CIGI QUALITA MOSIM 2023
A GRASP algorithm for scheduling of open-pit phosphate mine extraction

SOUFIANE AALLAOUI[1], AHLAM AZZAMOURI[1], LIBO REN[2,3] NIKOLAY TCHERNEV[1,2]

[1] EMINES, School of Industrial Management, Mohammed VI Polytechnic University
43150 Benguerir, Morocco
{soufiane.aallaoui, ahlam.azzamouri}@emines.um6p.ma

[2] LIMOS (UMR CNRS 6158), Clermont Auvergne University
1 rue de la Chebarde, 63177 Aubière Cedex, France
nikolay.tchernev@uca.fr

[3] CLeRMa EA3849, Clermont Auvergne University,
11 boulevard Charles de Gaulle, 63000 Clermont-Ferrand, France
libo.ren@uca.fr

***Résumé* – Cet article traite un problème d'ordonnancement des opérations et affectation des machines dans une mine à ciel ouvert pour atteindre les objectifs de production. Tout cela est contraint par : séquencement des opérations, accessibilité aux couches, compatibilité, disponibité et mouvements des machines. Le problème est modélisé comme un problème du Job Shop Flexible avec des time-lags génériques. Cet article introduit une modélisation du problème basée sur le graphe disjonctif et une méthode approchée Greedy Randomized Adaptive Search Procedure (GRASP) pour la résolution. Une comparaison entre cette dernière et un programme linéaire en nombre entier montre que cette méthode fournit des solutions de meilleures qualités.**

***Abstract* – This paper presents a solution for scheduling open-pit mining operations and machinery affectation to meet production goals. The problem is complex due to various constraints such as operation sequencing, machinery compatibility, machinery availability, and accessibility to layers. The solution uses a disjunctive graph and a GRASP algorithm to solve the problem modeled as a flexible job shop problem with generic time-lags between operations. The algorithm performance has been benchmarked against a Mixed Integer Linear Program and provides high-quality solutions via numerical experiments.**

***Mots clés* – Ordonnancement, Job Shop Flexible, Time-lags, Métaheuristique, GRASP.**
***Keywords* – Scheduling, Flexible Job Shop, Time-lags, Metaheuristic, GRASP.**

1 INTRODUCTION

This article analyses the problems posed by the operational decisions related to ore extraction in an open-pit mine. Motivated by a scheduling problem in OCP Group, a phosphate ore extraction and fertilizer production world leader, this research deals with a short-term scheduling problem. It proposes an original modeling approach based on a flexible job shop and minimum time-lags. The Greedy Randomized Adaptive Search Procedure (GRASP) heuristic approach is used to solve the problem.

We start by providing the characterizing features of the studied problem (§2) before presenting the suitable formalization for a theoretical scheduling problem and the related literature review (§3). Next, the framework based on a disjunctive graph model and the GRASP algorithm is proposed (§4). Section §5 deals with the computational evaluation of the framework, including industrial-based benchmarks, before concluding and promising directions for future research (§6).

2 PROBLEM DEFINITION

The integrated supply chain of OCP Group links the different processes of ore extraction, ore blending, phosphoric acid, fertilizer production, and export. It comprises three independent axes: the north, center, and south. This study focuses on the Ben Guerir mine in the center axis, where the phosphate deposit is composed of the accumulation of layers of different geological and chemical compositions natures. The sedimentary nature of the mine tolerates the extraction of a layer only if the upper level is already removed. As shown in Figure 1, from a transverse view, the deposit contains several panels, each panel is composed of a group of trenches, and each trench is made up of parcels (or blocks): rectangles of 4000m^2 (40m * 100m) that contain alternate phosphate and waste blocks. Particular layers might not exist in some parcels due to geological factors.

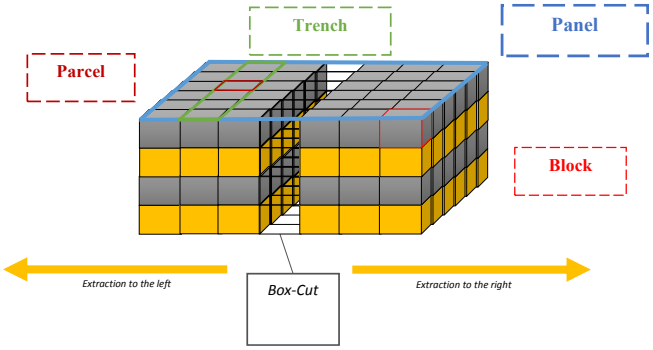

Figure 1. Open-pit block model

The extraction process results from the iteration of a sequence of elementary operations $O_{j,o}$ performed on blocks by specialized or multipurpose machine $M_{j,o,m}$ from a set of machines $M_{j,o} = \{M_{j,o,1}, M_{j,o,2}, \ldots, M_{j,o,M}\}$. These resources are limited and urge the planner to find the best spatio-temporal machinery affectation. The decisions to be taken by the planner relate to [Azzamouri et al., 2018]: (*i*) The choice of parcels to be extracted knowing that operations are in progress in several parcels; (*ii*) The allocation of the machines available to perform these operations. The extraction process starts with recovering the ore and moving the waste from the trench in the middle of a given panel, which creates a void called a "Box-cut" (Figure 1). Then, the extraction continues simultaneously on the adjacent trenches by pushing the waste blocks into the void and recovering the ore from transporting it to the plant. For example, extracting a phosphate block covered by a single block of waste (e.g., a parcel that contains two blocks) means that the block will be extracted based on seven elementary operations. These operations are grouped into two stages, using the available machines as follows: (*i*) The first stage is dedicated to waste block recovery that consists of: $\{$*Site preparation for drilling*: $O_{j,1}\}$, $\{$*drilling*: $O_{j,2}\}$, $\{$*blasting*: $O_{j,3}\}$, $\{$*site preparation for stripping*: $O_{j,4}\}$, and $\{$*stripping* : $O_{j,5}\}$; (*ii*) The second stage dedicated to phosphate block recovery involving $\{$*stacking*: $O_{j,6}\}$, and $\{$*loading*: $O_{j,7}\}$. In this paper, it is assumed that all the panels have a "Box-Cut," and the block extraction order has been made. The extraction process is constraint by several factors:

- Precedence: That comes from geological nature of the mine dictates the necessity of the upper blocks' removal to access the lower ones, the staircase extraction method that secures the process and the relations between elementary operations;
- Resource availability: As a consequence of the multi-purpose nature of the machinery, a machine may be busy executing an operation;
- Machines motions: Each machine should return to the waiting area after processing an operation to free up space for another device. Depending on operation locations, it can also travel from one parcel to another.

