100. Khan, A. and Niemann-Delius, C., *Production Scheduling of Open Pit Mines Using Particle Swarm Optimization Algorithm.* Advances in Operations Research, 2014. **2014**: p. 9.