Máster Universitario en Ingeniería de Análisis de Datos,
Mejora de Procesos y Toma de Decisiones

Departamento de Estadística e Investigación Operativa Aplicadas y Calidad

Matheuristics for the parallel machine scheduling problem with resource needs during setups

Zhang Ling

Supervisor: Federico Perea Rojas Marcos

Contents

1 Introduction **1**

2 Scheduling Problems **4**

2.1 Introduction to Scheduling Problems 4

2.1.1 Some Areas of Application 5

2.1.2 Classification 6

2.2 Parallel Machines 8

2.2.1 Identical Parallel Machines 10

2.2.2 Uniform Parallel Machines 12

2.2.3 Unrelated Parallel Machines 15

2.2.4 Unrelated Parallel Machines Scheduling with Setup Times (UPMS) 17

2.2.5 Unrelated Parallel Machines Scheduling with Resources in Setups (UPMR-S) 22

3 Formal Definition of the Problem **26**

3.1 Sets, Variables and Parameters 26

3.1.1 Sets 27

3.1.2 Variables 27

3.1.3 Parameters 28

3.2 MILP Model 28

4 Matheuristics Proposed 31

4.1 Algorithm 1 33

4.1.1 Model 1 of Algorithm 1: The Constructive Phase 34

4.1.2 Model 2 of Algorithm 1: The Repairing Phase . 35

4.2 Algorithm 2 36

4.2.1 Model 1 of Algorithm 2: The Constructive Phase 36

4.2.2 Model 2 of Algorithm 2: The Repairing Phase . 38

5 Computational Experiments 39

5.1 Solver and Computer 39

5.2 Generation of Instances 40

5.3 Results 40

6 Conclusion 46

Abstract

This thesis proposes two algorithms for the Unrelated Parallel Machines Scheduling problem with resources in setups (UPMR-S). The objective of UPMR-S is to minimize the completion time (makespan). This problem is a NP-hard problem. The problem consists of assigning a number of jobs to a group of unrelated parallel machines in order to minimize the makespan. We first introduce a mixed integer linear programming (MILP) model that has been proposed in an earlier research. Then, we propose two two-stage matheuristics for the UPMR-S. Some computational experiments are carried out and the results verify the effectiveness of the algorithms proposed.

Keywords: Unrelated parallel machine scheduling, limited resources, setup time, makespan

Chapter 1

Introduction

The Unrelated Parallel Machines Scheduling problem with resources in setups (UPMR-S) is a problem that has not been paid much attention in the literature.

A real manufacturing system is mostly made up of several parallel machines that can process jobs at the same time. Because of the complexity of parallel machine scheduling problems, exact methods may not find the solutions in a reasonable amount of time. Therefore, heuristics are needed to obtain the approximate (near) optimal schedules for these problems.

(Lenstra et al., 1990), propose a polynomial algorithm and a polynomial approximation scheme for Unrelated Parallel Machine Scheduling problem. Both results are about the theorem of the relationship of a class of integer programming problems and their linear programming relaxation.

There are many published papers on parallel machine scheduling problems. To cite a few, (Ruiz & Andrés-Romano, 2011) use heuristic approaches for Unrelated Parallel Machines Scheduling problem with resource-assignable sequence dependent setup times. In this paper, the length of setups can be changed according to the amount of resources settled for the setups. For this problem, the objective function used is a linear combination of total completion time and the total amount of resources allocated.

The paper of (Vallada & Ruiz, 2011) proposes a genetic algorithm which consists of a rapid local search and a local search enhanced crossover operator for the unrelated parallel machine scheduling problem with setups.

The paper of (Edis & Oguz, 2011) presents a Parallel Machine Scheduling problem with extra resources which uses a Lagrangian based constraint programming approach to minimize the makespan. The author first introduces an integer programming model of the problem, then uses the nonnegative Lagrangian multipliers to relax the resource constraint and obtain a maximum lower bound. Finally, a constraint programming model is used to generate feasible schedules with efficient upper bounds.

In this thesis, we propose two two-stage matheuristics to find solutions for the Unrelated Parallel Machines Scheduling problem with resources in setups (UPMR-S). The rest of this thesis is organized as follows: Chapter 2 presents scheduling problems. Chapter 3 presents a MILP model for the UPMR-S. Chapter 4 introduces two two-stage matheuristics, which are the main contribution of this thesis and Chapter 5 presents computational experiments. Finally, some conclusions are presented in Chapter 6.