Indeed, the problem described above corresponds to the Hybrid Flow Shop Scheduling Problem (HFSP) [Pinedo, 2016]. However, in classical HFSP, each machine belonging to a given stage is used only at this stage. However, in the industrial problem considered, many machines are used at different stages during the parcel treatment. Therefore, the problem is modeled as a Flexible Job Shop Scheduling Problem (FJSP) in which time constraints restrict the minimum time distance between two successive operations belonging to the same job or two different jobs.

3 LITERATURE REVIEW: INSIGHTS & CONTRIBUTIONS

The production scheduling and short-term planning issues for open-pit mines have not received as much attention in the academic literature as the medium- and long-term perspectives have [Blom et al., 2019]. Yet, over the past few decades, a considerable amount of study has been done on scheduling [Pinedo, 2016]. An open-pit mine production schedule related to block sequencing is heavily studied in the literature, but the issue of determining the resources that enable such a schedule didn't get as much attention. In this context, recent research in mine scheduling has focused on developing methods that can handle the complexity and uncertainty of real-world mining operations. Thus, [Kozan et Liu, 2016] proposed a mixed integer program to tackle a complex operational problem of multi-resource multi-stage mine production timetabling. The proposed approach maximizes the utilization of mining equipment and mining productivity. The authors demonstrate its effectiveness through a numerical case study and a practical implementation based on real-life mining data. However, [Lamghari et al, 2016] presented a progressive hedging approach for mine scheduling. The authors proposed a new method that considers the mining operations' uncertainty and reserve quality. They found that this approach can improve the efficiency of the mining operations and reduce the environmental impact. And for fleet management, [Mohtasham et al, 2021] present a mixed integer linear program to allocate trucks and shovels in an open-pit mine. They formulated this model to maximize fleet performance by maximizing production and minimizing the fuel consumption of the trucks. A copper mine case study has shown this model's effectiveness. These are examples of methods among many in the literature that addresses this problem. Furthermore, [Newman et al., 2010] present a review of operations research's application to mine planning. In many equipment routing and selection models, optimization is used, i.e., integer programming, to determine mine fleet size and allocation. In addition, many publications are about loading and haulage operations: shovel and truck allocation [Moradi Afrapoli et Askari-Nasab, 2019]. Simulation is also used to address problems like this; for example, [Azzamouri et al., 2018] used discrete event simulation to generate short and medium-term planning and machine allocation for an open-pit mine.

The problem addressed in this study is classified as a difficult FJSP because of the generic time-lags constraints that must be considered between operations from the same and different jobs [Lacomme et al., 2011]. This complexity comes from the fact that the FJSP is an extension of the hard problem JSP, for which the literature suggests both exact and approximation methods [Blażewicz et al., 1996]. [Roy et Sussmann, 1964] established the disjunctive graph model, which is frequently the foundation of successful solutions approaches, which effectively models the relationship between operations. Additional constraints can be considered to represent the reality of the scheduling problem, like the paper of [Lacomme et al., 2011] which addresses this problem with generic time-lags that models general timing relations between jobs. Also, [Caumond et al., 2008] propose a solution to the same problem using a disjunctive graph and a memetic algorithm. The FJSP arises when one operation may be executed on many machines. Thus, a review of the existing solution methods is presented by [Chaudhry et Khan, 2016] and [Xie et al., 2019], and classify them into exact algorithms, heuristics, and metaheuristics. Improved versions of the GRASP metaheuristic are used to solve the problem, like [Kemmoé-Tchomté et al., 2017] who propose GRASP with a multi-level evolutionary local search (mELS) that provides

valuable results in terms of computation times and quality.

4 FLEXIBLE JOB SHOP PROBLEM WITH GENERIC TIME LAGS (FJSPGTL)

4.1 FJSPGTL in the literature

The FJSP problem, first introduced by [Brucker et Schlie, 1990], has been widely studied in the literature and has many industrial applications. The FJSP with Generic Time-Lags (FJSPGTL) can be found in many applications since they result from technological or organizational constraints. Indeed, time-lags can be used as restrictions enforcing minimal or maximal delays between two operations to model general timing relations between jobs, like start-start relations between jobs [Lacomme et al., 2011]. A FJSP scheduling problem is an extended form of a classical job shop scheduling problem (JSP), which is NP-hard [Garey et al., 1976]. The particularity of FJSP is that each operation related to a given job is processed by a machine selected in a set of compatible and available ones. The complexity of the FJSP suggests the adoption of heuristic methods producing reasonably good schedules in an acceptable execution time instead of seeking an exact solution.

4.2 FJSPGTL settings

Without loss of generality, each parcel can be considered as a job *j*. Therefore, the scheduling of open-pit phosphate mine extraction using the FJSPGTL can be formulated as follows: a set of *J* jobs that must be processed on a set of *M* machines. Each one of these jobs *j* involves a set of operations $O_j = \{O_{j,1}, O_{j,2}, \dots, O_{j,k_j}\}$ where k_j refers to the number of operations of the job *j*. An operation O_{j,k_j} must be processed in a pre-determined order, and no pre-emption is allowed. Each operation is processed by one and only one machine from the set $M_{j,o} = \{M_{j,o,1}, M_{j,o,2}, \dots, M_{j,o,M}\}$ of the compatible machines. The processing time $p_{j,o}$ that depends on the machine $M_{j,o,m}$ allocated to $O_{j,o}$ and the geological nature of the block in question.

