NEGOTIATIONS FOR AGENT-BASED CONCURRENT PROCESS SCHEDULING OF RESOURCES IN HIGHLY DYNAMIC ENVIRONMENTS

Gabriel Alejandro Icarte Ahumada

Dissertation
zur Erlangung des Grades eines
Doktors der Ingenieurwissenschaften
–Dr.-Ing. –

Vorgelegt im Fachbereich 3 (Mathematik und Informatik)
der Universität Bremen im

Tag der mündlichen Prüfung: 15.10.2021

Gutachter: Prof. Dr. Otthein Herzog
Universität Bremen

Prof. Dr.-Ing. habil. Klaus-Dieter Thoben
Universität Bremen

DEDICATION

This dissertation is dedicated to my wife Úrzula, for taking care of me and our children. Her care of me and our children made it possible for me to complete this work.

Also, I dedicate this thesis to my children Gabriel, Mateo, and Pedro. They are the main motivation in my life.

Finally, I dedicate this work to my parents and siblings, for supporting me from the beginning of my academic career.

ACKNOWLEDGEMENTS

First, I want to thank my advisor Prof. Dr. Otthein Herzog, for his remarkable support and suggestions during my research. I have been extremely lucky to have an advisor who cared about my work and responded to my questions and queries promptly. I very much appreciate the time and valuable remarks on my research. I also wish to express my thanks to Prof. Dr.-Ing. habil. Klaus-Dieter Thoben, who kindly agreed to review this thesis. Furthermore, I want to thank Dr. Jürgen Pannek for inviting me to discuss my research with his research group.

I also want to thank Dr.-Ing. Ingrid Rügge, Managing Director of the International Graduate School for Dynamic in Logistics (IGS), for his notable support. From the beginning of my research, I felt his support and encouragement. She and my colleagues from IGS were an important support to carry out this research.

My sincere thanks also go to Doña Ines de Collahuasi Mining Company, for allowing me to analyze the material handling process in its open-pit mine. I want to thank Marcos Márquez, Pablo Letelier, and Gonzalo Nuñez for sharing their knowledge and information on mining processes.

Finally, I want to thank the Chilean National Agency for Research and Development (Agencia Nacional de Investigación y Desarrollo de Chile - ANID) and the German Academic Exchange Service (Deutscher Akademischer Austauschdienst - DAAD) for their financial support during my research and stay in Germany. I also thank the Arturo Prat University and the University of Bremen for their support. The completion of this thesis would not have been possible without the strong support of these institutions.

Gabriel Icarte Ahumada

Bremen, Germany, 2021

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

The following abbreviations are used in this dissertation:

ABS	Agent-Based Simulation
ACL	Agent Communication Language
ADP	Approximate Dynamic Programming
CCA	Central Cooperating Agent
CFP	Call For Proposal
CMDIC	Compañía Minera Doña Inés de Collahuasi
DARP	Dial-A-Ride Problem
EV	Electric Vehicle
FIPA	Foundation For Intelligent Physical Agents
FIPA-ACL	Agent Communication Language Associated with FIPA's Open Agent Architecture
GP	Goal Programming
ILP	Integer Linear Programming
ILS	Iterated Local Search
JADE	Java Agent Development
KQML	Knowledge Query and Manipulation Language
LCD	Liquid-Crystal Display
MAB	Multi-Armed Bandit Algorithms
MAS	Multiagent System
MAS-TD	Multiagent System for Truck Dispatching
MILGP	Mixed-Integer Linear Goal Programming
MILP	Mixed-Integer Linear Programming
MIP	Mixed-Integer Programming
MRP	Material Resource Planning
OC	Operation Combination
PI	Performance Impact
PlaSMA	Platform for Simulations with Multiple Agents
SBRO	Scenario-Based Robust Optimization
SSA	Satellite Scheduling Agent
TDS	Truck Dispatching System

TZI Institute for Information and Communication Technologies

UAV Unmanned Aerial Vehicles

UPMSP-ST Unrelated Parallel Machine Scheduling Problem with Sequence-Dependent Setup Times

VBA Visual Basic Applications

VRP Vehicle Routing Problem

VRPTW Vehicle Routing Problem with Time Windows

1 INTRODUCTION

The extraction and processing of minerals are an essential part of how the world and its various civilizations function and interact. For a long time, humanity has used minerals to build tools, as raw materials for construction and jewelry, ornamentation, fertilizer, to manufacture various products, among other uses. Nowadays, it is not easy to find a sector of the economy or an industry that does not regularly use a product created with a mineral. By 2050, the demand for certain mineral resources will increase significantly. For instance, as Figure 1.1 shows, some mineral demand will increase for the electronics and energy technology industry. This means that the mining industry will face many challenges from different perspectives such as environmental, technological, societal, geopolitical, and geographical to face this higher demand (World Economic Forum, 2015). For this purpose, mining companies are looking for ways to improve their processes.

Like many other industries, mining has many processes that must be monitored to be improved or solved for different reasons. Some reasons are environmental care, the uncertainty of the prices of the minerals, safety, efficiency, or future demands. For these reasons, mining companies must either increase their investment or become more efficient with the same resources. In this context, cost efficiency, agility, and robustness are essential. Besides, processes must have higher flexibility and adaptability to the dynamics of the industry and the marketplace.

Current advances in technology can improve and solve some of these problems in mining processes. For instance, technologies such as automation and drones can provide more safety to mining operations. Others, like Artificial Intelligence, the Internet of Things,

smart sensors, and Machine Learning, can improve process efficiency by increasing productivity, decreasing costs, or both.

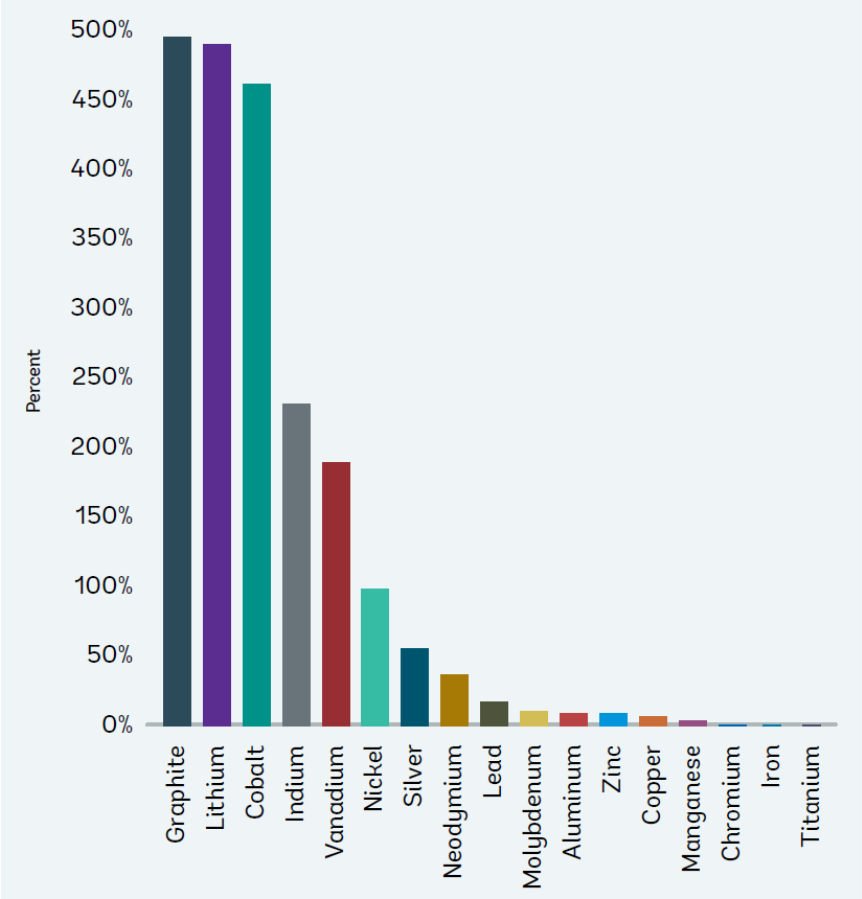

Figure 1.1: Demand for minerals by 2050 (The World Bank, 2020, p.73).

An important and complex process in the mining industry is the material handling process. This process includes continuous processing of information about the status of the equipment involved and the environment where it is performed, and it must react to disruptions or unexpected events to ensure process continuity. Besides, current centralized solutions that support this process are inefficient, since trucks' queues are built in front of shovels often, while other shovels wait for the arrival of trucks.

This thesis addresses the improvement of the material handling process in an open-pit mine by utilizing a multiagent system. A multiagent system is a system compound of intelligent agents that have to interact with each other to achieve their own objectives. Hence, these systems are inherently distributed. Besides, they can react to dynamic changes both caused by the environment and in the equipment respective states.

This chapter is organized as follows: Section 1.1 provides some personal motivation to do this research. Section 1.2 provides some background on the material handling process

can improve the truck dispatching in open-pit mines.

Section 1.3 of this work describes the problem to be solved. Section 1.4 provides the pursued objectives of this research. Section 1.5 presents the scope and limitations of the research. Section 1.6 introduces the research questions and a hypothesis. Section 1.7 presents the methodology to accomplish the research objectives and answers to the research questions. Finally, Section 1.8 gives an overview on the chapters of this thesis.

1.1 Personal Motivation

Mining is one of the most important economic activities for my country, and Chile is one of the world's most important mineral producers. Despite how important the mining industry is for the country and being one of the countries that exports the most minerals in the world, we have not developed too much mining technology. The most important mining technology is imported from abroad. For instance, trucks, shovels, even most software packages used in the Chilean mining industry are bought from foreign countries (Katz, Cárdenas, and Cáceres, 2000).

In order to be a world mining industry leader and help the country's economic development, we need to develop our own mining technology. In this way, we would not depend on foreign technology, it would allow for the study of problems specific to the country and for developing solutions for them. It would also generate new and more qualified jobs.

From my computer science background, I want to contribute to the development of the Chilean mining industry. Artificial intelligence is one of the fields in computer science that can help to solve or improve mining industry processes. For instance, smart sensors can detect rock pieces that are too big for the crusher and could cause a prolonged downtime, and autonomous haul trucks can make operations safer.

Finally, this thesis is also supported by one of the biggest copper companies in the world. This company provided the data and its mining knowledge, which are very useful for the thesis' development. Researching with the support of a world-class company is also motivating.

1.2 Background

Mining is an activity that extracts minerals that have accumulated in the soil and subsoil in the form of deposits. If the extraction of minerals is carried out from the ground, it is

called open-pit mining. If the minerals are extracted from the subsoil, it is called underground mining. Figure 1.2 shows both types of mines.

a) Open-pit mine (Chilena, 2019)

b) Underground mine (Geobrugg, 2020)

Figure 1.2: Types of mining.

During the exploitation stage of an open-pit mine, various processes are carried out. In summary, the exploitation stage considers the blasting, the extraction and material handling, the processing to obtain minerals, and delivering the minerals to the clients. This thesis focuses on the material handling process. Figure 1.3 shows a diagram of the main mining processes during the exploitation stage of an open-pit mine and its sequence.

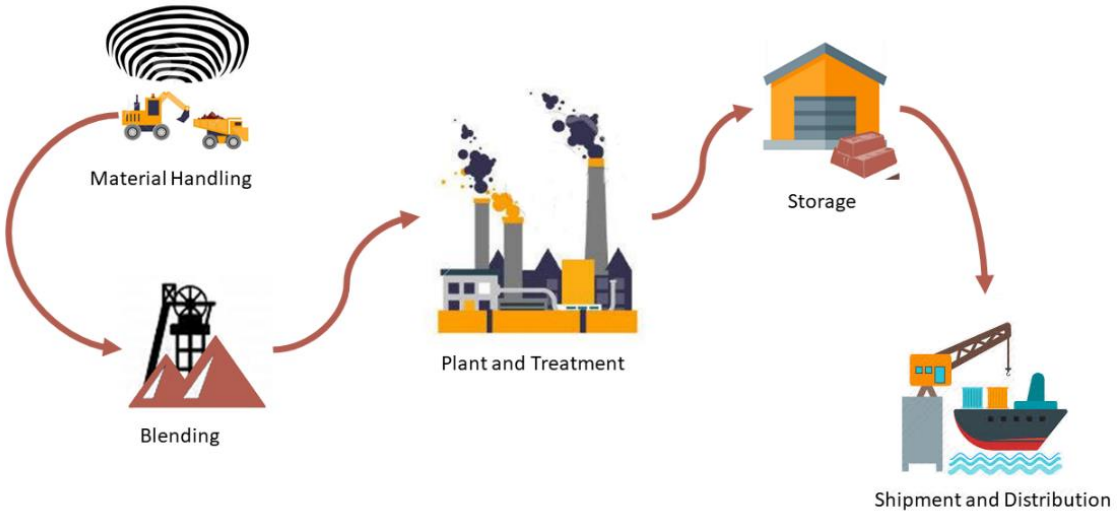

Figure 1.3: Mining process flow.

The material handling process is an important process for an open-pit mine since it represents up to 50% of its operating costs (Alarie and Gamache, 2002). In this process, big shovels and trucks must be organized to extract and transport material to different destinations at the mine. This process is carried out in a dynamic environment that may

affect the performance or availability of the involved equipment items. Therefore, it may affect the achievement of the objectives pursued. For example, changes of weather conditions or of the state of the haulage routes, or equipment breakdowns are some reasons that may cause delays in material handling (Adams and Bansah, 2016). Due to these influences, this is a complex process in open-pit mines.

Several solutions have been developed to support the material handling process, which indicate a trucks' follow-on destination taking into account two important objectives: to maximize the production and to minimize the transportation costs. These solutions consider several aspects such as mine status, information from the equipment items (trucks, shovels, crushers), production plans, and road information. Most of these solutions use a centralized approach based on an allocation model and methods from operations research, simulation modeling, or heuristic procedures.

Despite using these solutions, material handling in open-pit mines is not performed efficiently. For instance, trucks may queue up in front of shovels or crushers while other shovels wait for trucks. This generates inefficiency, high costs, not achieving production targets, etc.

There are different reasons that generate problems in the material handling process (Icarte and Herzog, 2018). Some of them are the use of an allocation model and methods that do not consider the dynamics of the environment, the use of estimated information about the involved equipment, and following a centralized system approach for global solutions only.

1.3 Statement of the Problem

Instead of using an allocation model to organize a truck fleet's activities, a schedule model would be more appropriate: Even if the allocation model is very useful in a dynamic environment since computation time is short, however, it has a myopic view since the truck allocation is only based on the current state of the mine and the equipment. This makes it hard to predict the global performance of the material handling process for an entire shift. In contrast, a schedule model has a global view of the problem and can organize the fleet efficiently. However, the computation time increases with the problem's size, which could not be appropriate in dynamic environments.

Since different entities are involved in the material handling process, each with their own objectives, specific knowledge, and behaviors, it is possible to use an approach based on intelligent agents to generate schedules. This approach would enable more decentralized

decision-making with more specific and realistic information. Using this approach, several issues are crucial, e.g., collaboration, coordination, negotiation, and learning.

In the context of the material handling process, agents can represent shovels, trucks, and crushers and negotiate with each other to generate their schedules. One way to organize the negotiation is by making sequential negotiation rounds. However, two disadvantages arise: On the one hand, if there are too many agents, the negotiation could take too much time. On the other hand, an agent that signs a contract with another agent could find a more attractive offer in a subsequent round. That would have two effects: if it decides to keep the previous contract, the agent loses the opportunity to settle on a better agreement, but if it decides to break the previous contract, it will be necessary to initiate a new negotiation round for the broken agreement. This could increase the computation time of the entire negotiation process. Besides, if an unpredicted event occurs during the execution of the schedules, the agents must update the schedules quickly by negotiating again.

In summary, the problem is how to enable the agents to negotiate efficiently to generate schedules and how to update them quickly when major events occur in the environment.

1.4 Objectives

The thesis aims to enhance the material handling process in open-pit mines using a multiagent system through improved scheduling. The challenge is to change the organization of the involved equipment in the process from an allocation model to a scheduling model. To do this, the agents in the multiagent system must negotiate to generate the schedules. The following objectives must be accomplished to achieve the project aims:

1. The generation of a model for concurrent negotiations in multiagent systems that takes the dynamics of the mining environment into account.
2. To develop a multiagent system to support a material handling process in open-pit mines with the model generated in 1.
3. To compare the performance of the material handling process supported by the system generated in 2. against a process supported by an operations research method, and against a heuristic procedure in order to determine the quality of the developed multiagent system.

1.5 Limitations of this Study

The first limitation is related to the type of material handling used in mines. There are different methods to transport the material in mining, e.g., trains, conveyor systems, or truck-shovel systems. This research considers only truck-shovel systems for material handling. In addition, because the data used in this thesis is from an open-pit mine, the findings might not apply to underground mines.

The second limitation is related to the multiagent technology which is still an active research area. However, this research will be only focused on the negotiation processes in a dynamic environment because of their importance to achieve acceptable solutions.

The third limitation is related to the scope of the multiagent system that is developed in this thesis. The system is a prototype that does not meet all the requirements of all open-pit mines. The material handling process is not performed in the same way in all open-pit mines. Some differences depend on many factors, such as location, equipment. However, the developed prototype follows the material handling process of one of the biggest copper mines in the world and thus is representative for other (smaller) open-pit mines.

1.6 Research Questions and Hypothesis

This research investigates how a multiagent approach can improve the material handling process in an open-pit mine specifically, how agents can negotiate efficiently to generate schedules in a dynamic environment. This means delegating the planning, controlling, and monitoring processes to the digital representatives of the objects themselves. The research will be focused on the following three major research questions:

1. How can a material handling process in an open-pit mine be supported more efficiently?
2. Which communication and interaction mechanism among agents allows for generating schedules?
3. How can an agent make an efficient decision during concurrent negotiations in dynamic environments?

The research hypothesis is the following:

A multiagent system with agents able to make efficient decisions during a negotiation process in a dynamic environment can provide better equipment schedules for the material handling process in open-pit mines than current approaches.

1.7 Methodology

To achieve the project's objectives, to answer the research questions, and to validate the hypothesis, a four-stage methodology was developed. Figure 1.4 shows the stages and the sequence of the methodology steps.

1. **Systematic literature reviews**: This stage aims to analyze and understand the truck dispatching problem and to study how multiagent systems work. To achieve this purpose, a systematic literature review on current solutions for truck dispatching systems in open-pit mines was done. Besides, a literature review on multiagent systems was performed.
2. **Development of a computational model**: This stage aims to generate a computational model for the different kinds of interactions between the agents, especially those related to negotiations. This computational model must include the dynamics of the environment and concurrent negotiations between the agents, based on the systematic literature reviews obtained in the previous stage. This stage was iteratively performed together with the following stage (development of the multiagent system) since the next stage's feedback can enhance the computational model.
3. **Implementation of a multiagent system**: This stage aims to implement the prototype of a multiagent system for truck scheduling in open-pit mines using the computational model generated in the previous stage. The system is implemented on PlaSMA (Platform for Simulations with Multiple Agents). This stage was iteratively performed together with the previous stage.
4. **Evaluation of the system**: This stage aims to evaluate the developed prototypical system against two centralized system approaches. To do this, an operations research method and a heuristic method were selected and implemented to compare their results against the results obtained by the multiagent system. In order to make this comparison, common scenarios based on real-world data were created.

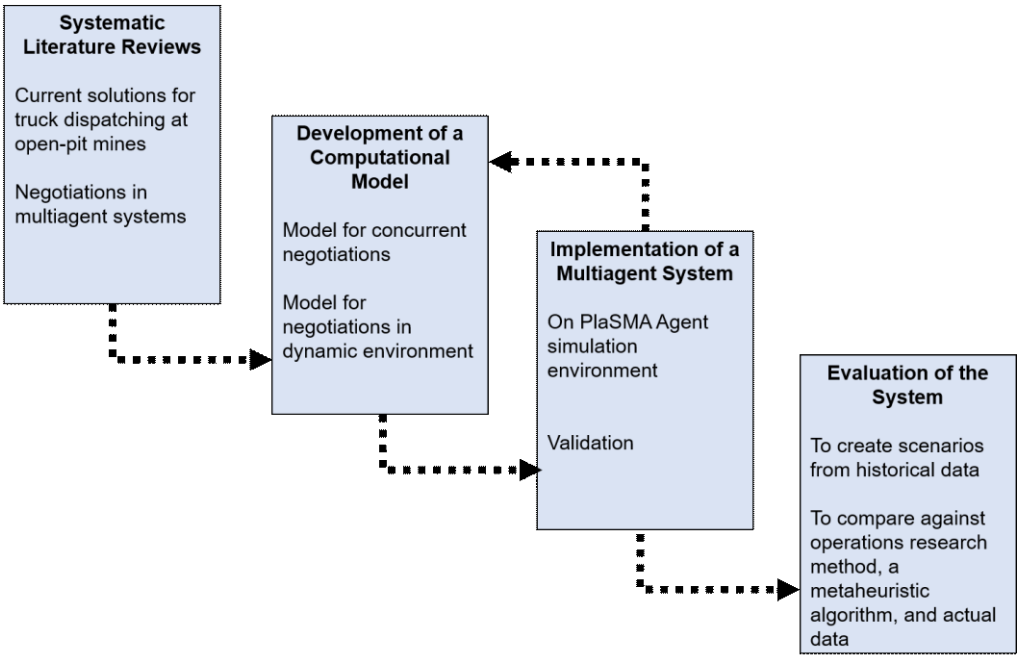

Figure 1.4: Research Methodology.

1.8 Structure of the Thesis

The thesis is structured as follows: Chapter 2 introduces the material handling process in open-pit mines. In particular, it focuses on the truck dispatching problem and the current approaches and methods to solve it. The chapter summarizes the state-of-the-art of the material handling process for open-pit mines.

Chapter 3 presents the foundations of multiagent technology and scheduling theory. It describes different multiagent system ("MAS") structures, agent architectures, communication, and interaction mechanisms that agents can employ. The scheduling theory lays out the different types of scheduling problems and a notation to model them. The end of the chapter shows how a MAS can be applied to solve scheduling problems.

Chapter 4 introduces the developed multiagent system to solve the truck dispatching problem in open-pit mines (MAS-TD). It describes how the MAS-TD generates schedules to organize the equipment involved in the material handling process. Besides, experiments with scenarios based on actual data are performed to determine the appropriate configuration setting to achieve the MAS's best performance.

Chapter 5 provides the evaluation of the prototypical MAS. This evaluation is performed in two parts. The first one evaluates the MAS-TD by comparing it against other methods that generate schedules. The second one compares the MAS-TD against an allocation system used in a real mining company. Both evaluations consider rescheduling.

Chapter 6 summarizes and concludes the research. It discusses how the material handling process in an open-pit mine can be improved by the MAS technology approach. It then returns to the initial research questions and the corresponding answers. The chapter finishes by providing new research perspectives resulting from the present findings, which may further improve both, the material handling process in open-pit mines, and concurrent negotiations in dynamic environments.

2 State of the Art of Truck Dispatching in Open-pit Mines

Material handling is an important logistic process in open-pit mines since it can account for up to 50% of the operational cost (Alarie and Gamache, 2002). In this process, shovels and trucks must work together to extract materials and transport them to different destinations at the mine. This process is performed in an environment that affects the performance of the involved equipment items, e.g., extreme weather conditions can affect the velocity of trucks and/or the availability of shovels. In order to achieve an efficient material handling process, the trucks must be dispatched correctly and as close to optimality as possible.

Truck dispatching means to answer the question of what is the next destination of a truck. To answer this question, it is necessary to consider several aspects such as the status of the mine, the performance of the equipment items and the environment where the operations are performed. Thus, deciding the next destination of a truck becomes a complex problem (Çetin, 2004).

In order to determine dispatch decisions, many systems have been developed. Most of them follow a centralized system approach, in which about information from equipment items and the status of the mine is collected centrally, and then determines and provides the destination to the trucks at a global level. Generally, these systems are based on methods such as mathematical programming, heuristic procedures, and simulation modelling (Icarte and Herzog, 2018). The strengths of these methods are their maturity and their well-known implementation. However, weaknesses can be observed in addressing the dynamics of a mine, not being able to provide a precise solution, using estimated information, and generating a dispatching solution in a timely manner when the model is too complex (Icarte, Berrios, Castillo, and Herzog, 2020).

Truck dispatching is a logistic decision in different industries. For example, in ports, cranes unload containers from vessels that must be transported by trucks to different destinations at the port yard. Another example is a modern warehouse: autonomous

vehicles collect items from shelves and move them to other places. Therefore, the solutions and proposals that arise from this thesis could be used in these and similar contexts.

The objective of this chapter is to analyze the problem of truck dispatching in open-pit mining and establish an approach to solve it. Firstly, Section 2.1 describes in detail the material handling process focusing primarily on shovel and truck operations. Section 2.2 describes truck dispatching and the different strategies and approaches used. Section 2.3 presents some representative proposed solutions. Section 2.4 discusses the proposed solutions. It particularly discusses the limitations of the current approaches and proposes a new approach to support open-pit truck dispatching in dynamic environments. Finally, Section 2.5 concludes the chapter.

2.1 Material Handling in Open-pit Mining

Material handling is an important logistic process in open-pit mines since it can account for up to 50% of the operational costs (Alarie and Gamache, 2002). In this process, several equipment items are involved, especially shovels and trucks. In simple words, big shovels extract materials that must be transported by huge trucks to different destinations at the mine. If the extracted material is ore, it must be transported and unloaded into a crusher or onto a stockpile. In the case that the extracted material is waste, it must be transported and unloaded onto a waste dump. In general, material handling is done during previously established shifts (8 or 12 hours) in order to meet the production targets established for the shift. Figure 2.1 depicts a shovel loading a truck.

In the material handling process, there are several aspects, such as the involved elements, the characteristics of the environment and the operations of trucks and shovels, which are explained in more detail in the following sections.

Figure 2.1: Shovel loading a truck (Chilena, 2015).

2.1.1 Elements of Material Handling

Although trucks and shovels are the most important equipment involved in material handling, there are other elements involved in this process with different characteristics. These elements are described in the following list:

- Trucks: They are huge machines able to transport a large amount of materials to different destinations at the mine. The transport capacity varies between 30 to 500 tons and their average speed can be 50 km/h.
- Shovels: They are big machines (generally larger than trucks) that excavate the material from the workbenches and load the trucks. Their extraction and loading capacity, as well as the time it takes for these operations, vary depending on their size.
- Crushers: They are machines used to crush the material. The crushers are fed from the top by trucks and unload the crushed material at the bottom. Crushers have mineral mixing requirements; therefore, they can receive material from different shovels. Only processable material (mineral) is crushed by the crushers.
- Stockpiles: They are storage places for minerals. Sometimes the material cannot be unloaded immediately into the crushers, so the material must be unloaded onto a stockpile. For example, trucks transport the materials to different destinations at the mine. If the crushing speed of a crusher cannot

keep up with the speed of transportation, trucks can be dispatched to a stockpile to avoid a truck queue in front of the crusher,.

- Waste dumps: They are storage places for non-processable material (e.g., rocks and sand). Often, the mineral cannot be extracted immediately by the shovels because the mineral is covered by sand and rocks, which must be removed. Those removed materials must be unloaded in waste dumps.
- Roads: Used by the trucks to transport the materials to some destination at the mine.
- Production plan: It is a plan that points out the amount of material that must be extracted and transported from shovels to unloading points (crusher, stockpile or waste dump).

These elements are present in most open-pit mines, although they may vary in the way they are used. For example, in some mines, waste dumps may receive more than one truck at a time to unload materials. In other mines, it is prohibited for trucks to simultaneously unload in these places. Another example is connected to road usage: while in some mines it is allowed for a truck to overtake another one, in other mines that action is not authorized[1].

2.1.2 The Truck Cycle

The truck cycle is a set of activities that a truck must perform during a shift in a repetitive manner. Figure 2.2 shows the stages of the truck cycle. Basically, it corresponds to the following actions (Guilherme Sousa Bastos, 2010; Chaowasakoo, Seppälä, Koivo, and Zhou, 2017b; Ta, 2015):

1. Truck loading: A shovel extracts material from a workbench and loads a truck that has done the spotting[2]. In some mines, it is possible for a shovel to simultaneously load two trucks (one on each side of the shovel) with the same material or even with two different materials.
2. Loaded travel: Once loaded, the truck transports the material to an unloading

[1] Private communication with an open-pit mine expert on 19.03.2017

[2] It is a mining term that refers to the movements of a truck from leaving its queue position, and moving towards the shovel until it reaches the position for loading.

point. The destinations of the material extracted by shovels are usually predefined in the production plan. On some occasions, the destination may change. This is the case, for example, if there is a problem at the destination that does not allow for unloading.

3. Queuing at the unloading site: This action will happen if there is another truck unloading.
4. Spotting at the unloading site.
5. Unloading: Depending on the place of unloading, this can be done simultaneously with other trucks, for example, at stockpiles or waste dumps.
6. Empty travel to the loading site: The empty truck travels to a shovel to be loaded there.
7. Queuing at the loading site: This action will happen if the shovel is loading another truck.
8. Spotting at the loading site.

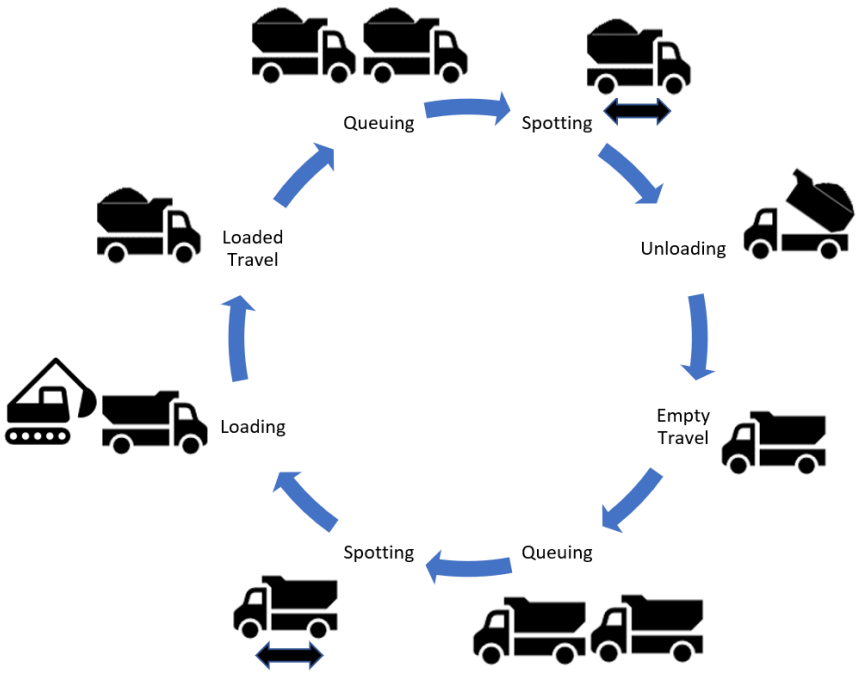

Figure 2.2: The truck cycle. (Adapted from Icarte et al., 2020, p. 2)

The truck cycle is an important tool to optimize material handling. For example, determining the time that each truck takes for different activities of the cycle, it can help to identify bottlenecks or equipment underutilization. It also allows for implementing some heuristics to determine the destinations of the trucks after an unloading activity is

finished. For instance, minimizing the total cycle time of trucks is a heuristic that maximizes the utilization of the trucks.

2.1.3 Dynamic Environment

Many open-pit mines are located in places with extreme climatic and geophysical conditions that generate a dynamic environment and, therefore, could affect the performance and availability of the equipment items involved in the material handling process. For example, changes in weather conditions or to the state of the routes as well as equipment failures are some of the reasons for delays in material handling (Adams and Bansah, 2016). Among the main factors that affect and/or hinder material handling are:

- Weather conditions: They can affect both infrastructure and equipment items. For example, low temperatures can freeze truck engines, which causes truck unavailability. In the case of fog, trucks must decrease their speed, which causes an increase in travel times.
- The geophysical variability of the mineral: Often it is easy for the shovels to extract materials from working benches. However, sometimes it is harder to extract materials due to the size of the rocks or the material hardness. This results in a shovel spending more time for the extraction of the materials, and therefore, it causes delays in truck loading. It could even produce a shovel failure.
- The topography: The mines are on surfaces that often change over time, mainly due to blasting and material extraction. For this reason, roads could change, new roads are created, and others eliminated.
- Equipment failures: Equipment may fail or decrease its performance due to two main factors: the fatigue of its components or poor operation. In both cases, the main effects are delays and/or unavailability of the equipment item.
- Other delays: There are a variety of situations external to material handling that indirectly affect it. For example, shift changes may take longer than planned. The same happens with the lunchtime breaks of the operators, or when blasting causes damage or unexpected consequences.

2.2 Truck Dispatching in Material Handling

Truck dispatching is the decision to assign a destination to a truck. In other words, it consists of answering the question: “where should this truck go now?” (Alarie and

Gamache, 2002, p. 5). In response, a destination must be assigned to a truck taking two objectives into account: increasing productivity and reducing operating costs (Alarie and Gamache, 2002). When a truck is working, there are two main moments that a truck requires a destination: after the truck finalizes a loading activity or after the truck finalizes an unloading activity. Sometimes, when a truck is traveling to a destination, the destination could change on the fly. For instance, if there is a problem at a crusher, a new destination must be sent to the trucks that are traveling to that crusher.

When a truck finalizes a loading activity at a shovel, the destination of the truck is determined by the production plan. Therefore, truck dispatching is not a decision at this moment. However, it is occasionally necessary to decide on a new destination, for instance when the destination established in the production plan is not available, or when the material extracted is different from what was planned. In those moments, it is necessary to decide on a new destination for the truck. When a truck finalizes an unloading activity at an unloading point (e.g., crusher, stockpile, or waste dump) the truck requires a new destination. At this moment it is necessary to decide on which destination is most appropriate for the truck.

In order to decide on the next destination of the truck, several procedures can be applied. For example, in small mines, maintaining fixed dispatching in which a truck travels between a shovel and a destination throughout the shift could be a good option. On the other hand, in large mines that use several trucks and shovels, dynamic dispatching would be more productive, in which the trucks are dispatched to different shovels.

In recent decades, different systems have been developed to support truck dispatching in open-pit mines. Many of these systems are based on mathematical programming, heuristic procedures, and simulation modeling. In addition, thanks to advances in information technologies, these systems have implemented different levels of automation. The following sections describe how truck dispatching in open-pit mines can be implemented from the perspectives of strategy, methods, and automation.

2.2.1 Strategies for Truck Dispatching in Open-pit Mines

When deciding what the next destination of a truck is, there are three strategies to follow (Alarie and Gamache, 2002). Figure 2.3 depicts these strategies. They differ in the scope of the elements considered to make the decision:

- 1 Truck for *n* shovels: In this strategy, when a truck requires a new destination, the *n* shovels are considered as destinations and are evaluated following some

selection criteria (cf. Section 2.2.2). The best-evaluated shovel is the next destination of the truck. This strategy is the oldest and one of the most suggested ones in the literature (Alarie and Gamache, 2002).

- *m* trucks for 1 shovel: In this strategy, the decision is made considering the other *m* trucks that will be dispatched soon. First, the shovels are ordered according to their level of production, placing those shovels with the lowest level of production at the top of the list. Considering this order, the truck that reduces the level of production is dispatched to the first shovel in the list.
- *m* trucks for *n* shovels: This strategy simultaneously considers the other *m* trucks that will be dispatched to the *n* shovels in operation, and it allows for making decisions according to a global view of the system.

2.2.2 Methods for Truck Dispatching in Open-pit Mines

Different methods can be applied in order to determine the next destination of a truck. The application of these methods depends on the complexity of the material handling process and the computation and communication of available resources. For instance, in a big mine with a big fleet of shovels and trucks and a powerful computation and communication platform, a method based on mathematical programming could be a good solution. The following sections explain the most common methods used in truck dispatching in open-pit mines: mathematical programming, heuristic procedures, and simulation modeling.

Methods Based on Mathematical Programming

A mathematical programming problem is a class of decision problem that deals with the efficient use of limited resources to meet desired objectives (Sinha, 2006). The problem to be solved must be modeled mathematically considering an objective function and constraints.

In the context of truck dispatching in open-pit mines, some of the methods applied are linear programming, integer programming, dynamic programming, and queuing theory. These methods can be applied in two ways: according to the plan-driven dispatching approach or the constrained assignment approach (Bissiri, 2002).

In the plan-driven dispatching approach there are two stages: The first stage is a planning stage, in which a guideline is determined that will be used by the second stage which is an operational stage (Munirathinam, Yingling, Munirathinarn, and Yingling, 1994).

Generally, the first stage determines the production plan for every shovel, and the second stage dispatches trucks in real-time in order to minimize the deviation from the production plan suggested by the first stage (Tan and Takakuwa, 2016).

In the constrained assignment approach there is no planning stage, only the operational stage. The dispatching decisions are made considering the targets of the production plan (Munirathinam et al., 1994).

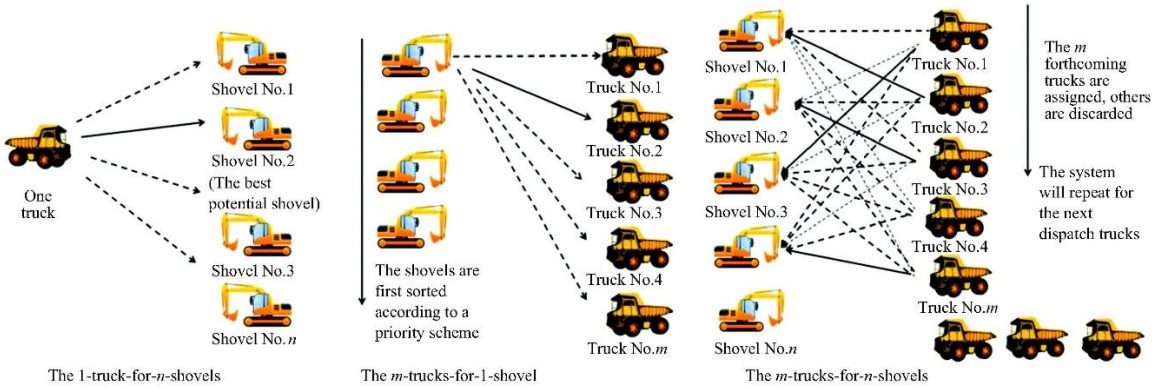

Figure 2.3: Strategies for truck dispatching in open-pit mines (Chaowasakoo et al., 2017a, p. 230).

Methods Based on Heuristic Procedures

Heuristics are a problem-solving method that uses shortcuts to produce good enough solutions to answer to a complex question or problem (Crawford and Barker, 2018). The methods based on heuristic procedures could be used when the material handling process is complex and difficult to model mathematically. Dispatch systems based on heuristic procedures are easy to implement and do not require a big computational effort to make dispatch decisions in real-time (Munirathinam et al., 1994) and are discussed in the following list:

- Fixed truck Assignment: Each truck is only assigned to transport materials from a shovel to a destination and repeats this cycle during the entire shift. Due to the variability present in the system, trucks are often queued at some shovels.
- Minimizing Truck Waiting Time: A truck is assigned to the shovel that would generate the shortest waiting time for the truck to be dispatched again. For this, the expected travel time of the truck from its unloading point to the shovel is considered as well as the time required to load all the trucks previously assigned to that shovel, including the truck being loaded, the queued trucks, and those traveling towards the shovel at the time of dispatch. A problem with this heuristic

is that it can favor those shovels that are located close to the dumpsites, and therefore generate idle time for the shovels that are located farther from the dumpsites.

- Maximize Truck: A truck is assigned to a shovel that will load it as soon as possible. This heuristic is similar to the previous one since it reduces the idle time of trucks and prevents the formation of long queues. However, it can cause imbalances in the production of the shovels.
- Minimizing Shovel Waiting Time: A truck is assigned to the shovel with a higher idle time or the one that is expected to be available first. There is a production cost associated with its implementation because of the long travel time that some trucks will require to reach the shovel, even when there is another shovel nearby but with a lower idle time.
- Minimizing Shovel Saturation: A truck is assigned to the shovel that has the lowest degree of saturation. The degree of saturation is defined as the rate between the number of trucks previously assigned to a shovel and the number of trucks that should be assigned to the shovel. The latter is calculated by the average travel time for a truck between the dispatch point and the shovel, divided by the average loading time for the trucks at that shovel.
- Minimizing Shovel Production Targets: A truck is assigned to the shovel which is most behind schedule. This strategy may be convenient if there are operational restrictions that must be met, such as a range of mineral laws required by the processing plant.

Methods Based on Simulation Modeling

Simulation modeling concerns the creation of a model as accurate as needed to mimic the behavior of a system and to reproduce observations for subsequent analysis (Kin and Chan, 2011) . Under some conditions, it is not possible to make the desired changes in a real system and monitor the resulted effects. This is where simulation modeling is most practical and allows the researcher to make modifications and get feedback at a low cost (Hashemi and Sattarvand, 2015).

Analysis of truck dispatching in open-pit mines using simulation is a well-established procedure since it allows for incorporating the inherent variability and complexity of the material handling process (Cetin, Erarslan, and Okuducu, 2001). In this context, simulation modeling is used for different purposes, such as to support decision-making,

to test methods, to determinate truck and shovel fleet size, to evaluate an optimization algorithm applied at the operational level or capturing traffic of trucks (Moradi Afrapoli and Askari-Nasab, 2017). The most applied simulation techniques for truck dispatching in open-pit mines are Monte Carlo simulation and discrete-event simulation.

2.2.3 Truck Dispatching Systems

As mentioned above, material handling can be a complex process. This complexity increases with the size of the fleets involved. For this reason, a Truck Dispatching System (TDS) is an important tool that addresses this complexity. There are different types of TDS that differ in the level of automatization: manual, semi-automated and full automated (Lizotte, Bonates, and Leclerc, 1987) which will be described in the following:

- Manual dispatching systems: In these systems, the dispatch decisions are made by a person (the dispatcher) without computational support. The dispatcher is located in a strategic position to observe the trucks and decide on the next destinations of the trucks. The decisions are made based on his/her experience and judgment.
- Semi-automated dispatching systems: In these systems, computers assist the dispatcher with the dispatch decision. The computer processes information, saves and generate reports, and interacts with the dispatcher, who then provides the dispatch decision to the truck operators
- Automated dispatching systems: These systems are composed of devices such as computers, communication equipment, sensors, and LCD screens. A computer receives information from truck sensors, determines the dispatch decision by applying a method (for instance a mathematical programming or heuristic procedure), and provides the dispatch decisions to the truck operators via LCD screen mounted in the truck.

2.3 Proposed Solutions

In the last decades, many papers have been published on the material handling process in open-pit mines. According to Icarte and Herzog (2019), these papers present solutions based on methods from mathematical programming, heuristic procedures, or simulation modelling. In the cases in which mathematical programming and heuristic procedures were used, the objective was to maximize the production, minimize the costs, or both. In the cases in which simulation modeling was used, the aim was to support the decision-

makers which dispatching strategy to use. These studies show two approaches to model the problem: the allocation model and the scheduling model. Most of these studies follow an allocation model that assigns trucks to trips between shovels and unloading points (Patterson, Kozan, and Hyland, 2017). However, this research found few studies represent the problem with a scheduling model. The following sections outline these studies from the perspective of these two approaches.