Chapter 2

Scheduling Problems

2.1 Introduction to Scheduling Problems

Scheduling is an area of research that can be classified within the field of computer science and operations research. Scheduling problems are formulated in different environments, computer systems, manufacturing and project scheduling. These processes include complex activities to be scheduled. We can model scheduling problems through jobs, machines, relationships between jobs and machines, processors, additional resources and other parameters. The purpose of modeling is to apply the most appropriate algorithm to find optimal or sub-optimal schedules in the sense of a given criterion. The definition of scheduling is the temporary assignment of resources to activities, in order to achieve some desirable goals, and determining the order of execution of actions in a behavior description (Leung, 2004). A couple of examples of scheduling problems are:

- Assignment of workers to machines in a factory to increase productivity.
- Allocation of processor and communication computations to network links to minimize application execution time.

Solution techniques for scheduling problems include constraint planning, mathematical programming, heuristics and meta-heuristics, and more. However, most of these technologies are designed for particular problems, and they require a lot of adjustments to be applied to different problems. The goal of machine scheduling is to assign a set of jobs to a group of machines, in order to optimize an objective function which leads to minimizing operating times and increasing productivity (Ouazene et al., 2014).

2.1.1 Some Areas of Application

Scheduling problems have been faced at many levels and in many areas. In general, we can apply them to manufacturing production and computer systems or project management. Lately, researchers have been working to solve problems in new applications, such as logistics, airport operations, management processes, decentralized systems and self-centered organizations, grid computing and scheduling issues in bioinformatics (Toksarı et al., 2014).

2.1.2 Classification

The classification scheme we will use consists of three parts $\alpha\backslash\beta\backslash\gamma$ (Kayvanfar et al., 2014), where:

- α : The environment of machines.
- β : Characteristics of jobs or resources.
- γ : Objective functions.

We now explain in more detail each of these parts.

Machine Environment α

1. α_1 represents the type of machines.
    - $\alpha_1 = P$: Identical Machines.
    - $\alpha_1 = Q$: Uniform Machines.
    - $\alpha_1 = R$: Unrelated Machines.
2. α_2 represents the number of machines.

Characteristics and Constraints β

- β_1 represents restrictions of preemption.
- β_2 represents additional resources.
- β_3 represents restrictions of precedence.
- β_4 describes available time.
- β_5 describes the type of processing time.

- β_6 describes the type of due dates.
- β_7 represents eligibility restrictions.

Objective Function γ

We can divide the optimization goals into two categories: completion time and costs. For example, some optimization goals mainly measure the time of completion of the last job (makespan), or measure lateness of jobs in relation to a deadline. The cost-related optimization objectives mainly measure the expenses of using machines, and the waiting cost before and after the machine processing. In the literature, the goal most commonly encountered is the ”minimax” criteria, which is used to calculate the maximum completion time of all the jobs: min C_{max}, where C_{max} is the total length of a schedule: the completion time of the last job. Another theory is the ”minisum” criteria, such as ”minisum” of the total tardiness of jobs. (T'kindt & Billaut, 2006)

We now enumerate some ”minisum” criteria and ”minimax” criteria.

1. minisum
   - Total completion time (makespan).
   - Total flow time.
   - Total tardiness.
   - Total number of tardy jobs.
2. minimax

- Lateness.
- Tardiness.
- Makespan.

2.2 Parallel Machines

In general, Parallel Machines Scheduling implies the assignment of machines and resources to jobs to complete all the jobs under the imposed restrictions. The following are some of the restrictions in the classic scheduling theory:

1. Each job is to be processed by only one machine.
2. Each machine can only handle at most one job at a time.
3. Once a machine starts processing a job, it will continue to run on that job until the job is completed (no preemption).

The Parallel Machines Scheduling problem is an optimization problem, which includes decision variables, constraints on the decision variables, and objective functions defined by decision variables. Decision variables are unknowns that need to be found. In any standard scheduling problem, there are a set of jobs $N = \{1, 2, ..., n\}$ and a set of machines $M = \{1, 2, ..., m\}$. A job $j \in N$ needs p_{ij} time units to process on machine $i \in M$. We assume that a job will not be interrupted when this job begins processing on a particular machine (no preemption). The problem we consider consists of scheduling a sequence of jobs on machines to minimize the maximum completion time (makespan) of the jobs. That is, we want to decrease the time before the last job is completed to the minimum..

The Parallel Machines Scheduling is subject to feasibility constraints and optimization objectives. Some additional constraints may be related to the resources required during the setup process and setup times between a pair of jobs. Typically, the optimization objective of such problems is to find a schedule that minimizes the total amount of time or expenses required to finish all the assignments.

According to its job processing speed, three types of parallel machines can be distinguished. If all machines have the same job processing speed, we call them Identical Parallel Machines. When machine speeds are different, two cases may happen. If the speed of each machine is constant and independent of the job, they are named Uniform Parallel Machines; however, if the speed of the machine depends on the specific job being processed, they are called Unrelated Parallel Machines.