A time-lag can be defined between the finish time of a given operation $O_{j,o}$ (denoted by $ft_{O_{j,o}}$) and the start time of another operation $O_{j',o'}$ (denoted by $st_{O_{j',o'}}$) using the following equation:

$$TL^{min}{}_{O_{j,o},O_{j',o'}} \leq st_{O_{j',o'}} - ft_{O_{j,o}} \leq TL^{max}{}_{O_{j,o},O_{j',o'}} \qquad (1)$$

With $TL^{max}{}_{O_{j,o},O_{j',o'}} \geq TL^{min}{}_{O_{j,o},O_{j',o'}}$

In this formula $TL^{min}{}_{O_{j,o},O_{j',o'}}$ represents the minimal time-lag and $TL^{max}{}_{O_{j,o},O_{j',o'}}$ is the maximal time-lag. The first part of this formula $TL^{min}{}_{O_{j,o},O_{j',o'}} \leq st_{O_{j',o'}} - ft_{O_{j,o}}$ means that $O_{j',o'}$ cannot start before at least $TL^{min}{}_{O_{j,o},O_{j',o'}}$ units after the end of $O_{j,o}$. The second part of the formula $st_{O_{j',o'}} - ft_{O_{j,o}} \leq TL^{max}{}_{O_{j,o},O_{j',o'}}$ means that $O_{j',o'}$ cannot be started later than $TL^{max}{}_{O_{j,o},O_{j',o'}}$ units after the end of $O_{j,o}$. When there is a classical precedence constraint between two operations, the value of minimum time-lag is set to 0, and the value of maximum time-lag is set to ∞. When there is no constraint between two operations, both the minimum and the maximum time-lags are set to ∞.

According to the $\alpha|\beta|\gamma$ notation introduced by [Graham et al., 1976] the problem can be represented by $FJc|l_{O_{j,o},O_{j',o'}}|C_{max}$.

Time-lags between the start and completion times of different activities have to be observed in numerous scheduling problems, including the Resource Constrained Project Scheduling Problem (RCPSP), where resource consumption is addressed [Brucker et al., 1999]. They result from technological or organizational constraints in practice. Besides minimum time-lags, maximum time-lags might be given, which occurs in the chemical and food industries.

5 A GRASP ALGORITHM FOR MINING SCHEDULING

The framework is based on a GRASP algorithm (for job sequence generation on machines) coupled with a powerful local search procedure. The problem is modeled as a non-oriented disjunctive graph. Since a job sequence on machines is generated, obtaining an oriented disjunctive graph is possible. A Bellman like longest path algorithm permits to compute the earliest completion time of the last operation qualified as the makespan C_{max} (Figure 2).

5.1 Graph modeling

This section aims to propose a graph modeling of open-pit phosphate mine extraction using the graph model proposed by [Dauzère-Pérès et Paulli, 1997] for the FJSP, which is based on the [Roy et Sussmann, 1964] disjunctive graph for the job scheduling problems. The time-lags constraints are also included.

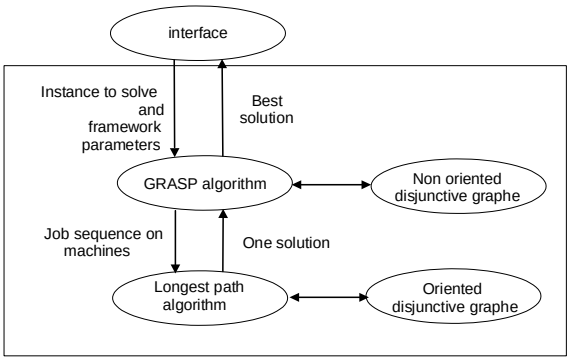

Figure 2. Description of the framework for FJSPGTL

5.1.1 Assumptions

As stated in the introduction, the extraction process is based on seven principal elementary operations. The first five operations allow the removal of the waste block, and the last two ones are dedicated to ore recovery. In addition, we consider that:

- The job j corresponds to the accumulation of parcels containing the set of layers (waste and phosphate layers);
- The existence of a waiting machine area: a location where machines are located during their idle time and from where they move to the operations to be processed;
- The time-lag is the travel time needed by a machine to return to the waiting area after processing an operation liberating the working area for further machine which will perform the next operation.
- The switching time: which corresponds to the machine $M_{j,o,m}$ ($M_{j,o,m} = M_{j',o',m}$) travel from the operation $O_{j,o}$ to the operation $O_{j',o'}$ locations;
- Classical shop scheduling assumptions;
- The objective function is to minimize the ***makespan*** C_{max} (the earliest completion time of the last operation).

5.1.2 Non-oriented disjunctive graph

The FJSPGTL instances can be represented by a disjunctive graph $G = (V, A, E)$, where V is a set of nodes, A is the set of arcs that represent the conjunctions (oriented) and E is the set of edges that represent disjunctive (non-oriented) arcs. The sets V, A and E are formed as follows: A node is created in the set V for each operation $O_{j,o}$ in the problem, plus a node O (named the source) that represents a fictitious operation performed before all others, and the node * (named the sink) is a fictitious operation performed after all others.

For all successive operations of the same job, an arc is created in the set A; this arc length is $p_{j,o} + TL^{min}{}_{(j,o),(j,o+1),m}$, where $p_{j,o}$ represents the processing time of the operation $O_{j,o}$. These arcs represent the precedence and minimum time-lags constraints between operations of the same job. Minimum time-lags constraints between operations $O_{j,o}$ and $O_{j',o'}$ belonging to different jobs, are modeled by extra arcs and it is weighted with $p_{j,o} + TL^{min}{}_{(j,o),(j',o'),m}$. For every two operations that can be processed by the same machine, an edge is created in the set E, i.e. this edge represents the disjunction between these operations which is a part of the sub-set of E. This sub-set contains edges linking all the operations performed by the same machine. After orientation, the length of these arcs is equal to the processing time plus the switching time modeled as minimum time-lag $p_{j,o} + TL^{min}{}_{(j,o),(j',o'),m}$.