2.3.1 Solutions Based on an Allocation Model

Upadhyay and Askari-Nasab (2016) developed a mixed-integer linear goal programming (MILGP) model to optimize the operations of the equipment items based on four desired goals: maximize production, minimize deviations in head grade, minimize deviations from the desired feed in tonnage feed to the processing plants, and minimize operating cost. The model is based on an allocation model. Since the model considers goals with different priorities and dimensions, a normalization of them is performed before combining them together. The normalized goals are then multiplied with weights to achieve the desired priority. In addition, the model uses a short-time (1 month) mine production schedule. In this way, the model links the operational level with the tactical level.

The authors validated their model in the case study of Gol-E-Gohar iron ore complex, a medium-size mine located in the south of Iran. Their results show an average plant utilization above 99%, average truck utilization above 92%, and the average shovel utilization above 95%.

Bajany et al. (2017) present a mixed-integer linear programming (MILP) model which optimizes route choices and minimizes the fuel consumption of both trucks and shovels in the case of an under-trucked mine. The model uses an *m* trucks-for-*n*-shovels dispatching strategy (cf. Section 2.2.1) to allocate trucks to shovels and takes into account the demands of dump sites, payload of trucks, loaded capacity of shovels, and fuel consumption. The model provides the optimal number of trips that each truck must do on each route of the pit, and the optimum idle time of each shovel. The MILP model is solved using the "Intlinprog" algorithm of Matlab[3].

[3] see: https://la.mathworks.com/products/matlab.html?s_tid=hp_products_matlab. (last visited on: 26.01.2021)

In order to evaluate the model, they used a hypothetical small downgraded open-pit mine with two unloading points, three shovels and ten trucks of 56t, 65t, and 96t, and follow a fixed assignment strategy. The results show that 4.64% saving of fuel is achieved with the proposed energy optimization model. Applying a fixed assignment strategy is inefficient since the stochastic nature of the material handling process generates queues at particular shovels, whereas other shovels are waiting for the arrival of trucks (Koryagin and Voronov, 2017).

Mohtasham et al. (2017) improved on the chance-constrained goal programming model presented by Micholopulos and Panagiotou (2001). The model determines an optimal truck allocation plan for open-pit mines and decreases the waiting times of the equipment items. The goal programming (GP) model considers four goals: maximizing shovel production, minimizing deviations in head grade, minimizing deviations in tonnage feed to the processing plants from the desired feed, and minimizing truck operating costs.

The model was implemented with Visual Studio using the C# programming language, and the CPLEX (IBM ILOG, 2009) optimizer software was incorporated. In order to validate the model, a case study of the Sungun copper mine with shifts of 8 hours, 9 loading points and 35 trucks with different capacities was used (medium-size mine). The results obtained indicate that the presented model increases mine production by more than 20.6%, and provides more effective use of the equipment items. Besides, the case study demonstrated the efficiency of the model to work as the upper stage of a multi-stage truck dispatching system.

Bakhtavar and Mahmoudi (2020) developed a two-phase scenario-based robust optimization (SBRO) model by considering the maximization of production, control of ore grade sent to the crusher, minimization of waiting times for trucks and shovels, and trucks with different capacities. The model can select the shortest path between loading and dumping points among several transportation paths. The first phase model minimizes the operational costs of trucks based on the shortest path between loading and dumping. The second phase model minimizes the number of trucks allocated to shovels based on shovel output, the number of available trucks, and the duration of a shift. Solving the model by the SBRO minimizes the high difference in the hauling costs under different scenarios. In order to validate the model, its results were compared against the results of the current dispatching system of a real medium-size mine. The comparison shows that the model increases the production of the shovels and decreases the operational costs per ton of ore.

Araújo and Souza (2011) developed a heuristic algorithm to determine the number of trips of each truck, and the working bench where the shovels must be allocated in order to meet the production and quality targets and to achieve the best utilization of the available fleet. The heuristic algorithm uses an Iterated Local Search (ILS) and was implemented with Visual Delphi 7.0.

In order to evaluate the algorithm, they compared their results against a mathematical model implemented on LINGO optimizer 1.0 (Lindo Systems Inc, 2013) and used real data to create a scenario with 11 working benches, 30 trucks (15 with capacity of 50 tons and 15 others of 80 tons) and 8 shovels with different capacities (medium-size mine). The results produced by ILS are of better quality than those produced by the LINGO optimizer, which is remarkable since the optimizer always selects the best solution. The only explanation of this result is that the algorithm and the mathematical model do not follow the same constraints. The solutions produced by the ILS were obtained without much computational effort, unlike those generated by the mathematical programming optimizer. In relation to the trucks and shovels, the good utilization of them was verified.

Mendes et al. (2014) presented a multi-objective evolutionary algorithm that considers two objectives: to minimize the truck waiting times and the traveled distance by the trucks. The algorithm generates an initial solution set and a crossover operator. A solution represents a chain of truck allocations for a shift. Given the complexity and variety of the restrictions considered for the truck dispatching problem, a random generation of a set of solutions results in a population composed basically of non-viable solutions. For this reason, they developed a heuristic called *OptNonfeasibleSolution*. The heuristic minimizes the total value of the restrictions violated by a non-viable solution of the initial population. The heuristic performs several searches in the vicinity of a nonviable solution, trying to make it viable. In this way, the initial solution set is of better quality than a random initial solution set.

In order to validate the algorithm, the authors used data of a hypothetical medium-size mine with different scenarios. Its results were compared with those of a previous version of the algorithm that evolves a random initial solution set. The comparison shows that the algorithm that applies the *OptNonfeasibleSolution* heuristic provides better solutions than the algorithm that uses a random initial solution set.

Alexandre et al. (2015) formulated the truck dispatching problem as a multi-objective mathematical model that allocates trucks to pits. The model includes two objectives: the first one is to maximize production at the mine, be it ore or waste rock. The second one

is to minimize the number of trucks in operation. They implemented the model by three evolutionary algorithms: SPEA2, NSGA-II, and MILS. An experiment was performed to compare the algorithms in terms of hypervolume and empirical attainment function values. The experiment used data from a real medium-size Brazilian mine. The results show that the of NSGA-II and SPEA2 algorithms perform better than the MILS algorithm for the problems considered, with the NSGA-II being marginally better than the SPEA2 algorithm.

Mendes et al. (2016) presented a multi-objective genetic algorithm that implements a heuristic for generating the initial solution set that uses local search operators for scanning the neighborhood of non-feasible solution sets, and for exploring the neighborhood of feasible solution sets. This work is based on the previous work (Mendes et al., 2014). The model follows an allocation model of trucks to pits.

In order to validate the developed algorithm, two experiments were performed. The aim of the first experiment was to compare the heuristic for generating an initial solution set against a random generation of initial solution sets. The second experiment compared the proposed algorithm against other evolutionary algorithms that employ other crossover operators. The obtained results show that the proposed algorithm is competitive as it produces good solutions in most instances for all scenarios.

Koryagin and Voronov (2017) presented a heuristic algorithm to dispatch trucks after the latter completes unloading. The algorithm uses priority parameters for choosing the shovel the truck will be allocated to, and the corresponding route for material transportation is considered. In addition, the algorithm applies a minimizing truck waiting time criterion.

In order to validate the algorithm, the authors used real data from a Russian open-pit coal mine. The mine works with two shovels and 15 trucks (small-size mine). They make a comparison between the result provided by their algorithm and a fixed assignment mode. The comparison shows that their algorithm enables significant cost savings by reducing the idle time of the mining equipment and the number of operating trucks.

Hashemi and Sattarvand (2015) developed a dispatching simulation model with the objective function of minimizing truck waiting times. An ore grade blending control unit was introduced into the model which enabled monitoring the portion of material excavated from different operating benches and control truck dispatching rules.

The simulation model was implemented in Arena software[4] and validated by comparing the output with the real data for truck cycle times. The real data are obtained from a small copper mine in Iran. Productivity assessment scenarios were established for the constructed model, and the output shows that by modifying the match factor of the haulage system, the production rate in the working benches increases by 40%. The term match factor is usually defined as the ratio of truck arrival rate to loader service time (Burt and Caccetta, 2007).

Chaowasakoo et al. (2017a) illustrated the differences between the dispatching strategies by conducting a stochastic simulation study based on the data gathered from an actual big-size mine. The simulations based on a class of heuristic methods (minimizing shovel waiting time, minimizing truck cycle time, minimizing truck waiting time, and minimizing shovel saturation and coverage) show that the dispatching decisions made according to a global view outperform, in terms of production, the commonly used methods, where only one truck or one shovel is taken into account for each dispatching decision.

Ozdemir and Kumral (2018) developed an agent-based Petri net simulation model to check whether production targets are feasible, and the extent to control head grade in mineral processing by considering the uncertainties in the mining operation. Moreover, the fuel consumption of haul trucks is tracked in the model. The agent-based model provides a range of probable realizations regarding some results (i.e., ore-waste quantities, ore grade, and fuel consumption) in such a way as to assess risks in a given period. Since the model considers the static and dynamic attributes of the agents, it provides a more precise estimation of the duration of the activities.

In order to validate the model, a case study was carried out based on data from a Canadian open-pit mine. The results show that the model could assist in capacity installation, mineral processing design, fuel tracking and, helping engineers to have a better understanding of material handling in an open-pit mining operation.

Xu et al. (2019) developed an Approximate Dynamic Programming (ADP) algorithm based on Q-Learning. The algorithm implements two models: a static model, and a dynamic model. The static model allocates trucks to a loading and an unloading job. The model employs an event-driven method to define a stage when an idling truck appears at any time (maybe more than a truck). The dynamic model allocates trucks considering the

[4] see https://www.arenasimulation.com/ (last visited on 05.02.2021)

dynamic and stochastic factors. These factors are translated into change of truck speed. When the system receives information about the speed at a certain time, it needs to adjust the assignments according to the current truck operation. This adjustment follows the principle of change minimization, i.e., the new assignment result should not change the original assignment scheme too much, in order to ensure the sustainability and stability of production.

Numerical experiments were designed for small, medium, and large scale open-pit mines. Their results demonstrate that the ADP algorithm can be applied for the truck problem in open-pit mines. The dynamic model can get the near optimal assignment plan, and it can make an effective allocation for real-time operation.

2.3.2 Solutions Based on a Scheduling Model

Chang et al. (2015) addressed the truck assignment problem as a scheduling problem and proposed two methods to solve it: a mixed-integer programming model (MIP) and a heuristic algorithm with improvement strategies. The objective function of the mathematical model maximizes the total transport value, that is, the sum of transport revenue of all trucks unloaded within a time horizon. The model provides a schedule of truck trips for a time horizon. In order to strengthen the model, a few valid inequalities are deduced, and two upper bounds of the problem are proposed. The model was implemented in CPLEX, an optimization software (IBM ILOG, 2009). The heuristic algorithm has two stages. The first stage provides an initial solution that is improved in the second stage through two improvement strategies: the shovel capacity rebalancing strategy and the neighborhood swapping strategy. The algorithm was implemented using VC++ 2010.

The authors performed numerical experiments to test the performance of the formulated mathematic model and the proposed heuristic approach. The data used in the experiments came from a hypothetical open-pit mine and involved two different mine path configurations. They considered different fleet sizes with homogeneous trucks. Their results show that the mathematical model is valid but not applicable to medium and large problem sizes, and that the developed heuristic algorithm with improvement strategies is effective and efficient.

Patterson et al. (2017) presented a mixed integer linear programming (MILP) scheduling model of the excavation and haulage activities with the purpose of minimizing energy consumption. Their work is based on the model formulated by Chang et al. (2015). The

model provides a schedule of truck trips for a time period to reach the targets in the production plan. To produce a truck schedule, the model uses a sequence of loading 'slots' per shovel to order the trucks being loaded at each shovel and also to order their unloading at the destination. Among the main aspects considered by the model are: heterogeneous equipment, shovel allocation, an explicit calculation of waiting time, and production targets.

The authors implemented their model in CPLEX (IBM ILOG, 2009) and it was tested for hypothetical mines with different fleet sizes. However, it was not possible to get a solution for medium and big-size mines in an appropriate time. To solve the model computation quickly, they implemented a Tabu Search algorithm. Finally, the model was tested in an open-pit coal mine in South East Queensland, Australia, with three shovels and 20 trucks (small size-mine). The solution generated by the model represented an increase in energy efficiency of 17.6%.

Tan and Takakuwa (2016) propose a simulation modeling approach enhanced by VBA programming to achieve an effective truck dispatching system in an open-pit mine. The model creates and tests a dynamic dispatching control table to meet a mining plan in an open-pit mine. The model was implemented in Arena Software[5]. They evaluated the model in a case study of a medium-size real open-pit mine. The results show that using this model, the transportation performance of the trucks could be improved, leading to a reduction in transportation costs.

2.4 Discussion

The previous sections have described the material handling process and the truck dispatching problem in open-pit mines. Besides, a description of the representative publications was presented. Given this background, this section discusses the following issues: the different models used to represent the problem, the centralized approach followed by truck dispatching systems, and the methods utilized by these systems to determine the dispatch solutions. Table 2.1 gives an overview of investigated proposed solutions for truck dispatching in open-pit mines. It shows their more relevant aspects and allows for comparing them in a structured way.

At first sight, truck dispatching in open-pit mines seems to be a kind of vehicle routing

[5] see https://www.arenasimulation.com/ (last visited on 05.02.2021)

problem (VRP) (Dantzig and Ramser, 1959) because of the presence of vehicles (trucks), pickup nodes (shovels), delivery nodes (crusher, stockpiles) and routes. However, truck dispatching in mines presents some characteristics that are not reported in the usual truck dispatching literature (Alarie and Gamache, 2002). Although there are some similarities, there also are differences. For instance, there is no start and end node (depot), the trucks must go to a pickup node and then to a delivery node and repeat this sequence during the entire shift, and the frequency of demand by the pickup points is large (Subtil, Silva, and Alves, 2011). These characteristics can require that a truck visits a node more than once. The travel times between nodes are usually short and the number of nodes is lower than the number of trucks, which produces waiting times at nodes. These differences make it difficult to apply a pure VRP model. In fact, all the reviewed related work, as well as the most applied commercial systems, do not apply a pure VRP model.

A model that has been widely applied is the allocation model (Patterson et al., 2017). An allocation model determines a real-time dispatching decision (or allocation) of trucks to shovels. Most of the time the allocation is required by a truck when it completes an unload. Although this model provides useful results, their dispatch solutions are of a "local maximum" because the model considers only the status of the mine and the equipment items at that moment. This decision cannot guarantee synchronization of the shovel and truck operations (e.g., loading operations) for a bigger time horizon (e.g., a shift of 8 hours).

.

Table 2.1: An overview of investigated proposed solutions for truck dispatching in open-pit mines.

Authors	Approach[6]	Type of Method[7]	Specific Method[8]	Type of Model	Evaluation			cf. Page
					Mine	Size[9]	Comparison against	
Upadhyay and Askari-Nasab (2016)	C	MP	MILGP	Allocation	Real	M	Key performance	22
Bajany et al. (2017)	C	MP	MILP	Allocation	Hypothetical	S	Fixed assignment	22
Mohtasham et al. (2017)	C	MP	GP	Allocation	Real	M	Key performance	23
Bakhtavar and Mahmoudi (2020)	C	MP	SBRO	Allocation	Real	M	Key performance	23
Araújo and Souza (2011)	C	HP	ILS	Allocation	Real	S	Mathematical Programming	24
Mendes et al. (2014)	C	HP	EA	Allocation	Hypothetical	D	Evolutionary algorithm	24
Alexandre et al. (2015)	C	HP	EA	Allocation	Real	D	Evolutionary algorithm	25
Mendes et al. (2016)	C	HP	EA	Allocation	Real	D	Evolutionary algorithm	25
Koryagin and Voronov (2017)	C	SM	Ad Hoc	Allocation	Real	S	Fixed assignment	26
Hashemi and Sattarvand (2015)	C	SM	DE	Allocation	Real	S	Key performance	26
Chaowasakoo et al. (2017a)	C	SM	DE	Allocation	Real	L	Key performance	26
Ozdemir and Kumral (2018)	C	MP	AB	Allocation	Real	M	Key performance	27
Xu et al. (2019)	C	MP	DP	Allocation	Hypothetical	S, M, L	Key performance	27
Chang et al. (2015)	C	SM	MILP	Scheduling	Hypothetical	S, M, L	Heuristic algorithm	27
Tan and Takakuwa (2016)	C	SM	DE	Scheduling	Real	M	Key performance	28
Patterson et al. (2017)	C	MP	MILP	Scheduling	Real	S, M, L	Metaheuristic algorithm	28

[6] Approach: (C) centralized, (D) distributed/decentralized

[7] Type of Method: (MP) Mathematical Programming, (HP) Heuristic procedure, (SM) Simulation Modeling

[8] Specific Method: (MILP) Mixed Integer Linear Programming, (MILGP) Mixed Integer Lineal Programming, (GP) Goal Programming, (ILS) Iterated Local Search, (EA) Evolutionary Algorithm, (DE) Discrete Event.

[9] Size: (L) Large, (M) Medium, (S) Small

A few researchers have proposed another model called the scheduling model (Chang et al., 2015; Patterson et al., 2017; Tan and Takakuwa, 2016). These models allow for generating schedules with all the operations that the involved equipment must perform with the start and end times of the operations. In addition, schedule models allow for synchronization between the operations of shovels and trucks. This kind of solution could be a more precise solution than those provided by an allocation model because they take into account a time horizon, that means that the solution is a "global solution". In their papers, the authors demonstrate the applicability of their models using data from a real open-pit mine and from a hypothetical one. However, they do not mention how their models could deal with a major change occurring in the mine.

All the reviewed papers propose models and methods that follow a centralized approach, which means that a central node receives information from the other nodes and takes all the decisions and then communicates these decisions to the nodes. This approach has several advantages such as being based on traditional proven structures and methods, robust optimization routines and centralized data storage. However, it has some disadvantages such as a high investment, low flexibility and scalability, and low tolerance to failures. In the context of truck dispatching in open-pit mines, a central node manages the information from/to trucks and shovels and determines and provides the destinations to the trucks. On one hand, the maintenance of these systems is quite simple but on the other hand, any failure in the central node, or a disruption in the communication between the central node and the trucks, generates the non-delivery of the destinations to the trucks. In this case, the fleets continue working with manual dispatching (by a human dispatcher,) or they perform a loop between the last shovel where the truck was loaded, and the last destination provided by the system.

Another approach that could be applied is a distributed approach, in which there are several nodes communicating with each other by a network and without a central node. Each node can take decisions and communicates its decisions to the others. Among the advantages of this approach are a better fault-tolerance than a centralized system, better performance, it allows for a more diverse and more flexible system, and a better scalability than the centralized approach. Among its disadvantages are a higher maintenance costs, inconsistent performance when not properly optimized and a higher communication effort. In the context of truck dispatching, the equipment items could organize their operations by themselves. In

addition, if one of the equipment items fails, the system would continue operating.

Regarding the methods in the literature applied to provide a dispatch solution, it can be observed that three types of methods are mostly used: mathematical programming, heuristic procedures, and simulation modeling. Regarding mathematical programming, although it determines the best solution, there are some problems: mines with large fleets of trucks and shovels represent complex stochastic networks that are very difficult to model (Sáez, 2014). In addition, in order to deal with the complexity of the problem, assumptions or estimations have to be made and, as a consequence, the performance of these methods can be limited (Temeng, Otuonye, and Frendewey, 1997).

The reviewed articles that used mathematical programming to solve an allocation model show good results. However, several researchers mention some disadvantages of these methods. One of the main disadvantages is that these methods use estimated pieces of information (Chang et al., 2015; Costa, Souza, and Pinto, 2005; Krzyzanowska, 2007; Newman, Rubio, Caro, Weintraub, and Eurek, 2010). For instance, they simplify that the equipment items have the same performance. However, this is not true in reality because this depends on the operations of the truck driver, the state of the routes and the state of the equipment. The use of estimated data cannot guarantee precise solutions. Another disadvantage mentioned is that these methods do not consider the dynamics of the environment (Bastos et al., 2011). In the case that the problem is modeled as a scheduling model, applying mathematical programming to solve it would be impracticable for big size mines (Chang et al., 2015; Icarte, Rivero, and Herzog, 2020; Patterson et al., 2017). In addition, when a major change occurs in the mine, there is not enough time to recompute the whole solution from scratch by an optimal solver.

Regarding heuristic procedures, despite its easy implementation, some authors (Munirathinam et al., 1994) highlight two weaknesses. The first is that when evaluating the performance of only one truck at a time it seems a myopic strategy, that is, the decision to dispatch each truck is made ignoring its impact on subsequent shipments. The second weakness is that it does not consider multiple measures of system performance when dispatching a truck. Some heuristics are weak since they are not able to consider restrictions, such as those imposed by the mineral treatment plant. In the case that the dispatch problem is modeled as a scheduling model, it is possible to solve it by applying a heuristic procedure even for big size mines (Chang et al., 2015; Icarte, Rivero, et al., 2020; Patterson et al., 2017),

although the solution provided by the method could be not optimal.

The reviewed articles that used heuristic procedures show good results for allocation models and scheduling model. However, some researchers mention the same disadvantages as for mathematical programming: the use of unspecific or estimated pieces of information for the equipment items (Chang et al., 2015; Newman et al., 2010), and that they do not consider the dynamics of the environment (Bastos et al., 2011). Heuristic procedures maybe preferable, in many cases, to mathematical programming since those methods require a large amount of calculation time (Gurgur, Dagdelen, and Artittong, 2011). However, they do not guarantee an optimal solution (He, Wei, Lu, and Huang, 2010; Jaoua, Riopel, and Gamache, 2009).

From the literature review and the discussion above, the following gaps became obvious:

- Lack of evaluation about whether the methods used in the scheduling model to generate the solutions are appropriate when a major event occurs in the mine.
- Finding out if the implementation of a scheduling model following a distributed approach can produce better solutions than the centralized approach.

2.5 Conclusions

This chapter introduced the material handling process and the truck dispatching problem in open-pit mines. The chapter starts with a description of the material handling process, pointing out the main components involved and the dynamic environment where the process is performed. Then the truck dispatching problem is presented. The different dispatch strategies are explained as well as the main methods applied to get a dispatch solution. Next, the description and classification of some representative truck dispatching systems are provided. Finally, some related work is described followed by a discussion.

A first conclusion is that, although the truck dispatching problem seems to be a Vehicle Routing Problem (VRP), a pure VRP model cannot be applied. This is because some characteristics of Truck Dispatching (e.g., visit a node more than once) do not allow for applying a pure VRP model. This conclusion is supported by the fact that all the proposed reviewed papers and commercial software do not apply a pure VRP model.

A second conclusion is regarding which model is more appropriate to represent the problem. In order to get a more efficient dispatch solution, a scheduling model must be preferred instead of an allocation model. A scheduling model allows for generating schedules for the

equipment involved for a time horizon pointing out the start and end times. This is a more precise and global solution that considers the synchronization of the activities of the trucks and shovels. In order to solve this type of model, mathematical programming, heuristic procedures, and simulation modeling could be applied. To select the appropriate method, the size of the problem is a characteristic that must be taken into account. For instance, mathematical programming is not able to deal with medium and large size problems (Chang et al., 2015; Icarte et al., 2020; Patterson et al., 2017). However, if a system follows a distributed approach, each node could use mathematical programming to solve a part of the problem.

The last conclusion is about what is the most efficient approach to deal with the problem. In this case, a distributed approach seems to be a better option than a centralized one due to the following reasons:

- The nature of the process is distributed (several equipment items working together).
- Avoidance of the dependency on a central node.
- The use of specific local data to make the decision in each node of the system (and to generate more precise solutions).
- To provide more robustness and flexibility than a centralized approach.

3 MULTIAGENT SYSTEMS AND SCHEDULING

Chapter 2 described the material handling process and the truck dispatching problem in open-pit mines. The conclusions of that chapter result in the statement that, in order to solve the problem more efficiently than current solutions, a solution should model the problem as a scheduling problem and deal with it with a distributed approach.

A multiagent system can be used for a distributed approach, composed of agents, and is able to deal with complex and dynamic problems (H. J. Müller, 1997). Some advantages of multiagent systems are flexibility, adaptability, scalability, and robustness, which are achieved by the proactive, reactive, and adaptive behavior of intelligent agents (Gath, 2015). Multiagent systems have been applied to different problems, including scheduling and transport. The characteristics of the multiagent technology make it suitable for the truck dispatching problem in open-pit mines.

This chapter outlines multiagent technology and its application to solve scheduling and transport problems. Firstly, Section 3.1 presents intelligent agents as the fundamental parts of multiagent systems. Section 3.2 describes the multiagent technology, focusing on the interactions between the agents. Section 3.3 shows how multiagent systems are applied to solve scheduling problems. Section 3.4 presents representative work related to the application of multiagent systems in scheduling and in the vehicle dispatching domain. Section 3.5 discusses the proposed solutions. Finally, Section 3.6 concludes the chapter.

3.1 Intelligent Agents

Agents are the main parts of multiagent systems. In the last decades, agent technology has received increasing attention from researchers and has been applied in many application areas (Müller and Fischer, 2014) because the inherent distribution of those areas allows for a natural decomposition of the system into agents able to interact with each other to achieve a

common objective. The following sections give an overview of agent definitions, a characterization of the environment where the agents could perform their operations, and the different architectures in which the agents could be implemented.

3.1.1 Agents and Environments

In general, there are many definitions of the term agent even in computer science (Gath, 2015). In the context of computer science, one of the most cited ones is that an agent is a software entity situated in an environment, able to perform actions in an autonomous way to achieve its objectives (Wooldridge, 1997). Russell and Norvig (2010) define an agent as "something that perceives and acts in an environment".

Independently of the definitions, there are some properties that are usually attributed to agents to solve particular problems. Nwana (1996) and Franklin and Graesser (1996) point out that autonomy, social skill, rationality, reactivity, pro-activity, adaptability, mobility, and accuracy are properties that should be present in agents, perhaps to different degrees.

Autonomy refers to the fact that the agents can operate without the direct intervention of humans or other agents. Social Skill refers to the fact that agents are able to interact with other agents (human or non-human) through a communication language. Rationality means that an agent can reason about perceived data in order to calculate a solution. Reactivity refers to the fact to that agents are able to perceive their environment and act upon it. Pro-activity means that agents can act guided by their own objectives or that they can take the initiative. Adaptability refers to the ability of the agent to change its own behavior through a learning process. Mobility refers to the ability of an agent to move through a network. Accuracy means that an agent cannot knowingly communicate false information.

An agent performs its actions in an environment. Determining in which environment the agent will perform its operations helps in the designing and implementation of the agent. For instance, it can help to decide which is the most appropriate architecture for the agent. Russell and Norvig (2010) suggest the following environments:

- Fully vs. partially observable. The environment is fully observable if the agent can get all aspects relevant to its decision-making. Otherwise, the environment is partially observable.
- Static vs. dynamic. If the environment can change when the agent is deliberating, then

the environment is dynamic; otherwise, it is static.

- Single agent vs. multiagent. If only one agent is involved in an environment, the environment is called a single-agent environment. If there are more agents in the environment, it is called a multiagent environment.
- Deterministic vs. stochastic. If the next state of the environment is entirely determined by the current state and the action taken by the agent, then the environment is deterministic, otherwise it is stochastic.
- Discrete vs. continuous. If in an environment there is a finite number of percepts and actions that can be performed within it, then such an environment is called a discrete environment, otherwise it is called a continuous environment.
- Episodic vs. sequential. In an episodic environment, each episode is independent of the previous one, while in a sequential environment, each agent action can have consequences for all future activities.
- Known vs. unknown. An environment is considered to be "known" if the agent understands the laws that govern the environment's behavior.

3.1.2 Agent Architectures

A key aspect of the agents is deciding which of its actions it should perform in order to achieve its objectives. In order to do this, an agent must possess certain elements and organize them in some way. The way to organize the agent elements is called agent architecture and it can depend on the environment where the agent is situated or on the aim of the agent.

Agent architectures are software architectures for decision-making systems embedded in an environment (Weiss, 1999). In general, the architecture makes the percepts from the sensors available to the agent and feeds the agent's action choices to the actuators as they are generated. Figure 3.1 depicts an architecture for a utility-based agent. On the basis of the architecture used by the agents, Russell and Norvig (2010) suggest the following classification:

- Simple reflex agents. These agents select actions on the basis of the current percept, ignoring the rest of the percept history.
- Model-based reflex agents. These agents use their percept history and their internal memory to make decisions about an internal model of the world around

it.

- Goal-based agents. These agents have a goal to achieve. Their decisions are based on how to achieve the goal.
- Utility-based agents. These agents act based not only on how to achieve the goal but, in addition, they look for the best way to reach that goal.
- Learning agents. These agents are capable of learning from their experiences and, in this way, can improve their decisions.

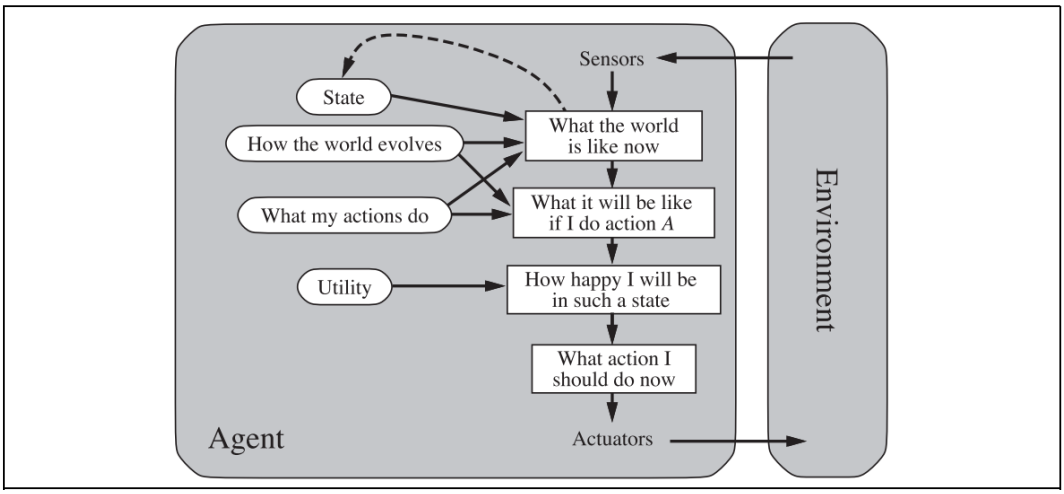

Figure 3.1: General architecture for a utility-based agent (Russell and Norvig, 2010, p. 54).

3.2 Multiagent Systems

Agent technology was started during the late ‘70s and early ‘80s with some works related to the cooperation of multiple systems to solve problems (Mas, 2005). When a problem is too complex to be managed by an agent, it may be considered to splitting the problem among several agents, in which each agent would manage a less complex problem. With this idea in mind, multiagent systems emerged (Huhns and Singh, 1998), systems made up of several agents that interact with each other, seeking to satisfy their own objectives and the objectives established for the system. The following sections describe what a multiagent system is and how the agents interact with each other.

3.2.1 Characteristics of Multiagent Systems

A multiagent system consists of a set of agents which interact with each other, typically by exchanging messages through some network infrastructure. In the most general case, the agents in a multiagent system represent or act on behalf of users or owners with different goals and motivations. In order to interact successfully, the agents require the ability to cooperate, coordinate, and negotiate with each other, in a similar way that humans cooperate, coordinate, and negotiate (Woodridge, 2001).

In a multiagent system, many aspects must be taken into account, and the range of these aspects generates different configurations of multiagent systems. For instance, the structure of the system and the environment where the agents act, are attributes that differentiate one multiagent system from another. Table 3.1 shows an overview of some attributes with their potential range.

Table 3.1: An overview of some attributes of multiagent systems (Weiss, 1999, p. 4).

	Attribute	Range
Agents	Number	Range from two upward
	Uniformity	Homogeneous . . . heterogeneous
	Goals	Contradicting . . . complementary
	Architecture	Reactive . . . Deliberative
	Abilities (sensors, effectors, cognition)	Simple . . . advanced
	Frequency	Low . . . high
	Persistence	Short-term . . . long-term
	Level	Signal-passing . . . knowledge-intensive
Interaction	Pattern (flow of data and control)	Decentralized . . . hierarchical
	Variability	Fixed . . . changeable
	Purpose	Competitive . . . Cooperative
	Predictability	Forseeable . . . unforseeable
	Accessibility and knowability	Unlimited . . . limited
Environment	Dynamics	Fixed . . . variable
	Diversity	Poor . . . rich
	Availability of resources	Restricted . . . ample

Other characteristics of multiagent systems are: provision of an infrastructure specifying communication and interaction protocols, which are often open, have no centralized design and contain agents that are autonomous and distributed, and possibly self-interested or cooperative (Weiss, 1999). In cooperative environments, the agents cooperate to achieve a global overall goal or at least a common sub-goal. In a competitive environment, agents try to achieve their own objectives or maximize their benefits.

According to Weiss (1999), multiagent systems are suitable for applications that show an

inherent distribution and/or an inherent complexity. For instance, problems that show distributed features, in their elements or data, such as spatial distribution (arise from different places) and/or temporal distribution (arise at different times), or are structured into clusters based on meaning (semantic distribution), and/or structured into clusters based on functionality are suitable for multiagent systems. Regarding complexity, whether the application is too large to be solved by a single agent (centralized) because of software or hardware limitations, a multiagent system would be more appropriate. In addition, multiagent systems offer several desirable properties such as efficiency, robustness, reliability, scalability, flexibility, and reusability (Weiss, 1999).

3.2.2 Agent Communications

For two or more agents to communicate with one another, it is necessary for them to understand the syntactic and semantic aspects of the messages that are exchanged. A similarity could be established between the communication process among agents and the one among humans. On the one hand, there is the message itself which is transmitted. On the other hand, there is the meaning that the sender wants to give to that message and, finally, the action that the receiver performs when he receives the message. This communication model, known as the Speech Act Theory (Searle, 1969), allows to consider these three aspects and is used in the area of multiagent systems. In this way it is possible to indicate the meaning of a message through performatives, allowing to restrict the semantics of the communicative act.

KQML (Fininet al., 1994) and FIPA-ACL (Foundation for Intelligent Physical Agents, 2002a) are agent communication languages that establish a formal structure for computer-readable messages. These languages follow the considerations mentioned by Searle (1969). The Foundation for Intelligent Physical Agents[10] (FIPA) is an organization that establishes standards and specifications for agent communication and interactions. FIPA has adopted and worked on specifications for architectures, communications languages, content languages, and interaction protocols. Table 3.2 shows a subset of FIPA performatives, and

[10] see https://www.fipa.org (last visited on 01.07.2021)

Figure 3.2 shows an example of a FIPA-ACL message.

Table 3.2: Subset of FIPA performatives. (Based on Foundation for Intelligent Physical Agents, 2002).

Performative	Description
REQUEST	The sender requests the receiver to perform some reaction.
AGREE	The action of agreeing to perform some action, possibly in the future
REFUSE	The action of refusing to perform a given action and explaining the reason for the refusal.
NOT UNDERSTOOD	The sender of the act (for example, *i*) informs the receiver (for example, *j*) that it perceived that *j* performed some action, but that *i* did not understand what *j* just did.
INFORM	The sender informs the receiver that a given proposition is true.
FAILURE	The action of telling another agent that an action was attempted but the attempt failed.

```
(request
    :sender (agent-identifier :name alice@mydomain.com)
    :receiver (agent-identifier :name bob@yourdomain.com)
    :ontology travel-assistant
    :language FIPA-SL
    :protocol fipa-request
    :content
      ""((action
          (agent-identifier :name bob@yourdomain.com)
           (book-hotel :arrival 15/10/2006
                       :departure 05/07/2002 ... )
      ))""
)
```

Figure 3.2: Example of FIPA-ACL message (Bellifemine et al., 2007, p. 18).

3.2.3 Interaction Protocols

Previously, it has been described how agents can send simple messages using a standard communication language. But more complex is the fact of establishing conversations where

different simple messages are exchanged between different agents in a certain order. A conversation between agents is modeled in an interaction protocol, which establishes the possible sequences of messages between a sender and a receiver in a conversation. In this way, both can follow the course of a conversation analyzing the expected sequence of messages.

When designing a protocol, especially those related to negotiations, certain criteria should be taken into account in order to evaluate the quality of the protocol. The mechanism design area is the design of protocols for governing interactions in multiagent systems (Woodridge, 2001) and is related to game theory (Morgenstern and Von Neumann, 1953). There are several criteria to consider in order to evaluate a protocol. Weiss (1999) proposed the following criteria:

- Social welfare: It measures the global overall utility of the agents. Nevertheless, the agents could get a low utility in a solution that maximizes social welfare (Gath, 2015).
- Pareto efficiency: In a solution, no agent can increase its utility without decreasing the utility of all other agents.
- Individual Rationality: Taking part in a negotiation is individually rational for an agent if there is an incentive to take part in it.
- Stability: The mechanism should be designed to be non-manipulable.
- Computational efficiency: The mechanisms should be designed in a manner that requires little computation effort for the agent.
- Distribution and communication efficiency: The mechanism should be designed in a manner that it avoids single points of failure and bottlenecks.

FIPA has developed a set of interaction protocols that represent a large set of possible coordination situations. Furthermore, these protocols can be modified, refined or used as components of more complex protocols. Table 3.3 shows the FIPA interaction protocols.

Table 3.3: FIPA Interaction Protocols.

Title	Description	Reference
FIPA Request Interaction Protocol Specification	It allows one agent to request another to perform some action.	Foundation for Intelligent Physical Agents (2002g)
FIPA Query Interaction Protocol Specification	It allows one agent to request to perform some kind of action on another agent.	Foundation for Intelligent Physical Agents (2002e)
FIPA Request When Interaction Protocol Specification	It allows an agent to request that the receiver perform some action at the time a given precondition becomes true	Foundation for Intelligent Physical Agents (2002h)
FIPA Contract-Net Interaction Protocol Specification	It is a minor modification of the original contract-net IP pattern in that it adds rejection and confirmation communicative acts.	Foundation For Intelligent Physical Agents (2002b)
FIPA Iterated Contract-Net Interaction Protocol Specification	It is an extension of the basic FIPA Contract-Net IP, but it differs by allowing multi-round iterative bidding.	Foundation for Intelligent Physical Agents (2002c)
FIPA Brokering Interaction Protocol Specification	It is designed to support brokerage interactions in mediated systems and in multiagent systems	Foundation for Intelligent Physical Agents (2002a)
FIPA Recruiting Interaction Protocol Specification	It is designed to support recruiting interactions in mediated systems and in multiagent systems.	Foundation for Intelligent Physical Agents (2002f)
FIPA Subscribe Interaction Protocol Specification	It allows an agent to request a receiving agent to perform an action on subscription and subsequently when the referenced object changes.	Foundation for Intelligent Physical Agents (2002i)
FIPA Propose Interaction Protocol Specification	It allows an agent to propose to receiving agents that the initiator will do the actions described in the "propose" communicative act when the receiving agent accepts the proposal.	Foundation for Intelligent Physical Agents (2002d)

Auctions

According to Noriega and Sierra (1998), auctions are a mechanism for pricing in which the negotiation follows an extremely simple process of coordination. The auction is usually carried out by an auctioneer who presents the goods to be exchanged and makes (or receives) offers. The auction follows some pre-established bidding convention that determines the sequence of offers, the manner in which the winner is designated and the amount at which the good is awarded. Woodridge (2001) establishes that auctions may vary in the following dimensions:

1. The winner determination: It defines who wins the auction. The good is allocated to the agent with the best bid. Such protocols are known as first-price auctions. Another way is to assign the good to the agent that has the highest bid, but this agent pays only

the amount of the second-highest bid. Such auctions are known as second-price auctions.

2. Common knowledge. This dimension is related to the knowledge that agents have on the bids of the other agents. If each agent can see the other bids, then the auction is called an open bid. If the agents cannot determine the bids of the other agents, then the auction is called a sealed-bid auction.
3. Bidding procedure. It is the mechanism that defines the bidding process. A well-known mechanism is one in which the auctioneer starts with a low price for the good (the reservation price) and increases the price in successive rounds. This mechanism is known as ascending. The opposite is a descending auction in which the auctioneer starts with a price and this price decreases each time that a round finalizes without a winner. A simple manner is with single round bidding. This mechanism is known as one shot bidding.

Woodridge (2001) also describes how the value of the auctioned item can be determined:

- Private value: the value of the item depends only on the agent's own preferences.
- Common value: the value that an agent gives to an item depends entirely on the value that other agents give to it.
- Correlated value: The value assigned by an agent depends partly on their own preferences and partly on the valuations of other agents.

Some popular auctions are English, First-price sealed-bid, Dutch, Vickrey, and All-pay. In the English auction, the bidders propose bids. The bids are public and increase until there are no more bids. So, the bidder that proposes the highest bid is the winner at the price of the bid. In the first-price sealed-bid auction, the participants present private bids. The winner is the person who presents the highest bid. In the Dutch auction, an auctioneer starts the auction with a high price for the auctioned item and continuously lowers the price until a bidder accepts the current price. In the Vickrey auction, bidders present private bids. The winner is the person who presents the highest bid at the price of the second-highest bid. All-pay auction is an auction in which each bidder must pay its bid whether they win the auction or not. The winner is the person who presents the highest bid.

One way to classify auctions is to regard the manner that the goods are auctioned in. In this context, auctions can be categorized as combinatorial auctions, parallel auctions, and

sequential auctions. In combinatorial auctions, several items are auctioned and the participants present their bids for combinations of goods following their own interests. The auctioneer determines the winner by the evaluation of the submitted bids. If the number of goods auctioned is high, the computation time will be high for the bidders since they must consider all the combinations of goods. In addition, determining the winning bid is also a problem because the winner determination problem is either of exponential size or NP-hard, although is possible to use adequate heuristics (Koenig et al., 2006).

In parallel auctions, each participant presents a bid for each good at the same time. The auctioneer selects the best bid for each good auctioned. In sequential auctions, several single-good auctions are performed. In each auction the bidders provide a bid for the current good auctioned. Both, in parallel and sequential auctions, the computation and communication effort are lower than in combinatorial auctions (Gath, 2015).

The Contract-Net Protocol

The Contract-Net protocol (Smith, 1980) is a well-known mechanism for task assignment that has proved to be a flexible and low communication interaction protocol (Schillo et al., 2002). In this protocol, an agent can play the role of a manager or contractor. The general idea of the protocol is the following: the manager splits a complex task into less complex subtasks easier to solve, announces to other agents that there is a sub-task expecting to be executed, and waiting for the offers of the contractors. The contractors receive this announcement of tasks and evaluate their skills and availability to perform them. If the contractor can perform the task, it makes an offer. After receiving the offers, or until a deadline is expired, the manager selects the most appropriate offer and assigns this sub-task to this agent, for which a contract is created. When the contractors have won bids, they must reserve the resources required for its execution, perform the tasks assigned to them, and generate reports about the progress of these tasks and their final results. Note that a contractor may further partition a task and award contracts to other nodes, by acting as the manager of that task. The control of the whole task assignment is distributed because processing and communication are not focused on particular nodes (Smith, 1980). Figure 3.3 depicts the Contract-Net protocol standardized by FIPA (Foundation For Intelligent Physical Agents, 2002b).