For Identical Parallel Machines, all machines have the same speed, and the processing times of the job p_{ij} are the same on distinctive machines, $p_{ij} = p_j$, and the processing speed will not be changed because of changes in machine.

For Uniform Parallel Machines, machines have different speeds and processing times of a job varied by speed factors, $p_{ij} = p_j/s_i$, where s_i is the speed factor of machine i. Lastly, for Unrelated Parallel Machines, p_{ij} is arbitrary and does not have special features (Blazewicz et al., 2007).

2.2.1 Identical Parallel Machines

The Identical Parallel Machines problem is denoted by $(P_m||C_{max})$. For the Identical Parallel Machines problem, every job requires the same processing times on each machine; that is, all machines have the same job processing speed. In this case, the processing times are the same on all machines, $p_{1j} = p_{2j} = p_{3j} = ... = p_{mj}$; therefore, minimizing completion time (makespan), is the equivalent of balancing workload in the machines. In manufacturing industry, balancing the workload among the machines is vital to decrease the idle times and work-in-process. It helps also to get bottlenecks out in manufacturing systems (Ouazene et al., 2014).

There are the following assumptions in this Identical Parallel Machines problem :

1. All machines are identical and are able to process all jobs.
2. Each machine can only process one job at a time.
3. Jobs are only allowed to be processed on one machine.
4. At time zero, all jobs are available to be processed (German et al., 2016).

Example of Identical Parallel Machines

We now show an example with three Identical Parallel Machines. Table 2.1 shows the processing times p_{ij}, where we can see the processing times of each job j (each column) are the same on all three machines.

	Job 1	**Job 2**	**Job 3**	**Job 4**
Machine 1	4	6	3	9
Machine 2	4	6	3	9
Machine 3	4	6	3	9

Table 2.1: p_{ij} for three identical parallel machines.

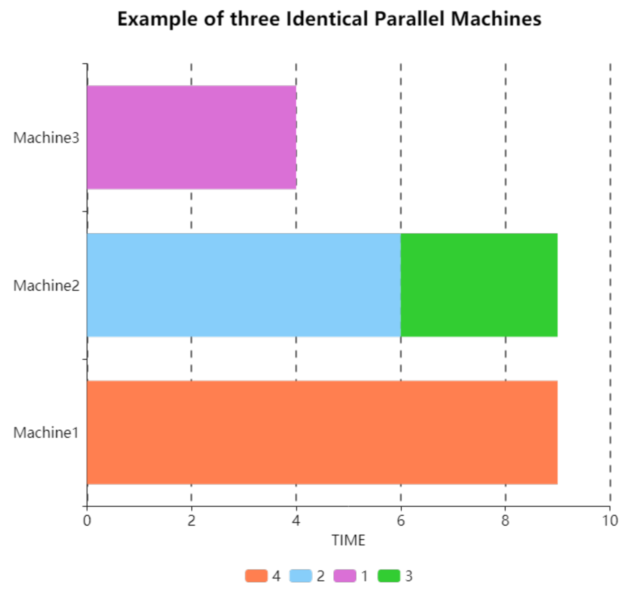

Figure 2.1: Example of three identical parallel machines.

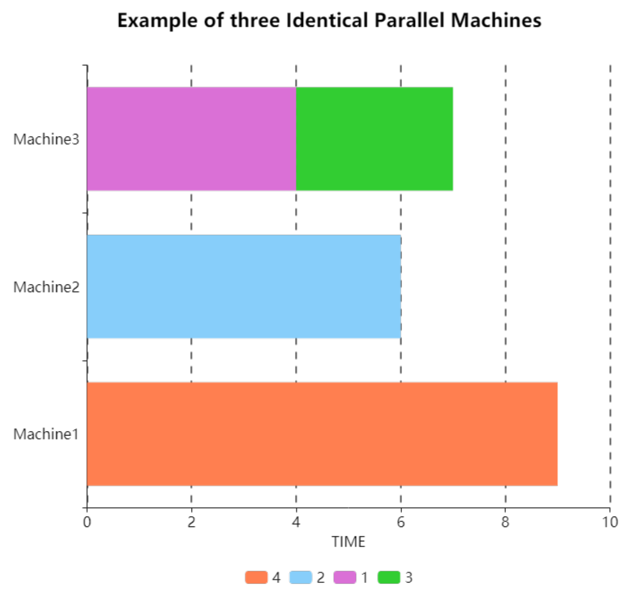

Figure 2.2: Example of three identical parallel machines.

In this three Identical Parallel Machines (IPM) problem, Figure 2.1 and Figure 2.2 show the same makespan (9 units of time) of the problem by two different schedules. It can be seen that the job with the longest processing time is usually the bottleneck, and generates the most influential constraints of the optimization goal.