Maximal time-lags constraints are represented by negative arc cost in the disjunctive graph from one operation to the previous one. The negative length of the arc is equal to the duration of the previous operation plus the maximal time-lag value $-(p_{j,o} + TL^{max}{}_{(j,o),(j',o'),m})$.

Without loss of generality and for more clarity, let us consider a simple example of FJSPGTL representing the problem studied which is composed of 6 jobs which have 7 operations per each. The specific minimal time-lags between operations plus the machine processing time are described in Table 2. And the maximal time-lags are all set to ∞. The Table 1 gives the set of compatible machines (ranging from M_1 to M_7) and the processing times $p_{j,o}$ for all operations $O_{j,o}$ for $1 \leq j \leq 6$ and $1 \leq o \leq 7$.

Table 1. Example of a FJSP. Each couple $M_{m(p_{j,o})}$ refers to the possible machine assignment for an operation and the corresponding processing time

Operation Plan	Compatible machines $M_{m(p_{j,o})}$
$O_{j,1}$	$M_{1(26.7)}, M_{3(88.9)}$
$O_{j,2}$	$M_{5(66.7)}$
$O_{j,3}$	$M_{7(20)}$
$O_{j,4}$	$M_{1(26.7)}, M_{3(88.9)}$
$O_{j,5}$	$M_{2(21.6)}, M_{4(15.9)}$
$O_{j,6}$	$M_{1(13.3)}$
$O_{j,7}$	$M_{6(8.5)}$

The minimal time-lags between operations jobs are as follows:

- For all the jobs, a minimal time-lag between each consecutive operations is $TL^{min}{}_{(j,o),(j',o'),m}$. This value depends on the machine m assigned to the origin operation $O_{j,o}$. Therefore the conjunction arc is valuated as follows: $\tau_{(j,1),(j,2),m} = p_{(j,1)} + TL^{min}{}_{(j,1),(j,2),m}$.
  For example, the minimal time-lag from the end of $O_{j,1}$ to $O_{j,2}$ is $TL^{min}{}_{(j,1),(j,2),m}$ the value depends on the machine m assigned to $O_{j,1}$:
  - M_1 is the assigned machine: $TL^{min}{}_{(j,1),(j,2),M_1} = 0.22$; $\tau_{(j,1),(j,2),M_1} = 26.7 + 0.22$
  - M_3 is the assigned machine: $TL^{min}{}_{(j,1),(j,2),M_3} = 0.20$; $\tau_{(j,1),(j,2),M_3} = 88.9 + 0.20$
- For the job $j \in \{1,2,4,5\}$, a minimal time-lag from the end of $O_{j,5}$ to $O_{j+1,1}$ is $TL^{min}{}_{(j,5),(j+1,1),m}$ and the corresponding arc is valued $\tau_{(j,5),(j+1,1),m} = p_{(j,5)} + TL^{min}{}_{(j,5),(j+1,1),m}$
- For the job $j \in \{1,2,4,5\}$, a minimal time-lag from the end of $O_{j,7}$ to $O_{j+1,6}$ is $TL^{min}{}_{(j,7),(j+1,6),m}$ and the corresponding arc is valued $\tau_{(j,7),(j+1,6),m} = p_{(j,7)} + TL^{min}{}_{(j,7),(j+1,6),m}$
- For the job $j \in \{1,4\}$, a minimal time-lag from $O_{j+2,5}$ to $O_{j,6}$ is $TL^{min}{}_{(j+2,5),(j,6),m}$ and the corresponding arc is valued $\tau_{(j+2,5),(j,6),m} = p_{(j+2,5)} + TL^{min}{}_{(j+2,5),(j,6),m}$.

Table 2 lists the Roman numerals and the corresponding values of minimal time-lags that depend on the machines.

Table 2. The weight values of the arcs between operations

$\tau_{(j,1),(j,2),m}$	$\{M_1: 26.92, M_3: 89.1\}$
$\tau_{(j,2),(j,3),m}$	$\{M_5: 66.98\}$
$\tau_{(j,3),(j,4),m}$	$\{M_7: 20.21\}$
$\tau_{(j,4),(j,5),m}$	$\{M_1: 26.92, M_3: 89.1\}$
$\tau_{(j,5),(j,6),m}$	$\{M_2: 21.82, M_4: 18.1\}$
$\tau_{(j,6),(j,7),m}$	$\{M_1: 13.51\}$
$\tau_{(j,5),(j+1,1),m}$	$\{M_2: 21.82, M_4: 18.1\}$
$\tau_{(j,7),(j+1,6),m}$	$\{M_6: 8.71\}$
$\tau_{(j+2,5),(j,6),m}$	$\{M_2: 21.83, M_4: 20.1\}$

A part of the non-oriented disjunctive graph illustrating this example is given in Figure 3. In this graph, an arc (in full line) between two successive operations ($O_{j,o}$, $O_{j,o+1}$) represents the precedence (routing) constraint of the Job j. It is not weighted by any distance between these operations as no machines are assigned yet. Each pair of disjunctive arcs is represented with a dotted edge and represents the constraint between two operations sharing the same possible machine resource.