The contract-Net protocol is an anytime algorithm since its process can be stopped before

getting an optimal solution. This means that the algorithm can be stopped at any time and still a valid solution will be returned. This characteristic is useful in a dynamic and stochastic environment where the available running times are not known in advance (Gath, 2015).

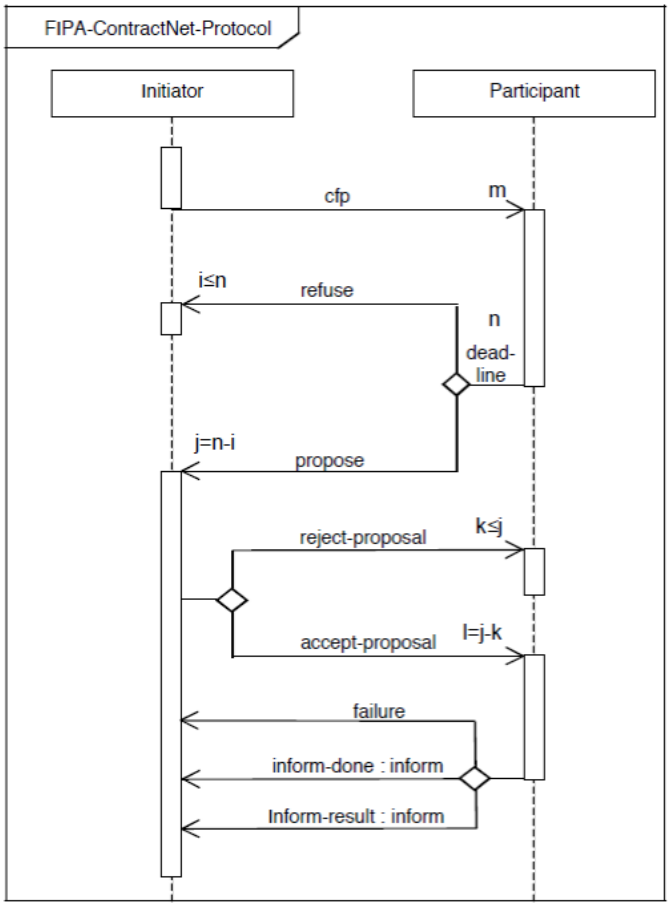

Figure 3.3: Contract-Net Protocol (Foundation For Intelligent Physical Agents, 2002b, p. 2).

3.2.4 Multiagent Organizational Structures

In a MAS with cooperative agents, they must select their activities regarding their joint performance in order to achieve their common objectives. To do this, they must be organized, be aware of each other and their capacities as well as of the interactions that they can make. Horling and Lesser (2004) mention many organizational structures, which are compared in Table 3.4 and described in the following list:

- **Hierarchies:** Agents are organized in a tree structure. In the highest level of the tree, they have a more global view than those below them. Only interactions between connected agents are allowed. This structure can be applied in environments where a natural decomposition makes sense.
- **Holarchies:** Agents are organized in holons, which act as corporate entities. A holon is a distinct entity, which appears as a whole while being potentially part of some larger entity consisting of several holons.
- **Coalitions:** Coalitions are groups of agents that work together to accomplish their tasks. They are short-lived with goal-oriented structures that solve a specific predefined problem. A coalition can be dissolved when its purpose no longer exists, or by the departure of agents.
- **Teams:** An agent team consists of a number of cooperative agents which have agreed to work together toward a common goal.
- **Congregations:** A congregation is a long-lived group of agents that is not formed to reach a pre-defined goal. It is an organization in which the agents can get additional benefits. Agents can freely enter and leave the group in comparison to coalitions.
- **Societies:** A society of agents is a long-lived and open structure. Different types of agents can join and leave the society at any time.
- **Federations:** A federation is a group of agents which have ceded some amount of autonomy to a single delegate which represents the group.
- **Markets:** In a market structure, the agents behave similarly to consumers or merchants with the goal of increasing individual utility. In this structure, agents use auctions and negotiations mechanisms to interact with one another.
- **Matrix:** Matrix structures relax the one-manager restriction presented in hierarchical structures, by permitting many managers or peers to influence the activities of an agent.
- **Compound Organizations:** Compound organizations are architectures that may include different characteristics of control.

Table 3.4: Comparison of organizational structures for multiagent systems (Horling and Lesser, 2004, p. 26).

Paradigm	Key Characteristic	Benefits	Drawbacks
Hierarchy	Decomposition	Maps to many common domains; handles scale well	Potentially brittle; can lead to bottlenecks or delays
Holarchy	Decomposition with autonomy	Exploit autonomy of functional units	Must organize holons; lack of predictable performance
Coalition	Dynamic, goal-directed	Exploit strength in numbers	Short term benefits may not outweigh organization construction costs
Team	Group level cohesion	Address larger grained problems; task-centric	Increased communication
Congregation	Long-lived, utility-directed	Facilitates agent discovery	Sets may be overly restrictive
Society	Open system	Public services; well defined conventions	Potentially complex, agents may require additional society-related capabilities
Federation	Middle-agents	Matchmaking, brokering, translation services; facilitates dynamic agent pool	Intermediaries become bottlenecks
Market	Competition through pricing	Good at allocation; increased utility through centralization; increased fairness through bidding	Potential for collusion, malicious behavior; allocation decision complexity can be high
Matrix	Multiple managers	Resource sharing; multiply-influenced agents	Potential for conflicts; need for increased agent sophistication
Compound	Concurrent organizations	Exploit benefits of several organizational styles	Increased sophistication; drawbacks of several organizational styles

3.3 Multiagent Scheduling of Resources

The solution proposed at the end of chapter 2 for the truck dispatching problem in open-pit mines proposes to use a scheduling model. For this reason, the following sections address topics related to scheduling theory, such as problem classification, notation, and rescheduling. In addition, a Section describes scheduling from the perspective of multiagent systems.

3.3.1 Scheduling

Scheduling is a decision-making process that deals with the allocation of resources to jobs over given time periods considering the optimization of one or more objectives (Pinedo, 2008). Scheduling problems can be identified in different areas of application such as logistics processes, manufacturing industry, transport. Scheduling theory is an active research area containing a great number of scheduling models (Agnetis, Billaut, Gawiejnowicz, Pacciarelli, and Soukhal, 2014).

A scheduling problem can be described by the characteristics of the resources, jobs, and the schedule objective to pursue. Resources and jobs can take many forms. In the case of resources, some examples are machines in a workshop, runways in an airport, and bricks in construction. In addition, resources can have different properties such as the speed of processing, location, and capacity. In the case of jobs, some examples are takeoffs and landings at the airport, stages of a construction project, or processes of an operating system. Jobs may have different properties such as priority levels, start time, arrival time, delivery time, etc. The objectives can also take many different forms. For instance, one objective can be the minimization of the makespan (the completion time of the last job to leave the system), minimize lateness, and/or minimize total weighted completion time.

Another way in which scheduling problems differ from one another is regarding information availability. While in some problems all the information is available a priori, in other problems the information becomes available only later. Some examples are the arrival of new jobs or changes in the machines such as failures or delays.

Scheduling problems can be difficult to solve due to technical issues as well as in respect to implementation (Pinedo, 2008). From the technical point of view, the difficulties are similar to other forms of a combinatorial optimization problem. Research in scheduling has tried to develop efficient methods to solve scheduling problems in polynomial time. However, many scheduling problems are not possible to be solved in polynomial time. These kinds of problems are the so-called NP-hard problems. Most of the complexity results for scheduling problems can be found at http://www.informatik.uni-osnabrueck.de/knust/class/ (Liaoet al., 2014).

From the implementation point of view, difficulties can arise due to the reliability of the input data that are needed and in respect to the accuracy of the model used for the analysis of the

actual scheduling problem (Pinedo, 2008).

Due to its usefulness in many industrial applications and its complexity, scheduling has been widely studied by applying different methods such as exact methods, evolutionary algorithms, heuristics, and metaheuristics (Zweben and Fox, 1994). Due to exacts methods are not practical for complex scheduling problems, search methods are used to solve these problems. However, for large complex problems are more appropriated to use local search algorithms instead of search methods (Shen, Member, Wang, and Hao, 2006). In the last decades, many researchers have applied agent technology to resolve scheduling problems, especially in large and complex problems due to the advantages that the agent technology offers such is parallel processing and natural decomposition of problems (Shen et al., 2006). More details on how MAS have been applied to scheduling problems are described in Section 3.3.4.

3.3.2 Rescheduling

In dynamic and stochastic processes, not only the generation of high-quality schedules is important but also the ability to update them quickly when unexpected events occur. For instance, machine failures, unexpected availability of resources, or delays are common events that may affect the operation guided by schedules.

In this context, the application of methods to solve only static problems is impractical when preliminary schedules become obsolete (Ouelhadj and Petrovic, 2009). In the context of truck dispatching for open-pit mines, preliminary schedules must be reviewed because of the dynamics of the environment and the stochasticity of the material handling process. Some examples that imply reviews of the schedules are equipment failures, sick operators, increased travel times, and changes in the state of the routes. These unforeseen events or changes affect the operations, imply a review of the current schedules, and the necessity to update them.

Approaches, Strategies, and Methods for Rescheduling

The research on rescheduling provides different approaches, strategies, and methods to be applied to rescheduling problems (Ouelhadj and Petrovic, 2009; Vieira et al., 2003). One approach is “Dynamic Scheduling”: In this approach, no schedules are generated, and

decisions are made locally in real-time by applying priority dispatching rules. This approach is used by most current truck dispatching systems in open-pit mines. Another approach is "Predictive-Reactive Scheduling": In this approach, schedules are revised (and updated, if necessary) in response to real-time events.

Regarding strategies for rescheduling, there are two main strategies: schedule repair and complete rescheduling. These strategies are applied only in the Predictive-Reactive Scheduling approach. For schedule repair, the schedules affected by an unforeseen event are revised and updated in case it is necessary. For complete rescheduling, new schedules are generated from scratch, and the tasks are re-scheduled from the moment on when the unforeseen event occurs.

For the strategy of rescheduling, when to reschedule is another important issue. Regarding this issue, three policies can be found in the literature (Ouelhadj and Petrovic, 2009; Vieira et al., 2003): periodic, event-driven, and hybrid. In the periodic policy, schedules are generated at regular intervals. The dynamic scheduling problem is broken down into a series of static problems that can be solved by using classical scheduling algorithms. The schedule is then executed and not revised until the next period begins, where the planning horizon is renewed considering the current information at that moment. For the event-driven policy, rescheduling is triggered in response to an unforeseen event that affects some equipment or resource used by the schedule. For the hybrid policy, the schedules are rescheduled periodically and also when an exception occurs.

Regarding schedule repair, some methods (Ouelhadj and Petrovic, 2009; Vieira et al., 2003) are: right-shift schedule repair, match-up schedule repair, and partial schedule repair. The right-shift method shifts the remaining operations of the schedule forward in time by the amount of downtime in the event of machine failure or delay. The match-up schedule repair method reschedules to matchup with the pre-schedule at some point in the future, and the partial schedule repair reschedules only the failed operations.

3.3.3 Notation

In all scheduling problems, the number of machines and jobs is finite. The number of jobs is identified by n and the number of machines by m. Usually, the subscript j refers to a job while subscript i refers to a machine. If a job requires multiple processing steps or operations,

then the pair (*i, j*) indicates the operation (processing step) of job *j* on machine *i*. The following data are associated with each job *j*:

- Processing time (p_{ij}). Represents the processing time of job *j* on the machine *i*.
- Release date (r_j). Represents the time job *j* arrives in the system, i.e., the earliest time at which job *j* can start its processing.
- Due date (d_j). Represents the committed shipping (date of completion, shipment or delivery), the deadline at which the job must be finished.
- Weight (w_j). Represents a priority factor. It points out the relative importance of job *j* regarding other jobs in the system.

A scheduling problem can be described using the standard triplet α, β, γ notation introduced by Graham et al. (1979) . The α component represents the machine environment including a single machine (1), identical machines in parallel (*Pm*), machines in parallel with different speeds (*Qm*), unrelated machines in parallel (*Rm*), flowshop (*Fm*), flexible flow shop (*FFc*), job shop *(Jm*), flexible job shop *(FJc*), or open shop (*Om*).

The β field indicates conditions applying to jobs. Usual entries in this field are time-related conditions, processing-time-related conditions, batch-related constraints, availability constraints and precedence.

The γ field is related to the objective function to be minimized. Some entries of this field are makespan (*Cmax*), maximum lateness (*Lmax*) and total weighted completion time ($\sum w_j C_j$). $FFC / r_j \mid \sum w_j C_j \mid$ is an example for the triplet field notation. In this case, it denotes a flexible flow shop in which the jobs have release and due dates and the objective is the minimization of the total weighted tardiness. Pinedo (2008) provides an extensive description of scheduling notations.

3.3.4 Multiagent Scheduling Problem Description

Most classical scheduling research supposes that the objective pursued is the same for all jobs considered in a scheduling problem. However, in many real life applications, a set of jobs could have different objectives (Perez-Gonzalez Framinan, 2014). For instance, in some problems, the jobs have to be finished after their arrival plus their processing times, i.e., the jobs must be processed in a fixed time. So, the remaining jobs should be scheduled considering that during a time interval, a machine will be busy (processing the former jobs).

In this context, multiagent scheduling is an approach by which a scheduling problem is decomposed into smaller parts that are solved by agents that probably have different (and/or conflicting) objectives and coordinate their sub-solutions using a communication mechanism (Toptal and Sabuncuoglu, 2010). In this way, agents can quickly solve their part of the general problem, and as a consequence of this, can come up with a faster solution for the overall scheduling problem. In addition, this overall solution is more responsive to dynamic events (Toptal and Sabuncuoglu, 2010).

In order to design a multiagent system to solve a scheduling problem, three design factors must be considered (Toptal and Sabuncuoglu, 2010): For the whole system, for an individual agent, and for inter-agent relations. From the system perspective, the following issues must be addressed:

- How the scheduling problem is decomposed;
- Who the independent decision-makers are, and
- How the independent decision-makers are assigned to solve subproblems.

From the perspective of an individual agent, the following issues must be taken into account:

- What information is prevalent to each agent;
- What are the tradeoffs between local and global objectives;
- How is the local scheduling problem solved, and
- How the solution is updated to eliminate conflicts.

Finally, from the perspective of inter-agent relations, the following issues must be considered:

- What command structure exists between these decision-makers, and
- How the partial schedules are communicated to identify conflicts.

Rescheduling based on Multiagent Systems

Multiagent systems have been applied to many rescheduling problems. The two main MAS architectures implemented especially in the manufacturing domain are autonomous and mediator architectures (Ouelhadj and Petrovic, 2009).

In autonomous architectures, agents representing machines, resources, or jobs interact with each other to generate schedules and to reschedule them, if necessary. This architecture

usually faces problems in providing optimized schedules and predictability in the presence of a large number of agents (Ouelhadj and Petrovic, 2009). Figure 3.4a shows an autonomous multiagent architecture.

The mediator architecture extends the autonomous architecture by adding mediator agents to coordinate the results of the interactions of the agents. The agents in charge of generating/repairing the schedules maintain their decision-making autonomously, but may request advice from the mediator agents. The mediator agents can advise, impose, or update decisions taken by the agents in charge to generate/repair the schedules to satisfy the global objectives and resolve the conflict situations. Figure 3.4b shows a mediator multiagent architecture.

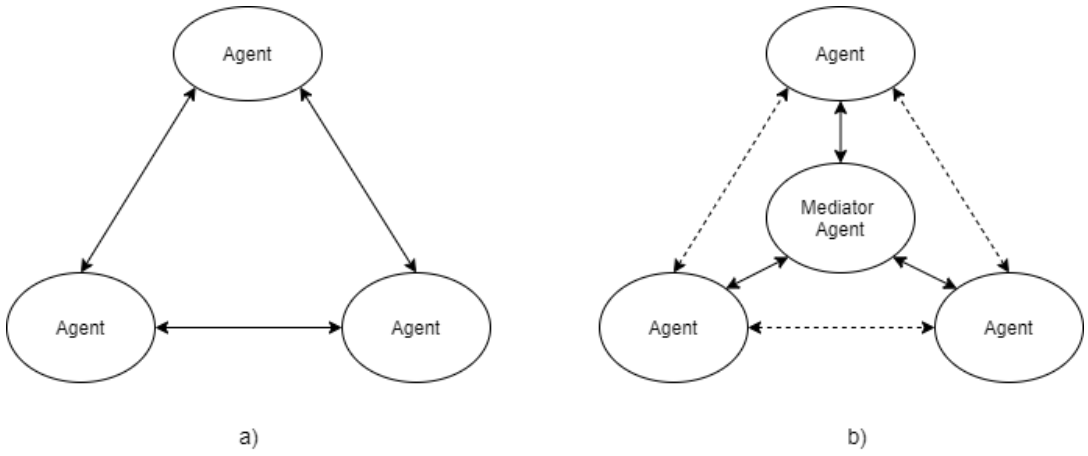

Figure 3.4: Multiagent scheduling architectures (Adapted from Lou et al., 2010, p. 3893).

3.4 Proposed Solutions

The research on the application of multiagent systems in scheduling problems has gained attention because these kinds of problems are computationally hard, making exact optimization methods impractical (Fazlirad and Brennan, 2018). This section describes some representative results related to the application of multiagent systems to scheduling problems, especially those that are applied to the transport domain.

3.4.1 Multiagent Systems for Scheduling Problems in Transport Domain

Granichin et al. (2013) present a multiagent system for real-time adaptive truck allocation, scheduling, and optimization of long-distance transportations of cargo. Agents represent orders and trucks. The agents are able to reschedule their schedules in response to upcoming events. When a new order arrives, a request for its allocation is sent to all truck agents. Truck agents evaluate the request and send an answer to the order agent. After receiving the proposals, the order agent selects the proposal that maximizes its profit.

In order to evaluate the solution proposed, simulations of different models of cargo transportation were implemented. The results show that by using the multiagent system the benefits increase up to 40-60%. In addition, an analysis was performed on the number of trucks required. The analysis results show that by using adaptive scheduling in real-time it is possible to execute the same number of orders with less trucks (up to 20% less).

Feng et al. (2015) describe a multiagent system capable of scheduling large quantities of satellites and large-scale scenes. Each satellite is modeled as a Satellite Scheduling Agent (SSA) to schedule plans according to its own goals, and there is a Central Cooperating Agent (CCA) which exchanges information with SSAs to adjust their results, thus achieving an optimum of the overall results. The agents use the contract-net protocol to generate schedules. SSAs generate bids, and send bids to the CCA according to its status and the ability to schedule the mission. A CCA receives bids, evaluates the combinations of bids that are composed by selecting a bid from each SSA, and then selects a combination to sign a "contract". The method of evaluation for bids used by the CCA is based on a genetic algorithm.

In order to validate the MAS, a simulation of the multi-satellite mission scheduling process was implemented using data from the satellite orbits database. In the simulation, the solution proposed is compared with a previous version of the system in which the CCA agent uses a depth-first search to evaluate the bids. The comparison shows that the solution proposed can reduce computing time. However, it cannot guarantee to find the best solution. Finally, the conclusion is that the previous MAS is suitable for a small number of agents, while the solution proposed is appropriate for a large number of agents or for large-scale problems.

Madhyastha et al. (2017) present an agent-based system for a Dial-a-Ride Problem (DARP). Vehicle agents and passenger agents use procurement auctions in which vehicle agents play

the role of sellers and passenger agents play the role of buyers. In a procurement auction, the roles of the buyers and the sellers are reversed as the sellers compete to obtain business from the buyers. Each vehicle agent bids on passengers with a bid value which is an inverse function of the time it would take the vehicle to reach the passenger. The system also considers a scheduling algorithm which incorporates traffic information in order to ensure that the fastest vehicle to reach the passenger is assigned to this task.

In order to validate the proposed solution, a comparison against a FIFO algorithm was performed using simulations of a vehicle fleet in two different locations. The results show that the system improves the maximum and average waiting times of the passengers, as well as the fuel costs for the vehicle fleet. In addition, the agent-based system reduces the response time to customer requests compared to a centralized system.

Whitbrook et al. (2018) describe a solution to deal with two common problems with heuristic task allocation approaches: Solution trapping in local minima and the static structure. To deal with these problems, they added two modules to an existing distributed task allocation algorithm called Performance Impact (PI). The first module extends the algorithm to permit dynamic online rescheduling in real-time, and the second one boosts performance by introducing an additional soft-max action-selection procedure that increases the algorithm's exploratory properties. PI is a distributed task allocation algorithm that runs separately on each agent. It uses a combinatorial auction-based approach, but introduces the concept of PI as a score function to build bundles iteratively by adding and removing tasks and testing the impact on global cost. There is a local task allocation phase in which each agent generates a bundle of tasks, and a task consensus phase that resolves conflicts through local communication between connected agents. The two phases are repeated until convergence, i.e., until a conflict-free task assignment is reached.

To validate the proposed solution, they developed a multiagent system to organize a fleet of unmanned vehicles that deliver food and medicine to survivors of a disaster. Each survivor requires either food or medicine and their delivery constitutes the completion of a task. Agents represent vehicles and the PI algorithm runs in each agent. Their results demonstrate the effectiveness of the dynamic rescheduling module and show that the average time taken to perform tasks can be reduced by up to 9% when the soft-max module is used. In addition, the solution to some problems that the baseline PI cannot handle is enabled by the second module.

Skobelev et al. (2018) present a multiagent system for unmanned aerial vehicles (UAVs) group action planning. The MAS is organized through communication between individual modules on each UAV, connected to the peer-to-peer network. The jobs to perform are observations of specific sub-areas that are part of a greater area. Each UAV has a separate single-board computer with MAS software on board. In addition, UAVs can send messages wirelessly. In this context, scheduling is the process of negotiation between these agents in order to define the activities agreed to be carried out by the agents. A centralized mechanism is responsible for the initial distribution of the jobs between individual UAVs of the group. Then, the UAV agent has to build a flight path to perform UAV area supervision. The UAV agent analyzes the list of available jobs and evaluates alternative sets of paths based on the set of planning criteria. During the planning process and performance of in-flight jobs, each UAV can re-allocate jobs according to agent key performance indicators. During coordination of the flight plan in the group, each UAV provides a set of possible options for changing positions. A dynamic scheduling mechanism provides adaptive relocation of individual observation areas between UAVs in the group to minimize the difference between completion times between different UAV devices. Thus, an adaptive balancing mechanism ensures that the mission is performed in the shortest time, even in cases in which UAVs vary their performance and unplanned events occur.

Series of experiments were performed to evaluate the effectiveness of job distribution by UAVs, to assess the formed UAV flight plan, to estimate time required for planning and preparing flight plans, and to assess system reactions to tasks changing during planning. Their results show that the MAS allows for planning and correcting actions of the UAV group when some events occur through the interaction of their individual computing modules.

Chargui et al. (2019) describe a reactive multiagent system for online scheduling and rescheduling of resources in port container terminals. The MAS objective is to maximize the productivity of the terminal while assigning simultaneously vessels, cranes, trucks, and workers. Trucks, cranes, and vessels are represented by agents. A scheduler agent uses a heuristic procedure to generate the entire schedule with the information provided by the agents. An initial schedule is built before the beginning of the scheduling horizon. If a real-time perturbation occurs, the agent generates an updated schedule starting from the instant the perturbation occurs.

In order to validate the proposed solution, the results of the heuristic integrated into the MAS

were compared with an exact method (a constraint programming model) implemented with CPLEX solver (IBM ILOG, 2009). The comparison shows that the heuristic implemented in the MAS is able to provide schedules even for large-size problems. In contrast, the solver runs out of memory when the problem is too large. Then, they compared two approaches: In one approach, planners prepared a schedule using the solution given by a heuristic, called "planned solution". Rescheduling is done after perturbations occur, resulting in an "actual solution". In the second approach, planners could also use the robust schedule provided after simulating the previous approach. Then, beginning with this solution, rescheduling is done as well resulting in an "actual solution". To measure the ability of the MAS in reacting against real-time events, they evaluated the gap between the "planned solution" and the "actual solution". Simulation results show that the second approach is suitable for planning stability with small gaps, and it can react to real-time perturbations without deviating too much from the initial schedule.

Seitaridis et al. (2020) propose an agent-based simulation to solve the problem of scheduling (and rescheduling) Electric Vehicle (EV) charging within a set of multiple charging stations. Each station has a number of chargers and an amount of available energy. Each EV and each charging station is represented by an agent. EV agents send their charging requests to the stations and each station agent computes an optimal solution using Integer Linear Programming (ILP) with the objective of maximizing the amount of charged energy and the number of charged EVs. Then, station agents send an offer (based on the calculated schedule) to each EV request. EV agents evaluate the offers, select one and reject the others or, they can require better offers from the station agents. In the next round, the stations recompute their schedules, considering solely the EV agents that accepted their offer and those that demanded a better offer in the previous round. If a station still cannot charge an EV, it computes an alternative proposal that is as close as possible to the EV's initial preferences.

Two situations are considered in the problem: static scheduling and dynamic scheduling. The static scheduling occurs at the beginning of the simulation when all the EV agents send their charging requests simultaneously. The stations compute their charging schedules at once and the previously described negotiation is started. In the dynamic scheduling case, EV agents may send a charging request at an arbitrary time point and the station agents increasingly compute their charging schedules. This applies in situations in which an EV is delayed. Through empirical evaluation, it was shown that the use of alternative proposals improves

the metrics of a single station and brings them closer to those of the optimal scenario. It is observed that when more stations are added in the system, the negotiations have less contributions and the static scenarios show better results than the dynamic one.

3.4.2 Multiagent Systems for Scheduling Problems in Other Domains

Martin et al. (2016) propose an agent-based distributed framework to solve combinatorial problems. Each agent is autonomous and can execute different metaheuristic/local search combinations with different parameter settings. The agents continuously adapt themselves during the search process using a direct cooperation protocol based on reinforcement learning and pattern matching. The agents cooperate using asynchronous messages.

The proposed solution was evaluated using two combinatorial problems: Permutation Flow-shop Scheduling and Capacitated Vehicle Routing. In both cases, the solution proposed performed at least as well as the state-of-art.

Wang et al. (2018) present a multiagent system to generate real-time scheduling for flexible job shops. The MAS consists of agents that represent tasks and machines, an agent that manages a pool of unprocessed tasks, an agent that monitors real-time production execution information, and an agent that uses a mathematic model. In addition, there is a bargaining-game-based algorithm to generate an optimal schedule according to the real-time data.

In order to validate the proposed solution, a simulated case study was implemented. The results of the simulation were compared to several traditional dynamic scheduling methods, including the complete reactive scheduling method. The comparison shows that the proposed MAS outperforms the traditional dynamic scheduling strategies in terms of makespan, critical machine workload, and total energy consumption.

Lin et al. (2018) present a mechanism based on multi-armed bandit algorithms (MAB) to optimize rescheduling intervals due to the occurrence of uncertain events in well development projects in the oil and gas industries. The proposed mechanism is designed for a multiagent-based framework of well scheduling, in which the optimization is performed by taking into account cumulative delays and the number of completed tasks in well development. Reinforcement learning is used to solve the MAB problem, in which each arm of the bandit problem is represented as a possible interval for rescheduling the well tasks. In the MAS, a scheduler agent receives the set of tasks to be scheduled and generates packages

using combinatorial auctions, then generates a schedule by a bidding mechanism with agents that represent service providers.

In order to validate the proposed solution, experiments based on real data were conducted in a simulation environment. Their results demonstrate the effectiveness of the proposed model and the multiagent framework. The exploration and exploitation properties of the MAB algorithm helped to optimize the intervals of rescheduling while achieving higher operational efficiency.

Lopes Silva et al. (2019) developed a multiagent framework for optimization using metaheuristics to solve combinatorial optimization problems. In this approach, the agents encapsulate a heuristic or metaheuristic and have the function of seeking the solution for a given combinatorial optimization problem. Each agent acts autonomously in this environment and interacts cooperatively with it and with the other agents. The interaction between the agents allows a metaheuristic hybridization. In addition, the agents use Reinforcement Learning to improve their self-adaptive skills.

In order to validate the proposed solution, they used two theoretical problems: the Vehicle Routing Problem with Time Windows (VRPTW) and the Unrelated Parallel Machine Scheduling Problem with Sequence-Dependent Setup Times (UPMSP-ST). Their results confirm that the learning capacities of the agents and the interaction between them influence the quality of solutions. The results also strengthen the issue of the scalability of the framework, since, with the addition of new agents, there is an improvement of the solutions obtained.

Gehlhoff and Fay (2020) present a multiagent system to generate plans for small and medium size manufacturing companies. The plan considers transport and production operations and points out pickup times and starting dates. The MAS uses a heterarchical structure and agents represent physical entities such as orders, vehicles, and machines. The agents interact with each other using the contract-net protocol. The order agent initiates the negotiation and receives production proposals from the resource agents. Each production proposal received by the order agent constitutes the base for a possible operation combination (OC). To determine the best OC, each possible path is calculated and selected according to defined criteria. After the best possible path has been determined, the order agent accepts the proposals that are included within the best path and rejects the rest.

In order to validate the MAS, a scenario based on an industrial application with an on-line

scheduling case was developed. The plans created by the MAS were feasible in the sense that all physical constraints were met.

Further reviews on the application of multiagent systems to scheduling problems can be found in Fazlirad and Brennan (2018) and Skobelev (2011).

3.5 Discussion

Chapter 2 concluded proposing a solution for the truck dispatching problem in open-pit mines based on a distributed approach and modeling the problem as a scheduling problem. The previous sections have described the multiagent technology and scheduling methods as parts of the solution for the truck dispatching problem in open-pit mines. In addition, a review of related results was presented. Table 3.5 shows these results and their more relevant aspects, allowing a comparison between them in a structured way.

Given this background, this section discusses several subjects. The first one is related to how agent technology has been applied to scheduling problems. The second subject addresses how the truck dispatching problem in open-pit mines can be formulated as a scheduling problem. The third subject is regarding how the MAS must be designed to be applied to the truck dispatching problem in open-pit mines from the perspective of a scheduling problem.

3.5.1 Scheduling Problems and Agent Technology

A problem in which jobs must be scheduled considering resources and optimization criteria is called a scheduling problem. In some problems, sets of jobs have to be scheduled considering specific optimization criteria and another set of jobs have to be scheduled by other optimization criteria. This is the subject of multiagent scheduling (Agnetis et al., 2014). In addition, machines in a scheduling problem can achieve different optimization criteria. For instance, in the case of truck dispatching in open-pit mines, shovels try to maximize their production, so a schedule with shovel operations with makespan (*Cmax*) as optimization criteria would be more appropriate. However, trucks try to minimize their cost, so a schedule of truck operations should try to reduce the processing time of the operations. Therefore, a truck schedule with minimization of the *Total Weighted Completion Time* as optimization criterion would be more appropriate.

Table 3.5: Overview on investigated scheduling methods.

Authors	Type Problem[11]	Domain	Decision Making[12]	Structure[13]	Agent Representation	Interaction	Cf. Pag.
Chargui et al. (2019)	S, R	Port container terminals	Centralized	Hierarchy*	Hybrid	Messages	57
Lopes Silva et al. (2019)	S	Unrelated Parallel Machine with Sequence-Dependent Setup Times	Distributed	Society*	Function	Messages	59
Whitbrook et al. (2018)	S, R	Unmanned aerial vehicles	Distributed	Society*	Physical	Combinatorial Auctions / Messages	55
Lin et al. (2018)	R	Well projects in oil and gas industry.	Distributed	Society*	Hybrid	Combinatorial Auctions	59
Seitaridis et al. (2020)	S, R	Charge of electric vehicles in multiple stations	Distributed	Market*	Hybrid	Auctions	58
Madhyastha et al. (2017)	S	Dial-a-Ride Problem	Distributed	Market*	Physical	Auctions	55
Skobelev et al. (2018)	R	Unmanned aerial vehicles	Distributed	Society*	Physical	Messages	56
Gehlhoff and Fay (2020)	S, R	Manufacturing	Distributed	Market*	Physical	Auctions	60
Wang et al. (2018)	S, R	Flexible Job Shop	Centralized	Hierarchy*	Hybrid	Messages	59
Granichin et al. (2013)	R	Truck cargo	Distributed	Market*	Physical	Auctions	54
Feng et al. (2015)	S	Multi-satellite mission	Distributed	Market*	Physical	Auctions	55
Martin et al. (2016)	S	Permutation Flow-shop Scheduling	Distributed	Society*	Function	Messages	58

[11] S=schedule, R=Reschedule

[12] Central=one agent generates the schedule, Distributed=More than one agent generate schedules

[13] * No explicit information provided

Traditional centralized methods such as analytical, heuristic, metaheuristic, and distributed methods have been applied to solve scheduling problems. The former ones are well-known methods and the most applied ones, specifically in those problems with single optimization criteria for all jobs. However, they are difficult to apply in real-world situations (Shen et al., 2006), especially to those with a big problem size and a set of jobs with different optimization criteria.

Agent-based technology has been demonstrated to be an alternative solution to scheduling problems, regardless the complexity of the problems, and if the jobs have different optimization criteria. This is the result of the specific characteristics of agent technology such as flexibility, parallel processing, decomposition, and reactive and proactive behavior (Shen et al., 2006). The latter does not mean that MAS are the best alternative to apply to all kinds of scheduling problems. For instance, in a static problem, with centrally available data and appropriate size, traditional methods are more suitable than MAS. However, most real-world scheduling problems require a method to generate solutions in a flexible, adaptive and robust way. It is often necessary to quickly generate valid schedules, e.g., when a machine has a failure or new jobs are arriving. Sometimes there is not enough time to generate an entirely new schedule, so updating a part of the schedule could be enough.

As shown in Section 3.4, there are several multiagent systems for scheduling problems. Most of them are applied to manufacturing (Gehlhoff and Fay, 2020; Lopes Silva et al., 2019; Martin et al., 2016; Wang et al., 2018) and vehicles (Chargui et al., 2019; Granichin et al., 2013; Madhyastha et al., 2017; Seitaridis et al., 2020; Skobelev et al., 2018; Whitbrook et al., 2018). All works mention the good performance of MAS in generating schedules and, in some of the them, the MAS positive reaction when an unforeseen event occurs (Chargui et al., 2019; Gehlhoff and Fay, 2020; Lin et al., 2018; Seitaridis et al., 2020; Skobelev et al., 2018; Wang et al., 2018; Whitbrook et al., 2018). In addition, some works combine MAS with other techniques such as genetic algorithms (Feng et al., 2015), reinforcement learning (Lin et al., 2018; Lopes Silva et al., 2019; Martin et al., 2016), and metaheuristics (Lopes Silva et al., 2019; Martin et al., 2016). These results demonstrate the applicability of MAS to complex scheduling and rescheduling problems.

3.5.2 Truck Dispatching in Open-pit Mines as a Scheduling Problem

In order to formulate the truck dispatching problem in open-pit mines as a scheduling problem, it is necessary to analyze it from the perspective of jobs, resources, and the scheduling objectives pursued.

From the perspective of jobs, "general jobs" are known before the shift starts and are described in the production plan. This plan points out the amount of material that must be extracted by shovels and transported by trucks to unloading points (crusher or stockpiles). Each "general job" must be split into many "small jobs" in which a shovel extracts an amount of material and a truck transports those materials to an unloading destination. In this context, a difference arises in comparison to common scheduling problems since it is not possible to know the number of necessary "small jobs". This happens because the processing times of the "small jobs" depend on truck traveling times, and this depends on the changing location of the trucks in the mine.

From the perspective of resources, trucks and shovels can be considered as machines in which some job operations will be performed, and crushers and stockpiles as resources with constraints to be used. In shovels, extraction is the only operation that must be performed. Shovels can be considered as unrelated machines in parallel with different speeds (Patterson et al., 2017). Regarding trucks, the operations that must be performed are the loaded travels between a shovel and an unloading point, the empty travels from an unloading point to a shovel and, the unloading of materials at a crusher or stockpile. Since the travels of a truck depend on the previous trip, this can be considered as sequence-dependent setup times.

Regarding crushers and stockpiles, these resources have a limitation on the number of trucks that can unload materials simultaneously. Regarding the scheduling objective of the problem, shovels and trucks have different objectives for their schedules. On the one hand, shovels want to achieve the target pointed out in the production plan and, after achieving it, shovels want to maximize their production. Therefore, a shovel schedule should consider a makespan (*Cmax*), i.e., minimize the completion time of the last job. A minimum makespan usually implies a good utilization of the machine (Pinedo, 2008). On the other hand, the objective of trucks is to minimize their costs. In this case, it is possible to use a weight to represent the cost of operation (travel times). Therefore, a minimization of the Total Weighted Completion Time should be the scheduling objective for the truck schedules.

Regarding rescheduling in a completely reactive approach, since no schedules are used, the assignments are dispatched by applying criteria. The solution quality is poor due to the myopic view of the dispatching rules and the difficulty of predicting system performance (Ouelhadj and Petrovic, 2009). This approach is the most used in the solutions proposed for truck dispatching in open-pit mines. In contrast, the predictive-reactive approach reviews schedules (and updates them if necessary) allowing to generate better system performance because the adjustments in the schedules are made considering a global view of the problem. The literature mentions two strategies for rescheduling: schedule repair and complete rescheduling. In the schedule repair strategy, the current schedule is modified, and in the complete rescheduling strategy, the schedules are regenerated from scratch. At first sight, complete rescheduling might reach better results than schedule repair because all the schedules are regenerated considering the new conditions, unlike schedule repair, which only considers a subset of the schedules (the affected ones) under the new conditions. However, schedule repair might be performed in a shorter time because only a subset of schedules is considered.

In MASs for rescheduling, the agents might review the schedules and repair or regenerate them by interacting with each other. The agents' local autonomy allows them to respond locally to the environmental variations, increasing the robustness and the flexibility of the system (Ouelhadj and Petrovic, 2009). The agent autonomy and reactivity make MAS more suitable for rescheduling problems than centralized systems.

Multiagent systems provide reliable, maintainable, flexible, robust, and stable architectures to solve rescheduling problems. Autonomous architectures are fully distributed, in which autonomous agents interact with each other to solve problems. A mediator architecture extends autonomous architectures adding mediator agents. Which architecture is more suitable for a problem depends on the characteristics of the problem. For instance, in a problem with too many agents, in which the number of interactions among the agents is high, a mediator architecture might be more suitable than an autonomous architecture, in order to decrease the number of interactions and improve the performance of the system.

3.5.3 MAS Design

The discussion about the MAS design is split into three aspects: The organizational structure

of the MAS, the design of the agents, and the interactions among the agents.

In the literature, the most frequently applied structures are hierarchical, team-centric, coalitions and markets due to their flexibility, easy implementation and ability to produce positive effects (Horling and Lesser, 2004). Hierarchical structures are well-known and simple structures applied to many MASs. They are specially appropriate for problems in which a natural decomposition makes sense since they allow for parallel execution. However, central decision-making can become a bottleneck affecting the agents at lower levels. In addition, its structure does not allow for flexibility. These situations also affect holarchies and federations.

In the case of coalition and congregation structures, agents can take advantage of more autonomy because they do not need to cede authority to other agents. However, difficulties can arise when the population of agents is too large or there is limited time to generate coalitions or congregations (Horling and Lesser, 2004). Compared to hierarchies and holarchies, interactions are more free in congregations and coalitions and the agents have more liberty to enter and leave the structure.

In market structures, the interactions of agents are based on negotiation mechanisms such as auctions and bargaining. The agents compete to maximize their utilities by achieving individually beneficial resource allocations. Thus, a market structure is inappropriate for maximizing the agents' social welfare because they fail to establish an equal distribution of resources among them (Berndt, 2015). However, if the agents have a cooperative behavior instead of a competitive one, this behavior enables fairness in the market.

In the reviewed works, there is no explicit information described on the structure used. However, after analyzing each work, it is observed that the market structure is mostly used. Those works that used a market structure use auctions as the main interaction mechanism among the agents. It is important to mention that no structure is better than the others, and that a selection of them depends on the requirements, objectives, and participants of the MAS.

Regarding what the agents represent, two approaches are used in the scheduling domain: The functional decomposition approach and the physical decomposition approach (Shen et al., 2006). In the functional decomposition approach, agents represent functions or procedures. For instance, agents encapsulate transportation management, scheduling procedures, and planning processes. This approach is useful when the agents must encapsulate the

functionality of organizations (such as departments or companies) or systems (such as MRP, databases or legacy systems). In the physical decomposition approach, agents represent real-world entities such as machines, orders, workers, or vehicles. There is an explicit relation between agents and physical entities. Using this approach enables the systems to be modeled closer to reality. In addition, agents can manage the information in a better manner, e.g., sharing it or using its specific data. However, communication overhead could arise if the system is too complex.

From the perspective of a scheduling problem in a MAS with agents using a functional approach, the schedules are generated by one agent that encapsulates the procedure to generate schedules. This agent receives information from other functionality agents, such as inventory management and production management to generate the schedule and inform the others. This approach is quite similar to a centralized approach. In the works of Lopes Silva et al. (2019) and Martin et al. (2016), agents follow a functional decomposition since each agent encapsulates a metaheuristic to solve combinatorial problems. A MAS with agents that follow a physical decomposition, the agents interact with each other to generate local schedules using their specific information and pursuing their own objectives. Then the agents can share the information related to their schedules in order to improve it or solving conflicts. The works of Whitbrook et al. (2018), Madhyastha et al. (2017), Skobelev et al. (2018), Gehlhoff and Fay (2020), Granichin et al. (2013) and Feng et al. (2016) use the physical decomposition approach where each agent represents a real-world entity.

A third alternative is a MAS with agents that follow a functional decomposition and agents that follow a physical decomposition. In this setting, agents that represent physical devices provide their specific information to an agent that encapsulates the procedure to generate schedules (the functional one). For instance, physical agents provide information related to their capacities, processing times, status and a functional agent generates the schedule using this information. The works of Chargui et al. (2019), Lin et al. (2018), Seitaridis et al. (2020) and Wang et al. (2018) use this approach.

Regarding interactions among agents, and from the perspective of scheduling problems, message sending and bidding are the mostly used interaction mechanisms. While message sending is more common in MAS with a more rigid structure such as hierarchy and holarchy, bidding is frequently used in MAS based on market structures. In the context of bidding, the

contract-net protocol or its modified version is the most commonly used for scheduling problems (Shen et al., 2006). However, if there are too many participants, the communication effort increases and the agents could spend more time managing messages than performing their actual work. To resolve this situation, some useful approaches are sending offers to a limited number of nodes and using agents that anticipate offers.