2.2.2 Uniform Parallel Machines

The Uniform Parallel Machines problem is denoted by $(Q_m||C_{max})$. The problem of Uniform Parallel Machines shows different proportions of processing speed between the different machines $(p_{ij} = p_j/s_i)$. For example, the job j may require one unit of processing times on machine 1 or two unit of processing times on machine 2, which depends on s_i, the speed factor of machine i. In an Identical Parallel Machines problem, jobs can be processed on any machine with the same speed factors; in a Uniform Parallel Machines problem, jobs can be processed on any machine with distinctive speed factors, which are in stable ratios with each other (Lin & Ying, 2017). The larger the processing factor s_i of the machine i, the faster processing speed and the shorter processing time of the same job j. In this situation, a good solution consists of processing longer jobs in faster machines to minimize the makespan.

Example of Uniform Parallel Machines

We now show an example of three Uniform Parallel Machines. Table 2.2 displays the processing times p_{ij}, We can see that the processing time p_{ij} of each job j has changed proportionally on all three machines. The goal is to decrease the maximum completion time to mimimum and produce a minimum makespan.

	Job 1	**Job 2**	**Job 3**	**Job 4**
Machine 1	1	2	4	3
Machine 2	3	6	12	9
Machine 3	4	8	16	12

Table 2.2: p_{ij} for three uniform parallel machines.

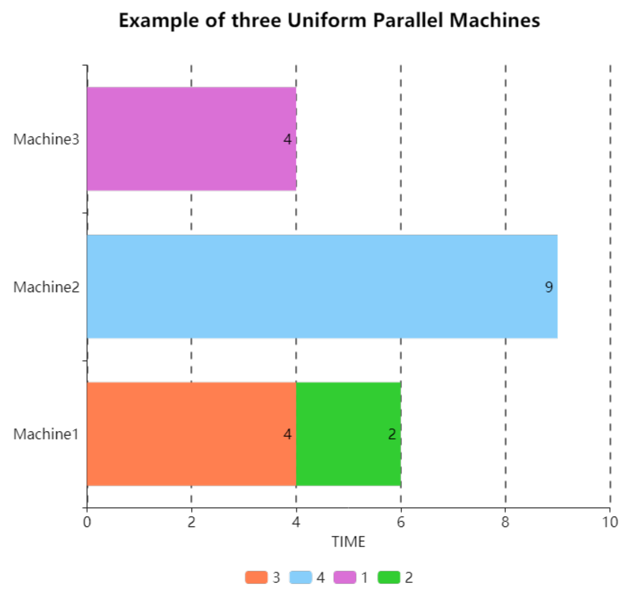

Figure 2.3: Example of three uniform parallel machines.

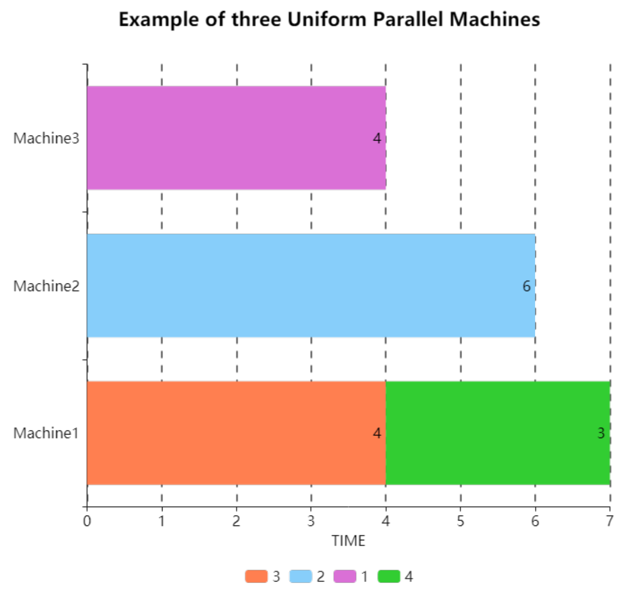

Figure 2.4: Example of three uniform parallel machines.

The processing time of machine 2 is three times that of machine 1, the processing time of machine 3 is four times that of machine 1 ($s_1 = 1, s_2 = 1/3, s_3 = 1/4$). In the schedule of Figure 2.3, the job 3 with the longest processing time is assigned to the fastest machine 1. However, the job 4, which also requires a long processing time, is assigned to machine 2 whose speed is relatively slow. The makespan of this scheduling is 9 time units. In Figure 2.4, job 3 and job 4 need longer time to process and are all assigned to the fastest machine 1. The makespan of this schedule is 7 units of time. In this problem, the speed factor of the machine i s_i is a key aspect. It is necessary to match the machine with the largest speed factor to the jobs that take the longest times to process.