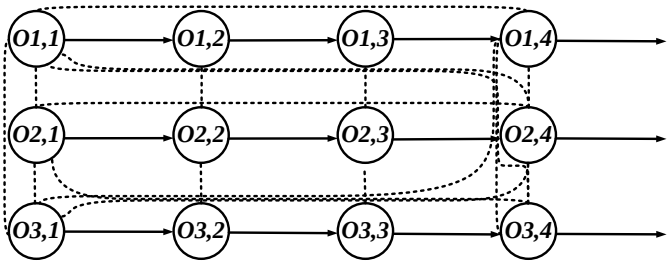

Figure 3. Illustration of a part of a problem with 6 jobs and 7 machines represented with disjunctive graph

A solution is an oriented conjunctive-disjunctive graph that includes arcs only. As the FJSP is an assignment and a scheduling problem, first, a machine assignment is done as shown in Figure 4. Such an assignment is represented by a

vector *MA* (Machine Assignment). Each element of *MA* is the number of the machine assigned to a given operation. Therefore, a possible representation of the assignment shown in Figure 4 would be in this way: {1 5 7 1 4 1 6 1 5 7 1 4 1 6 1 5 7 1 4 1 6 1 5 7 1 4 1 6 1 5 7 1 4 1 6 1 5 7 1 4 1 6}. This vector means that the first operation of Job 1 is processed on Machine M_1; the second operation of Job 1 is processed by machine M_5; the third operation of Job 1 is proceeded by machine M_7, and so on for all the jobs.

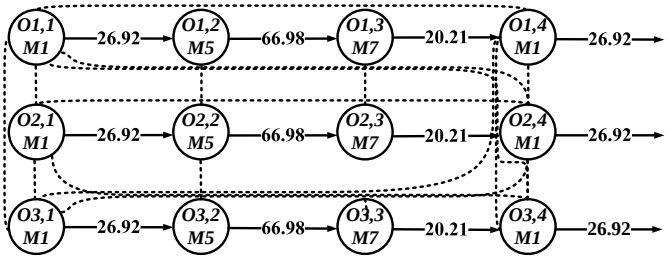

Figure 4. Illustration of a part of the 6 jobs and 7 machines problem with a disjunctive graph with the assignment of only one machine to each operation

Figure 4 shows a part of the new non-oriented disjunctive graph after machine assignments. The relevant edges to this assignment are still present, and the disjunctive edges that are not relevant have been removed since they are not involved in the schedule. Thus, $O_{1,1}$ and $O_{1,4}$ should be scheduled on the same machine M_1, so there is still an edge linking these operations.

When no time-lags are specified (for example between one operation and the dummy operation *O* of the graph), it is possible to assume, without loss of generality, to have null minimal time-lags and infinite maximal time-lags. The negative arcs representing infinite maximal time-lags are ignored in graph representation in the remainder of this article, since there is no interest in considering infinite maximal time-lags. The arc between successive operations of one job includes the processing time and minimum time-lags constraints as illustrates the Figure 4. The duration of this arc is equal to the processing time plus the minimal time-lags duration. Minimal time-lags are also included in disjunctive arcs: these arcs are weighted with the processing time of the operation at the beginning of the arc plus the minimum time-lag.

Finally, the acyclic conjunctive-disjunctive graph of a solution is given in Figure 5. In this graph, Arcs in bold constituted the critical path, and underlined numbers represent the earliest starting times of operations; all pairs of disjunctive edges are reduced to one arc and represent the sequence of operations processed on the same machine.

These disjunctive edges (arcs between operations of the same or different jobs which use the same machines) define the operations sequence on machines. The graph of Figure 5 is a solution in which:

- on machine M_1 the sequence is: $O_{1,1}$, $O_{4,1}$; $O_{1,4}$, $O_{2,1}$; $O_{4,4}$, $O_{5,1}$; $O_{2,4}$, $O_{3,1}$; $O_{5,4}$, $O_{6,1}$; $O_{3,4}$, $O_{1,6}$; $O_{6,4}$, $O_{2,6}$; $O_{4,6}$, $O_{5,6}$; $O_{3,6}$, $O_{6,6}$;
- no operations scheduled on machine M_2;
- no operations scheduled on machine M_3;
- on machine M_4 the sequence is: $O_{1,5}$, $O_{4,5}$; $O_{2,5}$, $O_{5,5}$; $O_{3,5}$, $O_{6,5}$;
- on machine M_5 the sequence is: $O_{1,2}$, $O_{4,2}$; $O_{2,2}$, $O_{5,2}$; $O_{3,2}$, $O_{6,2}$;
- on machine M_6 the sequence is: $O_{1,7}$, $O_{4,7}$; $O_{2,7}$, $O_{5,7}$; $O_{3,7}$, $O_{6,7}$;
- on machine M_7 the sequence is: $O_{1,3}$, $O_{4,3}$; $O_{2,3}$, $O_{5,3}$; $O_{3,3}$, $O_{6,3}$.