By definition, MAS actions establish concurrent negotiations. This means that several negotiations are happening simultaneously. Therefore, an agent can participate in more than one negotiation at the same time. In the case of negotiations that are based on the contract-net protocol, an agent can send proposals to different auctioneers. Since in the contract-net protocol the winner reserves its resources, it raises the possibility of losing a better offer. As a consequence of this, the entire scheduling solution could be of lower quality. However, one of the main advantages of MAS is the process parallelization and supporting concurrent negotiations must therefore be an issue to consider in the design of a MAS. In the reviewed works, not one mentions this situation.

Bargaining could be a useful mechanism to use in scheduling problems, especially when it is necessary to perform rescheduling, especially when a major unforeseen event occurs. In this case, and in order to avoid generating a new schedule from scratch, the use of bargaining as a one-to-one negotiation mechanism allows for the updating of local schedules. From the works reviewed, all of those which use a market structure use bidding as negotiation (Feng et al., 2015; Gehlhoff and Fay, 2020; Granichin et al., 2013; Madhyastha et al., 2017; Seitaridis et al., 2020) as a mechanism to generate an initial schedule and/or for updating a schedule (rescheduling).

3.6 Conclusions

This chapter has presented multiagent technologies and scheduling as components of a solution for the truck dispatching problem in open-pit mines. The chapter starts with a description of intelligent agents, their environments, and how they can communicate with each other to compose a multiagent system. Then some special issues related to scheduling are presented. Finally, representative results related to the application of multiagent systems to scheduling problems, especially to those where scheduling of vehicles are considered, are described followed by a discussion.

The first conclusion is that agent technology is an appropriate distributed approach to solve scheduling problems and therefore, should be applied to solve the truck dispatching problem. This is supported by the vast literature on the successful application of MAS to scheduling problems, including those related to vehicle scheduling due to the advantages offered by the agent technology: modularity, flexibility, scalability, and robustness. However, a successful application of the MAS will also depend on an appropriate design of the MAS in terms of system architecture, agent representation, and interaction mechanisms.

The second conclusion is related to the question how a MAS and its agents must be organized in order to solve the truck dispatching problem in open-pit mines from the perspective of a scheduling problem. First, the agents must represent physical devices such as trucks and shovels. In this way, the problem is modeled closer to reality, agents can follow specific objectives and they can use their specific information to make decisions. Secondly, the structure of the MAS must follow a market structure since it allows for a more flexible, decentralized, and autonomous behavior of the agents.

The third conclusion is related to applying the contract-net protocol as an interaction mechanism among agents to generate the initial schedules. The contract-net protocol is a suitable mechanism due to the following characteristics:

- Scalability: It does not need modifications if the number of agents increases that take part.
- Flexibility: The protocol can be used for different negotiations. For instance, shovels can play the role of initiators to negotiate with trucks on loading times and to generate schedules in this way. Trucks can also play the role of initiators to negotiate with other trucks, to exchange assignments and to decrease their travel times this way.
- Anytime algorithm: The contract-net protocol process may be stopped before the optimal solution is reached, and the best solution found up to then is returned. This characteristic is very important in a stochastic and dynamic environment.

In addition, the contract-net protocol is a well-known method that has been successfully applied in many scheduling problems. However, it is necessary to deal with the concurrency problem to take full advantage of parallel processing.

Finally, regarding rescheduling, both strategies, schedule repair and complete rescheduling

are suitable, and their application depends on the type of event that triggers the rescheduling. For instance, when a truck failure occurs, scheduling repair might be more suitable because the shovel affected by the cancelation of the loading assigned to the broken truck might be negotiated again. In this case, the affected shovel might start negotiations for the canceled assignments to reach its production level quickly again and without too many changes regarding the initial schedules. In the case of a shovel failure, the complete rescheduling strategy would be more suitable because trucks are more affected. Regarding the best moment for rescheduling, a hybrid policy is suitable because it covers more dynamic and stochastic situations. For instance, machine failures can be faced with an event-driven policy because its occurrence is unscheduled. In the case of machine delays, a periodic policy can be applied: If in specific intervals of time a delay affects a schedule, rescheduling should be initiated.

4 THE MULTIAGENT SYSTEM FOR TRUCK DISPATCHING IN OPEN-PIT MINES

The previous chapters have introduced the truck dispatching problem in open-pit mines, the requirements that must be satisfied for an improved solution in order to overcome the weaknesses of current solutions. In addition, and as part of the solution proposed in this thesis, the foundations of the multiagent technology and scheduling theory were presented.

The developed MAS uses and adapts suitable concepts and approaches presented in the previous chapter. The agents in the MAS are organized in a market structure and interact with each other to generate schedules, which constitute a more precise solution than current solutions to solve the truck dispatching problem in open-pit mines.

This chapter introduces the multiagent system for the truck dispatching problem in open-pit mines (MAS-TD), which allows for an autonomous and cooperative environment to generate a more efficient truck dispatching. Firstly, Section 4.1 provides an overview of the MAS-TD. Secondly, Section 4.2 describes the interaction protocol and provides a formal analysis of it as well as an analysis of the quality of the schedules generated by applying the protocol. Section 4.3 describes how the MAS-TD addresses the dynamics of the environment. Section 4.4 explains the decision-making of the agents and provides the performance analysis. Section 4.5 provides details of the implementation of the MAS-TD on a simulated platform. Section 4.6 reports on the experimental evaluation of the proposed MAS-TD. Section 4.7 presents a discussion on the design of the MAS-TD, the proposed protocol to manage the concurrent negotiations, and the performance of the system. Finally, Section 4.8 concludes the chapter.

4.1 An Overview of the Multiagent System for Truck Dispatching in Open-pit Mines

The objective of the MAS-TD is to improve the efficiency of the material handling process in open-pit mines. For more efficient material handling, the operations of the involved equipment must be organized as close to the optimum as possible. In order to organize the operations of the equipment, the MAS-TD generates schedules for each equipment considering to achieve the target of the production plan at the minimum cost. These schedules must guarantee efficient operation and synchronization of the equipment involved in the material handling process.

In order to achieve the objective of improving the efficiency of the material handling process in open-pit mines, the MAS-TD design must consider different aspects such as the structure to organize the agents, the agent representation, the interaction between agents, and the decision-making process of the agents. Regarding the structure, the main idea is to avoid centralized components in order to fully exploit the advantages of multiagent technology such as robustness, parallel processing, and flexibility. For this reason, a market structure (cf. 3.2.4) is used due to flexibility and by allowing for negotiations among the agents.

Regarding agent representation, the agents follow a physical representation of the entities involved in the material handling process in an open-pit mine, i.e., trucks, shovels, and unloading points are represented by agents. Table 4.1 shows the agents considered in the MAS-TD, their properties, and objectives.

Table 4.1: Agent description.

Agent	Real-world representation	Objective	Properties
truckAgent	Trucks	Create a schedule of the activities of the truck at minimum cost	Capacity, loaded velocity, empty velocity, spotting time and unloading time, layout of the mine.
shovelAgent	Shovels, Front Loaders	Create a schedule of the activities of the equipment that represents considering its target in the production plan	Capacity, dig velocity, load velocity and the destination of extracted material
unloadingPointAgent	Crushers, Stockpiles, Waste dumps	Create a schedule of the activities of the equipment that it represents	Number of trucks un-loading simultaneously

In order to generate the schedules, the agents negotiate among them by using an adapted contract-net protocol (cf. 4.2). In this protocol, a *shovelAgent* plays the role of the initiator and starts a negotiation process by sending a *call-for-proposal* message to *truckAgents*. In the *call-for-proposal* message, the *shovelAgent* offers the time when it will be available to load a truck. The *truckAgents* play the role of participants in the protocol, evaluate the offer sent by the *shovelAgent*, and respond with *propose* or *refuse* messages. Then, the *shovelAgent* evaluates the proposals received, selects the best one, and informs the participants about the result. In the case that a negotiation process finishes successfully, the *shovelAgent* and the *truckAgent* that win the bidding add the task to their schedules. This negotiation process is repeated until the *shovelAgent* generates a schedule that accomplishes the target of the operational plan or until the schedule reaches the end of the shift. The *shovelAgents* in the system start this negotiation process in parallel, therefore the protocol must manage concurrent negotiations.

The initial schedules are generated by the agents before the shift starts. When the shift starts, the equipment executes the operations according to the order and time established in their schedules. However, during the shift, events can affect the equipment. In this case, the agents must interact with each other to update the schedules.

The MAS-TD runs on PlaSMA (Warden et al., 2010), which is a platform to evaluate and simulate dynamic transport scenarios. PlaSMA uses the FIPA compliant Java Agent Development (JADE) framework (Bellifemine et al., 2007) as the underlying agent management platform.

4.2 Contract-Net Protocol with Confirmation Stage

In the previous section, it was mentioned that the agents interact with each other to generate the schedules using an adaptation of the well-known contract-net protocol (Smith, 1980). The adaptation addresses the concurrency in the negotiations. This Section describes the adaptation of the original contract-net protocol. The following sections give an overview of the protocol, a formal analysis, and an analysis of the quality of the schedules generated by using this protocol.

4.2.1 Overview of Protocol

In the context of the MAS-TD for truck dispatching in open-pit mines, the original contract-net protocol works as follows:

1. A *shovelAgent* initiates a negotiation process with *truckAgents* sending them a *call-for-proposals* message. The *call-for-proposal* points out the time when the shovel is available to load a truck.
2. When a *truckAgent* receives a *call-for-proposal*, it must decide whether or not to send a proposal. To do this, it asks the *unloadingPointAgent* about the prospective waiting time and, with the information of the *call-for-proposal*, it determines whether the offer fits in its schedule. If yes, it prepares a proposal, sends it to the *shovelAgent*, and waits for the answer. If not, it sends a *refuse* message to the *shovelAgent*.
3. When the *shovelAgent* receives the messages from all the participants (*refuse* or *propose*), or when the deadline is expired, it looks for the best proposal.
4. Then, the *shovelAgent* sends an *accept-proposal* message to the *truckAgent* that proposed the best proposal and sends a *reject-proposal* message to the other *truckAgents* that sent a *propose* message.
5. After this, the *truckAgent* that receives the *accept-proposal* message and the *shovelAgent* that initiated the negotiation add all activities to their schedules that must be performed and the negotiation is finished.

The *shovelAgent* starts a new negotiation process by applying this protocol until the target of the production plan is met or until it reaches the end of the shift. In the case that a *shovelAgent* ends a negotiation without a winner, it starts a new negotiation adding one minute to the loading time offered in the last negotiation. In this way, the *shovelAgent* avoids repeating the unsuccessful negotiation.

The adapted Contract-Net Protocol

The previous protocol has some limits. Given that the agents work in parallel, several negotiations are performed concurrently. Therefore, a *truckAgent* may receive several *call-for-proposal*'s. If the *truckAgent* has sent a proposal in a negotiation, and it is still waiting for the answer from a *shovelAgent*, the other received *call-for-proposal′* s will be rejected. In

this context, it could happen that the *truckAgent* refuses a *call-for-proposal* that is a better option than the *call-for-proposal* answered previously. This problem is also called "The Eager Bidder Problem" (Schillo et al., 2002).

One way to tackle this problem is by applying the contract-net protocol in a sequential manner, i.e., a single negotiation at a time. Sequencing several negotiation processes may make the participants miss some better contracts since its resources are committed in another contract (Aknine, Pinson, and Shakun, 2004). In addition, the run time of the entire system could increase. Another way is to allow the agent to break a contract awarded if the agent receives a better offer. As a consequence of this, the initiators must start again a negotiation process for those broken commitments (Aknine et al., 2004).

To enable an agent to manage concurrently several negotiation processes, the original contract-net protocol is adapted by including a confirmation stage: when a *shovelAgent* finalizes the evaluation of the proposals, it sends a *requestConfirmation* message to the *truckAgent* with the best proposal, requiring the confirmation of the sent proposal. The *truckAgent* that receives the message accepts the confirmation if it has not sent a better proposal to another *shovelAgent*. Otherwise, it refuses the *requestConfirmation* message. To compare the required proposal for confirmation and the other proposals sent, each *truckAgent* maintains a hash map with the *call-for-proposal's* received and their corresponding proposals.

If the *shovelAgent* receives a *refuse* message of the *requestConfirmation* message, it sends a new *requestConfirmation* message to the *truckAgent* that sent the second-best proposal. The *shovelAgent* will continue sending *requestConfirmation* message until it receives an acceptance of the confirmation or until there are no more proposals. In this way, the *truckAgent* might decommit a previous proposal sent. The communication primitives used by the *shovelAgent* and *truckAgent* are *requestConfirmation*, *acceptConfirmation* and *refuseConfirmation*. Figure 4.1 depicts the interaction between the agents using the contract-net protocol with confirmation stage. Table 4.2 shows a schedule example for a truck generated by the agents by applying this protocol.

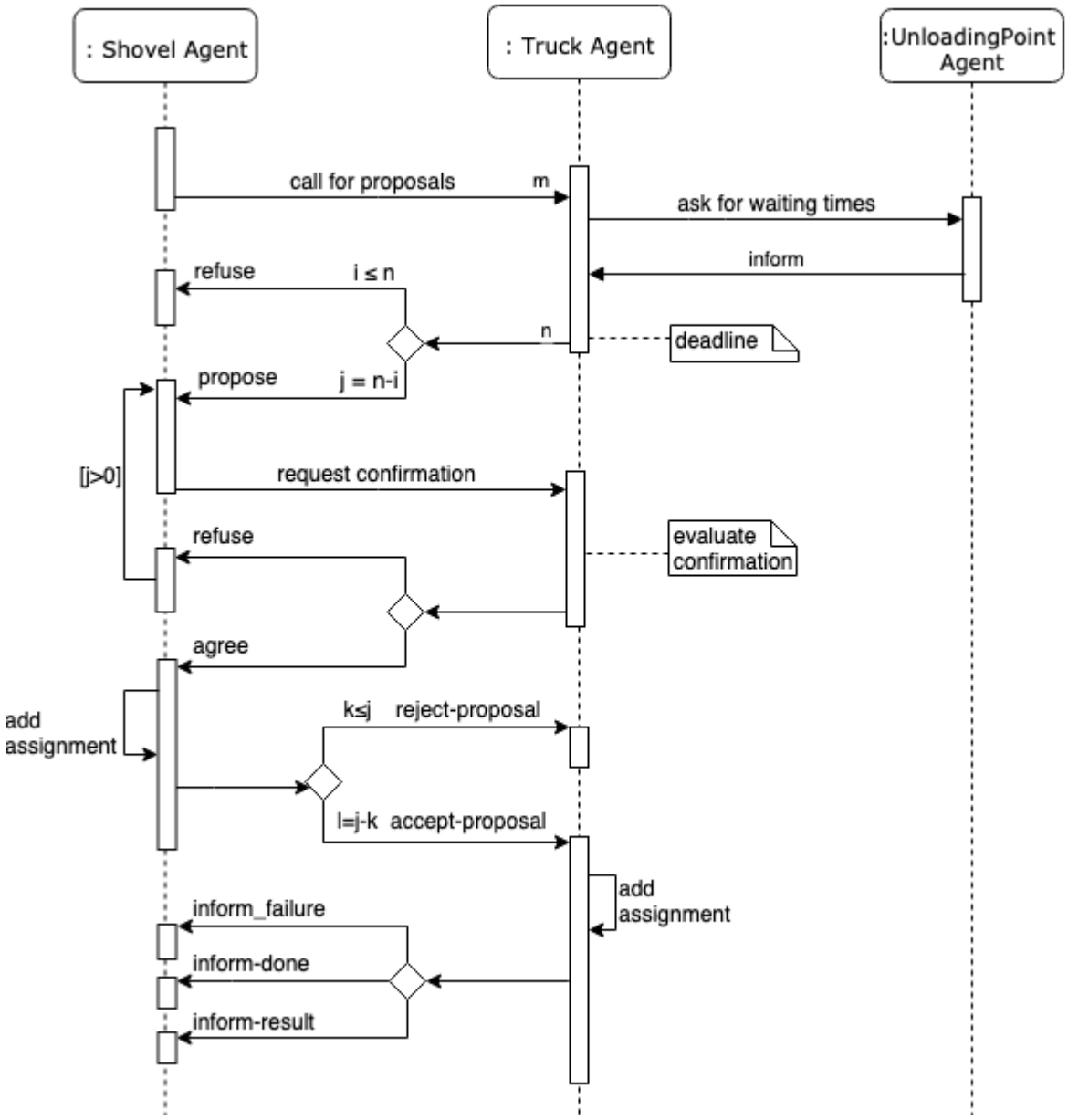

Figure 4.1: The interaction between the agents using the contract-net protocol with the confirmation stage.

Table 4.2: Example of schedule created for a truck.

Assignment	Destination	Start Time of the Trip	Arrival Time	Start Time of the Spotting	Start Time of the Loading or Unloading	End Time of the Assignment
0	Shovel.01	00:47:01	01:20:23	01:20:23	01:21:36	01:23:12
1	WasteDump.02	01:23:12	01:32:33	01:32:33	01:32:33	01:33:23
2	Shovel.04	02:10:39	02:18:47	02:18:47	02:20:00	02:21:12
3	WasteDump.03	02:21:12	02:26:38	02:26:38	02:26:38	02:27:28
4	Shovel.04	02:27:28	02:31:37	02:31:37	02:32:50	02:34:02
5	WasteDump.03	02:34:02	02:39:28	02:39:55	02:39:55	02:40:45

The Dynamic View of the Protocol

The state transitions in the negotiation process between a *shovelAgent* and *truckAgents* are depicted in Figure 4.2. The process starts when a *shovelAgent* sends a *call-for-proposal* message to *truckAgents*. After sending the *call-for-proposal*, the process passes to the first state "Evaluating CFP". In this state, the *truckAgents* evaluate the received *call-for-proposal* and decide on sending a proposal or a *refuse* message as the answer. Meanwhile the *shovelAgent* is waiting for the messages from the *truckAgents*.

When a *shovelAgent* receives a message from a *truckAgent*, the negotiation process passes to the state "Receiving proposals". In this state, the *shovelAgent* receives and stores the proposals sent by the *truckAgents* and keeps a count of the received messages from the *truckAgents*. The negotiation keeps in this state until a deadline expires or when the *shovelAgent* has received all the messages (*propose* or *refuse*) from the *truckAgents*. In the "Receiving proposals" state two situations can happen: having received proposals or not. If it has not received any proposals, the negotiation process finishes without a contract. If it has received proposals, the negotiation process passes to the state "Selecting best proposal".

In the state "Selecting best proposal", the *shovelAgent* selects the best proposal and sends a message with the performative *requestConfirmation* to the *truckAgent* that sent the best proposal, and the negotiation process passes to the state "Waiting for answer".

In the state "Waiting for answer", three situations can happen: receiving a message with the performative *acceptConfirmation*, receiving a message with the performative

refuseConfirmation, or a deadline expires. In the first case, the negotiation process passes to the state "Sending results". In the "Sending results" state the *shovelAgent* sends a *reject-proposal* message to those *truckAgents* that were not awarded, and the negotiation process ends with a contract. In the second and third cases, the negotiation process passes to the "Removing proposal" state. In this state, the *shovelAgent* removes the proposal from its storage. If there are more proposals, the negotiation process passes to the state "Selecting best proposal" and repeats the cycle. If there are no more proposals, the negotiation process ends without a contract.

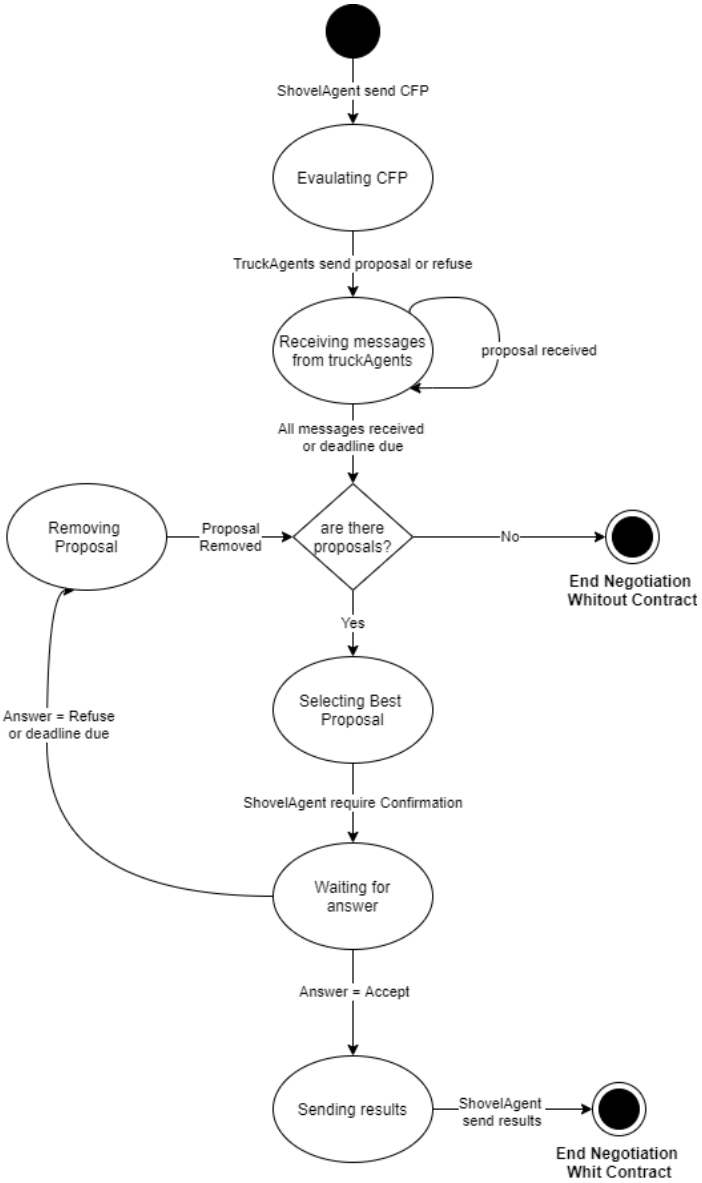

Figure 4.2: State machine diagram.

4.2.2 Negotiation Algorithms

As described previously, in the contract-net protocol with confirmation stage there are two classes of agents that interact with each other to generate the schedules: the *shovelAgents* and the *truckAgents*. Since they work in parallel, the algorithms that implement their behaviors must take this situation into account. Algorithm 1 implements the initiator behavior of the *shovelAgent* and algorithm 2 implements the participant behavior of the *truckAgents*.
The algorithms are designed to receive messages and manage them at any point in time. In addition, some Boolean variables (terminated with the word flag) are used to control the execution of some parts of the code.

Initiator Behavior Algorithm

The initiator behavior (Algorithm 1) receives an offer as an argument, which will be used later. The algorithm starts with the initialization of the variables and then executes the procedure *sendCFP* with the offer received as an argument. This procedure determines the participants (and the number of participants, which is stored in the variable *numberOfParticipants*), creates and sends the *call-for-proposal* message to the participants.
After execution of the procedure *sendCFP*, a loop starts, which terminates when the negotiation process ends (in lines 31, 41 or 49). In each iteration, the reception of a message is checked and retrieved with the procedure *receiveMessages*. The message received is stored in the variable *msg*.
If the message received corresponds to a *propose* message, the proposal is added to a list of proposals received (called *receivedProposals*) and the value of the variable *receivedMessages* is increased, which counts the number of messages received. Each time that a proposal is received, the arrival time and cost proposed are compared with their corresponding maximum and minimum values stored at that moment. The maximum and minimum values will be used later in the procedure *evaluateProposals*. If the message received is a *refuse* message, only the value of *receivedMessages* is increased.
The if block (lines 29-37) executes the code related to the selection of the best proposal. It is executed only once (controlled by the variable *confirmationFlag*) if a deadline expires or all the messages from the participant are received (*receivedMessages==numberOfParticipants*).

Algorithm 1 Initiator Behavior

```
1:  procedure INITIATOR(offer)
2:      receivedProposals, evaluatedProposals ← ∅
3:      confirmationFlag, evaluationFlag ← false
4:      receivedMessages, numberOfParticipants ← 0
5:      maxCost, maxArrival ← Very low value
6:      minCost, minArrival  ← Very high value
7:      SENDCFP(offer, numberOfParticipants)
8:      while true do
9:          msg ← RECEIVEMESSAGE
10:         if msg = proposal then
11:             receivedProposals.add(msg)
12:             receivedMessages ← receivedMessages + 1
13:             if msg.arrivalProposed > maxArrival then
14:                 maxArrival  ← msg.arrivalProposed
15:             end if
16:             if msg.arrivalProposed < minArrival then
17:                 minArrival ← msg.arrivalProposed
18:             end if
19:             if msg.costProposed > maxCost then
20:                 maxArrival ← msg.arrivalProposed
21:             end if
22:             if msg.arrivalProposed < minArrival then
23:                 minArrival ← msg.arrivalProposed
24:             end if
25:         end if
26:         if msg = refuse then
27:             receivedMessages ← receivedMessages + 1
28:         end if
29:         if evaluationFlag and (receivedMessages  = numberOfParticipants or deadline
            expires)  then
30:             if receivedProposals = ∅ then
31:                 return
32:             else
33:                 evaluatedProposals ← EVALUATEPROPOSALS(receivedProposals,
                    offer, maxCost, minCost, maxArrival, minArrival)
34:                 QUICKSORT(evaluatedProposals)
35:                 confirmationFlag ← true
36:                 evaluationFlag ← false
37:             end if
38:         end if
```

```
39:         if confirmationFlag then
40:             if evaluatedProposals = ∅ then
41:                 return
42:             else
43:                 SENDCONFIRMATION(evaluatedProposals.first)
44:                 confirmationFlag ← false
45:             end if
46:         end if
47:         if msg = acceptConfirmation then
48:             SENDREJECTIONS
49:             return
50:         end if
51:         if msg = refuseConfirmation or deadline expires then
52:             evaluatedProposal.remove
53:             confirmationFlag ← true
54:         end if
55:     end while
56: end procedure
```

If there are no proposals, the procedure ends without a contract (line 31). Otherwise, the procedure *evaluatedProposals* is executed with the *receivedProposals*, the offer, and the minimum and maximum values of the arrival time and cost proposed as arguments. Details of this procedure are in Section 4.4.1.

Next, the proposals evaluated are sorted by calling of the procedure *quickSort*. Due to the number of proposals received for the shovel is not high (the biggest mine works with no more than 200 trucks) *quickSort* is a suitable algorithm to sort the proposals. The best proposal will be the first proposal in the ordered *evaluatedProposals* list. Then, the *confirmationFlag* (line 36) becomes true to allow for the execution of the confirmation block (lines 39-46) and the *evaluationFlag* becomes false to avoid the execution of the block again.

The confirmation block is executed *receivedProposals* + 1 times. When there are no more proposals, the negotiation process ends (line 41) without a contract. If there is a proposal, the procedure *sendConfirmation* is executed with the first proposal stored in *evaluatedProposals* as an argument (line 43). To avoid repeating this block, the *confirmationFlag* becomes false. If an *acceptConfirmation* message is received, the procedure *sendRejections* is executed, which sends *reject-proposal* messages to the participants that were not awarded, and the

procedure finishes with a contract (line 49). If a *refuseConfirmation* message is received or the deadline expires, the first best proposal from the *evaluatedProposals* list is removed and the *confirmationFlag* variable becomes true to allow for executing one more time the confirmation block in the next iteration.

During the execution of this algorithm, the evaluation of the proposals and the sorting of them are the most cost-intensive operations. The procedure *evaluatedProposals* is done in *O(n)* time and space, where *n* corresponds to the number of proposals received. This algorithm is described and analyzed in Section 4.4.1. In the case of the *quickSort*, it runs in *O(n log n)* time in its best and average cases and runs in $O(n^2)$ in its worst case and is stable sort with space complexity of *O(n)*. Therefore, the algorithm 1 is done in $O(n^2)$ (due *quickSort*) in its worst case and its space complexity is *O(n)*.

Participant Behavior Algorithm

The participant behavior (algorithm 2) is a cycle that runs indefinitely and reacts when a message is received or some conditions are accomplished. The behavior starts initializing the variable *currentCFPs* (line 2), which is a list that stores the *call-for-proposals* of the negotiation process in which the agent is taking part. Then, it starts an infinite loop. In each iteration, the reception of a message is checked and retrieved with the procedure *receivedMessage*.

If the message received is a *call-for-proposal*, the function *DecideSendProposal* is executed. This function is described in Section 4.4.2. If the function returns true it means that the agent decides to send a proposal, therefore stores the *call-for-proposal* in the *currentCFPs* and sends a proposal (lines 7-8). In case the function returns false, the agent sends a *refuse* message (line 10).

If the message received is a *requestConfirmation*, the algorithm checks if the idle time provided in the *requestConfirmation* message is higher or equal than 1. This condition is necessary to avoid a special situation: if a *shovelAgent* receives only *refuseConfirmation* messages, it must finalize the negotiation process without a winner, and it starts a new negotiation process offering a new loading time (last offered time + one minute). However, it could happen that the negotiation ends again without a winner, and the *shovelAgent* would start another negotiation process adding another minute to the offered loading time. This

situation would generate idle time in the schedule of the shovel. To avoid this, the *truckAgents* consider the shovel idle time from the last loading.

Algorithm 2 Participant Behavior

```
1:  procedure PARTICIPANT
2:      currentCFPs ← ∅
3:      while true do
4:          msg ← RECEIVEMESSAGE
5:          if msg = CFP then
6:              if DECIDESENDPROPOSAL(msg.offer)=true then
7:                  currentCFPs.add(msg)
8:                  SENDPROPOSAL
9:              else
10:                 SENDREFUSE
11:             end if
12:         end if
13:         if msg = requestConfirmation then
14:             if idleTime ≥ 1 then
15:                 SENDACCEPT
16:             else
17:                 if ISTHEREBETTERCFP then
18:                     SENDREFUSE
19:                 else
20:                     SENDACCEPT
21:                 end if
22:             end if
23:         end if
24:         if msg = rejectProposal then
25:             currentCFPs.remove(msg)
26:         end if
27:         if msg = acceptProposal then
28:             currentCFPs.remove(msg)
29:             ADDTASKTOSCHEDULE
30:         end if
31:     end while
32: end procedure
```

If the shovel idle time from the last loading is lower than one minute, the algorithm executes

the function *isThereBetterCFP* to check if there is a better *call-for-proposal* received (line 17). This function is described in Section 4.4.2. If the function returns true, the agent sends a *refuseConfirmation* message, otherwise, sends an *acceptConfirmation* message. Nevertheless, if the shovel idle time is higher or equal to one minute, the *truckAgent* must send an *acceptConfirmation* message. In this way, the *truckAgents* prefer to achieve the goals of the production plan instead of decreasing their own cost.

If the participant agent receives a *reject-proposal* message, the *call-for-proposal* related to the proposal rejected is removed from the *currentCFPs* (line 25). If an *accept-proposal* message is received, the related *call-for-proposal* is removed from the *currentCFPs* and the task is added to the schedule.

During the execution of this algorithm, the functions *DecideSendProposal* and the *isThereBetterCFP* are the most cost-intensive operations. With *n* the number of proposal sent by the agent (or the number of negotiation process in which the agent is taking part), the function *DecideSendProposal* is done in *O(n)* time and space. In the case of the function *isThereBetterCFP*, it runs in *O(n)*. Therefore, the algorithm 2 is done in *O(n),* and its space complexity is *O(n).*

4.2.3 Analysis of Protocol

The protocol aims to manage concurrent negotiations. So far, the protocol has been defined informally with diagrams and algorithms. Now it is introduced the formal definitions, theorems, and proofs of the protocol.

Definition 4.1. *(Protocol for concurrent negotiations). Let T be a set of indivisible tasks and Tj denotes a task in T that can be performed by one agent. IA denotes a set of initiator agents where each one is able to initiate a negotiation process for a task Tj , and IAi denotes a specific initiator agent. PA denotes a set of participant agents able to perform a task Tj , and PAp denotes a specific participant agent. It is assumed that |PA| > 1, this ensures that at least two participant are engaged in a bidding. Finally, IAi(Tj) denotes the initiator agent of the negotiation process for the task Tj . The protocol for concurrent negotiations can be viewed as a 7- tuple:*

$$\text{protocol} = (P_{IA}, P_{PA}, \text{evaluatedProposals}, g(\text{evaluatedProposals}), f(T_j), \text{currentCFPs}, c(\text{currentCFPs}, m)) \quad (4.1)$$

- P_{IA} is a set of communication performatives from an IA_i to a PA_p. P_{IA} = {call-for-proposal, requestConfirmation, accept-proposal, reject-proposal}
- P_{PA} is a set of communication performatives from an PA_p to a IA_i. P_{PA}={propose, refuse, acceptConfirmation, refuseConfirmation}
- evaluatedProposals is 1 X |PA| array, where evaluatedProposals[p] containing the proposal of participant p for the task T_j .
- g(evaluatedProposals) is a function of the initiator agents to evaluate the proposals sent by the participant agents.
- $f(T_j)$ is a function that allows a participant to assess their capacities to respond to the notice of request for the performance of the task T_j.
- currentCFPs is 1 X |IA| array, where $currentCFPs_i$ containing the proposal sent by the participant in the negotiation process initiated by the IA_i agent.
- c(currentCFPs, m) is a function of the participant agents to evaluate a confirmation request m sent by an initiator agent.

Theorem 4.1. *Let S be a multiagent system composed of n agents and T a set of tasks announced. A negotiation process engaged by the agents of S using the protocol proposed ends after a finite set of steps.*

Proof. Observation of the automaton in Figure 4.2 describing the possible transitions of a negotiation between an initiator and its participant reveals the following loops and an indefinitely remain at a state:

1. Infinitely remain in “receiving proposal” state.
2. Infinitely remain in “waiting confirmation” state.
3. Sequence loop on states “Selecting best proposal”, “requiring confirmation”, “waiting answer” and “removing proposal”.

In case 1, two conditions avoid to infinitely remain in the “receiving proposal” state. The first

one is the number of messages received. The *shovelAgent* knows the number of possible *truckAgents* that will participate in the negotiation process. If the number of messages (*propose* or *refuse*) received is equal to the number of participants, the negotiation passes to the next state. It assumes that *truckAgents* only sent one proposal and always sent either a *propose* or *refuse* message. The second condition is a deadline. If a *truckAgent* has a failure and cannot send a *propose* (or *refuse*) message, the negotiation process passes to the next state even if not all the *propose* or *refuse* messages have been received.

In case 2, a deadline is included to avoid to infinitely remain in "waiting confirmation" state. If the *truckAgent* which a *requestConfirmation* message was sent to does not respond, and the deadline expires, the negotiation process passes to the next state.

In case 3, to avoid an infinite loop on states "Selecting best proposal", "Requiring confirmation", "Waiting answer" and "Removing proposal", the substate "is there a proposal?" is included. The idea of this loop is to continue sending a *requestConfirmation* message even if a *refuseConfirmation* message is received. However, if there are no more proposals, the cycle is removed with the substate.

Theorem 4.2. *Let S be a multiagent system composed of n agents and T a set of tasks announced. In a negotiation process carried out by the agents of S using the protocol, one winner will be determined at most*.

Proof. Observation of the automaton in Figure 4.2 describing the possible transitions of a negotiation process between an initiator and its participant reveals that only after receiving an *acceptConfirmation* message, the initiator agent sends the results to the other agents and the negotiation process ends. In the algorithm, this is implemented in the If block that manages the reception of an *acceptConfirmation* message (lines 47-50).

Theorem 4.3. *Let S be a multiagent system composed of I initiator agents and P participants. The protocol is done in* $O(P^2)$ *in time and O(P) space by an Initiator and P participants*.

Proof. The initiator sends P *call-for-proposal* messages to the participants. After that, it waits for the answer messages (*propose* or *refuse*). Independent of the time that it takes the

participants to send an answer, the maximum time is limited by a deadline. When the deadline expires, the initiator starts the evaluation of the proposals. The time that it takes for the evaluation of the proposals depends on the requirements of the specific domain, so, for this analysis of the protocol, it is considered that this time is constant *O(1)*. Then, the sort of received proposal is done, in the worst case, *in* $O(P^2)$. After this, the confirmation stage starts. In this stage, in the worst case, the initiator sends *O(P) requestConfirmation* messages. Independent of the time that it takes the participants to send an *acceptConfirmation* or a *refuseConfirmation* message, is limited by the deadline. Therefore, the higher run time is performed by the sorting operation and the number of messages sent depends on the number of participants *P*.

Another aspect to analyze is the quality of the generated schedules by applying the adapted contract-net protocol. To do this, the following definition and theorem are introduced.

Definition 4.2. *(Scheduling model). Let A be a set of agents indexed by i. Each agent manages its own schedule. The scheduling model can be viewed as following:*

- operations = {loading, unloading, loadedTravel, emptyTravel}
- $schedule_{i,t}$ = (operation, startTime, endTime, quantityOfMaterial) | i ∈ A, operation ∈ operations, startTime ∈ $\mathbb{N}$, endTime ∈ $\mathbb{N}$, quantityOfMaterial ∈ $\mathbb{R}$ and quantityOfMaterial >0 . It represents the schedules of agent i at time t.
- $b_{i,t}(schedule_{i,t})$ is the benefit of agent i on its schedule at time t, and it is defined as:

$$b_{i,t}(schedule_{i,t}) = \sum_{x \in schedule_{i,t}} (quantityOfMaterial) \; \forall \; i \in A \qquad (4.2)$$

- $\overrightarrow{b_t}$ = {$b_{1,t}$ ($schedule_{1,t}$), …., $b_{i,t}(schedule_{i,t})$} denoting the individual benefits of agents on their corresponding schedules at time t.

Theorem 4.4. *Let S be a multiagent system composed of n agents and T a set of tasks announced. A monotonical application of the adapted contract-net protocol generates pareto-optimal schedules*.

Proof. To probe the theorem, it is necessary to demonstrate that each successful negotiation by applying of the adapted contract-net protocol generates a schedule that is better for at least one agent, and not worse for any agent.

Let t_0 be the time when a negotiation starts and t_1 be the time when a negotiation ends. To probe the theorem, it is necessary to demonstrate

$$\overrightarrow{b_{t_1}} \geq \overrightarrow{b_{t_0}} \; \forall \; i \; \in A, t_1 > t_0 \tag{4.3}$$

By Theorem 4.2 is known that in a successful negotiation by applying the adapted contract-net protocol, only one winner agent is determined at most. This means that the winner agent (win) and the initiator agent (ini) must add an activity to their corresponding schedules. That means

$$\begin{aligned} schedule_{ini,t_1} = \; & schedule_{ini,t_0} \\ \cup \; & (operation, startTime, endTime, quantityOfMaterial) \end{aligned} \tag{4.4}$$

$$\begin{aligned} schedule_{win,t1} & \\ = \; & schedule_{win,t_0} \\ \cup \; & (operation', startTime', endTime', quantityOfMaterial') \end{aligned} \tag{4.5}$$

with *operation', startTime', endTime'* related to the operations that the truck must perform. Because Definition 4.2 a new *operations* involve a *quantityOfMaterial* ≥ 0, therefore

$$b_{ini,t_1}\left(schedule_{ini,t_1}\right) \geq b_{ini,t_0}(schedule_{ini,t_0}) \tag{4.6}$$

$$b_{win,t_1}\left(schedule_{win,t_1}\right) \geq b_{win,t_0}(schedule_{win,t_0}) \tag{4.7}$$

and therefore Eq. 4.3 is true and thus Theorem 4.4 holds.

4.3 Rescheduling

The efficient and fast reaction to dynamic events that occurs in the environment is one of the main characteristics of multiagent technology. In truck dispatching for open-pit mines, as

described in Section 2.1.3, many events occur in a mine that affects the performance and the availability of the equipment involved in the material handling process. In such a case, the agents must interact with each other to update the generated schedules. The following sections describe how the *shovelAgents*, *truckAgent*, and *unloadingPointAgents* update their schedules when a major event occurs at the mine.

4.3.1 Shovel Rescheduling

Shovels are affected due to different reasons. For instance, the hardness of the workbench affects the digging and extraction operations. Another example is when the electrical power fails; in this case, the shovel becomes unavailable for a while. These and other situations can cause delays or failures in a shovel.

In the case of a shovel failure, the *shovelAgent* that represents it cancels their assignments from the moment when the event occurs and informs the *truckAgents* and *unloadingPointAgents* about this situation. Consequently, the *truckAgents* and the *unloadingPointAgents* that have schedules with assignments related to that shovel must cancel their assignments. In this context, complete rescheduling is a suitable strategy for updating the schedules because of free slots time arise in the schedules of affected *truckAgents*, and they could perform other assignments.

In complete rescheduling, all the agents interact to generate new schedules. In this context, all the agents cancel their assignments from the moment when the event occurs, and *shovelAgents* start new negotiations by applying the adapted contract-net protocol. In this way, the agents reorganize all the schedules considering the new status of the material handling process.

In the case of a shovel delay, a schedule repair strategy is suitable because the schedules can be updated quickly, and only some schedules are affected. The *shovelAgent* that represents it accumulates this delay in a variable. The *shovelAgent* checks if this variable reaches a threshold (e.g., a delay limit). Only if the threshold is reached, the *shovelAgent* cancels the next assignment and start a new negotiation for the canceled assignment with updated information (new loading time).

4.3.2 Truck Rescheduling

As with shovels, trucks are affected by many reasons. For instance, when a tire bursts, the truck is not available for a while. Also, fallen rocks at roads because of blastings can delay trucks. These and other situations can cause delays or failures in trucks.

In the case of a failure truck, the *truckAgent* that represents it cancels the assignments in its schedule from the moment when the event occurs and sends a *failure* message to the *shovelAgents* and *unloadingPointAgents* related to their canceled assignments. The *shovelAgents* and *unloadingPointAgents* that receive the *failure* message cancel all the assignments related to the *truckAgent* that sends the *failure* message. Consequently, free slots arise in the schedule of the *shovelAgents*, and the scheduled production decrease. Besides, trucks can expect less at crusher because the affected truck will not arrive at the crusher. In this context, schedule repair and complete rescheduling strategies can be applied for updating the schedules.

In a schedule repair strategy, only affected *shovelAgents* start negotiation processes by applying the adapted contract-net protocol offering loading times. In this way, the affected shovels look to reach their production targets. In complete rescheduling strategy, all the agents cancel their assignment from the moment truck failure occurs, and all *shovelAgents* start new negotiations by applying the adapted contract-net protocol. In the case of a truck delay, a schedule repair strategy is suitable because the schedules can be updated quickly, and some schedules are affected. The *truckAgent* that represents it accumulates this delay in a variable. The *truckAgent* checks if this variable reaches a threshold (e.g., a delay limit) and, only if the threshold is reached, the *truckAgent* cancels the next assignment. Then, it informs the *shovelAgent* and *unloadingPointAgent* related to the canceled assignment on this situation. Both, the *unloadingPointAgent* and the *shovelAgent* cancel the correspond assignment. Finally, the affected *shovelAgent* starts a new negotiation for the canceled assignment.

4.3.3 Unloading Point Rescheduling

Waste dumps, stockpiles, or crushers are unloading points that can be affected by different events. For instance, rock hardness can generate delays at the crusher or even become a

crusher unavailable. Problems in paths can become waste dumps unreachable for trucks. These and other situations can cause delays or failures at unloading points.