2.2.3 Unrelated Parallel Machines

Unrelated Parallel Machines Scheduling (UPM) to minimize the makespan is denoted by $R_m||C_{max}$. This problem has different processing times, which depend both on the machine and the job. Unrelated Parallel Machines is the most general case of these types of problems. Each job can be processed on machines with different speed factors, which are not in fixed ratios with each other. In other words, the processing time of each job depends on the machine where it is processed (Fanjul-Peyro & Ruiz, 2010). Each machine performs differently for each job, some jobs may be slower, and other jobs may be faster. Unlike the uniform case, the processing times are not related.

Example of Unrelated Parallel Machines

We now show an example of three Unrelated Parallel Machines. Table 2.3 shows the processing times of different machines, job 1 runs slow on machine 1, and runs fast on machine 3. Conversely, job 4 runs fast on machine 1 and slow on machine 3. There is no specific relationship between these processing times.

	Job 1	**Job 2**	**Job 3**	**Job 4**
Machine 1	7	8	7	5
Machine 2	5	4	5	9
Machine 3	6	5	9	11

Table 2.3: p_{ij} for three unrelated parallel machines.

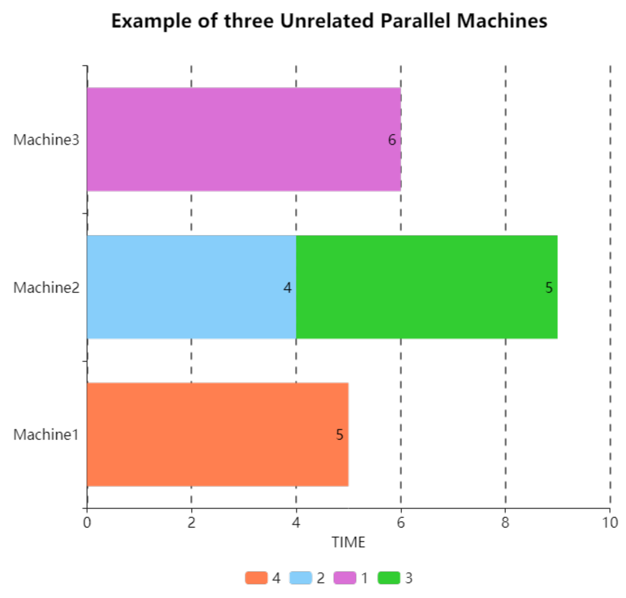

Figure 2.5: Example of three unrelated parallel machines.

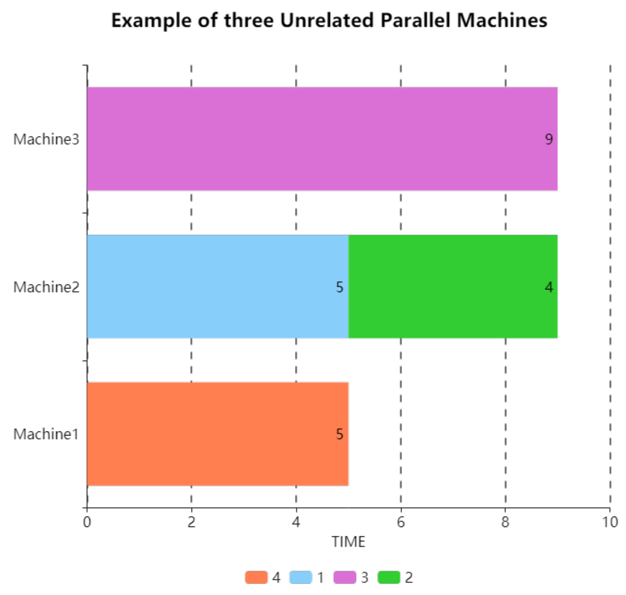

Figure 2.6: Example of three unrelated parallel machines.

In Figure 2.5 and Figure 2.6 above, we can see that the two assignments have the same makespan, (makespan equal to 9 unit time). Jobs loaded on both machine 2 and machine 3 take up the same amount of processing time. These two assignment methods are equivalent to exchanging job 1 and job 3. Although makepans are equal, the first assignment has idler time on the machines, which might be important.

2.2.4 Unrelated Parallel Machines Scheduling with Setup Times (UPMS)

In this section, we address on unrelated parallel machine problems with setup times associated with machines and job sequences

(UPMS). The setup is a non-production time period, usually simulating the operation to be performed on the machine after processing the job, in preparation for the processing of the next job in the sequence. Sequence-dependent setup times are usually incurred in some real industrial environments. Setups usually involve cleaning the production line when changing from one sort of job to another, or adding and removing mechanical parts and elements from the production line (production machine). The great majority of production processes need these setups (Fanjul-Peyro et al., in press).