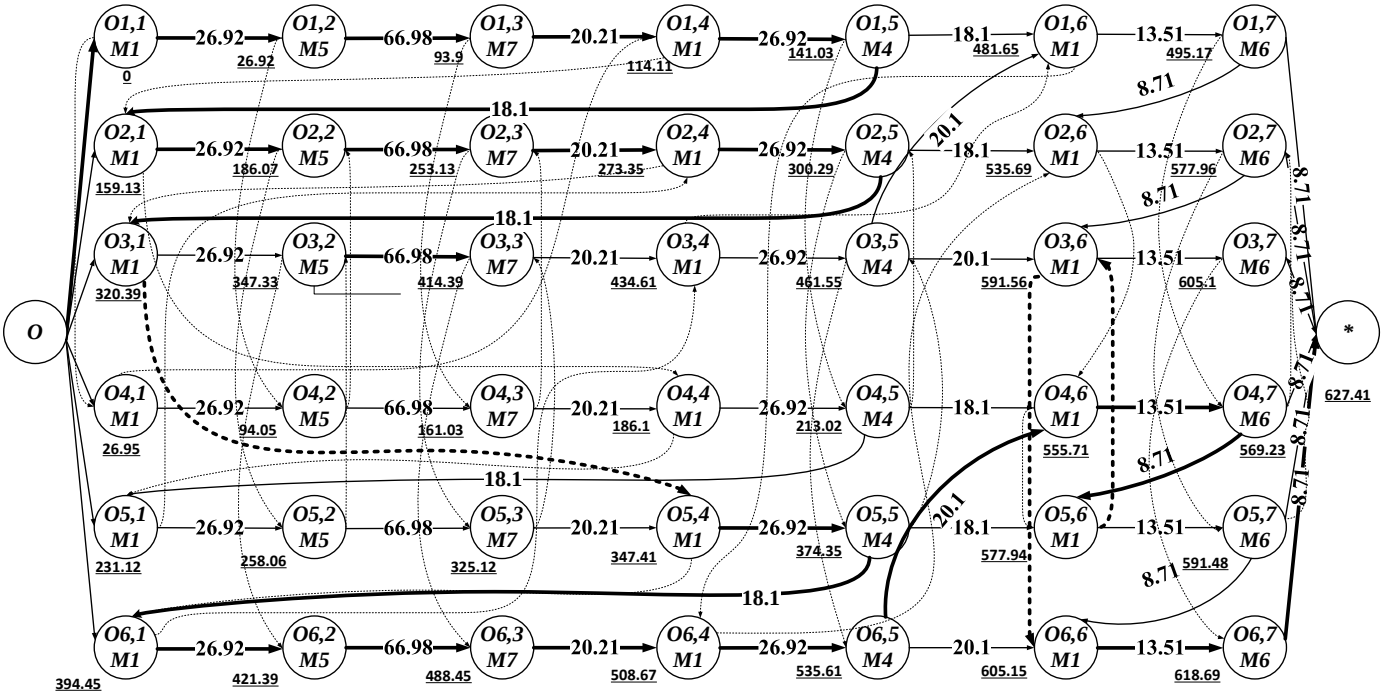

Figure 5. A schedule of the FJSP. Bold-face arcs show a critical path whose length, i.e.: the makespan, is 627.41

Since each operation belongs to one job only, a common representation of a solution consists in giving the job sequence on machines. With such notation, the previous solution is noted: Machine 1: job 1, job 4, job 1…; Machine 4: job 1, job 4…, etc. However, [Bierwirth, 1995] introduces an alternative representation as a sequence of job numbers. Based on his proposal, the solution of Figure 5 is encoded to: {1 4 1 1 1 4 1 2 4 4 4 2 5 2 2 5 2 3 5 5 5 6 3 3 6 3 3 1 1 6 6 6 2 2 4 4 5 3 5 3 6 6}. A representation that is known by: sequence with repetition. Such sequence is represented by a vector *OS* (Operation

Selection), read from left to right. In this vector, the first "1" corresponds to the first operation of Job 1; the second value "4," refers to the first operation of Job 4; the third value is the second operation of Job 1, followed by the third operation of this Job 1, and so on. A greedy algorithm or a metaheuristic can manage this sequence effectively because it makes it possible to create an acyclic-oriented disjunctive graph. Please note that the same oriented disjunctive graph can be represented by several such sequences. In the Figure 6, a part of these data structures are represented.

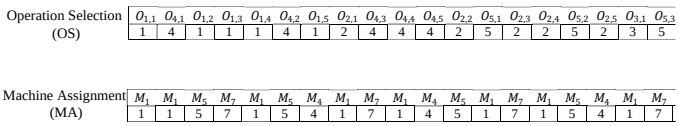

Figure 6. Coding of a solution

After presenting the disjunctive graph and solutions coding, the metaheuristic explained in the next section allows solution space exploration.

5.2 Greedy Randomized Adaptive Search Procedure (GRASP)

The GRASP metaheuristic explores the solution space. This metaheuristic was proposed by [Feo et Resende, 1995], and it's a multi-start metaheuristic. It consists of the generation of a solution using a randomized construction heuristic; then, this solution is improved using a local search. A classical GRASP is proposed (Algorithm 1). *f(s)* refers to the objective function optimized in the process as the makespan C_{max}.

Algorithm 1: GRASP.

```
Output
S*              :  Best found solution;
Variables
S               :  A temporary solution;
BEGIN
1.   WHILE stop criteria not met DO
2.     S := Construction_Phase;
3.     S := Local_Search_Phase;
4.     S*:= Best solution found;
5.   End WHILE
6.   Return S*;
END
```

5.2.1 Construction phase

This phase aims to build an initial solution by assigning and scheduling one operation at a time, using a greedy randomized heuristic. As described by [Binato et al., 2000] for the JSP, at each iteration, a Restricted Candidate List (*RCL*) is built from the set containing the operations already scheduled. An operation is selected from this set and added to the *RCL* under the criteria of "which of the already scheduled operations represent the smallest increase in the makespan". Let's consider $O_{j,o}$ the o^{th} operation of the job j and $O_{j,o}$ is processed on the machine $M_{j,o,m}$ from the set $M_{j,o}$ of the available machines for $O_{j,o}$. The set of scheduled operations is noted O^s, while candidate operations to be scheduled is noted O^c. The candidate list O^c of operations to be processed is built at each iteration from a list L of operations that are not scheduled yet. The next operation to be scheduled is chosen randomly from *RCL*, all the operations have equal probabilities to be chosen. The algorithm used to generate an initial solution is almost similar to the construction phase algorithm proposed by [Kemmoé-Tchomté et al., 2017]. Still, the difference is that we consider the time-lags and switching time constraints.

5.2.2 Evaluation phase: Longest path

The evaluation algorithm computes the earliest start time of each operation, then the end date of the last operation in the schedule. It's used in the local search phase to evaluate a possible solution after the initial solution perturbation. Algorithm 2 presents the evaluation of the longest path for the problem described above.