In the case of an unloading point failure (only at crushers), the *unloadingPointAgent* that represents it cancels their assignments and informs the *truckAgents* and *shovelAgents* about this situation. Consequently, *truckAgents* with assignments that involve the affected crusher must change the material destination in those assignments. Therefore, truck travel times change, and many assignments might be affected. In this context, complete rescheduling is appropriated because this event affects many pieces of equipment and their schedules.

In complete rescheduling, all the agents interact to generate new schedules from the moment that the failure at the unloading point occurs. Then, all the *shovelAgent* start new negotiations by applying the adapted contract-net protocol.

In the case of an unloading point delay, a schedule repair strategy is suitable because the schedules can be updated quickly, and some schedules are affected. The *unloadingPointAgent* that represents it accumulates this delay in a variable. The *unloadingPointAgent* checks if this variable reaches a threshold (e.g., a delay limit) and, only if the threshold is reached, the *unloadingPointAgent* cancels the next assignment. Then, it informs the *truckAgent* and *shovelAgent* related to the canceled assignment. Finally, the affected *shovelAgent* starts a new negotiation for the canceled assignment.

4.4 Decision Making

The decision-making process of an agent is one of its most important characteristics. A bad design of the decision-making process could cause the agent to make bad decisions or to take too much time to make them. As a consequence, the performance of the agent could decrease which would affect the entire MAS-TD performance. In the MAS-TD for truck dispatching in open-pit mines, the *shovelAgents* and *truckAgents* make the main decisions in the system. The following section describes the decision-making process of those types of agents.

4.4.1 Decision Making *shovelAgent*

A *shovelAgent* is the agent that initiates a negotiation process using the contract-net protocol with a confirmation stage as described in Section 4.2. In order to decide on the best proposal,

the *shovelAgents* evaluate all the proposals received using a utility function. This function promotes the proposals that propose to start loading on time and with the least time to perform all the activities. More formally:

Definition 4.3. *(Best Proposal-decision) Let Offer* $\in \mathbb{N}$ *represent the time offered by a shovel to start loading. P denotes a set of received proposals* $p_1 ... p_n$. *Each proposal is denoted by* p_i = *(arrivalTime, cost)* $|$ *arrivalTime* $\in$ $\{0 \cup \mathbb{N}\}$, *cost* $\in \{0 \cup \mathbb{N}\}$. *The decision is a multicriteria problem as*

$$Decision = arg_{min}\,(p_i \in P)\,\{U(p_1), .. U(p_n)\} \tag{4.8}$$

$$U(p_i) = U(p_i.arrivalTime)' + U(p_i.cost)' \tag{4.9}$$

Where *U(arrivalTime)'* is the normalized value of $U(p_i.arrivalTime)'$ and $U(p_i.cost)'$ is the normalized value of *U(cost)*. The normalization formula applied is

$$x' = \frac{x\ - min(x)}{max(x) - min(x)} \tag{4.10}$$

$$U(p_i.cost) = p_i.cost \tag{4.11}$$

$$\delta = p_i.arrivalTime - offer\ \in \mathbb{Z} \tag{4.12}$$

$$U(p_i.arrivalTime) = \begin{cases} \delta, if\ \delta < 0. The\ truck\ arrives\ earlier \\ 0, if\ \delta = 0.\ The\ truck\ arrives\ just\ on\ time \\ -2 * \delta, if\ \delta > 0.\ The\ truck\ arrives\ later \end{cases} \tag{4.13}$$

The algorithm 3 is a function that returns a list of evaluated proposals. The algorithm receives a list of proposals, the time offered by the shovel to start loading, and maximum and minimum values for the arrival time and the cost proposed by the *truckAgents*. Each proposal is evaluated following the equations described previously. The evaluation of the arrival time proposed is done using Eq. 4.13 (lines 4-7). The evaluation of the cost proposed is done using Eq. 4.11 (line 8). The maximum and minimum values are used to normalize the values using Eq. 4.10 (line 9-10). The utility value of the proposal is determined using Eq. 4.9 (line 11). Finally, the evaluated proposal is stored in the list *evaluatedProposals*. After evaluating the last proposal, the algorithm returns *evaluatedProposals* (line 15). The algorithm 3 is done

in *O(n)* time and space because of the for clause.

Algorithm 3 Evaluate Proposals

```
1:  procedure EVALUATEPROPOSALS(proposals, offer, maxCost, minCost, maxArrival,
        minArrival )
2:      evaluatedProposals ← ∅
3:      for each p_i ∈ Proposals do
4:          U (arrivalTime) ←  (arrivalTime - offer)
5:          if U (arrivalTime) > 0 then
6:              U (arrivalTime) ← 2 * U (arrivalTime)
7:          end if
8:          U (cost) = cost
9:          U (arrivalTime)' ← (arrivalTime - minArrival) / (maxArrival - minArrival)
10:         U (cost)' ← (cost - minCost) / (maxCost - minCost)
11:         U (p_i) = U (arrivalTime)' + U (cost)'
12:         p_i.value ← U (p_i)
13:         evaluatedProposals ←  evaluatedProposals ∪ p_i
14:     end for
15:     return evaluatedProposals
16: end procedure
```

In this thesis, the utility function considers both production maximization and cost minimization. To maximize the production, the shovels must extract material as much as possible. A manner to get this is by decreasing the idle time of the shovel. Thus, the utility function promotes the proposals that propose to arrive on time instead of those that propose to arrive later. To minimize the costs, the utility function considers the times that the truck takes to perform all the operations (i.e., empty travel from unloading point to the shovel, spotting, loading, loaded travel to unloading point and unloading). Other potential utility functions might be used to evaluate the proposals. For instance, the minimization of the idle time in shovels, the maximization of the truck production, or even based on criteria described Section 2.2.2. The reason for the current utility function is because it considers production and cost simultaneously.

4.4.2 Decision Making *truckAgent*

A *truckAgent* must take decision in two different situations. The first one is when it receives

a *call-for-proposal* message from a *shovelAgent* and must decide whether to send a proposal. The second one is when it receives a *requestConfirmation* message from a *shovelAgent* and must decide whether to confirm the sent proposal.

Deciding Whether to Send a Proposal

As described in Section 4.2, a *shovelAgent* sends a *call-for-proposal* message to *truckAgents* offering the time when will be available to load a truck. A *truckAgent* that receives this offer, must decide whether the offer received can be performed by the truck. To do this, the *truckAgent* checks the assignments in its schedule. Briefly, the *truckAgent* must perform the following steps:

1. Check whether the truck will be busy performing an operation at the offered time by the shovel. If yes, it sends a *refuse* message. If not, it continues with the next step.
2. Check whether there is enough time to perform all the operations between the end time of the previous assignment at the time offered by the shovel and the next assignment. If not, it sends a *refuse* message. If yes, it continues with the next step.
3. Determine the times to perform each operation in the assignment.
4. If the destination of the material extracted by the shovel is a crusher, it sends a message to the *unloadingPointAgent* that represent the crusher asking for the waiting time at the arrival time calculated in the previous step.
5. Update the times with the information of the previous step.
6. Send a proposal to the *shovelAgent*.

The previous steps are implemented in the algorithm 4. The algorithm 4 receives the time offered by the shovel as an argument. The algorithm starts checking if at the time offered by the shovel the truck will be busy executing the function *bussyAtLoadingTime*. This function looks for in the truck schedule and determines if the truck is busy or not. If yes, the algorithm sends a *refuse* message and ends (line 15). If not, it continues determining the previous and next assignment to the time offered by the shovel with the functions *getPreviousAssignment* and *getNextAssignment* (lines 3-4). Then it calculates the total time to perform all the operations related to the offer by executing the function *calculateTotalTime* (line 5). This is the most expensive operation because it must calculate the travel times. To calculate the travel times the function executes the Dijkstra algorithm to determine the shortest path for its

travel. Once the total time is calculated, it checks if the free time slot (*nextAssignment.startTime - previousAssignment.endTime*) is enough to perform all the operations (line 6). If not, it sends a *refuse* message and the algorithm ends (line 12). If yes, it sends a message to the *unloadingPointAgent* where the minerals extracted by the shovel must be unloaded to know the prospective waiting time. With this information, a proposal with all the times of the operations that the truck must perform is generated and sent (line 7-10) and the algorithm ends. Figure 4.3 depicts the different situations mentioned in this paragraph.

Algorithm 4 Deciding to send proposal

```
1:  function DECIDESENDPROPOSAL(offer)
2:      if (not BUSSYATLOADINGTIME(offer.loadingTime)) then
3:          previousAssig ← GETPREVIOUSASSIGNMENT(offer.loadingTime)
4:          nextAssig ← GETNEXTASSIGNMENT(offer.loadingTime)
5:          totalTime ← CALCULATETOTALTIME(offer)
6:          if totalTime ≤ (nextAssig.startTime - previous.endTime) then
7:              proposal ← CALCULATEPROPOSAL(offer)
8:              waitingTimeAtCrusher ← ask to crusherAgent for waiting time
9:              UPDATEPROPOSAL(proposal, waitingTimeAtCrusher)
10:             return true
11:         else
12:             return false
13:         end if
14:     else
15:         return false
16:     end if
17: end function
```

As it was mentioned, the most cost intensive operation is *calculateTotalTime* since it must use a *Dijkstra* algorithm, which is done in *O(n log n)* time , being *n* the number of nodes in a graph that represents a road network. Each of the functions *bussyAtLoadingTime*, *getPreviousAssignment*, and *getNextAssignment* is done in *O(assignments)* , being *assignments* the number of assignments in the truck schedule. Therefore, the run time of algorithm 4 is linear.

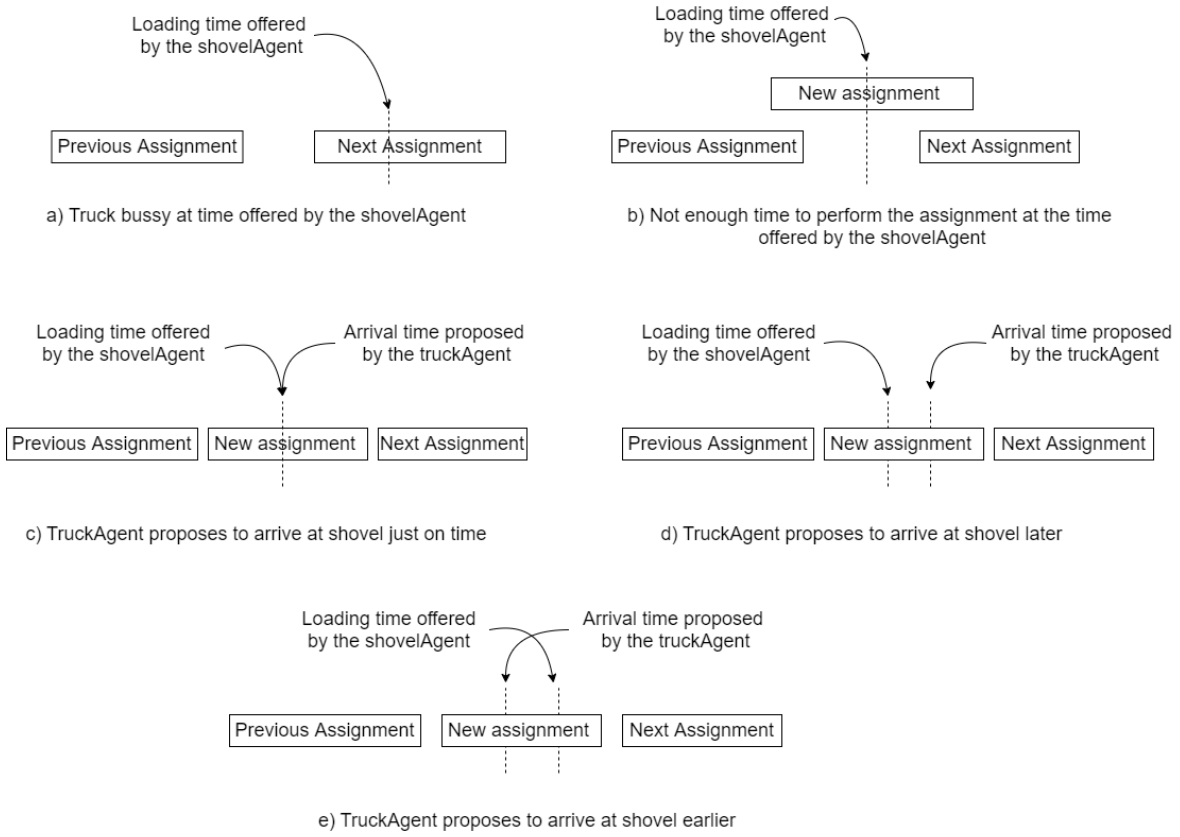

Figure 4.3: Situation to check in the truck schedule.

Deciding Whether to Confirm the Sent Proposal

When a *truckAgent* receives a *requestConfirmation* message from a *shovelAgent*, firstly, it must check if the shovel idle time mentioned in the *acceptConfirmation* message is greater than 1. If yes, the *truckAgent* must send an *acceptConfirmation* message to the *shovelAgent*. If not, the *truckAgent* must decide on sending a *refuseConfirmation* message or sending an *acceptConfirmation* message. To do this, the function *isThereBetterCFP* is executed. The function checks whether there are proposals sent to *call-for-proposals* received from other *shovelAgents* that overlap with the proposal that is being required to confirm. If there is a proposal that overlaps with a better utility for the truck (lower cost), the function returns true, otherwise it returns false.

The complexity of *isThereBetterCFP* function is *O(n)* because of iterates on a list of *n call-for-proposals* stored and only check with if-clauses the overlapping.

4.5 Implementation

Since it is not possible to test the MAS-TD for truck dispatching in open-pit mines in a real-world scenario, the MAS-TD is implemented on the Platform for Simulation of Multiple Agents (PlaSMA) (Warden et al., 2010). PlaSMA is an agent-based event-driven simulation platform developed by the Institute for Information and Communication Technologies (TZI) at the University of Bremen. PlaSMA allows for the simulation and evaluation of scenarios of the logistic domain where autonomous agents perform planning and decision processes. The platform is based on the FIPA-compliant Java Agent Development Framework (JADE) (Bellifemine et al., 2007) which offers components for agent communication and coordination. In addition, JADE provides abstract agent behaviors that can be extended, e.g., simple behaviors, parallel behaviors, or cycle behaviors. Each agent implemented in the platform has its own process in the operating system, therefore the number of agents that can run simultaneously depends on the number of the available cores in the machine.

PlaSMA extends JADE as a simulation middleware for discrete event time progression which allows for precise simulations of processes with small simulated time intervals and ensures correct synchronization and reproducibility (Gehrke, Schuldt, and Werner, 2008). The platform is capable of evaluating large real-world infrastructure with more than 300,000 routes, e.g., roads, motorways, trails, and waterways, and 200,000 traffic junctions on a laptop computer (Gath, 2015). Figure 4.4 depicts the user interface of a PlaSMA container transport simulation.

4.6 Performance Evaluation

In order to evaluate the performance of the developed MAS, experiments based on real data were performed. The first experiments aimed to find suitable values for the parameters of the negotiation process: the deadline for the *call-for-proposal* and the deadline for the confirmation stage. Then, other experiments were performed to evaluate the generation of the schedules and the reaction of the agents when a major event occurs. The following Section explains the experiments and the used scenarios.

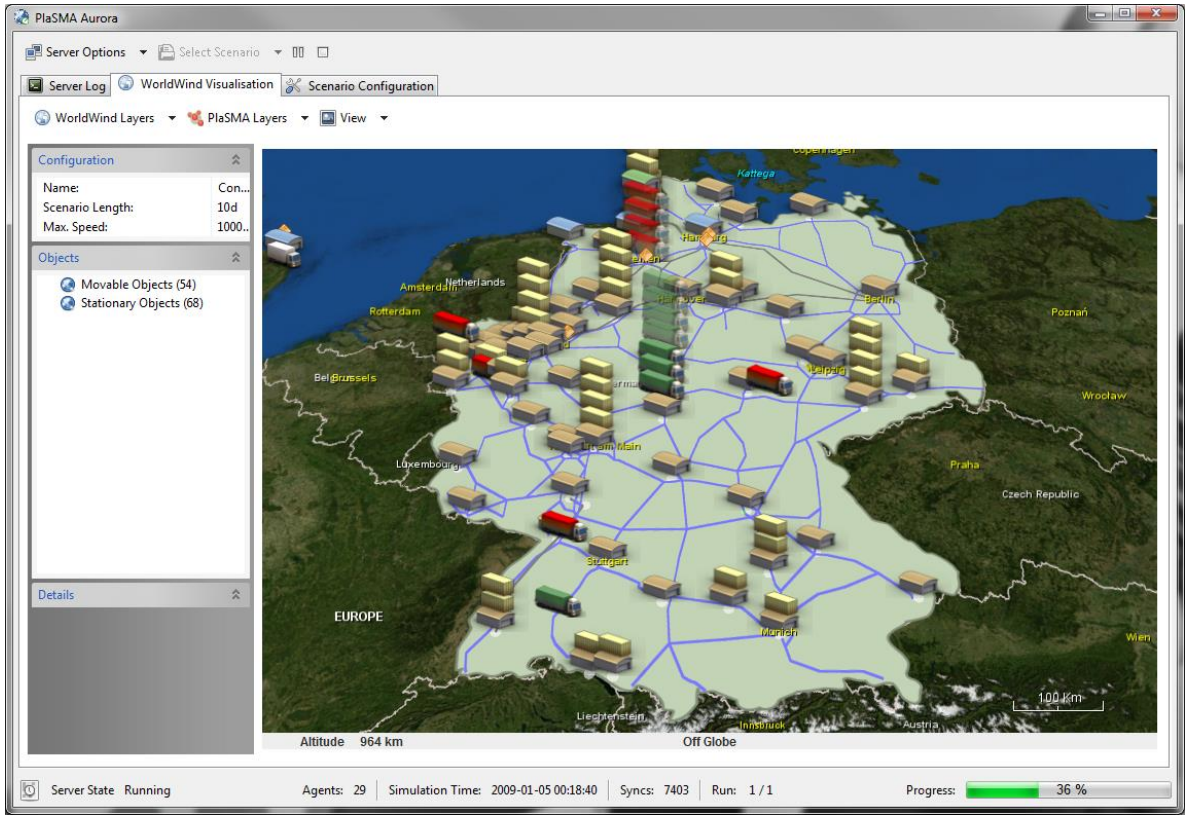

Figure 4.4: The Platform for Simulation of Multiple Agents – PlaSMA (TZI, 2011).

4.6.1 Experimental setup

The experiments used scenarios based on actual data from an open-pit copper mine in Chile. Table 4.3 show the scenarios with the number of shovels and trucks considered in each one and the duration of the shift (column H). The actual data, such as truck velocities and shovel capacities, is used to set the properties of agents. Table 4.4 shows the properties of the agents with the range of possible values. The experiments were performed by simulations run on PlaSMA (cf. 4.5) and on a laptop computer with an Intel Xenon 3 Gigahertz CPU, 32 gigabytes of RAM, and Windows 10.

Table 4.3: Scenarios set for the experiments.

Equipment	Property	Unit	Min Value	Max Value
Trucks	Velocity loaded	[km/hr]	20	25
	Velocity empty	[km/hr]	40	55
	Capacity	[tons]	230	370
	Spotting time	[sec]	20	80
	Current load	[tons]	0	370
Shovel	Capacity	[tons]	35	80
	Load time	[sec]	8	30
	Dig time	[sec]	8	20
	Destination	Location at mine (crusher, stockpile or waste dump)		
Crusher	Equipment discharging	[number of trucks]	1	1
Stockpile	Equipment discharging	[number of trucks]	1	20
Waste Dumps	Equipment discharging	[number of trucks]	1	20

Table 4.4: Property values for the simulations.

Scenario	H	Shovels	Trucks
1	1	1	10
2	3	3	25
3	6	5	40
4	9	7	60
5	12	10	90

4.6.2 Parameter Configuration

In the multiagent system for truck dispatching in open-pit mines, the length of the negotiation process to generate schedules and the quality of these schedules depends on two parameters: the deadline set by a *shovelAgent* to receive messages from *truckAgents*, and the deadline of the confirmation stage. This section investigates the effects of varying these parameters to identify suitable configurations for application. With this purpose, the scenario number 5 (with 90 trucks and ten shovels and a 12-hours shift) is used, which corresponds to a big open-pit mine scenario.

Deadline in *call-for-proposal* Message

A *shovelAgent* set a deadline in the *call-for-proposal* message to avoid an infinite waiting of responses (*propose* or *refuse*) from the *truckAgents* in a negotiation process. After a

shovelAgent sends the *call-for-proposals* to the *truckAgents*, the *shovelAgent* waits for their responses. When it receives all the responses, the *shovelAgent* passes to the next stage in the negotiation process: the evaluation of the received proposals. However, if a *truckAgent* cannot send a response to the *shovelAgent*, the *shovelAgent* would be waiting infinitely for that response. To avoid such a situation, a deadline is set to receive responses to the *call-for-proposal*. After the deadline is reached, the proposals' evaluation will start even if they have not received all the responses from the participants.

The value of the deadline affects the length and the results of the negotiation process. A long deadline would allow the *shovelAgent* to receive many as possible responses from the *truckAgents* and, in this way, to evaluate more proposals. However, the length of the negotiation would be higher than a short deadline. In a short deadline, the initiator would receive a lower number of proposals to evaluate than a long deadline. Still, the length of the negotiation process would be smaller.

Before analyzing the effect of the *call-for-proposal* deadline parameter on the negotiation process, it is necessary to set their maximum and minimum values allowed. For this, experiments without the *call-for-proposal* deadline were performed. In this way, it was possible to observe the moment when a *shovelAgent* received the first response and the moment when it receives all the responses from the *truckAgents*. In the experiment was assumed that all *truckAgents* send a response.

The results show that the first response is received in 750ms. This value was set as the minimum value for the deadline. All the responses from the *truckAgents* were received before 1550ms. This value was used as a maximum value for the deadline.

Deadline for the *requestConfirmation* Message

The confirmation stage allows a *truckAgent* to reject a *requestConfirmation* message from a *shovelAgent* if it is participating in another negotiation whit a better potential agreement. As the deadline for the *call-for-proposal* messages, a high value would allow the participant to have more alternatives to evaluate and get a better agreement but would increase the length of the negotiation process. In the case of a short deadline, the length of the negotiation process would decrease; however, the participant could lose a better potential deal.

To analyze the effect of the deadline in the *requestConfirmation* message in the negotiation

process, it was necessary to set the maximum and minimum values. To set these values, experiments without deadlines in the *requestConfirmation* message were performed to establish the moment when a participant received the first *requestConfirmation* message from a *shovelAgent*. This time was set as the minimum value. For the maximum value, the last received message (*requestConfirmation* or *refuseConfirmation*) from a *shovelAgent* whit a potential agreement was considered.

The results show that the first message is received in 950ms. This value is set as the minimum value for the deadline. In the case of the maximum value, the last received message from a *shovelAgent* whit a potential agreement was received in 4000ms.

Deadlines in *call-for-proposals* and *requestConfirmation* Messages

Whit the minimum and maximum values set for the deadline in the *call-for-proposal* message and the deadline in the *requestConfirmation* message, many pairs of these deadlines arise to evaluate. Some of these pairs were set in the experiments. Figure 4.5 shows the results of these pairs.

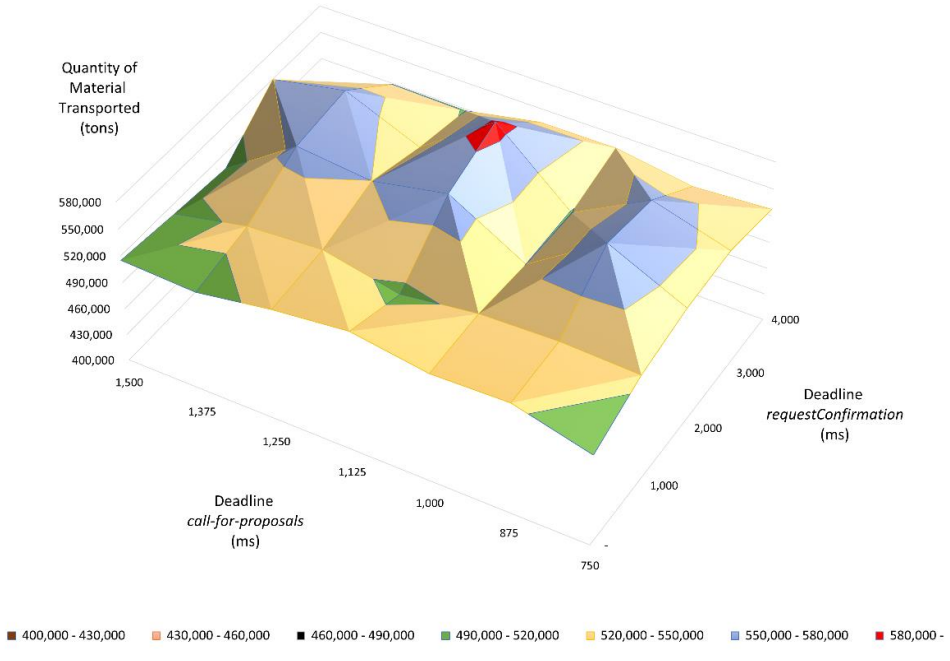

Figure 4.5: Quantity of material transported according to the value of the deadlines for the call-for-proposal and requestConfirmation messages.

The results show that for a value of 1125ms for the deadline in the *call-for-proposals* messages and value of 3000ms for the deadline in the *requestConfirmation* message, the agents generate schedules with the higher production (589,596 tons, red zone) by applying the adapted contract-net protocol. This production is achieved in an intermediate value of the deadline in the *call-for-proposal* message and a high value of the confirmation deadline.
By contrast, the schedule with lower production is achieved with the lowest values of the deadline for the *requestConfirmation* message and both the lowest value and higher value of the deadline for the *call-for-proposal* messages. These can be interpreted that the value of the confirmation deadline affects more than the *call-for-proposal* deadline in the generation of schedules.

4.6.3 Generation of Schedules

In this experiment, several simulations run for all scenarios with the same data, i.e., the same mine layout, and trucks and shovels with the same characteristics. Table 4.5 shows the performance results of the MAS-TD in terms of calculation time for the schedules. The results show that the required time for the MAS-TD to generate the schedules increases with a bigger problem. However, the required time for the last scenario, which is a common scenario of a big-size open-pit mine, is a practical timeframe to generate schedules in the context of the mining industry.

Table 4.5: Required time to generate the schedules by the MAS-TD.

Scenario	H	Shovels	Trucks	Time (seconds)
1	1	1	10	0.04
2	3	3	25	0.43
3	6	5	40	2.45
4	9	7	60	4.52
5	12	10	90	13.28

4.6.4 Dynamic Solution Update

The objective of this experiment was to evaluate different strategies to update the schedules when a major event occurs at the mine. With this purpose, scenario number 5 was used (90 trucks and ten shovels and a twelve-hours shift). In this scenario, two trucks have a failure at

the beginning of the shift and are not available anymore. Truck failures are simulated instead of shovel failures because they are a common situation in open-pit mines.

The evaluated strategies were a simple schedule update (cancelation of the affected assignments), a schedule repair, and a complete reschedule strategy. Table 4.6 shows, for each strategy, the performance of the new schedules (in terms of production level and times that the trucks need to perform all the involved operations) and the required time to update the schedules.

Table 4.6: Performance of rescheduling strategies.

	Production (tons)	**Cost (hours)**	**Required time (seconds)**
Initial schedule	512,049	844.05	796.80
Simple schedule update	467,949	781.35	0.01
Schedule repair	494,633	818.34	15.21
Complete rescheduling	496,927	821.12	785.50

These results show that the simple schedule update achieves the lowest production level in a very short period. This was predictable because the agents update only their schedules canceling the affected assignments. It was calculated in order to contrast it with schedule repair and complete rescheduling strategies.

The schedule repair strategy increases the production level by 5.7% and requires more time to regenerate the schedules than the simple schedule update. This happens because the affected *shovelAgents* start the negotiation processes with the cancelation of some loading assignments to achieve their production goals.

The complete rescheduling strategy achieves the highest production level. However, the required time to generate all schedules is much higher than the schedule repair strategy. This happens because all *shovelAgents* cancel all their assignments and initiate the negotiation processes to regenerate the schedules from scratch. In schedule repair strategy, only a subset of *shovelAgents* (the affected ones) start the negotiation processes. Besides, in complete rescheduling strategy, since the *truckAgents* do not have schedules, they take more time to evaluate and decide on whether to send a proposal. In schedule repair strategy, as the *truckAgents* have schedules, they are more restricted to send proposals. Therefore, it takes them less time to evaluate and decide on whether to send a proposal. Basically, a negotiation

process takes more time in complete rescheduling strategy than in schedule repair strategy.
It looks like schedule repair is the most suitable strategy to apply by the MAS-TD when a major event occurs at an open-pit mine since, in a short period, the schedules are repaired. However, complete rescheduling is also a suitable alternative to be applied by the MAS-TD despite the longer time it requires. This is because anytime algorithms implement the adapted contract-net protocol, i.e., complete rescheduling returns a valid schedule even if it is interrupted before it ends.

4.7 Discussion

The previous sections have described the multiagent system for truck dispatching in open-pit mines (MAS-TD), focusing on the interaction and decision-making of the agents. The agents in the MAS-TD generate (and regenerate) schedules for the equipment involved in the material handling process by applying an adapted contract-net protocol. The adaptation of the contract-net protocol allows the agents to manage concurrent negotiations.
This section discusses several subjects. The first one is related to the design of the MAS-TD and how it is suitable for solving the truck dispatching problem. Next, the proposed protocol is discussed, focusing on its parameters. The third aspect discussed is the decision-making process and how this process affects the performance of the agents. The final discussed aspect is related to the performance of the MAS-TD.

MAS-TD Design

The MAS-TD uses a market structure to organize the agents. This structure allows for autonomous decisions, and communication is limited to the exchange of proposals. However, some drawbacks can arise, especially with selfish agents. Collusion and malicious behavior are some examples.
In the context of MAS-TD, the agents decide autonomously on take part in a negotiation process by sending *propose* (or *refuse*) messages. Besides, the agents can decide autonomously to refuse a previous proposal sent in the confirmation stage of the negotiation process. Since the agents are designed to exhibit a cooperative behavior, the drawbacks mentioned before do not arise in the proposed MAS-TD.
The agents in the MAS-TD follow a physical representation. This representation allows for

a representation of the problem closer to reality. However, new agents might be included in the MAS-TD with a functional representation. For instance, an agent with the objective of providing key performance indicators to users on the material handling process. This agent would send messages to the agents requiring information on their performances to calculate KPIs of the entire system.

Contract-Net Protocol with Confirmation Stage

The proposed protocol is designed to address concurrent negotiations, which allows for decreasing the negotiation length (in comparison to sequential negotiations) and, therefore, determining a fast scheduling (and rescheduling) solution. This is especially important in dynamic environments since a fast reaction is necessary when a major event occurs. Its design allows for stability in the sense that it is impossible to manipulate the outcomes of the negotiation process.

In the protocol, the values of the deadline in the *call-for-proposal* messages and the deadline for the *requestConfirmation* messages must be set. These parameters affect the performance of the MAS-TD in two aspects: the length of the negotiation processes and the quality of the schedule solution. For instance, a high-value for these parameters would suppose obtaining better schedules and increase the time to get it. This could be inappropriate to regenerate schedules when a major event occurs. In contrast, low-value for these parameters would suppose faster negotiation processes and a lower quality schedule solution.

Agent Decision Making

Regarding the decision-making of the agents, these are quite simple and efficient in time and space. All the decisions are taken autonomously. The decision of the *shovelAgents* on the best-received proposal is optimal since it selects the best proposal received. This decision is based on a utility function that considers both productivity and costs. However, other potential utility functions could be considered. A *shovelAgent* might have different utility functions and use them depending on the status of the material handling process. For instance, if there are too many trucks (this situation is called over-trucked), the *shovelAgent* might use a utility function based only on costs. In this way, *shovelAgents* would have more adaptation capacity.

In the case of a *truckAgent*, its decisions are optimal because it selects the best available option. In the case of the confirmation stage, its decision to confirm or refuse a previously sent proposal will depend on the deadline in the *requestConfirmation* message. With a high deadline, the *truckAgent* has more time to receive *requestConfirmation* messages from other *shovelAgents*, and to get an agreement with a higher benefit. However, the length of the negotiation processes would increase (and the entire performance of the MAS-TD). Other mechanisms might be included in the decision-making of the *truckAgent* to avoid increasing the length of the negotiation process. For instance, a mechanism based on probabilities to manage the risk of losing a contract required to confirm vs. the probability of winning in a negotiation process with a more potential benefit. Another alternative would be to exchange information on the proposals with the other participants in the negotiation processes. In this way, it evaluates if it has a chance to win the negotiation process.

MAS-TD Performance

To meet the requirements of any real-world processes, the systems developed to support them must generate solutions in a short time and react fast to any changes that affect the processes. In the context of MAS-TD, the performance measurement must focus on the required time to generates schedules and rescheduling when a major event occurs at the mine. With this purpose, experiments based on actual data were performed. The results demonstrate that the MAS-TD generates pareto-optimal schedules in a reasonable time for the mining industry, even for those open-pit mines with a big fleet of trucks and shovels. Regarding rescheduling, the experiments show that the MAS-TD can reschedule the operations of the fleet in a short time by applying the adapted contract-net protocol, whether complete rescheduling or schedule repair strategies. These results are because multiagent technology characteristics allow for parallel processing, autonomy, and the use of specific data.

4.8 Conclusions

This chapter has presented the multiagent system for truck dispatching in open-pit mines (MAS-TD). The chapter starts with an overview of the system. Next, the main components of the MAS-TD are described: the adapted contract-net protocol, the decision-making process of the agents, their implementation. Then, the evaluation of the MAS-TD

performance is shown. Finally, a discussion of the characteristics of the MAS-TD is presented.

The first conclusion is on the MAS-TD design. Agents that represent physical entities in the real world that interact with each other in a market structure, it has demonstrated to be a suitable design to solve the truck dispatching problem in open-pit mines because:

- It allows for representing the problem closer to reality.
- It provides flexibility to the agents to keep or leave the system.
- It gives the agents the autonomy in their decision-making.

The second conclusion concerns the adapted contract-net protocol. The confirmation stage included in the protocol allows participant agents for efficient management of concurrent negotiations. The protocol is efficient in time and space, and it allows the agents to generate and regenerate pareto-optimal schedules.

The third conclusion regards the decision-making process of the agents. The decision-making processes of the agents are efficient in time and space. However, to keep this efficiency, the cost-intensive tasks involved in the decision-making process must be implemented as efficiently as possible. Otherwise, it could negatively affect the decision-making process.

The last conclusion considers MAS-TD performance. The MAS-TD provides a solution to deal with the truck dispatching problem in open-pit mines in practical frame time. Independent of the domain-dependent requirement, the developed MAS-TD allows for breaking down a scheduling problem in multiple smaller problems, which are solved concurrently. Due to its characteristics, the MAS-TD provides flexibility and robustness to solve a dynamic scheduling problem.

5 EVALUATION

The previous chapter has presented the multiagent system for truck dispatching in open-pit mines (MAS-TD). It also demonstrated that MAS-TD is an alternative to current solutions for the problem. This is because the MAS-TD can generate optimal schedules and regenerate them quickly when a major event occurs at the mine. However, it is still pending to get to know how the MAS-TD performs in comparison to current solutions.

To compare the MAS-TD to alternative solutions, an evaluation of the MAS-TD was performed by comparing it with other methods solving the truck dispatching problem. This chapter describes this evaluation. Firstly, Section 5.1 describes the evaluation design, providing the objectives, methods, and used resources for the evaluation. Secondly, Section 5.2 introduces the Chilean mining company that provided the data for the evaluation. After a brief look at the company history, the section focuses on how they face the material handling process, and on the generated scenarios based on the data provided by the company. Section 5.3 and Section 5.4 present two methods to generate schedules: mathematical programming and a metaheuristic algorithm. These two alternative methods are used in the evaluation. Next, Section 5.5 describes the MAS-TD evaluation from the perspective of scheduling and rescheduling, comparing the generated schedules by the MAS-TD against the generated schedules of the methods mentioned previously. Then Section 5.6 describes the evaluation of MAS-TD by comparing the performance of its schedules against the actual performance of a material handling process supported by an allocation system. Finally, Section 5.7 presents the conclusions of the chapter.

5.1 Evaluation Design

This section describes the evaluation design. The following sections describe the simulation purpose and parameters, the measurements which are evaluated during the simulations, and the tools and resources utilized.

5.1.1 Objectives and Purpose of the Evaluation

The evaluation seeks to compare the MAS-TD method solving the truck dispatching problem in open-pit mines to other methods. Due to the fact that there are two approaches used by the current methods to solve the problem (the scheduling approach and the allocation approach), the evaluation of MAS-TD was split into two parts.

The first part evaluates the MAS-TD from the scheduling approach perspective. The evaluation was performed by comparing the generation of schedules between MAS-TD and two other methods: mathematical programming (cf. 5.3) and the metaheuristic algorithm (cf. 5.4). In the evaluation, rescheduling is also considered.

The second part evaluates if the MAS-TD generates a better solution than an allocation system. In the evaluation, the actual performance of a material handling process, which is supported by an allocation system, is compared against the performance that would have been obtained by following the schedules generated by the MAS-TD. The evaluation also considers events that occur at an open-pit mine.

5.1.2 Output Performance Measures

The following output performance measures are compared in the evaluations:

- Production. It is the total material transported by the trucks during a shift. It is measured in tons.
- Costs. The truck travel times are considered as costs. The actual data does not have any information on costs such as operator salaries, fuel costs, and maintenance costs; therefore, another way to consider costs was necessary. Truck travels are one of the mostly performed and expensive operations during a shift. In truck travel, fuel consumption and tire use are the most important costs. The longer the travel, the higher are the fuel consumption and tire use. For this reason, truck travel times are considered as costs. They are measured in hours.
- Required time to generate schedules. It is the computation time that a method requires to generate schedules. Only the first evaluation considers this measurement. It is measured in minutes.
- Efficiency: The ratio between all the material transported and the costs to

transport these materials. Because travel times are considered as costs, this output performance is measured in tons per hour.

5.1.3 Simulation of the Material Handling Process

The best way to evaluate the MAS-TD would be in a real-world open-pit mine. In this way it would be possible to observe directly how the MAS-TD influences the material handling process. However, evaluating in the real-world is quite difficult because it could involve high investment, costly time of the staff, and other aspects of the mining area such as an unforeseen stop of operations. These and other aspects make it not practical to perform a real-world evaluation, but for these reasons, computational simulations are better suitable for the comparison.

The simulation technique used to evaluate the MAS-TD is Agent-based simulation (ABS). In an ABS model, the decision processes of simulated actors at the micro-level are explicitly described. Structures emerge at the macro level due to the agents' actions and their interactions with other agents and the environment (Siebers and Aickelin, 2008). Therefore, ABS is suitable for identifying and evaluating the MAS-TD characteristics before its implementation in a real-world open-pit mine.

In each simulation, a scenario is run. A scenario corresponds to a 12-hours shift, a fleet of trucks and shovels, and a layout of the open-pit mine. All the scenarios are based on actual data provided by a Chilean Mining company. Details of the data used are in Section 5.2.2. All simulations run on PlaSMA (cf. 4.5). Figure 5.1 shows a simulated open-pit mine based on actual data.

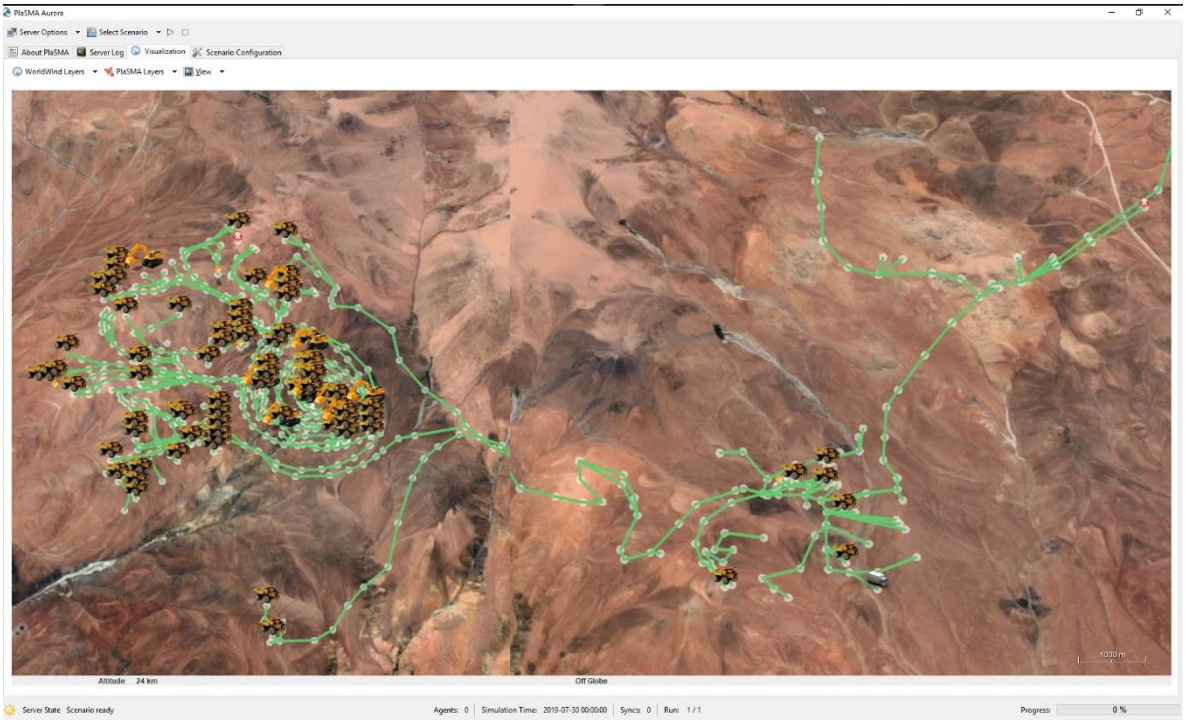

Figure 5.1: An open-pit mine simulated on PlaSMA.

5.2 Real-World Scenarios

All the simulations were performed using scenarios, which are based on actual data. The following sections present

- the mining company that provides the data,
- how the material handling process is performed,
- the current system that supports the truck dispatching, and
- the considered data in the simulations.

The section specializes on the data, how it was obtained, and how it was managed.

5.2.1 Compañía Minera Doña Inés de Collahuasi (CMDIC)

CMDIC is a Chilean mining company dedicated to the extraction and production of copper concentrate and molybdenum concentrate. Its production levels, mineral resources, extension, and location place CMDIC within the world's six main copper producers and the second largest in Chile. It has one of the largest copper deposits, with 10,380 million tons.

The shareholders are Anglo American plc (44%), Glencore (44%), and Japan Collahuasi Resources B.V. (12%), who are represented on the board of directors.