Unlike the previous UPM problem, in this case, there is a machine setup phase between two consecutive jobs. The setup time may be different depending on which jobs are to be processed. Therefore, it is not only necessary to assign jobs to machines, but also to decide on the sequence of jobs to be processed. Setup times vary depending on the precedence of the job, which are different from the processing times. We define s_{ijk} to represent the setup time between job j and job k needed to adjust machine i.

The followings are assumptions about the UPMS problem:

Assumptions

- At time zero, all the jobs are available to be assigned and all the machines are available to process jobs.
- Each machine can process only one job at a time.
- Each job must be processed only once.

- The process time of each job on each machine differs from each other.
- The setup time for each job on each machine depends on the sequence.
- It is not allowed to preempt the operation of each job.
- Machines are not related and any available machine can process any job.
- Machines are available during the entire schedule.

Example of UPMS

Table 2.4 below shows the processing times p_{ij} for each job on Machine 1 and Machine 2. These machines are all unrelated. Table 2.5 and Table 2.6 show the setup times on each machine, assuming the column job is processed after the row job. These setup times depend on the order of the job sequences. In Table 2.5, on machine 1, processing job 2 after job 1 requires only one unit of setup time, but processing job 1 after job 2 requires five units of setup time. For example, suppose these jobs are coloring the products. Job 1 is paint white and job 2 is paint black. If you execute job 1 first, it takes very little time to clean up the machine. Even if some of the residual color is not cleaned, there is no effect on the next job 2. However, if you run job 2 first, it will take a lot of time to clean up the machine, otherwise it will affect the quality of the next job 1.

	Job 1	**Job 2**	**Job 3**	**Job 4**
Machine 1	3	2	2	5
Machine 2	4	3	1	3

Table 2.4: p_{ij} for two unrelated parallel machines with setup times.

	Job 1	**Job 2**	**Job 3**	**Job 4**
Job 1	0	1	4	2
Job 2	5	0	4	5
Job 3	3	2	0	3
Job 4	5	3	1	0

Table 2.5: s_{1jk} setup time between a pair of jobs j and k on machine 1.

	Job 1	**Job 2**	**Job 3**	**Job 4**
Job 1	0	2	1	2
Job 2	4	0	4	3
Job 3	2	3	0	4
Job 4	6	3	3	0

Table 2.6: s_{2jk} setup time between a pair of jobs j and k on machine 2.

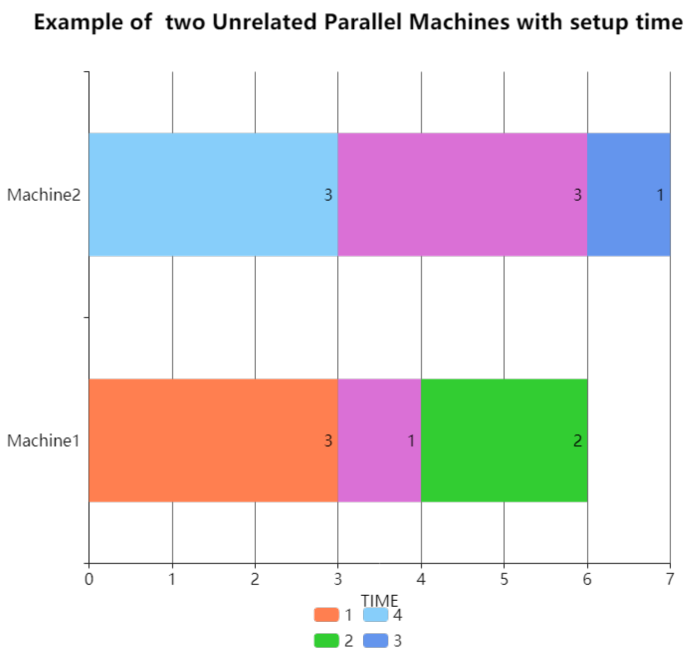

Figure 2.7: Example of two unrelated parallel machines with setup times.

Figure 2.7 above shows the sequences of jobs on machines for the example. The purple bar in the figure shows the time taken by the setup phase. In this example, swapping the jobs running on two machines, or swapping the sequence of jobs on a machine will increase the makespan. If the number of machines exceeds two, the problem is more complicated. Whereas in problems without setups it is just needed to assign jobs to machines, in the UPMS we also need to decide the sequences on the machine. Therefore, adding setups implies adding one more decision to make, which in turn implies an increase in the problem complexity.