Algorithm 2: Evaluation.

```
Input
Data                :  Problem information;
B                   :  Bierwirth vector;
BM                  :  Affectation vector;
Output
Cmax                :  Makespan;
PERE                :  Operations predecessor
                       that conditions their
                       start date;
OperationsMachine   :  The operations machine
                       affectation;
Variables
EarliestStart       :  Operations earliest start
                       date;
PERE                :  Operations predecessor
                       that conditions their
                       start date;
OperationsMachine   :  The operations machine
                       affectation;
MachineOperation    :  The latest operation on a
                       machine;
(j,o),(k,l)         :  job number, operation
                       number
etapeJob            :  Operation to be scheduled
                       for each job
machine,m           :  The machine assigned to
                       the current operation
d,dPD               :  End date of conjonctif
                       and disjonctif
                       predecessor respectively;
pere, Dpere         :  Conjonctif and
                       disjonctive predecessor;
BEGIN
1.   Initialization of all variables;
2.   FOR i:=0 TO the total number of operation
     DO
3.      d = 0, dPD = 0;
4.      pere=(-1,-1), Dpere=(-1,-1);
5.      j = B[i], o = etapeJob[j];
6.      machine = BM[i];
7.      For all (k,l)conjonctifs
        Predecessors of (j,o) DO
8.         m = OperationsMachine[j,o];
9.         IF d < (EarliestStart[k,l]+
           ProcessingTime[k,l,m] +
           TimeLags[k,l,j,o,m]) DO
10.           d = (EarliestStart[k,l]+
              ProcessingTime[k,l,m] +
              TimeLags[k,l,j,o,m];
11.           pere = (k,l);
12.        END IF
13.     END FOR
14.     IF (j,o) got a disjunctive
        predecessor DO
15.        (k,l)=MachineOperation[machine];
16.        Dpere = (k,l);
17.        dPD =(EarliestStart[k,l]+
           ProcessingTime[k,l,m] +
           SwitchingTime[k,l,j,o,m];
18.     END IF
19.     IF dPD > d DO
20.        pere = Dpere;
21.        d = dPD;
22.     END IF
```

```
23.      EarliestStart[j,o] = d;
24.      PERE[j,o] = pere;
25.      MachineOperation[machine]=(j,o);
26.   END FOR
27.   Compute Cmax; //The end date of the last
      operation.
28.   RETURN Cmax,PERE,OperationsMachine,
      EarliestStart;
END
```

5.2.3 Local search phase

After generating the initial solution, a local search is applied to improve the quality of the solutions. It's based on the critical path, where we exchange the order of two consecutive operations processed by the same machine and/or change the machine allocation of an operation in the critical path. The algorithm 4 (Local_Search_Dis) explores the critical path from the sink node * to the source node O. The possible permutations that could improve the solution are saved, if the evaluation of the new solution (the permutation is applied) indicates an improvement of the solution, this new solution is the best solution and so on. The evaluation of FJSP relies on a longest path computation (Algorithm 2).

Algorithm 3: Local_Search.

```
Input/Output
Data              :   Problem information;
S                 :   Initial solution;
Variables
(j,o)             :   operation on the critical
                      path;
S1,S2             :   temporary solutions;
BEGIN
1.   Local_Search_Dis(S,Data);
2.   (j,o)= The last operation of
     the critical path;
3.   WHILE (j,o)!=(0,0) DO
4.      IF (Number of available machines for
        (j,o)>1) DO
5.         S1 = S;
6.         FOR EACH machine different
           Than the current one DO
7.            Change the machine
              affected to op in S1;
8.            Local_Search_Dis(S1,Data);
9.            IF f(S1)<f(S) DO
10.              S = S1;
11.              (j,o) = The last operation of
                 the critical path of S;
12.           ELSE
13.              (j,o)= The predecessor of
                 (j,o)on the critical path of S;
14.           END IF
15.        END FOR
16.     ELSE
17.        (j,o) = The predecessor of (j,o)on
           the critical path of S;
18.     END IF
19.  END WHILE
   END
```

For this described GRASP metaheuristic, 500 is the iteration number chosen to obtain a solution. The choice of this number comes from obtaining good solutions in terms of quality and computation time, starting from 300 or 500 iterations, depending on the instance.

Algorithm 4: Local_Search_Dis.

```
Input/Output
S                 : initial solution;
Data              : Problem information;
Variables
(j,o),(k,l)       : operation and predecessor on
                    critical path;
S_temp            : temporary solution;
m                 : machine assigned to the
                    operation (j,o) in the critical
                    path;
BEGIN
1.   Cmax = f(S);
2.   (j,o)= The last operation of the critical
     path;
3.   S_temp = S;
4.   WHILE (j,o)!=(0,0) DO
5.      m = Machine assigned to (j,o);
6.      IF (Number of critical path operations
        processed by m >1) Do
7.         (k,l)= disjunctive predecessor of
           (j,o);
8.         Apply permutation of (j,o) and (k,l)
           in S_temp;
9.         Evaluation of S_temp;
10.        IF f(S_temp)<f(S) DO
11.           S = S_temp;
12.           (j,o)= The last operation of the
              critical path;
13.        END IF
14.     ELSE
15.        (j,o) = predecessor of (j,o) on the
           critical path;
16.     END IF
17.  END WHILE
END
```

6 RESULTS AND ANALYSIS

In this section, we present a comparison between the results obtained by the Mixed Integer Linear Programming (MILP) [Aallaoui et al., 2022] and the GRASP metaheuristic described in the previous section based in the instances described in Table 3. The number of iterations chosen for the GRASP is 500.

Table 3. Generated instances description

Instance (Ins)	1	2	3	4	5	6
Jobs	6	12	12	18	24	24
Operations / job	7	7	14	7	7	7
Machines	7	7	10	7	7	10

The instances used to test the models are instances generated from the real problem. Table 3 below presents 6 instances used to compare the MILP and the GRASP in terms of the quality of the solution and the computation time (CPU).

The resolution of the MILP was done with CPLEX 20.1 with a time limit of one hour, and GRASP metaheuristic was coded on Python. The execution of both models was on a machine with a processor i7 ~ 2.8 GHz and 16 Go of Ram.