The company's industrial facilities and the "Rosario" and "Ujina" ore deposits are located at 4,400 meters above sea level in the high plateau of the Atacama Desert, northern Chile. This Andean area is characterized by having a rainy climate in summer and occasional snowfalls in winter. Its daily temperatures can range between -8 ° and 25 ° Celsius. Figure 5.2 shows the location of the CMDIC.

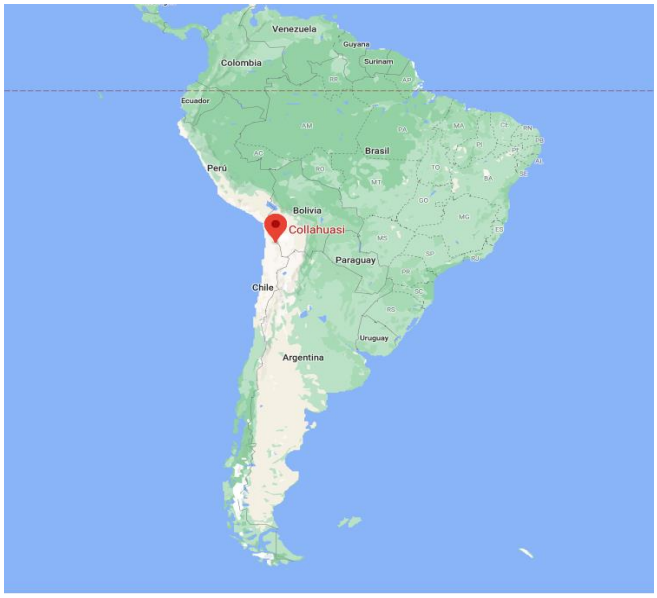

Figure 5.2: CMDIC location (Google, 2021).

The concentrator plant is located in the "Ujina" sector. In this sector a pipeline system of 203 km begins, whereby the copper concentrate is transferred to the CMDIC maritime terminal. From this port, products are shipped to international markets. In this place, there are also the molybdenum and concentrate filtering plants. Figure 5.3 shows the production processes of the company and its locations.

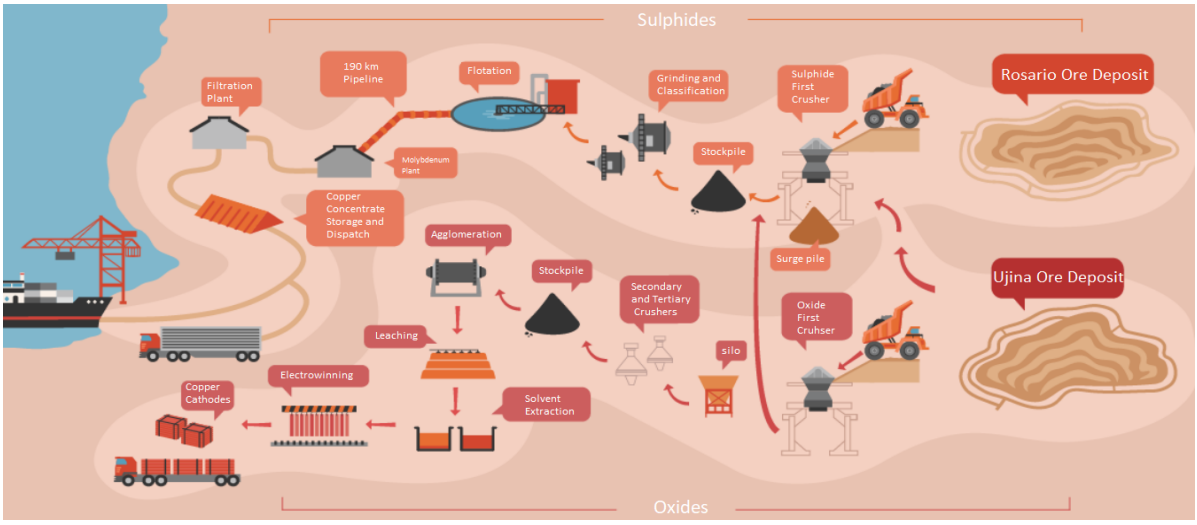

Figure 5.3: CMDIC's processes from ore deposits to the port (Collahuasi, 2016).

CMDIC History

The Collahuasi mining district's commercial activity began in 1880 with the exploitation of the high-grade copper-silver vein systems. This operation was interrupted in 1930 due to the global economic crisis. Activities in the area resumed in 1978, the year in which the key components of the "Rosario" ore deposit were identified. Later, in 1991, a combination of satellite imagery-based studies, ground surveys, and borehole drilling activities resulted in the discovery of the "Ujina" ore deposit. The feasibility and environmental impact studies of the Collahuasi Project were approved in 1995. At the end of 1996, CMDIC signed the agreements regarding financing and commercialization, and the development and construction phase began.

Material Handling in CMDIC

The material handling process in CMDIC is carried out utilizing a heterogeneous fleet of shovels and large trucks that work in 12-hour shifts. As described in Section 2.1, the shovels load trucks, which transport the materials to different destinations at the mine. Figure 5.4 shows different aspects during loading: trucks arriving at the shovel, a truck being loaded, a truck waiting for loading, and a loaded truck starting the travel to the unloading point. Table 5.1 shows how the fleet in CMDIC is constituted.

Figure 5.4: Shovel loading a truck. Other trucks arrive and wait to be loaded (La Segunda, 2015).

Table 5.1: CMDIC's fleet used in the material handling process.

Type of equipment	Equipment	Model	Capacity [yd3]	Number of equipment
Loading	Shovel	Bucyrus 495HR	73	4
		P&H 4100 XPC	74	4
		Caterpillar 495	79	1
		Komatsu PC5500	36	2
		Komatsu PC8000	42	1
	Front loader	Marathon LeTourneaun	36	3
Hauling	Truck	Komatsu 930 E	330	104
		Liebherr T 282C	373	7
		Caterpillar 794AC	330	4

Truck Dispatching in CMDIC

Material handling in CMDIC is supported by the Dispatch ™ system, which is developed by the Modular Mining company. This system is one of the most famous ones and is used in large mining (Moradi Afrapoli, Tabesh, and Askari-Nasab, 2019). Dispatch ™ is a real-time management tool that seeks to optimize truck allocation to shovels and their next destinations

(crushing or dumps). The system records the events that occur during the operation, and based on this information, the system automatically determines the optimal material haulage route (Alarie and Gamache, 2002).

In the first stage, linear programming is used to determine the optimal flows (in tons per hour) and the shortest paths between each shovel and each discharge point. The program considers the mine's current configuration, the maximum excavation rate at each shovel, and the maximum capacity at the discharge points. It uses an objective function that minimizes the sum of several pseudo-costs.

In the second stage, linear programming is also used with an objective function seeking to maximize the trucks' production. At this stage, the trucks are dispatched to meet the flows determined by the first stage. To do this, the second stage uses two lists: one for routes and one for trucks. The list of routes is ordered according to the current production lag in the route concerning the production determined for that route in the first stage. The most lagging route is the first item on the list. The truck list contains all trucks that are unloading or on the way to an unloading point. Therefore, trucks are assigned to shovels by matching the best truck to the most lagging route. The best truck is the one that minimizes the lost tons, which is a measure of non-productivity. This measure considers the resulting shovel's idle time, the truck waiting time, the additional travel time that the truck must perform to reach the shovel instead of the closest one, and the flows determined in the first stage. After assigning the best truck to the corresponding shovel of the most lagging route, the best truck is removed from the truck list, and the most lagging route is moved to the end of the route list, leaving the second most lagging route to be the new most lagging route. The pairing process repeats as long as the truck list is not empty. The dispatch decision process ends with dispatching the assignment for the truck that requires an immediate assignment. More details on the fundamentals of Dispatch ™ can be found in White and Olson (1986) and White, Olson, and Vohnout (1993).

Scenarios

The scenarios used in the evaluations were generated with the data obtained through PowerView™, which is a software that receives and stores the information generated in Dispatch™ for later consultation and review. Because not all data in PowerView™ is useful

for the generation of scenarios, and as some important data is also deleted after some time, it was necessary to generate a local database to store the relevant data. After analyzing the data, an Entity-Relation (ER) diagram was created to organize the relevant data for this thesis. Then, a local database was implemented based on the ER diagram shown in Figure 5.6. With ETL (Extract, Transform, and Load) processes, the useful data was copied from PowerView^{TM} to the local database.

The stored data in the local database cover the period from 12.31.2018 to 30.04.2019. In this period, 155 12-hours shifts are stored. Figure 5.5 shows the quantity of material transported per shift and the number of equipment pieces that operated during those shifts. From this data, the shifts with the following criteria were selected:

- the shift whit the highest production,
- the shift with the highest number of trucks,
- the shift with the highest number of shovels, and
- the shift with the highest number of trucks and shovels failures

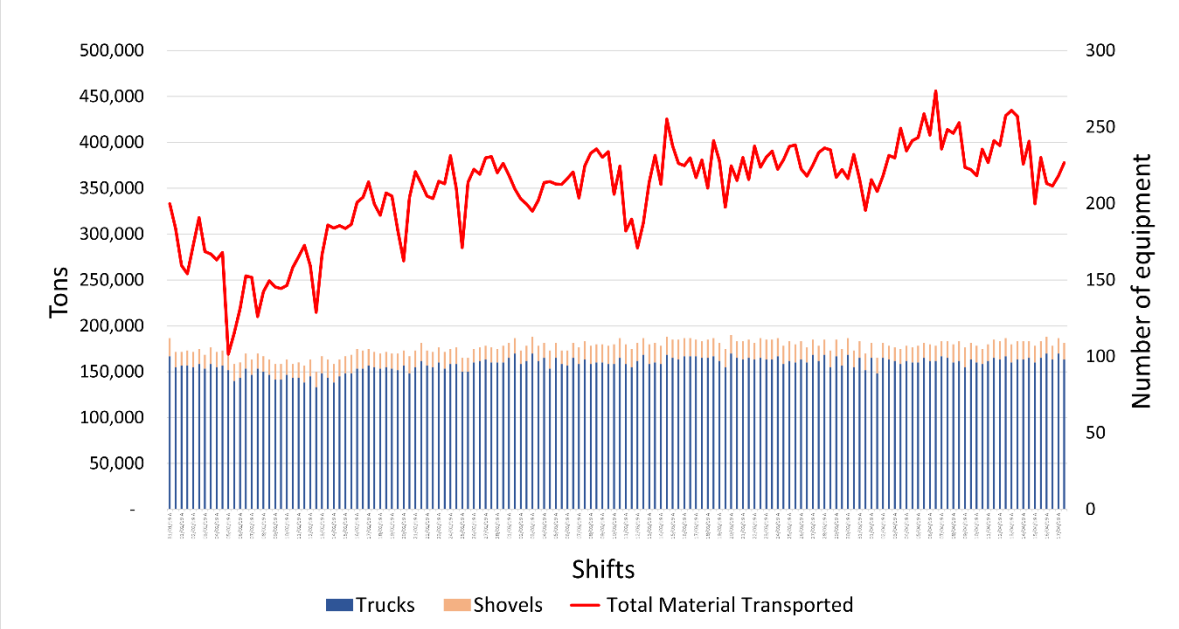

Figure 5.5: Material transported and equipment number per shift.

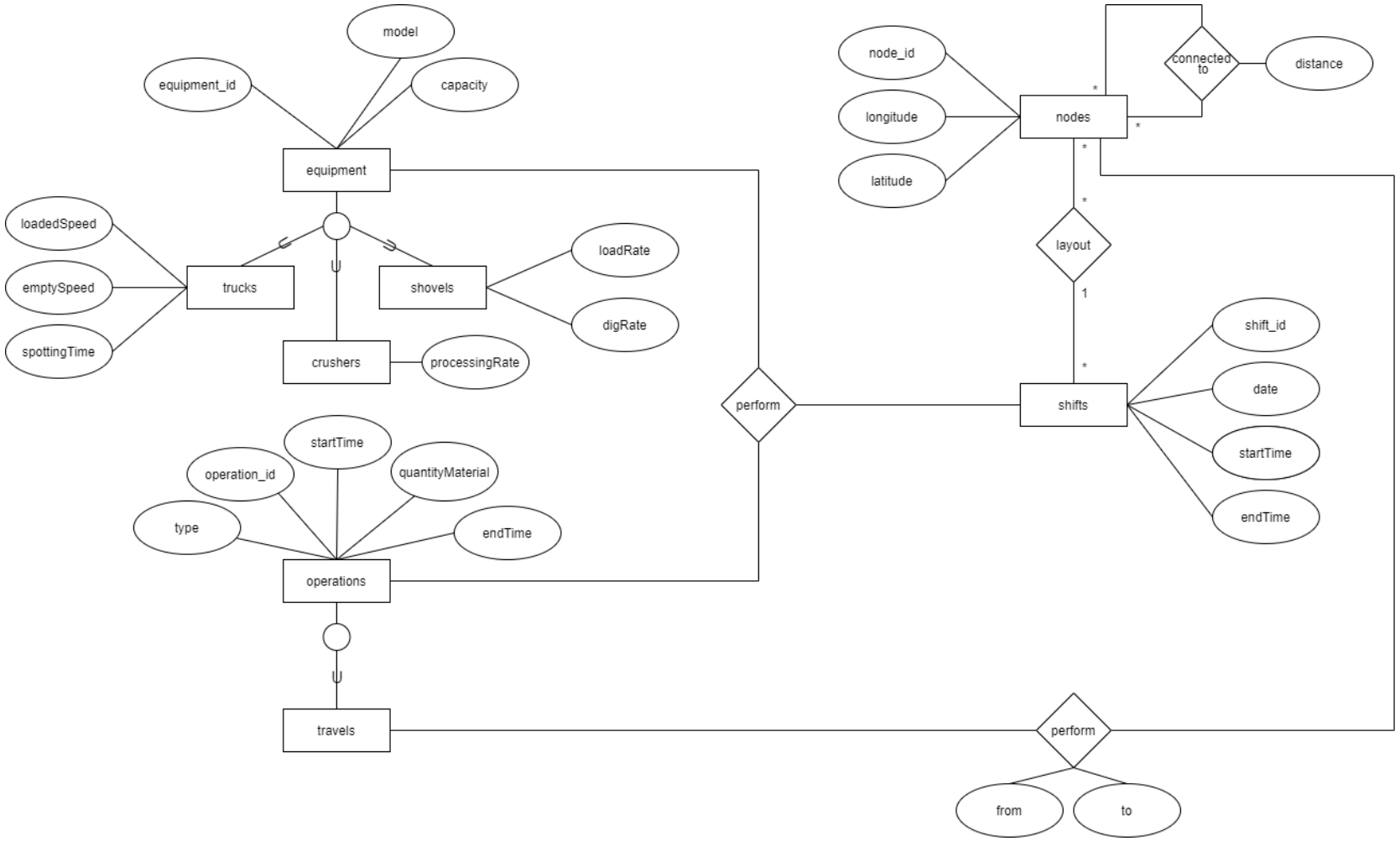

Figure 5.6: Entity-Relation diagram for material handling data.

Table 5.2 shows the selected scenarios for the simulation. As it can be observed, the scenarios do not follow any logic. For instance, in scenario two, the biggest fleet was used; however, its production is not the highest. Another example is by comparing scenarios one and three, in which their fleet size is quite similar. However, their production is quite different. These situations occur because different factors affect the fleet. Perhaps, bad weather in scenario 2 influences the velocity of the trucks; therefore, their velocity was lower. This could result in a smaller number of unloads. Another possible situation that might have happened is that a blasting took too much time; therefore, the equipment items are not productive for a longer time. Another reason could be the mine layout: it might have been that some shovels were closer to their corresponding unloading points; therefore, the trucks made more unloadings. Unfortunately, the actual data does not provide this kind of information. Therefore, it was not possible to know the actual reasons for these differences among the shifts.

Table 5.2: Shifts selected for the simulations.

Scenario ID	Shift	Number of Trucks	Number of Shovels	Material Transported (tons)	Travel Times (hours)	Brief Description
1	06/04/2019 B[14]	97	10	455,696.23	780.74	Highest production
2	20/03/2019 A	102	12	378,069.92	772.84	Highest number of trucks
3	19/03/2019 B	95	13	325,899.61	757.62	Highest number of shovels
4	08/03/2019 B	96	12	394,759.09	791.59	Many events (truck and shovel failures)

As described in Section 5.1.1, the evaluation of MAS-TD included rescheduling. Because scenario four contains many events (shovel and truck failures), it was selected to observe how the methods react when major events affect the material handling process. Table 5.3 shows the events in scenario four, the affected equipment, the event start and end time, and the duration.

The scenarios were generated with the local database and scripts that generate the scenarios in the format required by PlaSMA. Table 5.4 shows the components of a scenario and their corresponding data.

[14] Shift A: 08:00 to 20:00. Shift B: 20:00 to 08:00

Table 5.3: Truck and shovel failures in scenario four.

ID Equipment	Type of Equipment	Start Time	End Time	Duration
CA104	Truck	00:00:00	02:09:09	02:09:09
CA156	Truck	00:00:00	03:28:28	03:28:28
CA93	Truck	00:00:00	03:28:28	03:28:28
CA67	Truck	00:04:04	02:40:40	02:36:36
CA138	Truck	01:11:11	02:03:03	00:52:52
CA60	Truck	01:22:22	03:26:26	02:04:04
CA85	Truck	01:38:38	03:27:26	01:48:48
CA162	Truck	02:39:39	04:28:27	01:48:48
CA148	Truck	04:06:06	04:44:43	00:37:37
CA136	Truck	05:57:57	08:45:45	02:48:48
CA145	Truck	06:20:20	06:55:55	00:35:35
CA121	Truck	06:59:59	07:37:36	00:37:37
CA72	Truck	07:39:39	09:02:02	01:22:22
CA92	Truck	07:46:46	08:19:19	00:32:33
CA98	Truck	07:47:47	08:24:23	00:36:36
CA116	Truck	07:55:55	10:53:52	02:57:57
CA55	Truck	08:49:49	09:37:37	00:48:48
CA99	Truck	09:20:20	12:00:00	02:39:40
CA164	Truck	09:57:57	10:23:23	00:26:26
CA109	Truck	10:45:45	12:00:00	01:14:15
CA75	Truck	10:46:46	12:00:00	01:13:17
PA13	Shovel	01:22:22	07:42:42	06:19:20
PA06	Shovel	03:06:06	04:31:31	01:24:25
PA11	Shovel	07:28:28	12:00:00	04:31:32
CF06	Shovel	08:31:31	09:03:02	00:31:31
CF05	Shovel	00:37:37	05:59:58	05:21:21

Table 5.4: Components of a scenario for simulation.

Aspect	Component
Mine Layout	Vertices Paths Distances
Trucks	Location at the beginning of the shift Capacity Empty velocity Loaded Velocity Spotting time Dumping time Working Time
Shovels	Location at the beginning of the shift Capacity Load Time Dig Time Operational Target Destination of material
Unloading Points	Location Number of trucks unloading simultaneously

5.3 Mathematical Programming

This section describes a mathematical model that generates schedules for the truck dispatching problem in open-pit mines. To address the problem, the mathematical model is formulated based on Patterson's work (Patterson et al., 2017), which uses Mixed-integer linear programming (MILP) to minimize the energy consumption of the shovels and trucks, taking into account the targets of the production plan in an open-pit coal mine. The model uses a sequence of loading 'slots' per shovel to organize the activities of trucks and shovels.

In the model, trucks can be assigned to any shovel. The shovels are assigned to one pit, and the material extracted by a shovel must be transported to a destination throughout the shift. Shovels can load one truck at a time. At a crusher, one truck can unload at a time, whereas in a waste dump or a stockpile trucks can unload simultaneously.

The notation of sets, indices, parameters, and decision variables used in the model are shown in Table 5.5.

The objective function (5.1) is to minimize the costs, which are based on truck travel times. Restriction (5.2) ensures that at most one truck is assigned to each time slot l on each shovel.

Restriction (5.3) ensures that no more than one truck t is loaded on the shovel at a time.

Restriction (5.4) ensures that at crushers, no more than one truck can unload at a time.

Restriction (5.5) ensures that an unloading time ($\mu_{s,l}$), starts after the loading starts ($\succ_{s,l}$) plus the time it takes to perform the loading c_s and the travel time to the destination c_j^s.

Restriction (5.6) ensures that the following loading of a truck in l' must be after the truck finishes the unloading in l.

Restrictions (5.7) and (5.8) ensure that the sequence of each truck has one predecessor and a successor.

Restriction (5.9) ensures that the proposed loading targets for each shovel pointed out in the production plan are met. This restriction is activated or deactivated, depending on the simulation objective.

Restriction (5.10) ensures that a truck travel time to its first loading point must be considered at the beginning of the shift.

Restriction (5.11) ensures that all trucks start in a dummy pit. This pit represents the initial place where a truck is at the beginning of the shift.

Restriction (5.12) ensures that all trucks also end the shift in a dummy pit.

This mathematical model was implemented on CPLEX.

Table 5.5: Formula notation.

Name	Description
S	Set of shovels (index s)
T	Set of trucks (index t)
Ls	Slot time of the shovel s
J	Set of destination of the material extracted by shovel s (index j)
Cs	Loading time of the shovel s
Cj	Unloading time at the destination j
C_j^s	Travel time from destination j to shovel s
C_s^j	Travel time from shovel s to destination j
C_s^t	Travel time of truck t to shovel s (only at the beginning of the shift)
$C_j^{s'}$	Travel time from the destination j to next shovel s'
A_t	Capacity of truck t
δ_s	The target of extracted material by shovel s
M	Sufficiently large positive number
$x_{t,s,l}$	1 if the truck t loads at shovel s in the time slot l, otherwise 0
$\curlywedge_{s,l}$	Loading start time of shovel s in time slot l
$\mu_{s,l}$	Unloading start time of material extracted by shovel s in time slot l
$\curlywedge^{seq}_{t,s,l,s',l'}$	1 if truck r was loaded by shovel s in time slot l before being loaded in shovel s' and slot time l'. Otherwise, 0.

$$\text{Min} \sum_{\forall\, s,l} (C_s^j + C_s + C_j^s + C_j) \tag{5.1}$$

$$\sum_{\forall\, l} x_{t,s,l} \leq 1 \;\forall\, l,s \tag{5.2}$$

$$\curlywedge_{s,l+1} - \curlywedge_{s,l} \geq C_s \;\forall\, l,s \tag{5.3}$$

$$\mu_{s,l+1} - \mu_{s,l} \geq C_j \;\forall\, l,s \tag{5.4}$$

$$\mu_{s,l} \geq \curlywedge_{s,l} + C_s + C_j^s \;\forall\, l,s \tag{5.5}$$

$$\curlywedge_{s',l'} \geq \mu_{s,l} + C_j^{s'} + C_j - M\left(2 - \curlywedge^{seq}_{t,s,l,s',l'} - x_{t,s,l}\right) \;\forall\, t,l,s,s',l' \tag{5.6}$$

$$x_{t,s,l} = \sum_{\forall\, s,l} \curlywedge^{seq}_{t,s,l,s',l'} \;\forall\, t,l,s \tag{5.7}$$

$$x_{t,s,l} = \sum_{\forall\, s,l} \curlywedge^{seq}_{t,s,l,s',l'} \;\forall\, t,l,s \tag{5.8}$$

$$\sum_{\forall\, l} x_{t,s,l}\, A_t \geq \delta_s \;\forall\, s \tag{5.9}$$

$$\curlywedge_{s,1} \geq C_s^t - M\left(1 - \curlywedge^{seq}_{t,0,0,s,1}\right) \;\forall\, t,s \tag{5.10}$$

$$\sum_{\forall\, s,l} \searrow^{seq}_{t,0,0,s,l} = 1 \;\; \forall\, t \tag{5.11}$$

$$\sum_{\forall\, s,l} \searrow^{seq}_{t,s,l,0,0} = 1 \;\; \forall\, t \tag{5.12}$$

5.4 Metaheuristic Algorithm

Metaheuristics, particularly the Tabu Search (Glover, 1989), are commonly employed to overcome the complexity of scheduling formulations, especially for problems dealing with heterogeneous vehicle fleets (Koç, Bektaş, Jabali, and Laporte, 2016). The Tabu Search algorithm explores neighborhoods to find improved solutions starting from an initial solution. It stores recently visited solutions in memory (tabu list). The algorithm rejects neighborhood moves that result in a solution already in the tabu list to avoid cycling around local optima. During the algorithm's main loop, the best neighborhood that is not in the tabu list is selected and added to the tabu list. If it is the best solution found so far, it is recorded as such. The algorithm terminates after reaching a stop condition and returns the best solution found.

Algorithm 5 implements the Tabu Search described above. In lines 1-5, the initial setup of the algorithm is done. In *sBest* the initial solution provided by Disp-ALG (cf. 5.4.1) is stored, and it is set as the current solution in *bestCandidate*. Also, the tabu list is created and stored in *tabuList*. The main loop starts in line 6. This loop searches for a better solution until a time-out is reached. In line 7, the function *getNeighbors(bestCandidate)* generates new feasible solutions through small changes in the stored solution in *bestCandidate*. (Changes in a solution are explained in Section 5.4.2.) Lines 8 - 13 look for the best candidate among the neighbors that is not in the tabu list. If this best candidate is better than the best-found solution so far, it is set as the best solution (lines 14-16). The best candidate is added to the tabu list (line 17), and if the tabu list is full, the oldest element is removed (line 18-20). Finally, when the *stopTime* is reached, the algorithm returns the best-found solution.

Algorithm 5 Tabu Search

Input: executionTime
Output: improvedSolution

```
1:  sBest ← Disp-ALG
2:  bestCandidate ← sBest
3:  tabuList ← ∅
4:  stopTime ← currentTime+executionTime
5:  tabuList.push(sBest)
6:  while currentTime < stopTime do
7:      sNeighborhood ← getNeighbors(bestCandidate)
8:      bestCandidates ← NeighborHood.firstElement()
9:      for sCandidate ∈ sNeighborHood do
10:         if not (tabuList.contains(sCandidate)) and (fitness(sCandidate) >
            fitness(bestCandidate)) then
11:             bestCandidate ← sCandidate
12:         end if
13:     end for
14:     if fitness(bestCandidate) > fitness(sBest) then
15:         sBest ← bestCandidate
16:     end if
17:     tabuList.push(bestCandidate)
18:     if tabuList.size > maxTabuSize then
19:         tabuList.removeFirst()
20:     end if
21: end while
22: return sBest
```

5.4.1 Initial Solution

The algorithm 6 (Disp-ALG) builds feasible schedules for the equipment involved in the material handling process. These feasible schedules are used as the initial solution in the Tabu Search algorithm presented above. The Disp-ALG algorithm uses the following inputs:

- Travel Times Matrix: These are the travel times that trucks take to travel from one point to another in the mine.
- Production plan: It points out the quantity of material that must be extracted by the shovels and transported to a destination.
- Equipment data: Information about available shovels and trucks, such as capacity and velocity.

The algorithm output is a schedule of the operations that the equipment items must

perform. The notation of sets, indices, parameters, and variables used throughout the algorithm are shown in Table 5.6. More formally:

Table 5.6: Notations for Disp-ALG.

Name	Description
S	Set of shovels
T	Set of trucks (index t)
J	Set of destination of the material extracted by shovel s (index j)
C_s	Loading time of the shovel s
C_j	Unloading time at the destination j
C_j^s	Travel time from destination j to shovel s
C_s^j	Travel time from shovel s to destination j

$$plan = \{(s, j, x) \mid s \in S, j \in J, x \in \mathbb{N} \tag{5.13}$$

$$truckActivity = \{'emptyTrip' \wedge' loading' \wedge' loadTrip' \wedge 'unloading'\} \tag{5.14}$$

$$shovelSchs = \{(s, startTime, endTime, t) \mid s \in S, startTime \in \mathbb{N}, endTime \in \mathbb{N}, t \in T\} \tag{5.15}$$

$$truckSchs = \{(t, truckActivity, startTime, endTime, from, to) \mid t \in T, activity \in truckActivity, startTime \in \mathbb{N}, endTime \in \mathbb{N}, from \in (S \cup J), to \in (S \cup J)\} \tag{5.16}$$

The main algorithm (algorithm 6) consists of 3 steps: the first one looks for a shovel task defined as *(s, j, tll).* The second step looks for the best truck *t* to perform the shovel task. If the second step is successful, the algorithm proceeds with the third step, which consists of adding the activities and times to the schedules of the truck *t* and shovel *s*; otherwise, the shovel task is discarded. These steps are repeated until the last activity of the shovels exceeds the end of the shift. To determine whether these conditions are met, the algorithm invokes the Boolean function *isConditionsMeet.*

Algorithm 6 Disp-ALG

Input: S, T, Plan, H
Output: shovelSchds, truckSchds
1: shovelSchds ← ∅ ; truckSchds ← ∅;
2: **while not** isConditionsMeet **do**
3: s, j, tll ← FindJob(S, shovelSchds, H)
4: **if** s **not** null **then**
5: t, act, act' ← FindTruck(T, s, j, tll, truckSchds)
6: **if** t **not** null **then**
7: Schedule(t,s,j,tll,shovelSchds, truckSchds, act, act')
8: **end if**
9: **end if**
10: **end while**

The function *FindJob* (algorithm 7) is a function that returns a shovel task. To do this, the algorithm executes the function *SelectShovel*, which returns a shovel *s* whose activities have not reached the end of the shift. The function *SelectShovel* is modified depending on the objective of the simulated material handling process. For instance, if the objective is to achieve a production plan's targets, the function returns a shovel *s* whose target has not been met. Then, it executes the function *getDestination* to get the destination *j* of the material extracted by shovel *s*. Next, it proceeds with the function *getSchedule* to retrieve the schedule of the shovel *s*. If the *getSchedule* function returns null, i. e., the shovel *s* does not have a schedule yet, so function ends returning *(s, j, 0)* as the shovel task. Otherwise, the algorithm invokes the function *getEndTimeLastLoading* to get *tll* (end time of the last loading activity). If *tll* is higher than the shift's horizon time, the algorithm returns null, otherwise returns *(s, j, tll)* as the shovel task and the algorithm terminates. The next time that the function *FindJob* is invoked, the function selects another shovel following a round-robin method.

Algorithm 7 FindJob

Input: S, shovelSchds, H
Output: s, j, tll
1: tll ← ∅
2: s ← SelectShovel(S, H)
3: **if** s **is** null **then**
4: **return** null
5: **else**
6: j ← getDestination(s)
7: tempShovelSchd ← getSchedule(s)
8: **if** tempShovelSchd **is** null **then**
9: **return** s, j, 0
10: **else**
11: tll ← getEndTimeLastLoading(tempShovelSchd)
12: **if** tll > H **then**
13: **return** null
14: **else**
15: **return** s, j, tll
16: **end if**
17: **end if**
18: **end if**

After finding a shovel task, the function *FindTruck* (algorithm 8) is invoked with the parameters *s, j, tll* (a shovel task). The function returns the best truck *t* to perform the shovel task and its previous and following activities to *tll* (*act* and *act'* respectively). To do this, the function gets the trucks' schedules by executing the function *getSchedule* and looks for all the trucks that have a free time slot to perform the shovel task. Then, for each of these trucks, the algorithm calculates the time to perform all the operations necessary to perform the shovel task. The function returns the truck that performs all the activities in the least amount of time. The procedure *Schedule* (algorithm 9) updates the shovel and truck schedules adding all the operations with their start and end times.

Algorithm 8 FindTruck

Input: T, s, j, tll, truckSchds
Output: t, act, act'
1: t ← ∅
2: **for** each t ∈ T **do**
3: tempTruckSchd ← getSchedule(t)
4: **if** tempTruckSchd **is** null **then**
5: **return** t, null, null
6: **else**
7: **for** each act ∈ tempTruckSchd **do**
8: **if** $act_{activity}$ = "unload" **and** $act_{endtime}$ < tll **and** $act'_{startTime}$ > tll **then**
9: totalTime ← $C_s^j + C_s + C_j^s + C_j$
10: **if** totalTime < ($act'_{starTime}$ - $act_{endtime}$) **then**
11: **if** (tll+totalTime) < act'_{start} **then**
12: **return** t, act, act'
13: **end if**
14: **end if**
15: **end if**
16: **end for**
17: **end if**
18: **end for**
19: **return** null

Algorithm 9 Schedule

Input: t, s, j, tll, shovelSchds, truckSchds, act, act'
Output:
1: empTrip ← t, "emptyTrip", tll – $C^s_{act_{to}}$, tll, act_{to}, s
2: loadAtShovel ← t, "load", tll, C_s , s, s
3: tripToUnload ← t, "loadedTrip", tll + C_s, tll + C_s + C_s^j, s, j
4: Unload ← t, "unload", tll + C_s + C_s^j, tll + C_s + C_s^j + C_j , j, j
5: truckSchds truckSchds ← empTrip ∪ loadAtShovel ∪ tripToUnload ∪ Unload
6: shovelLoad ← s, tll, tll + C_s, t
7: shovelSchds ← shovelSchds ∪ shovelLoad

5.4.2 Neighborhoods

The solution space is explored by performing three operators to generate neighborhoods from the initial solution. Figure 5.7 shows examples of possible generated neighborhoods. The *swap* operator, shown in Figure 5.7(a), generates the first one. The *swap* operator swaps two truck assignments, ignoring unfeasible swaps. A feasible swap is only considered if the swap decreases the cost to perform the assignments. The second

neighborhood, shown in Figure 5.7(b), is generated by the *change* operator. This operator moves an assignment from one truck to another one only if the second truck can perform the assignment in a shorter time than the first truck. The *new assignment* operator generates the last neighborhood, shown in Figure 5.7(c). The *new assignment* operator adds a new assignment by seeking a free time slot to perform a loading operation in the shovel. Then it looks for a truck to perform the loading operation.

To select the best neighborhood, an evaluation is performed based on the efficiency of the schedules. The ratio *TotalMaterialToBeTransported / TotalCosts* determines the efficiency of all schedules. The neighborhood with the highest ratio is selected as the current solution for an iteration in the Tabu Search algorithm.

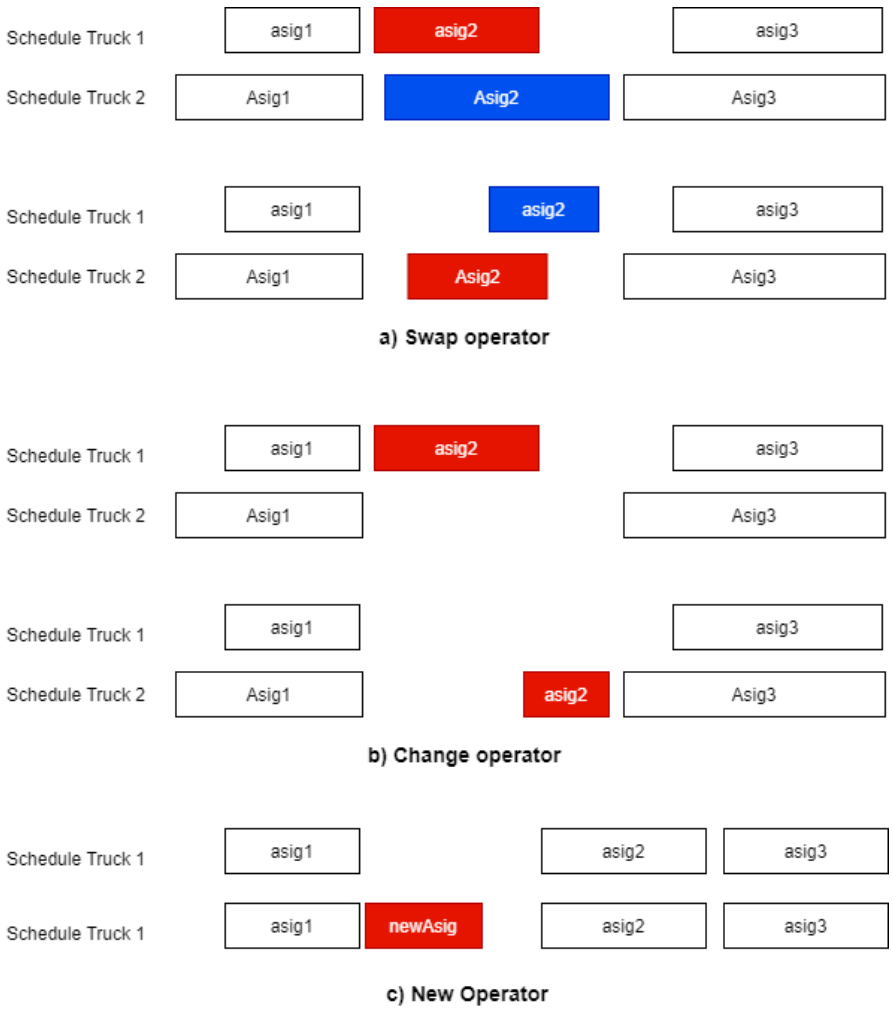

Figure 5.7: Operators to generate neighborhoods.

5.5 Evaluation of MAS-TD and Other Schedule Generating Methods

This evaluation was performed by the comparison of applying the MAS-TD, mathematical programming, and the metaheuristic algorithm to generate schedules and rescheduling. The scheduling evaluation was split into two parts. On the one hand, the methods were set to achieve the maximum production level. On the other hand, the methods were set to achieve the targets of a production plan. In both situations, the minimization of the costs was also considered. The rescheduling evaluation considers shovel and truck failures. The following sections provide more details on the evaluations and show the results obtained, followed by a discussion.

5.5.1 Results

Scheduling for Maximum Production Level

The simulations aimed to compare the computation time of the methods to generate the schedules, the reached production in the generated schedules, the costs to perform all the truck operations and the efficiency of the schedules. Tabu Search runs with different execution times (5, 10, 15, and 20 minutes). All the methods are set to get the maximum production level and to minimize the costs.

Regarding computation time to generate the schedules, only MAS-TD and Tabu Search generated schedules within practical time frame (for the mining industry). Mathematical programming was not able to generate the schedules, even after four hours of run time. Figure 5.8 shows the computation time of the methods to generate the schedules for each shift. MAS-TD generated the schedules in 20.48 minutes on average. Tabu Search generated the schedules in 5, 10, 15, and 20 minutes as it was pre-set. Also, Disp-ALG was included in the comparison since it generates the initial schedules used by Tabu Search. Disp-ALG was the fastest method to generate schedules. On average, Disp-ALG required 0.1 minutes to generate the schedules. Because Mathematical Programming could not generate schedules for the scenarios, the next comparisons only consider MAS-TD, Tabu Search, and Disp-ALG.

Regarding the production goals, the activities in the schedules generated by Disp-ALG move the lowest quantity of material in each of the four scenarios. After 20 minutes of run time, Tabu Search improves the schedules generated by Disp-ALG by increasing

them by 0.76% on average. MAS-TD was the method whose schedules could move the highest amount of material. The activities scheduled by it transported 9.93% more material than the activities in the schedules generated by Tabu Search. Figure 5.9 shows the quantity of the material transported in the schedules generated by the three methods.

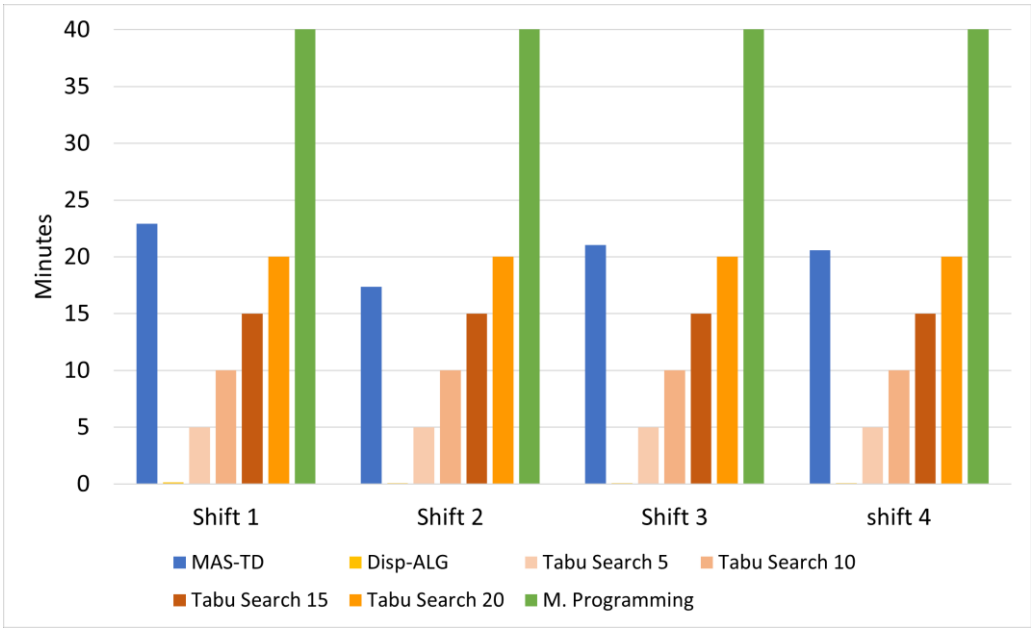

Figure 5.8: Computation time to generate schedules of MAS-TD, Tabu Search, Disp-ALG and Mathematical Programming.

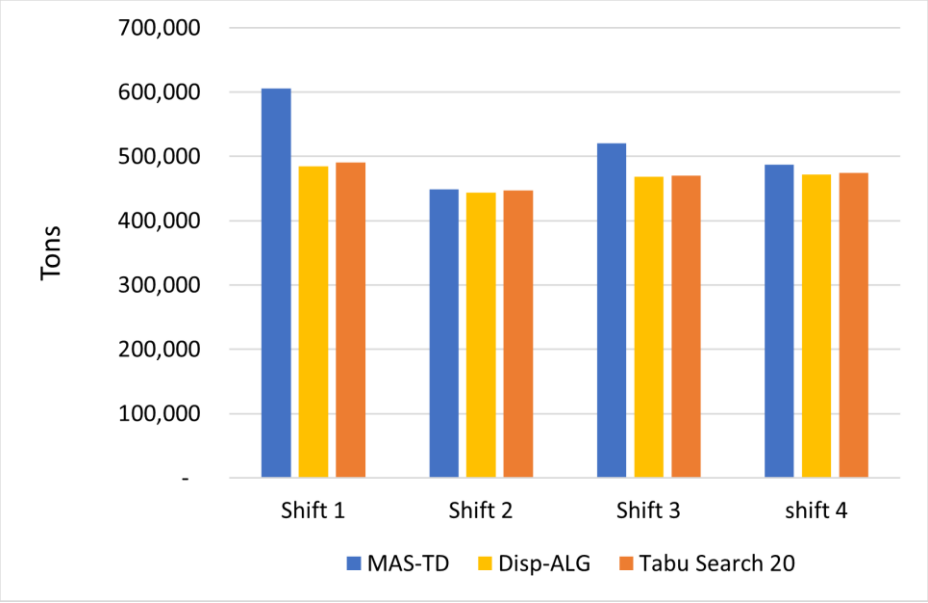

Figure 5.9: Production of MAS-TD, Tabu Search and Disp-ALG.