2.2.5 Unrelated Parallel Machines Scheduling with Resources in Setups (UPMR-S)

In this section we introduce the Unrelated Parallel Machines Scheduling with limited resources in setup problem, which is the topic of this piece of work. In this thesis, we do not consider that the resources assigned to each setup will influence the time required for the setup phase. In other words, the setup time is fixed. We consider allocating these limited resources to each setup to meet the required amount of resources for each setup. Due to the limited number of available resources, R_{max}, when the time of the setup phase on multiple machines overlaps, the available resources cannot satisfy the setups on multiple machines at the same time, so some setups have to be rearranged (Yepes, 2017).

Example of UPMR-S

This example of UPMR-S adds resource constraints to the example UPMS. Table 2.7 and Table 2.8 below show the resources r_{ijk} that are needed during the setup phase on machine i between job j and job k. $R_{max} = 3$.

	Job 1	**Job 2**	**Job 3**	**Job 4**
Job 1	0	2	2	2
Job 2	2	0	3	1
Job 3	3	3	0	3
Job 4	1	2	1	0

Table 2.7: r_{1jk} resource for setup between a couple of jobs j and k on the machine 1.

	Job 1	**Job 2**	**Job 3**	**Job 4**
Job 1	0	2	1	2
Job 2	3	0	2	3
Job 3	2	3	0	2
Job 4	1	2	3	0

Table 2.8: r_{2jk} resource for setup between a couple of jobs j and k on the machine 2.

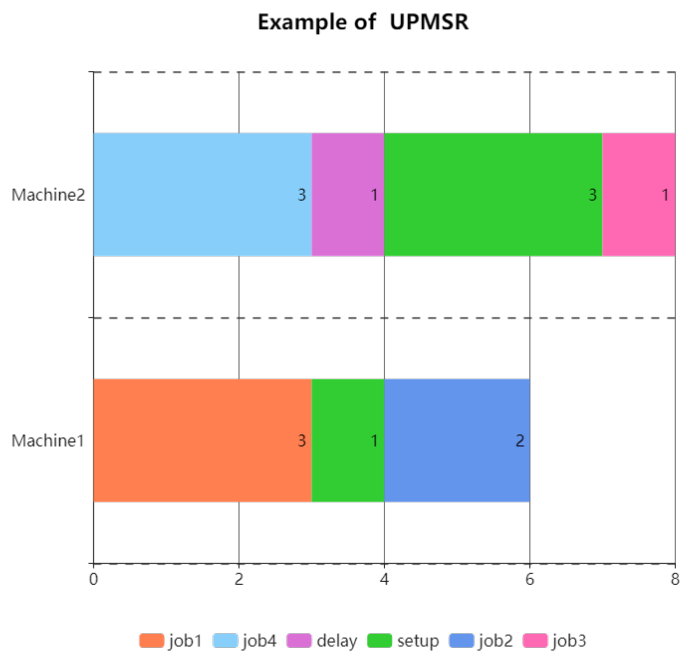

Figure 2.8: Example of two unrelated parallel machines with setup times.

In the UPMS example, the resources in the setup phase are unlimited. Even if the setup phases on both machines overlap, we do not have to check whether the resources are sufficient or not. In the UPMR-S example, R_{max} is equal to three units, and the two setup phases that overlap at time 3 require five units of resources. Therefore, the limited number of resources available cannot satisfy both setups simultaneously.

In Figure 2.8 above, we can see that on machine 2, the setup phase is delayed by one time unit until the setup phase on machine 1 ends. The resources of the two units are released and all three resources are available for the other setup. The makespan is equal to seven units of time. If the setup phase on machine 2 is done first, the setup phase on machine 1 needs to be delayed by three units of time, and the makespan is equal to eight units of time. This illustrates how the order of setups also affects the completion time. Adding resources to the setups implies that for the UPMR-S problem, three decisions have to be made:

1. Assignment (Like in problems without setups and resources).
2. Sequencing (Like in problems with setups).
3. Timing (We now need to decide when each job is processed, and when each setup is done).

Chapter 3

Formal Definition of the Problem

In this Chapter, a mathematical definition of the Unrelated Parallel Machine Scheduling problem with limited resources in the setups (UPMR-S) will be given, where the setup times are sequence-dependent. Furthermore, a mathematical model is introduced to understand the problem better (Yepes, 2017).

3.1 Sets, Variables and Parameters

In this section, we introduce the sets, variables, and parameters needed to formulate the UPMR-S problem.

3.1.1 Sets

- $M = \{1, \ldots, m\}$: A set of machines that is indexed with the letter i.
- $N = \{1, \ldots, n\}$: A number of jobs to process that is indexed with the letters j and k.
- $T = \{1, \ldots, t_{max}\}$: Set of instants.