Table 4 presents the results and the gap between the makespan obtained with MILP and the makespan obtained with GRASP calculated based on equation (5):

$$gap(\%) = \left(\frac{Cmax_{GRASP}}{Cmax_{MILP}} - 1\right) * 100\% \qquad (5)$$

Table 4. Results

Instance	MILP		GRASP		GAP
	Cmax	CPU Time	Cmax	CPU Time	
1	627.41	12.17	627.41	0.07	0
2	1205.56	172.3	1208.8	0.7	0.26875477
3	4467	3600	2540.7	3.8	-43.12290128
4	1782.19	1784.8	1897	1.32	6.442074077
5	2404.76	3600	2373.6	1.6	-1.295763403
6	4439.05	3600	2355.6	3.18	-46.93459186

The results show that the GRASP can reach optimal solutions in a low CPU time but only for some instances. In addition, the solutions obtained with CPLEX (MILP) within the 1-hour limit of execution are outperformed by the solutions obtained by the GRASP in terms of quality and computation time. However, for the instances that CPLEX found an optimal solution, the GRASP needed more iterations (more time) to explore the solution space.

7 CONCLUSION

In this study, a metaheuristic is proposed to tackle the Flexible Job Shop Problem with generic time-lags. It's a GRASP metaheuristic relying on a construction heuristic to generate a solution to minimize the makespan and a local search phase to improve this solution. The results show that the proposed metaheuristic provides valuable results for some instances and good results for other instances in terms of solution quality.

To enhance the results, several techniques can be studied to be applied to our case, such as improving the GRASP metaheuristic by adding a multi-level evolutionary local search with an estimation procedure. This technique will make the local search phase fast and will allow the solution space exploration rather fast.

Hence, in future studies, we will develop a multi start GRASP with a multi-level evolutionary local search metaheuristic to improve the results and test it using the FJSP classical instances.

8 REFERENCES

Aallaoui, S., Azzamouri, A., & Tchernev, N. (2022). Mixed Integer Linear Programming Model for Open Pit Mine Scheduling. IFAC-PapersOnLine, 55(10), 2276-2281.

Azzamouri, A., Fénies, P., Fontane, F., & Giard, V. (2018). Scheduling of open-pit phosphate mine extraction. *International journal of production research*, *56*(23), 7122-7141.

Bierwirth, C. (1995). A generalized permutation approach to job shop scheduling with genetic algorithms. *Operations-Research-Spektrum*, *17*(2), 87-92.

Binato, S., Hery, W. J., Loewenstern, D. M., & Resende, M. G. C. (2000). A greedy randomized adaptive search procedure for job shop scheduling. *Essays and Surveys in Metaheuristics, ATT Labs Research Technical Report*.

Błażewicz, J., Domschke, W., & Pesch, E. (1996). The job shop scheduling problem: Conventional and new solution techniques. *European journal of operational research*, *93*(1), 1-33.

Blom, M., Pearce, A. R., & Stuckey, P. J. (2019). Short-term planning for open pit mines: a review. *International Journal of Mining, Reclamation and Environment*, *33*(5), 318-339.

Brucker, P., & Schlie, R. (1990). Job-shop scheduling with multi-purpose machines. *Computing*, *45*(4), 369-375.

Caumond, A., Lacomme, P., & Tchernev, N. (2008). A memetic algorithm for the job-shop with time-lags. *Computers & Operations Research*, *35*(7), 2331-2356.

Chaudhry, I. A., & Khan, A. A. (2016). A research survey: review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, *23*(3), 551-591.

Dauzère-Pérès, S., & Paulli, J. (1997). An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research*, *70*, 281-306.

Fu, J., Taniguchi, T., & Karasawa, Y. (2004). The largest eigenvalue characteristics for MIMO channel with spatial correlation. *Electronics and Communications in Japan (Part I: Communications)*, *87*(12), 18-27.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, *1*(2), 117-129.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of discrete mathematics* (Vol. 5, pp. 287-326). Elsevier.

Kemmoé-Tchomté, S., Lamy, D., & Tchernev, N. (2017). An effective multi-start multi-level evolutionary local search for the flexible job-shop problem. *Engineering Applications of Artificial Intelligence*, *62*, 80-95.

Kozan, E., & Liu, S. Q. (2016). A new open-pit multi-stage mine production timetabling model for drilling, blasting and excavating operations. *Mining Technology*, *125*(1), 47-53.

Lacomme, P., Tchernev, N., & Huguet, M. J. (2011, September). Dedicated constraint propagation for Job-Shop problem with generic time-lags. In *ETFA2011* (pp. 1-7). IEEE.

Lamghari, A., & Dimitrakopoulos, R. (2016). Progressive hedging applied as a metaheuristic to schedule production in open-pit mines accounting for reserve uncertainty. *European Journal of Operational Research*, *253*(3), 843-855.

Mohtasham, M., Mirzaei-Nasirabad, H., Askari-Nasab, H., & Alizadeh, B. (2021). A multi-objective model for fleet allocation schedule in open-pit mines considering the impact of prioritising objectives on transportation system performance. *International Journal of Mining, Reclamation and Environment*, *35*(10), 709-727.

Moradi Afrapoli, A., & Askari-Nasab, H. (2019). Mining fleet management systems: a review of models and algorithms. *International Journal of Mining, Reclamation and Environment*, *33*(1), 42-60.

Newman, A. M., Rubio, E., Caro, R., Weintraub, A., & Eurek, K. (2010). A review of operations research in mine planning. *Interfaces*, *40*(3), 222-245.

Pinedo, M. L. (2012). *Scheduling* (Vol. 29). New York: Springer.Roy, B., & Sussmann, B. (1964). Les problemes d'ordonnancement avec contraintes disjonctives. *Note ds*, *9*.

Xie, J., Gao, L., Peng, K., Li, X., & Li, H. (2019). Review on flexible job shop scheduling. *IET Collaborative Intelligent Manufacturing*, *1*(3), 67-77.