Regarding costs, Tabu Search decreases the costs of the schedules generated by Disp-ALG by 0.92% on average. The schedules generated by MAS-TD are more expensive than the activities in the schedules generated by Tabu Search in 4.59% on average. This is because the MAS-TD generated schedules with increased production, and therefore more operations must be performed than by the schedules obtained by Tabu Search. Figure 5.10 shows the costs of the generated schedules by the methods for each shift.

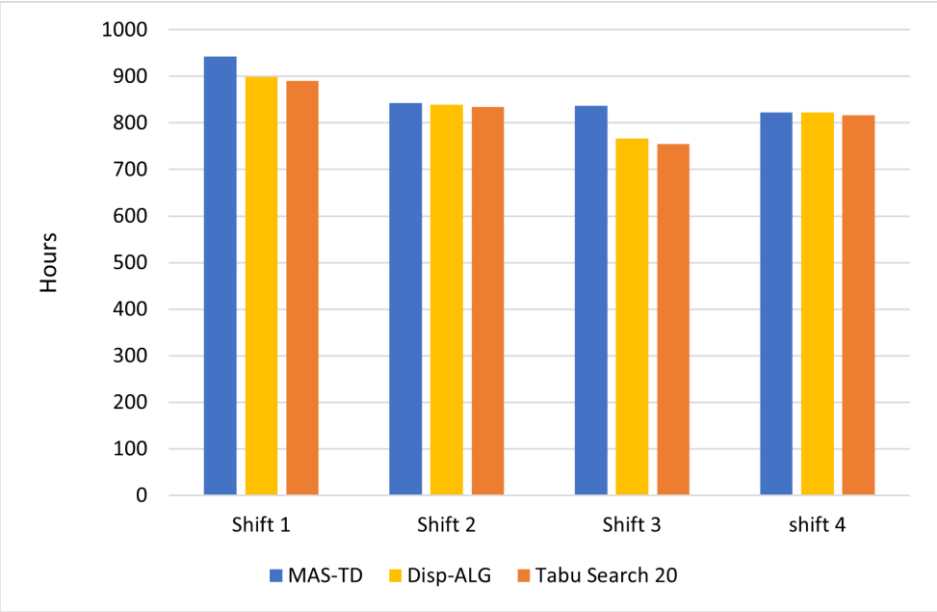

Figure 5.10: Costs of MAS-TD, Tabu Search and Disp-ALG.

Regarding the efficiency of the schedules, determined by the ratio of Quantity of Material Transported/Costs, the generated schedules by MAS-TD are more efficient than the ones obtained by Tabu Search by 5.26% on average. Tabu Search, through the improvement operations, increases the efficiency of the generated schedules by Disp-ALG by 0.93% on average. Figure 5.11 shows the efficiency of the generated schedules by the methods for each shift.

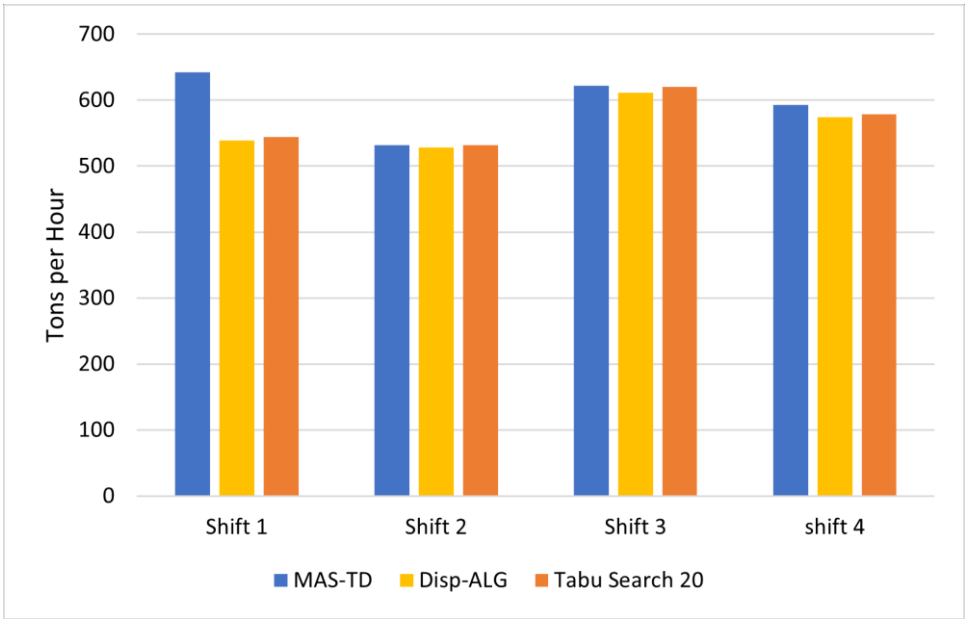

Figure 5.11: Efficiency of the generated schedules by the methods for each shift.

Scheduling with a Production Plan

The simulations aimed to compare the computation time of the methods to generate the schedules, the reached production in the generated schedules, the costs to perform all truck operations, and the efficiency of the schedules. Tabu Search ran with the different pre-set execution times (5, 10, 15, and 20 minutes). Differently from the previous evaluation, all the methods were set to achieve the targets of a production plan at minimum costs. The production plan is based on actual data.

Regarding computation times, the methods have the same pattern as the previous evaluation (for maximizing the production) which means that Mathematical Programming could not generate schedules, Disp-ALG is the fastest method to generate schedules, MAS-TD generates schedules in practical time frames (for the mining industry), and Tabu Search generated the schedules in 5, 10, 15, and 20 minutes as was pre-set. Compared to the previous evaluation, MAS-TD and Disp ALG generate the schedules in less time, because the methods are set to achieve a lower production level than in the previous evaluation. Figure 5.12 shows the computation time of the methods to generate schedules for each shift. Because Mathematical Programming could not generate schedules for the scenarios, the next comparisons only consider MAS-TD, Tabu Search, and Disp-ALG.

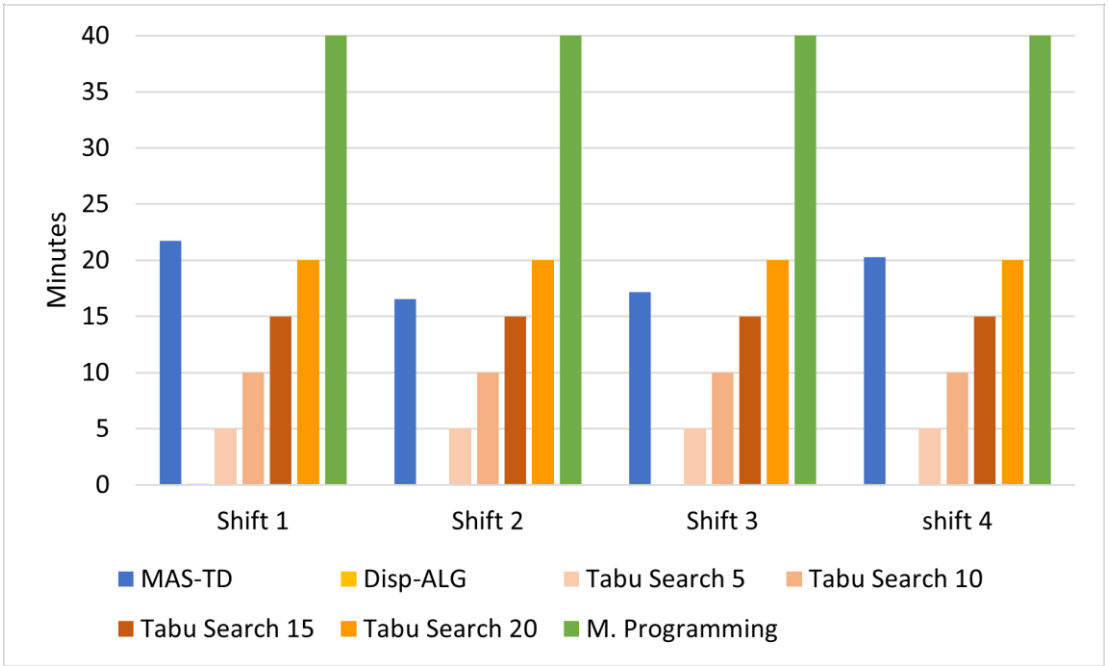

Figure 5.12: Computation time of the methods to generate schedules for each shift.

Regarding production, all the methods generate schedules that reach the production level in the production plan. Due to the fact that the schedules generated by Disp-ALG reach the production level pointed out in the production plan, Tabu Search does not improve these schedules (from the perspective of production level). Figure 5.13 shows the targets in the production plan, and the production level achieved for the methods for each shift.

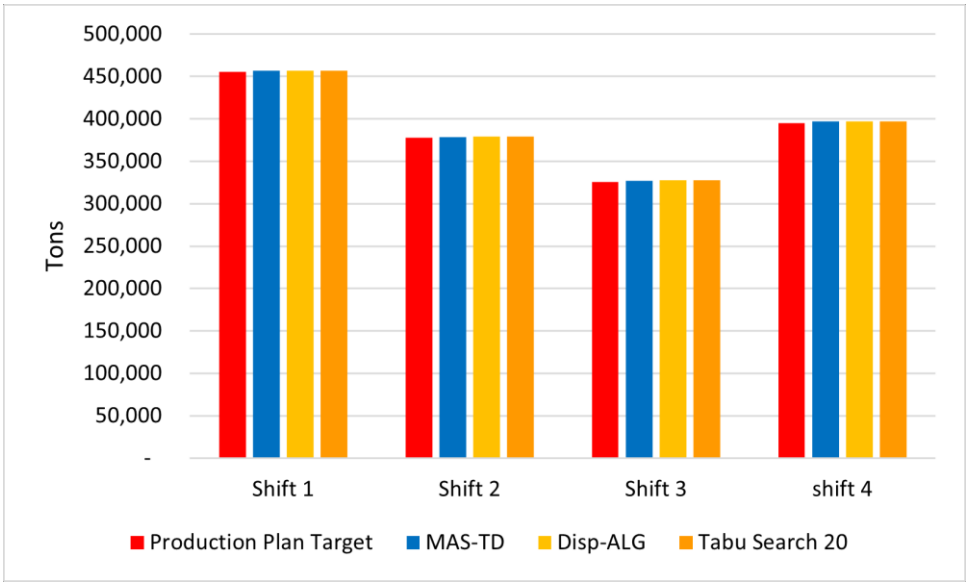

Figure 5.13: Production level planned and production level reached by the methods for each shift.

Regarding costs, Tabu Search decreases the costs of the schedules generated by Disp-ALG by 1.44% on average. The costs in the schedules generated by MAS-TD are lower than the costs in the obtained schedules by Tabu Search by 1.32% on average. Figure 5.14 shows the costs of the generated schedules by the methods for each shift.

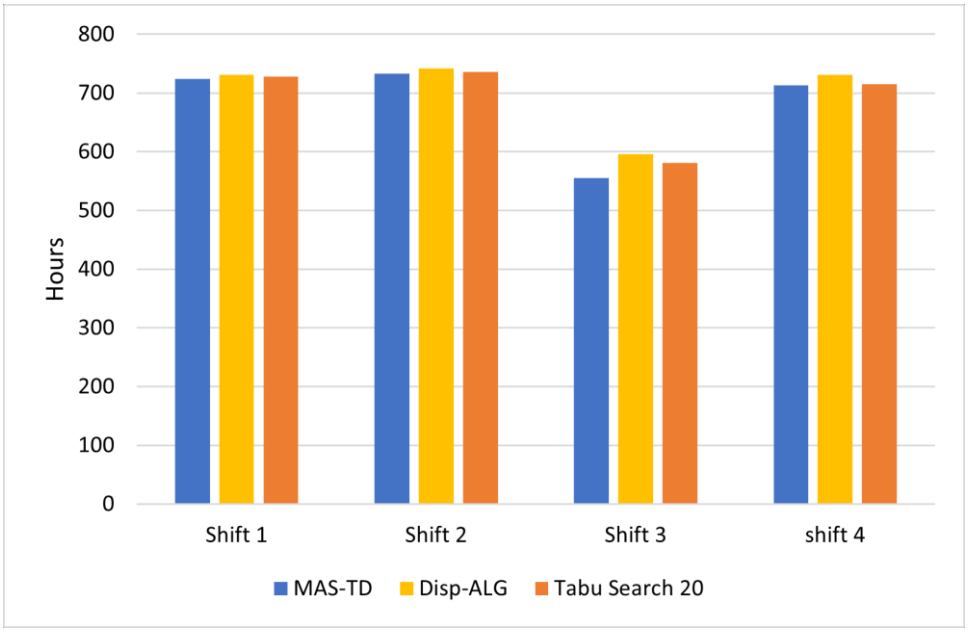

Figure 5.14: Costs of MAS-TD, Tabu Search and Disp-ALG.

Regarding the schedules' efficiency, on average, the generated schedules by MAS-TD are more efficient than the ones obtained by Tabu Search by 1.39%. Tabu Search increases the efficiency of the generated schedules by Disp-ALG by 1.46% on average because of the improvement operations. This efficiency increase is achieved due to the reduction of the costs found by Tabu Search. Figure 5.15 shows the efficiency of the generated schedules by the methods for each shift.

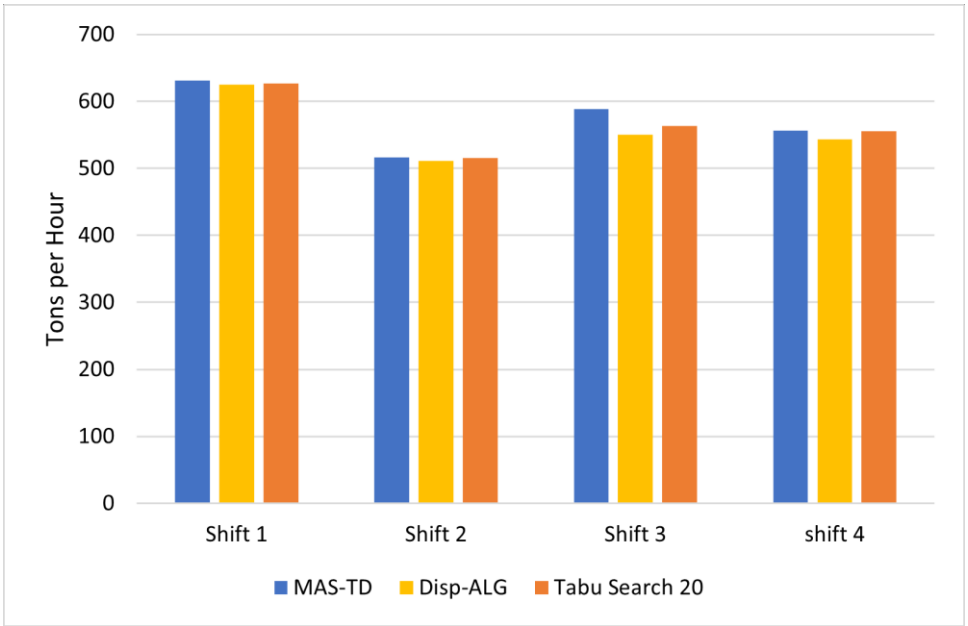

Figure 5.15: Efficiency of generated schedules by MAS-TD, Tabu Search and Disp-ALG.

Rescheduling

To evaluate rescheduling, scenario four was simulated. The simulations evaluated the capacity of MAS-TD, Tabu Search, and Disp-ALG to react when major events occur at the mine. Tabu Search was set to run for five minutes to get a quick solution. In this evaluation, the production reached by the updated schedules were compared. Also, the calculation time to update the schedules was considered in the evaluation.

Regarding computation time to update the schedules, Disp-ALG is the method that reacts fastest to mine events. MAS-TD took much more time than Disp-ALG and Tabu Search. When MAS-TD regenerated the schedules at the beginning of the shift, it took almost the same time that the initial generation of the schedules. However, as shift advances, the time required to regenerate the schedules decreases because the shift time horizon is shorter. This is demonstrated in Table 5.7 where the time required by the MAS-TD to regenerate the schedules decreased as the shift advanced.

Table 5.7: Computation time to update the schedules.

ID equipment	Type of Equipment	Type of Event	Event Time	MAS-TD	TS5	Disp-ALG
CA104	Truck	Start Failure	0:00:00	0:21:02	0:05:00	0:00:05
CA156	Truck	Start Failure	0:00:00		0:05:00	0:00:05
CA93	Truck	Start Failure	0:00:00		0:05:00	0:00:05
CA67	Truck	Start Failure	0:04:04	0:20:58	0:05:00	0:00:05
CF05	Shovel	Start Failure	0:37:37	0:20:02	0:05:00	0:00:05
CA138	Truck	Start Failure	1:11:11	0:19:07	0:05:00	0:00:05
CA60	Truck	Start Failure	1:22:22	0:18:53	0:05:00	0:00:05
PA13	Shovel	Start Failure	1:22:22		0:05:00	0:00:05
CA85	Truck	Start Failure	1:38:38	0:18:42	0:05:00	0:00:05
CA138	Truck	End Failure	2:03:03	0:18:24	0:05:00	0:00:05
CA104	Truck	End Failure	2:09:09	0:18:15	0:05:00	0:00:05
CA162	Truck	Start Failure	2:39:39	0:18:02	0:05:00	0:00:05
CA67	Truck	End Failure	2:40:40	0:18:03	0:05:00	0:00:05
PA06	Shovel	Start Failure	3:06:06	0:17:36	0:05:00	0:00:05
CA60	Truck	End Failure	3:26:26	0:17:25	0:05:00	0:00:05
CA85	Truck	End Failure	3:27:26	0:17:26	0:05:00	0:00:03
CA156	Truck	End Failure	3:28:28	0:17:22	0:05:00	0:00:03
CA93	Truck	End Failure	3:28:28		0:05:00	0:00:03
CA148	Truck	Start Failure	4:06:06	0:16:59	0:05:00	0:00:03
CA162	Truck	End Failure	4:28:27	0:16:39	0:05:00	0:00:03
PA06	Shovel	End Failure	4:31:31	0:16:30	0:05:00	0:00:03
CA148	Truck	End Failure	4:44:43	0:16:16	0:05:00	0:00:03
CA136	Truck	Start Failure	5:57:57	0:15:01	0:05:00	0:00:03
CF05	Shovel	End Failure	5:59:58	0:14:55	0:05:00	0:00:03
CA145	Truck	Start Failure	6:20:20	0:14:22	0:05:00	0:00:03
CA145	Truck	End Failure	6:55:55	0:13:52	0:05:00	0:00:02
CA121	Truck	Start Failure	6:59:59	0:12:24	0:05:00	0:00:02
PA11	Shovel	Start Failure	7:28:28	0:11:23	0:05:00	0:00:02
CA121	Truck	End Failure	7:37:36	0:10:49	0:05:00	0:00:02
CA72	Truck	Start Failure	7:39:39	0:10:28	0:05:00	0:00:02
PA13	Shovel	End Failure	7:42:42	0:10:22	0:05:00	0:00:02
CA92	Truck	Start Failure	7:46:46	0:09:13	0:05:00	0:00:02
CA98	Truck	Start Failure	7:47:47	0:09:13	0:05:00	0:00:02
CA116	Truck	Start Failure	7:55:55	0:09:05	0:05:00	0:00:02
CA92	Truck	End Failure	8:19:19	0:08:28	0:05:00	0:00:02
CA98	Truck	End Failure	8:24:23	0:07:35	0:05:00	0:00:02
CF06	Shovel	Start Failure	8:31:31	0:06:55	0:05:00	0:00:02
CA136	Truck	End Failure	8:45:45	0:06:39	0:05:00	0:00:01
CA55	Truck	Start Failure	8:49:49	0:06:28	0:05:00	0:00:01
CA72	Truck	End Failure	9:02:02	0:05:47	0:05:00	0:00:01
CF06	Shovel	End Failure	9:03:02	0:05:40	0:05:00	0:00:01
CA99	Truck	Start Failure	9:20:20	0:04:02	0:05:00	0:00:01
CA55	Truck	End Failure	9:37:37	0:03:19	0:05:00	0:00:01
CA164	Truck	Start Failure	9:57:57	0:03:08	0:05:00	0:00:01
CA164	Truck	End Failure	10:23:23	0:02:45	0:05:00	0:00:01
CA109	Truck	Start Failure	10:45:45	0:02:22	0:05:00	0:00:01
CA75	Truck	Start Failure	10:46:46	0:02:25	0:05:00	0:00:01
CA116	Truck	End Failure	10:53:52	0:02:00	0:05:00	0:00:01
PA11	Shovel	End Failure	12:00:00			
CA99	Truck	End Failure	12:00:00			
CA109	Truck	End Failure	12:00:00			
CA75	Truck	End Failure	12:00:00			

Regarding production, the evaluation compared two approaches. In an approach, the MAS-TD generates the schedules before the shift starts and performs a complete rescheduling (cf. 3.3.2) as a rescheduling strategy. The other approach is the compound of Tabu Search and Disp-ALG. In this approach, before the beginning of the shift, the schedules are obtained by Tabu Search. Then, for rescheduling, Disp-ALG is used. Tabu Search was not considered for rescheduling because it takes too much time (5 minutes) to react to the mine's new conditions. Although MAS-TD also takes time to regenerate all the schedules, it can quickly provide truck assignments. This is because the negotiation protocol that the agents use to generates the assignments is fast. Therefore, while the agents continue regenerating the entire schedules, the trucks have some assignments to perform already.

Figure 5.16 shows the accumulated production during the shift for each method. The filled lines represent the planned production pointed out in the schedules generated initially. Dotted lines represent the achieved production in the updated schedules. The graph shows that all the methods can update the schedules when major events occur at the mine. However, the updated schedules in MAS-TD achieve a higher production (7.19%) compared to Disp-ALG.

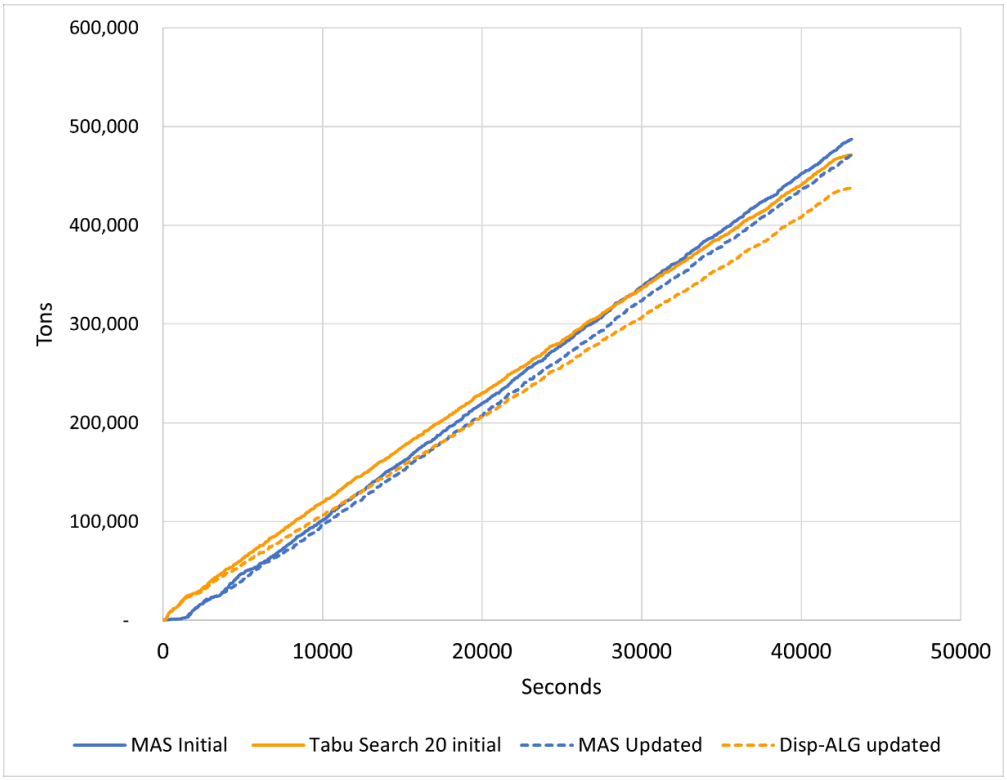

Figure 5.16: Accumulated production during the shift by the schedules.

5.5.2 Discussion

The results demonstrate that MAS-TD and Tabu Search can generate schedules in reasonable time frames (for the mining industry) whether it is for maximizing the production or within the parameters of a production plan. In contrast, Mathematical Programming could not generate the schedules after four hours of runtime for either of the four scenarios, because the truck dispatching problem is modeled as a scheduling problem that is NP-hard. Besides, because the simulation scenarios were big, the solver for the mathematical model could not find the optimal solutions.

MAS-TD and Tabu Search generate the schedules in a similar amount of time, and Disp-ALG was the fastest method to generate the schedules. However, it is important to note that with more CPU kernels so that each agent in MAS-TD could run in its own hardware environment, the time to generate the schedules would be considerably shorter.

From the perspective of production and considering the obtained results for maximizing the production, the schedules generated by MAS-TD achieve a higher production level than the production reached in the generated schedules by Tabu Search and Disp-ALG. This can also be observed in respect to the efficiency of the schedules. The better results in the scheduling solutions provided by MAS-TD are due to the fact that the matching between a shovel task and a truck is more precise. Tabu Search starts the improvement of the schedules generated by Disp-ALG, and Disp-ALG follows a sequential creation of the schedule: it looks for the best truck for a shovel task and then continues with the following shovel task. However, this selected truck may be more appropriate for another shovel task. This causes an inefficient use of the trucks and a lower quality of the schedules. MAS-TD avoids this situation thanks to the concurrent negotiation mechanism: The mechanism allows for a *shovelAgent* to decide on the most appropriate truck proposals. Also, a *truckAgent* can confirm (or not) a proposal sent previously to a shovel with high idle time. This allows for a more precise match between shovels and trucks.

From the perspective of costs, and considering the obtained results of the methods which were set to reach the production targets in a production plan, the costs of the schedules generated by MAS-TD were 1.32% lower than the costs generated by Tabu Search. This happens due to the fact that *truckAgents* have the chance to select a shovel with a lower cost. Disp-ALG also looks for a truck to perform the operations for a shovel in the shortest time. However, when it finds it, there is no chance of changing it in the future,

i.e., if there is another shovel task with less time, it is not possible to change it. Tabu Search might find these improvements, but this might also take more time.

Regarding rescheduling, MAS-TD, Tabu Search, and Disp-ALG can update schedules when major events occur at the mine. However, the production and computation time differs among them that the methods take to generate the schedules. Disp-ALG is the fastest method to update the schedules; however, its schedules are the worst in terms of production and efficiency compared to the generated schedules by MAS-TD and Tabu Search. Nevertheless, Tabu Search requires more time to improve the initial solution. This time is not appropriate when a major event occurs at the mine.

MAS-TD can apply two strategies for rescheduling: schedule repair or complete rescheduling. In the simulation, the complete rescheduling was applied because it reaches the maximum production level. This strategy takes time because it generates all schedules from the beginning. However, in a short time, the agents can agree on some truck assignments in the schedules. This happens due to two reasons: first, the algorithm that generates the schedules in MAS-TD is an anytime algorithm, i.e., it can be stopped at any time still generating a feasible schedule solution; second, the negotiation process is fast and within a few seconds, many assignments can be calculated. This allows the fleet to continue working based on the short-term assignment generated, while MAS-TD continues updating all schedules. Figure 5.17 shows that most of the successful negotiations, i.e., assignments set, are done in less than 1 second in all scenarios.

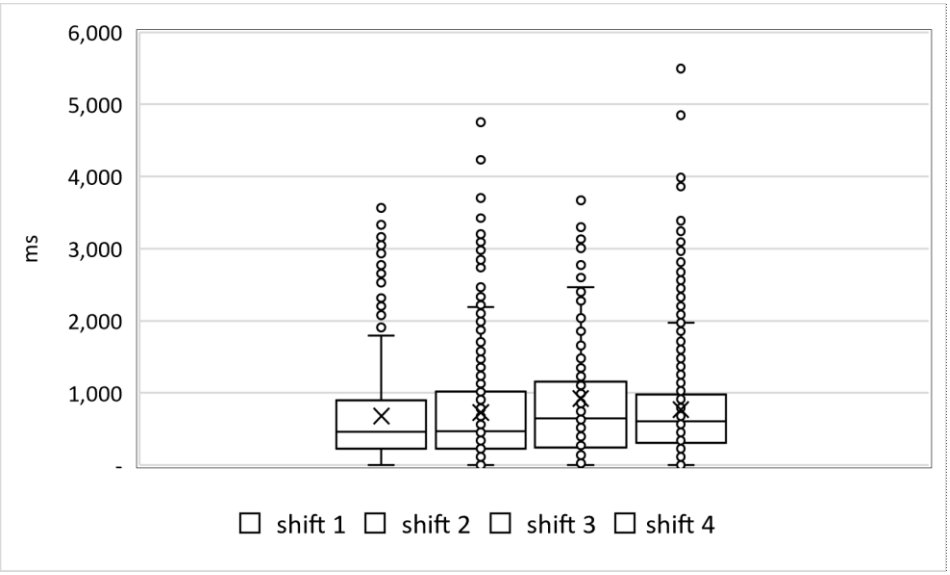

Figure 5.17: Box and whiskers diagram shows that the mean of the duration of the successful negotiations is around 0.5 seconds.

5.6 Evaluation of MAS-TD and an Allocation System

Most of the current systems for the truck dispatching process in open-pit mines are based on an allocation model (cf. Section 2.3). As described in Section 5.2, actual data is generated by an allocation system. This section describes two types of simulations to determine if the MAS-TD's scheduling approach manages a material handling process more efficiently than an allocation approach. The first simulations compare the generation of schedules, and the second simulations observe how the systems react to major events. Both types of simulations compare the obtained result by the MAS-TD against actual data.

5.6.1 Results

Initial Schedules

The first types of simulations compare the production and costs of the schedules generated by the MAS-TD against actual data. The schedules seek to maximize production and minimize the costs. Although actual data reflect the events that occur during the shift, and the generated schedules by MAS-TD do not, this comparison allows to find out how much the production and costs would have been if all the operations would have been organized in advance.

Regarding production, in all scenarios the schedules generated by MAS-TD reached a higher production level than the production achieved with the allocation system. On average, the production is higher by 33.65%. Figure 5.18 shows that the MAS-TD generates schedules with a high production level for the scenarios compared to actual data.

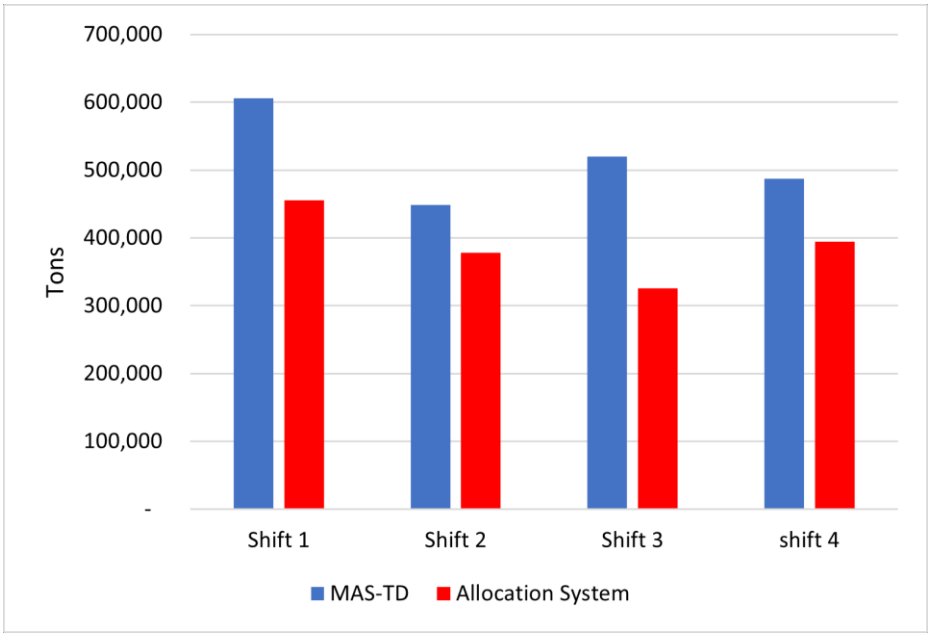

Figure 5.18: Production reached by the schedules generated by the MAS-TD and the production reached by the allocation system.

Regarding costs, MAS-TD generates schedules with higher costs than the costs reached by the assignments determined by the allocation system. On average, the MAS-TD costs are higher by 11.02% than the costs using the allocation system because the generated schedules by the MAS-TD have more operations to perform. Figure 5.19 shows a comparison of the costs of the generated schedules by MAS-TD and the costs associated to the determined assignments by the allocation system.

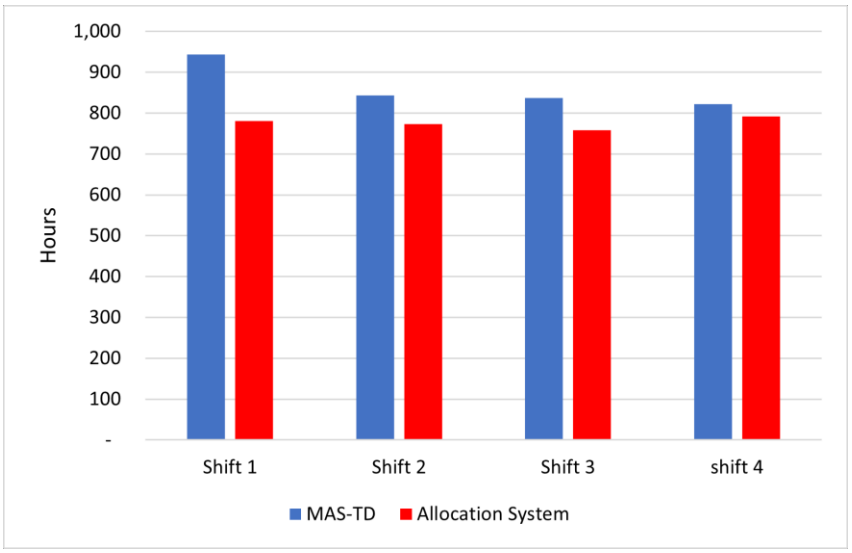

Figure 5.19: Costs reached in the schedules generated by the MAS-TD and the costs reached by the allocation system.

Scheduling with a Production Plan

In this evaluation, MAS-TD was set to generate initial schedules to reach the same production level as the allocation system. The idea was to observe if the MAS-TD generates more efficient schedules even for the same production level.
Regarding production, both MAS-TD and actual data obviously achieve the same production level. However, a marginally increasing production level in the schedules generated by the MAS-TD (0.34%) is observed. Regarding costs, MAS-TD generates schedules with lower costs than the allocation system. On average, the costs decrease by 12.24%. Figure 5.20 shows the cost of the initial schedules generated by MAS-TD and the cost reached by the allocation system.

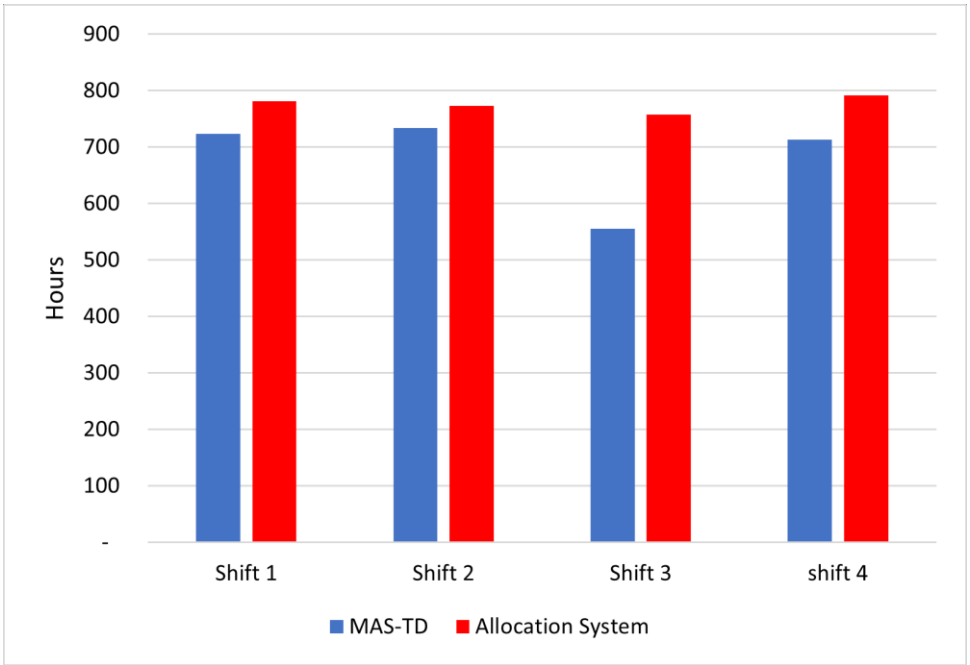

Figure 5.20: Costs reached for the schedules generated by the MAS-TD and the costs reached by the allocation system.

Rescheduling

To evaluate rescheduling, scenario four was simulated. The simulation compares the production, costs and efficiency reached by MAS-TD against the allocation system when important events occur at the mine. For rescheduling, the MAS-TD applies a complete rescheduling strategy (cf. 3.3.2), which means that new schedules are generated from scratch, and the tasks are scheduled from the moment an event occurs.

In the simulated scenario, MAS-TD generates initial schedules for the equipment before the shift starts. After the shift starts, a few trucks and shovels are not available; therefore, the MAS-TD regenerates all the schedules. Next, during the shift, the broken trucks and shovels become available again for work. Each time that a piece of equipment becomes available or unavailable, the MAS-TD regenerates the schedules.

Figure 5.21 shows the accumulated production during the shift. The filled blue line represents the production set for the initial schedules. A dotted blue line represents the accumulated production generated by MAS-TD of the updated schedules. The red line represents the accumulated production level achieved with the allocation system. The graph shows that MAS-TD updates the schedules when a major event occurs at the mine. These updates maintain the objective of maximizing the production and minimizing the costs and demonstrate that the schedules regenerated by MAS-TD reach around 15% more production than the allocation system.

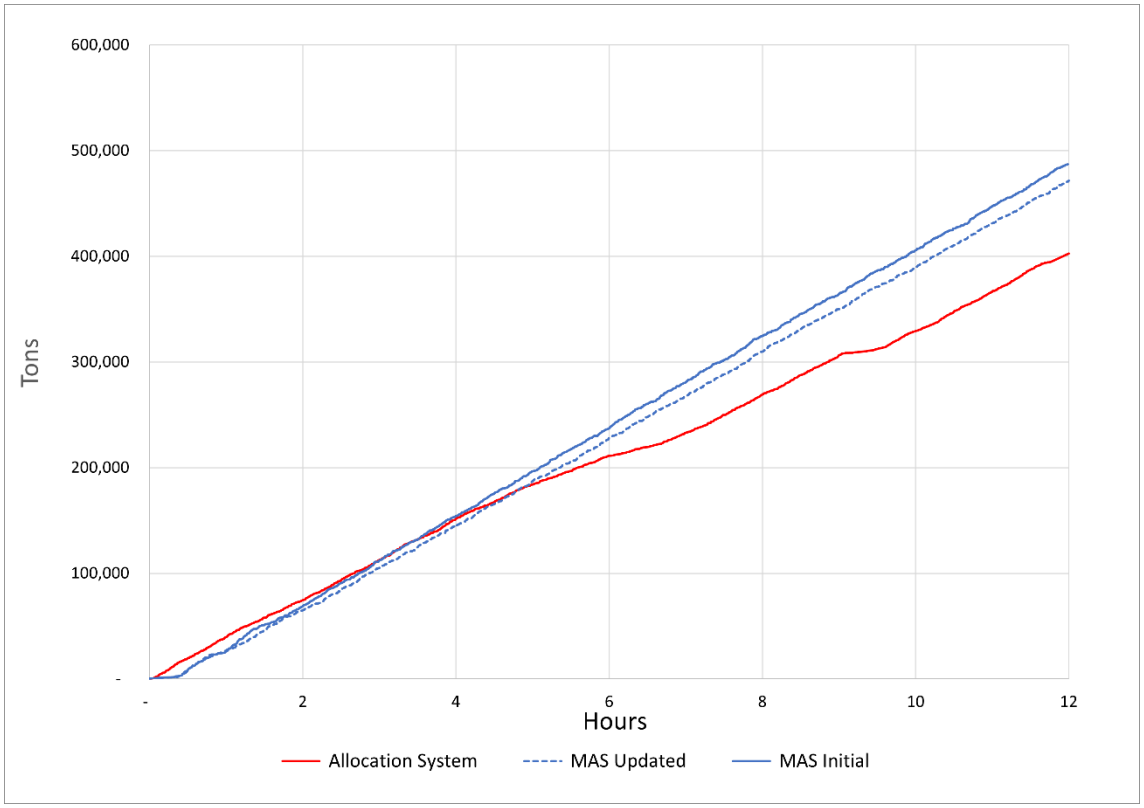

Figure 5.21: Accumulated production during the shift by the schedules generated. by MAS-TD vs. the allocation system.

At the beginning of the shift, the graph shows that the allocation system accumulates more production than the MAS-TD. This happens as in the data retrieved from the mining

company it becomes obvious that some trucks are on the way to unload, while in the simulation of the MAS-TD all the trucks start empty, and therefore are on the way to a shovel for loading.

Table 5.8 shows a comparison of the material handling performances by using the MAS-TD and the allocation systems. The performances are shown through the production, costs, and efficiency obtained by applying both methods. It observes that the MAS-TD (Updated schedules) reaches around 50,000 tons more production than the allocation system, representing an increase of 12.41%. Regarding the efficiency, the MAS-TD increases around 75,000 tons/hour, representing an improvement of 13.2% approx.

Table 5.8: Overview of the material handling process performance

	MAS-TD (Initial Schedules)	**MAS-TD (Updated Schedules)**	**Allocation System**
Production (tons)	487,214	459,142	402,745
Costs (hours)	822.08	783.91	791.59
Efficiency (tons/hour)	592.66	585.71	508.78

5.6.2 Discussion

Regarding scheduling, the results show that the schedules generated by the MAS-TD support the material handling process more efficiently. On the one hand, with the objective of maximizing the production, the MAS-TD generates schedules that improve the fleet operations; therefore, the obtained production is higher than the one achieved by the allocation system. On the other hand, when a production plan exists, the schedule costs to perform the operations are lower than the activity costs that follow from the assignments generated by the allocation system. The main reason for these savings is that the MAS-TD generated travel times of a truck are shorter than the travel times shown in the actual data since the MAS-TD agents use the shortest path for their trip. In contrast, the truck operators in the real world decide by themselves which path to follow. Another reason is the use of specific data that allows for better adapted calculations of each equipment item's activity times. In this way, the agents can create more appropriate and efficient schedules.

Regarding rescheduling, the MAS-TD regenerates the schedules appropriately. MAS-TD

maintains the objective of maximizing the production and minimizing the costs during each regeneration of the schedules; Figure 5.21 shows that when an important event occurs, MAS-TD keeps the production level trend. For instance, at times 05:33 (hh:mm) and 08:53, there were important failures. In both situations, the trend in the accumulated production obtained by the allocation system decreases more than the trend of the accumulated production in the schedules of the MAS-TD. This can be interpreted as an improvement provided by MAS-TD in obtaining a more resilient material handling process than the one provided by the allocation system.

Finally, the results demonstrate that MAS-TD achieves a better performance for the material handling process in open-pit mines than the allocation system because of the following reasons:

- Facing the truck dispatching problem as a scheduling problem instead of a dynamic scheduling problem (i.e., without schedules) allows for a more global view of the problem.
- The multiagent technology allows to use specific information. It enables agents to make precise calculations and improve their decision-making process based on these calculations.
- The negotiation mechanism allows the agents to choose the best option that helps them to achieve the system's objective, in this case, to maximize the production and minimize the costs.

5.7 Conclusions

This chapter has presented the evaluation of the Multiagent System for truck dispatching in open-pit mines. The chapter starts with the evaluation design, in which the purpose, methods, and materials are described. After this, the actual data used in the simulations, the mining company, and how the data was retrieved and managed is described. Two methods to generate schedules that were later used in the evaluations, are then discussed: Mathematical Programming and a metaheuristic algorithm. Later, two evaluations are presented: the first one compares the results obtained by the methods previously mentioned, vs. MAS-TD results. The second one compares the solution provided by the MAS-TD vs. a real-world system based on the allocation approach.

The first conclusion concerns the MAS-TD and the other methods to generate schedules and rescheduling. The comparison with Tabu Search and Mathematical Programming

demonstrates that MAS-TD is a better option than those methods, for both scheduling and rescheduling. The results show that MAS-TD achieves a higher production and efficiency level.

The second conclusion concerns the MAS-TD and the allocation system. The comparison was done not only between a centralized approach and a distributed one; it was also between a scheduling model and an allocation model (the model most currently used in open-pit mining). The comparison demonstrates that MAS-TD enhances the material handling process by generating schedules and the rescheduling of them when major events affect the fleet. The MAS-TD increases production levels (or decreases the costs if a production plan is used) compared to the production and the costs obtained by an allocation system.

Finally, returning to the main question that this chapter aimed to answer, the simulation results demonstrate that the MAS-TD is a better alternative than current methods to manage the material handling process in open-pit mines.

6 CONCLUSIONS AND OUTLOOK

The preceding chapters have presented the MAS-TD as a novel and superior solution to enhance the scheduling for the material handling processes of open-pit mines'. The MAS-TD generates schedules for the equipment involved in the processes by applying an adapted contract-net protocol that manages concurrent negotiations. The dynamic schedule update is also provided by the MAS-TD when major events affect the equipment.

This thesis covers the detailed problem description, the foundations of multiagent technology and scheduling theory to develop the MAS-TD concepts, and the evaluation of the multiagent solution compared with different methods. The contributions of the thesis go in two directions: On the one hand, and from the mining perspective, it provides a new way to organize mining fleet activities, replacing an allocation approach by a schedule approach. This is a novel way to dispatch trucks in the mining industry. On the other hand, it provides an adapted negotiation protocol between software agents that efficiently manages concurrent negotiations. This protocol can also be applied to different domains.

This chapter finalizes these contributions with a concluding summary and a discussion of future research potentials on concurrent negotiation in dynamic environments. To this end, Section 6.1 presents the summary of all preceding chapters and their conclusions. The section describes also how this thesis responds to the presented research questions in Chapter 1. Section 6.2 provides further application work focusing on the extension of the developed MAS-TD and its application on other domains. Finally, Section 6.3 provides further research directions.

6.1 Summary and Conclusions

The truck dispatching problem in the material handling process in open-pit mines was the real-world process that initiated this research. Starting from a problem statement, the

research defined objectives, research questions, hypotheses, and a four-stage methodology. All these aspects organizing the research are described in Chapter 1.

Chapter 2 summarizes the state-of-the-art of the material handling process of open-pit mines. In particular, it is focused on the truck dispatching problem and on the methods that solve it. Most of these methods follow a centralized approach and are based on an allocation model. These methods obtain information from the pieces of equipment and determine a truck's following destination each time it is required. In other words, no schedules are used for this approach which leads to the statement that material handling in open-pit mines is not performed efficiently. For instance, queues of trucks built up in front of shovels or crushers, and shovels must wait for the truck arrivals. This generates inefficiencies, high costs, and missed production targets. This leads to the first research question:

1. How can a material handling process in an open-pit mine be supported more efficiently?

The chapter's conclusions mention two ways to improve the material handling process efficiency. The first one is to use a scheduling model instead of an allocation model. The second one is to use a distributed approach instead of a centralized approach.

The reason to use a scheduling model is that it provides a better solution than an allocation model. The allocation model is very useful in dynamic environments because it quickly determines a dispatch solution each time it is required. However, the solution quality can be poor due to the myopic view and the difficulty of predicting system performance (Ouelhadj and Petrovic, 2009). In contrast, a scheduling model generates schedules (and updates them if necessary), which leads to a better system performance, because the schedules are constructed considering a global view. However, generating schedules takes more time, which could be not appropriate for dynamic environments, which require fast updates. Nevertheless, multiagent technology can handle this situation, too.

There are several reasons to use a distributed approach instead of a centralized approach: It avoids a dependency on a central node, takes advantage of parallel processing, uses specific local information, and its models are closer to reality. In the context of scheduling, a distributed approach can take advantage of parallel processing to calculate schedules in practical frame times. However, other aspects must also be considered, such as the required communication effort between software agents and their synchronization.

Chapter 3 presents the foundations of multiagent technology and scheduling theory.

These are the main elements of a solution to solve the truck dispatching problem in open-pit mines and to overcome the identified weaknesses in the current solutions discussed in Chapter 2. The chapter describes different aspects of multiagent technology, such as structures to organize agents, agent architectures, communication, and interaction mechanisms that agents can employ.

The scheduling theory section describes different types of scheduling problems and a notation to model them. Rescheduling is also covered by studying of approaches and strategies for rescheduling. The last part of the chapter shows how multiagent systems have been applied to solve scheduling and rescheduling problems. This chapter answers the second research question:

2. Which communication and interaction mechanism among agents allows for generating schedules?

The chapter's concludes that a contract-net protocol is a suitable interaction mechanism to generate schedules. The reasons are:

- Scalability: It does not need modifications if the number of agents increases that take part in the negotiation.
- Flexibility: The protocol can be used for different types of negotiations. For instance, one type of negotiation could be to generate schedules. In this case, *shovelAgents* and *truckAgents* negotiate with each other to generate the schedules. Another type of negotiation is to improve the schedules. In this case, *truckAgents* negotiate with each other to swap assignments. In both cases, the same protocol is applied.
- Anytime algorithm: The contract-net protocol exchange may be stopped before an optimal solution is reached, and the best solution found is returned. This characteristic is very important in a stochastic and dynamic environment.

The chapter's conclusions also state that the protocol must deal with the concurrency problem to take advantage of parallel processing.

Chapter 4 introduces MAS-TD, the developed multiagent system to solve the truck dispatching problem in open-pit mines. It describes how agents are organized in MAS-TD and how they negotiate to generate the schedules. Also, it shows an adapted contract-net protocol to cope with concurrent negotiations. In the protocol, *shovelAgents* play the role of initiators, and *truckAgents* play the role of participants. *ShovelAgents* send *call-*

for-proposal messages to *truckAgents* pointing out their next available time for loading operation. *TruckAgents* decide to respond with a proposal or refuse the *call-for-proposal* message. The *truckAgent's* proposal contains the information when the truck will arrive at the shovel and the cost to perform all operations (in terms of time). Finally, the *shovelAgent* decides on the best proposal by applying an evaluation function that promotes those proposals that arrive on time and at smaller costs.

In addition, the chapter presents experiments performed to get the appropriate configuration setting to achieve the best performance of the MAS-TD. The experiments were performed with scenarios based on actual data. This leads to the third research question:

3. How can an agent make a more efficient decision during concurrent negotiations in dynamic environments?

The chapter's conclusions describe that the adapted contract-net protocol is suitable to cope with concurrent negotiations in dynamic environments. The adaption considers a confirmation stage in which an initiator agent requires a *requestConfirmation* message from a participant agent who sent a proposal. The agent who receives the *requestConfirmation* message can reject the *requestConfirmation* message if it is taking part in another negotiation that may potentially be more beneficial. Additionally, the chapter demonstrates that the adapted contract-net protocol is suitable to update the schedules when major events occur in the environment.

In the context of MAS-TD, *truckAgents* can refuse the *requestConfirmation* message sent by a shovel if they are taking part in a negotiation, which could save costs. Besides, if an important event occurs at the mine, the protocol can be used to update the schedules. For instance, in the case of a truck failure, some *shovelAgents* are affected by the cancelation of the broken truck's loading operations. Therefore, these *shovelAgents* will start new negotiation processes to update their schedules (partial rescheduling). Now suppose that many failures occur in a short period of time. In that case, all *shovelAgents* and *truckAgents* will cancel their operations, and the *shovelAgents* will start new negotiation processes to regenerate all the schedules (complete rescheduling).

Chapter 5 describes the evaluation of the MAS-TD. The evaluation is performed in two parts as there are two main approaches to solve the problem: the scheduling and allocation approach. The first part evaluates the MAS-TD by comparing it with other methods that also generate schedules. The second one compares the MAS-TD with an

allocation system used in a real-world mining company. Both evaluations include also rescheduling.

In the first evaluation, MAS-TD is compared with Mathematical Programming and the Tabu Search algorithm. The evaluation demonstrates that MAS-TD generates schedules that reach a higher production level than the other methods. Besides, the computation time needed to generate the schedules is shorter than the one of the other methods except the Disp-ALG, which is an ad-hoc algorithm that provides an initial solution to the Tabu Search algorithm. However, the production levels achieved with its generated schedules was much lower (20% less) than the production levels reached by MAS-TD. These findings can also be observed when major events occur: MAS-TD reschedules its schedules in a more productive (7.19% more) and faster way than the other methods.

In the second evaluation, MAS-TD is compared with an allocation system used in a real-world open-pit mine. The initial schedules generated by MAS-TD are updated when a truck or shovel failure occurs. The results demonstrate that a scheduling-based system achieves a better material handling performance than an allocation system. This is true for the production levels as well as cost: The production levels reached are higher (12.41% more) than the ones of the allocation system. Regarding costs, for the same production levels the MAS-TD schedules can be implemented at lower costs (12.24% less).

As a general conclusion, the MAS-TD based on the scheduling approach and agents able to manage concurrent negotiations, improve the material handling process performance in open-pit mines versus methods applied today.

6.2 Future Application Work

The thesis describes the MAS-TD as a solution to deal with the truck dispatching problem in open-pit mines by generating schedules. In addition, the system can be extended to cover also other processes in a mining company. Furthermore, since scheduling is a task in many industries, MAS-TD can be adapted to different domains. The following section outlines these perspectives for future applications of MAS-TD.

6.2.1 Extension of MAS-TD

There are also other processes that influence the material handling processes in open-pit mines. For instance, processes in processing plants and equipment maintenance processes

affect the material handling process performance. In these cases, the MAS-TD can be extended by including new agents related to other processes. In this way, the MAS-TD would provide an even better reaction to the influences of other processes.
Another extension would be to add agents representing other machines and activities directly related to the material handling process. For instance, water trucks, graders, and other machines are used to maintain the routes in good conditions. With new agents representing these pieces of equipment, the MAS-TD would have more information to generate more precise and realistic schedules.

6.2.2 Application in Other Domains

Truck dispatching is a problem in many domains. Some of them are quite similar to the truck dispatching problem in open-pit mines. For instance, there are cranes in ports that take containers from a vessel and load trucks that transport the containers to different destinations inside the port. Before starting the shift, and considering incoming vessels' information, the MAS-TD could generate the trucks' and cranes' schedules.
Another quite similar process is in modern warehouses that use autonomous guided vehicles (AGVs). In these warehouses, a robot takes an item from a shelf and drops the item into an AGV. With the information of incoming orders, the MAS-TD could generate the schedules for the AGVs and the robot. Obviously, the MAS-TD must be adapted to these domains
Although the current MAS-TD is applied to transport processes, it is also possible to use it in other different domains as MAS-TD uses the scheduling approach; therefore, it can be adapted and used for scheduling problems, such as flow shop problems. In this case, the agents must represent machines, jobs, and resources as well as job orders. They can use the adapted contract-net protocol with the confirmation stage to generate schedules.

6.3 Future Research Directions

From a research perspective, several areas could be further investigated to improve the developed approach. Firstly, it would be interesting to investigate the subcomponents of the multiagent system. For instance, instead of using a contract-net protocol with confirmation stage to manage concurrent negotiations, it could be used alternative approaches. One alternative is based on probability analysis. In this alternative, the idea is to analyze the probability of winning a more attractive negotiation than another that is

already feasible. Another alternative is based on commitment level. This alternative allows an agent to participate in several negotiations sequentially. The agent can decommit earlier contracts when it finds out that a new contract is more attractive. In case it decommits a previous contract, a penalty must be considered. Maybe all these alternatives could be used together by the agents: they could decide which is more appropriate depending on the context.

From the perspective of rescheduling, this thesis has used both the partial and complete regeneration of schedules by applying the adapted contract-net protocol with confirmation stage. However, there are other ways to update the schedules, such as right-shift and match-up operations. These alternatives have been applied in other domains successfully with a centralized approach. Therefore, the investigation of these alternatives in a distributed approach such as MAS-TD could improve its performance.

Furthermore, it would be interesting to investigate how prediction and anticipation of future events, such as equipment delays and traffic problems, can be integrated. In this way, the agents could generate more robust schedules. In this context, the knowledge generated from historical data would be used. By such a learning mechanism, the proactive behavior of the system could be improved.

Finally, although JADE provides an appropriate platform for MAS-TD, it would be interesting to evaluate the MAS-TD performance on other platforms as there are many platforms for the development and the simulation of multiagent systems that could be better base options than JADE.

7 REFERENCES

Adams, K. K., and Bansah, K. K. (2016). Review of Operational Delays in Shovel-Truck System of Surface Mining Operations. *4th UMaT Biennial International Mining and Mineral Conference*, 60–65. Tarkwa, Ghana.

Agnetis, A., Billaut, J. C., Gawiejnowicz, S., Pacciarelli, D., and Soukhal, A. (2014). Multi Agent scheduling: Models and algorithms. In *Multiagent Scheduling: Models and Algorithms*. Berlin, Germany: Heidelberg Springer. https://doi.org/10.1007/978-3-642-41880-8

Aknine, S., Pinson, S., and Shakun, M. F. (2004). An extended multi-agent negotiation protocol. *Autonomous Agents and Multi-Agent Systems*, *8*(1), 5–45. https://doi.org/10.1023/B:AGNT.0000009409.19387.f8

Alarie, S., and Gamache, M. (2002). Overview of Solution Strategies Used in Truck Dispatching Systems for Open Pit Mines. *International Journal of Surface Mining, Reclamation and Environment*, *16*(1), 59–76. https://doi.org/10.1076/ijsm.16.1.59.3408

Alexandre, R. F., Campelo, F., Fonseca, C. M., and de Vasconcelos, J. A. (2015). A Comparative Study of Algorithms for Solving the Multiobjective Open-Pit Mining Operational Planning Problems. In A. Gaspar-Cunha, C. Henggeler Antunes, & C. Coello (Eds.), *Evolutionary Multi-Criterion Optimization. EMO 2015. Lecture Notes in Computer Science* (Vol. 9019, pp. 433–447). Springer, Cham. https://doi.org/10.1007/978-3-319-15892-1

Araújo, F. C. R., and Souza, M. J. F. (2011). A heuristic for the open-pit mining operational planning problem with dynamic truck allocation. *Rem: Revista Escola de Minas*, *64*(1), 69–76.

Bajany, D. M., Xia, X., and Zhang, L. (2017). A MILP Model for Truck-shovel Scheduling to Minimize Fuel Consumption. In H. Li, J. Yan, F. Sun, U. Desideri, & S. K. Chou (Eds.), *Energy Procedia* (Vol. 105, pp. 2739–2745). beijing, China: Elsevier Ltd. https://doi.org/10.1016/j.egypro.2017.03.925

Bakhtavar, E., and Mahmoudi, H. (2020). Development of a scenario-based robust model for the optimal truck-shovel allocation in open-pit mining. *Computers and*

Operations Research, *115*(March 2020), 104539. https://doi.org/10.1016/j.cor.2018.08.003

Bastos, Guilherme S, Souza, L. E., Ramos, F. T., and Ribeiro, C. H. (2011). A single-dependent agent approach for stochastic time-dependent truck dispatching in open-pit mining. *14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 1057–1062. IEEE. https://doi.org/10.1109/ITSC.2011.6082902

Bastos, Guilherme Sousa. (2010). *Methods for Truck Dispatching in Open-Pit Mining*. Aeronautics Institute of Technology, São José dos Campos.

Bellifemine, F., Caire, G., and Greenwood, D. (2007). Developing Multi-Agent Systems with JADE. In *Developing Multi-Agent Systems with JADE*. Chichester, UK: John Wiley & Sons. https://doi.org/10.1002/9780470058411

Berndt, J. O. (2015). *Self-Organizing Multiagent Negotiations*. Dissertation. Universität Bremen.

Bissiri, Y. (2002). *Application of Agent-Based Modeling to Truck-Shovel Dispatching Systems in Open Pit Mines*. University of British Columbia.

Burt, C. N., and Caccetta, L. (2007). Match factor for heterogeneous truck and loader fleets. *International Journal of Mining, Reclamation and Environment*, *21*(4), 262–270. https://doi.org/10.1080/17480930701388606

Çetin, N. (2004). Open-pit truck/shovel haulage system simulation. Middle East Technical University.

Cetin, N., Erarslan, K., and Okuducu, A. (2001). Computer simulation of track/shovel system at Tuncbilek coal mine using GPSS/H. *17th International Mining Congress and Exhibition of Turkey - IMCET 2001*, 715–718.

Chang, Y., Ren, H., and Wang, S. (2015). Modelling and optimizing an open-pit truck scheduling problem. *Discrete Dynamics in Nature and Society*, *2015*, 745378. https://doi.org/10.1155/2015/745378

Chaowasakoo, P., Seppälä, H., Koivo, H., and Zhou, Q. (2017a). Digitalization of mine operations: Scenarios to benefit in real-time truck dispatching. *International Journal of Mining Science and Technology*, *27*(2), 229–236. https://doi.org/10.1016/j.ijmst.2017.01.007

Chaowasakoo, P., Seppälä, H., Koivo, H., and Zhou, Q. (2017b). Improving fleet management in mines: The benefit of heterogeneous match factor. *European Journal of Operational Research*, *261*(3), 1052–1065. https://doi.org/10.1016/j.ejor.2017.02.039

Chargui, K., El fallahi, A., Reghioui, M., and Zouadi, T. (2019). A reactive multi-agent approach for online (re)scheduling of resources in port container terminals. *IFAC-PapersOnLine*, *52*(13), 124–129. https://doi.org/10.1016/j.ifacol.2019.11.163

Chilena, M. (2015). Grandes camiones y palas: Un match que apunta a mayor rendimiento. Retrieved February 1, 2021, from https://www.mch.cl/informes-tecnicos/grandes-camiones-y-palas-un-match-que-apunta-a-mayor-rendimiento

Chilena, M. (2019). Una de las palas más grandes del mundo dejó Chuqui y llega a DMH. Retrieved March 5, 2021, from https://www.mch.cl/2019/12/31/una-de-las-palas-mas-grandes-del-mundo-dejo-chuqui-y-llega-a-dmh/

Collahuasi. (2016). Collahuasi - Mucho mas que cobre. Retrieved April 25, 2021, from http://www.collahuasi.cl/que-hacemos/nuestra-operacion/

Costa, F. P. Da, Souza, M. J. F., and Pinto, L. R. (2005). Um modelo de programação matemática para alocação estática de caminhões visando ao atendimento de metas de produção e qualidade. *Rem: Revista Escola de Minas*, *58*(1), 77–81. https://doi.org/10.1590/S0370-44672005000100013

Crawford, H., and Barker, L. (2018). Bias in The Spotlight: Heuristics. Retrieved December 15, 2020, from https://www.researchworld.com/bias-in-the-spotlight-heuristics/

Dantzig, G. B., and Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science*, *6*(1), 80–91. https://doi.org/10.1287/mnsc.6.1.80

Fazlirad, A., and Brennan, R. W. (2018). Multiagent Manufacturing Scheduling: An Updated State of the Art Review. *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, 722–729. Munich, Germany: IEEE. https://doi.org/10.1109/COASE.2018.8560576

Feng, P., Chen, H., Peng, S., Chen, L., and Li, L. (2015). A method of distributed multi-satellite mission scheduling based on improved contract net protocol. *2015 11th International Conference on Natural Computation (ICNC)*, 1062–1068. Zhangjiajie, China: IEEE. https://doi.org/10.1109/ICNC.2015.7378139

Finin, T., Fritzson, R., McKay, D., and McEntire, R. (1994). KQML as an agent communication language. In R. A. Nabil, K. B. Bharat, & Y. Yelena (Eds.), *Proceedings of the Third International Conference on Information and Knowledge Management* (pp. 456–463). Gaithersburg, Maryland, USA: Association for Computing Machinery.

Foundation for Intelligent Physical Agents. (2002a). *FIPA Brokering Interaction*

Protocol Specification.

Foundation for Intelligent Physical Agents. (2002b). *FIPA Communicative Act Library Specification*.

Foundation for Intelligent Physical Agents. (2002c). *FIPA Iterated Contract Net Interaction Protocol Specification*.

Foundation for Intelligent Physical Agents. (2002d). *FIPA Propose Interaction Protocol Specification*.

Foundation for Intelligent Physical Agents. (2002e). *FIPA Query Interaction Protocol Specification*.

Foundation for Intelligent Physical Agents. (2002f). *FIPA Recruiting Interaction Protocol Specification*.

Foundation for Intelligent Physical Agents. (2002g). *FIPA Request Interaction Protocol Specification*.

Foundation for Intelligent Physical Agents. (2002h). *FIPA Request When Interaction Protocol Specification*.

Foundation for Intelligent Physical Agents. (2002i). *FIPA Subscribe Interaction Protocol Specification*.

Foundation For Intelligent Physical Agents. (2002a). *FIPA ACL Message Structure Specification. Standar. Document No. SC00061G* (p. 11). p. 11.

Foundation For Intelligent Physical Agents. (2002b). *FIPA Contract Net Interaction Protocol Specification. Standard. Document No. SC00029H* (p. 9). p. 9.

Franklin, S., and Graesser, A. (1996). Is It an Agent, or Just a Program?: A Taxomony of Autonomous Agents. In J.P. Müller, M. J. Wooldridge, & N. R. Jennings (Eds.), *Intelligent Agents III Agent Theories, Architectures, and Languages. ATAL 1996. Lecture Notes in Computer Science, vol 1193* (pp. 21–35). Budapest, Hungary: Springer Berlin Heidelberg.

Gath, M. (2015). *Optimizing Transport Logistics Processes with Multiagent-based Planning and Control*. Dissertation. Universität Bremen.

Gehlhoff, F., and Fay, A. (2020). On agent-based decentralized and integrated scheduling for small-scale manufacturing. *At-Automatisierungstechnik*, *68*(1), 15–31. https://doi.org/10.1515/auto-2019-0105

Gehrke, J. D., Schuldt, A., and Werner, S. (2008). Quality Criteria for Multiagent-Based Simulations with Conservative Synchronisation. In M. Rabe (Ed.), *13th ASIM Dedicated Conference on Simulation in Production and Logistics (ASIM 2008)* (pp.

545–554). Berlin, Germany: Berlin, Germany Fraunhofer IRB Verlag.

Geobrugg. (2020). Underground mining: how to increase productivity using advanced fortification systems for tunnels. Retrieved January 25, 2021, from https://www.geobrugg.com/es/Underground-mining-how-to-increase-productivity-using-advanced-fortification-systems-for-tunnels-123788.html?seite_neu=event_detail_es

Glover, F. (1989). Tabu Search - Part I. *Orsa Journal on Computing*, *1*(3), 190–206.

Google. (2021). Location, CMDIC. Retrieved May 18, 2021, from https://www.google.de/maps/place/Collahuasi/@-25.8222416,-65.5779907,4z/data=!4m8!1m2!2m1!1scollahuasi!3m4!1s0x9154bbcca9e7282f:0x3bf175ff12ea1bb!8m2!3d-20.9814192!4d-68.6379674

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, *5*(1979), 287–326.

Granichin, O., Skobelev, P., Lada, A., Mayorov, I., and Tsarev, A. (2013). Cargo transportation models analysis using multi-agent adaptive real-time truck scheduling system. In J. Filipe & A. Fred (Eds.), *Proceedings of the 5th International Conference on Agents and Artificial Intelligence - Volume 2* (Vol. 2, pp. 244–249). Barcelona, Spain: SciTePress. https://doi.org/10.5220/0004225502440249

Gurgur, C. Z., Dagdelen, K., and Artittong, S. (2011). Optimisation of a real-time multi-period truck dispatching system in mining operations. *International Journal of Applied Decision Sciences*, *4*(1), 57. https://doi.org/10.1504/IJADS.2011.038091

Hashemi, A. S., and Sattarvand, J. (2015). Simulation Based Investigation of Different Fleet Management Paradigms in Open Pit Mines-A Case Study of Sungun Copper Mine. *Archives of Mining Sciences*, *60*(1), 195–208. https://doi.org/10.1515/amsc-2015-0013

He, M. X., Wei, J. C., Lu, X. M., and Huang, B. X. (2010). The genetic algorithm for truck dispatching problems in surface mine. *Information Technology Journal*, *9*(4), 710–714. https://doi.org/10.3923/itj.2010.710.714

Horling, B., and Lesser, V. (2004). A survey of multi-agent organizational paradigms. *Knowledge Engineering Review*, *19*(4), 281–316. https://doi.org/10.1017/S0269888905000317

Huhns, M. N., and Singh, M. P. (1998). *Readings in Agents*. San Francisco, CA, United States: Morgan Kaufmann.

Hund, K., La Porta, D., Fabregas, T. P., Laing, T., and Drexhage, J. (2020). Minerals for Climate Action: The Mineral Intensity of the Clean Energy Transition. *Climate Smart Mining Initiative - The World Bank Group*, p. 110 pp. The World Bank.

IBM ILOG. (2009). IBM ILOG CPLEX V12.1 User's Manual for CPLEX. In *International Business Machines Corporation*. Armonk, NY, USA: IBM.

Icarte, G., Berrios, P., Castillo, R., and Herzog, O. (2020). A Multiagent System for Truck Dispatching in Open-pit Mines. In M. Freitag, H. Haasis, H. Kotzab, & P. J. (Eds.), *Dynamics in Logistics. LDIC 2020. Lecture Notes in Logistics* (pp. 363–373). Bremen, Germany: Springer, Cham. https://doi.org/10.1007/978-3-030-44783-0_35

Icarte, G., and Herzog, O. (2018). A comparison of mechanisms for optimal decisions in open-pit mining. In B. G. Lottermoser (Ed.), *Abstract volume First International Conference Mines of the Future : 23 -24 May 2018 / MRE Institute of Mineral Resources Engineering, RWTH Aachen University* (p. 74). Aachen, Germany: Verlag Mainz.

Icarte, G., and Herzog, O. (2019). A multi-agent system for truck dispatching in an open-pit mine. In B. Lottermoser (Ed.), *Abstracts of the Second International Conference Mines of the Future 13 & 14 June 2019, Institute of Mineral Resources Engineering, RWTH Aachen University*. Aachen, Germany: Verlag Mainz.

Icarte, G., Rivero, E., and Herzog, O. (2020). An Agent-based System for Truck Dispatching in Open-pit Mines. In A. Rocha, L. Steels, & J. van den Herik (Eds.), *Proceedings of the 12th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART* (pp. 73–81). Valleta, Malta: SciTePress.

Jaoua, A., Riopel, D., and Gamache, M. (2009). A framework for realistic microscopic modelling of surface mining transportation systems. *International Journal of Mining, Reclamation and Environment*, *23*(1), 51–75. https://doi.org/10.1080/17480930802351479

Katz, J., Cárdenas, K., and Cáceres, J. (2000). *Instituciones y Tecnología en el Desarrollo Evolutivo de la Industria Minera Chilena*. CEPAL.

Kin, W., and Chan, V. (2011). Foundations of Simulation Modeling. In J. J. Cochran (Ed.), *Wiley Encyclopedia of Operations Research and Management Science*. New York, USA.: John Wiley & Sons. https://doi.org/10.1002/9780470400531.eorms0336

Koç, Ç., Bektaş, T., Jabali, O., and Laporte, G. (2016). Thirty years of heterogeneous vehicle routing. *European Journal of Operational Research*, *249*(1), 1--21.

Koenig, S., Tovey, C., Lagoudakis, M., Markakis, V., Kempe, D., Keskinocak, P., … Jain, S. (2006). The Power of Sequential Single-Item Auctions for Agent Coordination. In A. Cohn (Ed.), *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2* (Vol. 2, pp. 1625–1629). Boston, MA, USA: AAAI Press.

Koryagin, M., and Voronov, A. (2017). Improving the organization of the shovel-Truck systems in open-pit coal mines. *Transport Problems*, *12*(2), 113–122. https://doi.org/10.20858/tp.2017.12.2.11

Krzyzanowska, J. (2007). The impact of mixed fleet hauling on mining operations at Venetia mine. *Journal of the Southern African Institute of Mining and Metallurgy*, *107*(4), 215–224.

La Segunda, O. (2015). Trabajadores del cobre inician movilizaciones por falta de acuerdo con Codelco. Retrieved November 25, 2020, from http://www.lasegunda.com/Noticias/Economia/2015/07/1018613/Trabajadores-del-cobre-inician-movilizaciones-por-falta-de-acuerdo-con-Codelco

Liao, T. W., Chang, P. C., Kuo, R. J., and Liao, C. J. (2014). A comparison of five hybrid metaheuristic algorithms for unrelated parallel-machine scheduling and inbound trucks sequencing in multi-door cross docking systems. *Applied Soft Computing Journal*, *21*(2014), 180–193. https://doi.org/10.1016/j.asoc.2014.02.026

Lin, F., Dewan, M. A. A., and Nguyen, M. (2018). Optimizing Rescheduling Intervals through Using Multi-Armed Bandit Algorithms. *2018 IEEE International Conference on Internet of Things (IThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 746–753. Halifax, Canada: IEEE. https://doi.org/10.1109/Cybermatics

Lindo Systems Inc. (2013). LINGO and optimization modeling.

Lizotte, Y., Bonates, E., and Leclerc, A. (1987). A design and implementation of a semi-automated truck/shovel dispatching system. In I. C. Lemmer, H. Schaum, & F. A. G. M. Camisani-Calzolari (Eds.), *APCOM 87: Proceedings of the 20th International Symposium on the Application of Computers and Mathematics in the Mineral Industries*: *Vol. I* (pp. 377–387). Johannesburg, South Africa: South African Institute of Mining and Metallurgy.

Lopes Silva, M. A., de Souza, S. R., Freitas Souza, M. J., and Bazzan, A. L. C. (2019). A reinforcement learning-based multi-agent framework applied for solving routing and

scheduling problems. *Expert Systems with Applications*, *131*(2019), 148–171. https://doi.org/10.1016/j.eswa.2019.04.056

Lou, P., Ong, S. K., and Nee, A. Y. C. (2010). Agent-based distributed scheduling for virtual job shops. *International Journal of Production Research*, *48*(13), 3889–3910. https://doi.org/10.1080/00207540902927918

Madhyastha, M., Reddy, S. C., and Rao, S. (2017). Online scheduling of a fleet of autonomous vehicles using agent-based procurement auctions. *2017 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, 114–120. Bari, Italy: IEEE. https://doi.org/10.1109/SOLI.2017.8120980

Martin, S., Ouelhadj, D., Beullens, P., Ozcan, E., Juan, A. A., and Burke, E. K. (2016). A multi-agent based cooperative approach to scheduling and routing. *European Journal of Operational Research*, *254*(1), 169–178. https://doi.org/10.1016/j.ejor.2016.02.045

Mas, A. (2005). *Agentes software y sistemas multiagente: conceptos, arquitecturas y aplicaciones*. Madrid: Prentice Hall.

Mendes, Joao Batista;, Maia, R. D., D'Angelo, M. F. S. V., and Vasconcelos, J. A. de. (2014). Algoritmo Evolucionario Aplicado ao Problema de Despacho de Caminhoes em Minas a Ceu Aberto. *Anais Do XX Congresso Brasileiro de Automática*, 4300–4307. Belo Horizonte, Brazil.

Mendes, Joao Batista, D'Angelo, M. F. S. V., Maia, N. A., and Veloso, R. R. (2016). A Hybrid Multiobjective Evolutionary Algorithm for Truck Dispatching in Open-Pit-Mining. *IEEE Latin America Transactions*, *14*(3), 1329–1334.

Micholopulos, T. N., and Panagiotou, G. N. (2001). Truck allocation using stochastic goal programming. *10th International Symposium on Mine Planning and Equipment Selection*, 965–970. New Delhi, India.

Mohtasham, M., Mirzaei Nasirabad, H., and Mahmoodi Markid, A. (2017). Development of a goal programming model for optimization of truck allocation in open pit mines. *Journal of Mining and Environment*, *8*(3), 359–371. https://doi.org/http://doi.org/10.22044/jme.2017.859

Moradi Afrapoli, A., and Askari-Nasab, H. (2017). Mining fleet management systems: a review of models and algorithms. *International Journal of Mining, Reclamation and Environment*, *33*(1), 42–60. https://doi.org/10.1080/17480930.2017.1336607

Moradi Afrapoli, A., Tabesh, M., and Askari-Nasab, H. (2019). A multiple objective transportation problem approach to dynamic truck dispatching in surface mines.

European Journal of Operational Research, *276*(1), 331–342. https://doi.org/10.1016/j.ejor.2019.01.008

Morgenstern, O., and Von Neumann, J. (1953). *Theory of games and economic behavior*. USA: Princeton university press.

Müller, H. J. (1997). Towards agent systems engineering. *Data and Knowledge Engineering*, *23*(3), 217–245. https://doi.org/10.1016/s0169-023x(97)00013-x

Müller, Jörg P., and Fischer, K. (2014). Application Impact of Multi-agent Systems and Technologies: A Survey. In O. Shehory & A. Sturm (Eds.), *Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks* (pp. 27–53). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-54432-3_3

Munirathinam, M., Yingling, J. C., Munirathinarn, M., and Yingling, J. C. (1994). A review of computer-based truck dispatching strategies for surface mining operations. *International Journal of Surface Mining, Reclamation and Environment*, *8*(1), 1–15. https://doi.org/10.1080/09208119408964750

Newman, A. M., Rubio, E., Caro, R., Weintraub, A., and Eurek, K. (2010). A review of operations research in mine planning. *Interfaces*, *40*(3), 222–245. https://doi.org/10.1287/inte.1090.0492

Noriega, P., and Sierra, C. (1998). Subastas y Sistemas Multiagente. *Revista Iberoamericana de Inteligencia Artificial*, *2*(6), 68–84. https://doi.org/10.4114/ia.v2i6.620

Nwana, H. S. (1996). Software Agents: An Overview. *The Knowledge Engineering Review*, *11*(3), 205–244.

Ouelhadj, D., and Petrovic, S. (2009). A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling*, *12*(4), 417–431. https://doi.org/10.1007/s10951-008-0090-8

Ozdemir, B., and Kumral, M. (2018). Appraising production targets through agent-based Petri net simulation of material handling systems in open pit mines. *Simulation Modelling Practice and Theory*, *87*(2018), 138–154. https://doi.org/10.1016/j.simpat.2018.06.008

Patterson, S. R., Kozan, E., and Hyland, P. (2017). Energy efficient scheduling of open-pit coal mine trucks. *European Journal of Operational Research*, *262*(2), 759–770. https://doi.org/10.1016/j.ejor.2017.03.081

Perez-Gonzalez, P., and Framinan, J. M. (2014). A common framework and taxonomy

for multicriteria scheduling problems with interfering and competing jobs: Multi-agent scheduling problems. *European Journal of Operational Research*, *235*(1), 1–16. https://doi.org/10.1016/j.ejor.2013.09.017

Pinedo, M. L. (2008). *Scheduling. Theory, Algorithms, and Systems*. New York, NY, USA: Springer. https://doi.org/10.1007/978-0-387-78935-4

Russell, S. J., and Norvig, P. (2010). Artificial Intelligence A Modern Approach Third Edition. In *Pearson*. Upper Saddle River, NJ, USA: Pearson. https://doi.org/10.1017/S0269888900007724

Sáez, N. E. (2014). *SIMULACIÓN ON-LINE PARA EL DESPACHO DE CAMIONES MINEROS EN OPERACIONES A CIELO ABIERTO*.

Schillo, M., Kray, C., and Fischer, K. (2002). The eager bidder problem: a fundamental problem of DAI and selected solutions. *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 2*, (January), 599–606. Bologna, Italy: Association for Computing Machinery. https://doi.org/10.1145/544862.544886

Searle, J. R. (1969). *Speech Act Theory*. Cambridge, MA, USA: Cambridge University Press.

Seitaridis, A., Rigas, E. S., Bassiliades, N., and Ramchurn, S. D. (2020). An agent-based negotiation scheme for the distribution of electric vehicles across a set of charging stations. *Simulation Modelling Practice and Theory*, *100*(2020), 102040. https://doi.org/10.1016/j.simpat.2019.102040

Shen, W., Member, S., Wang, L., and Hao, Q. (2006). Agent-Based Distributed Manufacturing Process Planning and Scheduling: A State-of-the-Art Survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *36*(4), 563–577. https://doi.org/10.1016/S0015-1882(98)90658-9

Siebers, P.-O., and Aickelin, U. (2008). Introduction to Multi-Agent Simulation. In F. Adam & P. Humphreys (Eds.), *Encyclopedia of Decision Making and Decision Support Technologies* (pp. 554--564). USA: IGI Global. https://doi.org/10.4018/9781599048437.ch062

Sinha, S. M. (2006). Mathematical Programming: Theory and Methods. In *Operations Research*. New Delhi, India: Elsevier Science. https://doi.org/10.1287/opre.8.1.101

Skobelev, P. (2011). Multi-agent systems for real time resource allocation, scheduling, optimization and controlling: Industrial applications. In V. Mařík, P. Vrba, & P. Leitão (Eds.), *Holonic and Multi-Agent Systems for Manufacturing. HoloMAS 2011.*

Lecture Notes in Computer Science (Vol. 6867, pp. 1–14). Toulouse, France: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-23181-0_1

Skobelev, P., Budaev, D., Brankovsky, A., and Voschuk, G. (2018). Multi-agent tasks scheduling for coordinated actions of unmanned aerial vehicles acting in group. *International Journal of Design and Nature and Ecodynamics*, *13*(1), 39–45. https://doi.org/10.2495/DNE-V13-N1-39-45

Smith, R. G. (1980). The Contract Net Protocol:High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, *C–29*(12), 1104–1113. https://doi.org/10.1109/TC.1980.1675516

Subtil, R. F., Silva, D. M., and Alves, J. C. (2011). A Practical Approach to Truck Dispatch for Open Pit Mines. In E. Y. Baafi, R. J. Kininmonth, & P. I (Eds.), *35th APCOM Symposium* (pp. 765–777). Wollongong, NSW, Australia: AusIMM.

Ta, C. H. (2015). *Optimal Haul Truck Allocation in the Syncrude Mine*. Master Thesis. University of Alberta.

Tan, Y., and Takakuwa, S. (2016). A practical simulation approach for an effective truck dispatching system of open pit mines using VBA. *2016 Winter Simulation Conference (WSC)*, 2394–2405. Washington, DC, USA: IEEE.

Temeng, V. a., Otuonye, F. O., and Frendewey, J. O. (1997). Real-time truck dispatching using a transportation algorithm. *International Journal of Surface Mining, Reclamation and Environment*, *11*(4), 203–207. https://doi.org/10.1080/09208119708944093

Toptal, A., and Sabuncuoglu, I. (2010). Distributed scheduling: A review of concepts and applications. *International Journal of Production Research*, *48*(18), 5235–5262. https://doi.org/10.1080/00207540903121065

TZI, C. for C. and C. T. (2011). PlaSMA Multiagent Simulation. Retrieved February 25, 2021, from https://plasma.informatik.uni-bremen.de/

Upadhyay, S. P., and Askari-Nasab, H. (2016). Truck-shovel allocation optimisation: a goal programming approach. *Mining Technology*, *125*(2), 82–92. https://doi.org/10.1179/1743286315Y.0000000024

Vieira, G. E., Herrmann, J. W., and Lin, E. (2003). Rescheduling manufacturing systems: A framework of strategies, policies, and methods. *Journal of Scheduling*, *6*(1), 39–62. https://doi.org/10.1023/A:1022235519958

Wang, J., Zhang, Y., Liu, Y., Wu, N., and Member, S. (2018). Multiagent and Bargaining-Game-Based Real-Time Scheduling for Internet of Things-Enabled Flexible Job

Shop. *IEEE Internet of Things Journal*, *6*(2), 2518–2531. https://doi.org/10.1109/JIOT.2018.2871346

Warden, T., Porzel, R., Gehrke, J. D., Herzog, O., Langer, H., and Malaka, R. (2010). Towards Ontology-Based Multiagent Simulations : Plasma Approach. In A. Bargiela, S. Azam Ali, D. Crowley, & E. J. Kerckhoffs (Eds.), *Proceedings of the 24th European Conference on Modelling and Simulation, ECMS 2010* (pp. 50–56). Kuala Lumpur, Malaysia.

Weiss, G. (1999). Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. In *Foundations*. Cambridge, USA; London, UK: The MIT Press. https://doi.org/10.1007/s10709-010-9480-x

Whitbrook, A., Meng, Q., and Chung, P. W. H. (2018). Reliable, Distributed Scheduling and Rescheduling for Time-Critical, Multiagent Systems. *IEEE Transactions on Automation Science and Engineering*, *15*(2), 732–747. https://doi.org/10.1109/TASE.2017.2679278

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

White, J. W., Olson, J. P., and Vohnout, S. I. (1993). On improving truck/shovel productivity in open pit mines. *CIM Bulletin*, *86*(973), 43--49.

Woodridge, M. (2001). *An Introduction to Multiagent Systems*. Chichester, UK: John Wiley & Sons.

Wooldridge, M. (1997). Agent-Based Software Engineering. *IEE Proceedings - Software Engineering*, *144*(1), 26–37.

World Economic Forum. (2015). Mining & Metals in a Sustainable World 2050. *World Economic Forum*, 1–44. Geneva, Switzerland.

Xu, T., Shi, F., and Liu, W. (2019). Research on Open-pit Mine Vehicle Scheduling Problem with Approximate Dynamic Programming. *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, 571–577. Taipei, Taiwan: IEEE. https://doi.org/10.1109/ICPHYS.2019.8780275

Zweben, M., and Fox, M. (1994). *Intelligent Scheduling*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.