For modeling purposes, we define the set $N_0 = \{0, 1, \ldots, n\}$ where 0 is a dummy job. We will assume that the sequence of jobs processed on machine i starts from the dummy job and ends with the dummy job.

3.1.2 Variables

We now introduce the variables needed for modeling the problem.

- C_{max}: Objective variable, makespan.
- X_{ijk}: Binary variable takes value 1 if the job k is processed right after j on machine i (with the corresponding setup), it takes value 0 otherwise. This variable designs the sequences of jobs on each machine.
- Y_{ij}: Binary variable takes value 1 if job j is processed on machine i, it takes value 0 otherwise. This variable designs the assignment of jobs to machines.
- H_{ijkt}: Binary variable takes value 1 if the setup, on the machine i, between consecutive jobs j and k, finishes at the time t. Otherwise, it takes value 0. This variable designs the timing of the setups.

3.1.3 Parameters

In this part, we will show the input data, which are known and different for each UPMR-S instance.

- s_{ijk}: Setup time between consecutive jobs j and k on machine i.
- p_{ij}: Processing time of job j on machine i.
- r_{ijk}: Resources needed for the setup between job j and job k on machine i.
- t_{max}: The upper bound of the completion time.
- R_{max}: Amount of resources available for setups.

3.2 MILP Model

In this section we introduce a Mixed-Integer Linear Programming (MILP) model, for solving the UPMR-S. Mixed Integer Linear Programming is rigorous, flexible, and has a wide range of modeling capabilities. This method is widely explored for scheduling problems, since a scheduling problem is essentially equivalent to a combination of several discrete decisions, such as equipment allocation and job assignment over time (Floudas & Lin, 2005).

The following is a Mixed-integer Linear Programming (MILP) Model for the UPMR-S problem, and it was presented in (Yepes, 2017).

$$\min \quad C_{\max} \tag{3.1}$$

$$\text{s.t.} \quad \sum_{k \in M} X_{i0k} \leq 1, \quad i \in M \tag{3.2}$$

$$\sum_{i \in M} Y_{ij} = 1, \quad j \in N \tag{3.3}$$

$$Y_{ij} = \sum_{k \in N_0, j \neq k} X_{ijk}, \quad i \in j \in N \tag{3.4}$$

$$Y_{ik} = \sum_{j \in N_0, j \neq k} X_{ijk}, \quad i \in M, k \in N \tag{3.5}$$

$$\sum_{t \leq t_{max}} H_{ijkt} = X_{ijk}, \quad \forall i \in M, j \in N0, k \in N, k \neq j \tag{3.6}$$

$$\sum_{t} tH_{ijkt} \geq \sum_{k' \in N_0} \sum_{t \in t_{max}} H_{ik'jt}(t + s_{ijk} + p_{ij}) - B(1 - X_{ijk}),$$
$$\forall i \in M, j \in N_0, k \in N, k \neq j \tag{3.7}$$

$$\sum_{i \in M, j \in N_0, k \in N, t' \in \left\{t, \ldots, t + s_{ijk} - 1\right\}} r_{ijk} H_{ijkt'} \leq R_{max},$$
$$\forall t \leq t_{max} \tag{3.8}$$

$$\sum_{t \leq t_{max}} H_{ijkt} \leq C_{max}, \quad \forall i \in M, j \in N_0, k \in N_0, k \neq j \tag{3.9}$$

$$X_{ijk} \geq 0, Y_{ij} \geq 0, H_{ijkt} \in \{0, 1\}$$

- Equation 3.1 indicates the objective to be minimized (completion time or makespan).
- Constraint 3.2 ensures that at most one job is assigned to the first position in the job sequence on each machine.
- Constraint 3.3 guarantees that each job is assigned to one and only one machine.
- Constraint 3.4 guarantees that each job j that is processed on machine

i only has one successor k.

- Constraint 3.5 guarantees that each job k that is processed on machine i only has one predecessor j.
- Constraint 3.6 indicates that for each machine i, and for each pair of consecutive jobs j and k on machine i, the setup between j and k must end before t_{max}.
- In Constraint 3.7, B is a large number. This ensures breaking the cycle of two consecutive jobs. The completion time of the previous job with the addition of processing time of the job j and plus the setup time between two successive jobs, job j and job k on the same machine is equal to the completion time of the current job.
- Constraint 3.8 ensures that the amount of resources used can not exceed R_{max} at any time.
- Constraint 3.9 imposes that the C_{max} must be greater than or equal to the final time of all the setups, including the final dummy setup between the last job and the dummy job 0.

As we will see in the experiments, this model can only solve very small instances of the UPMR-S. For this reason, in the next chapter we introduce more efficient approaches.

Chapter 4

Matheuristics Proposed

This chapter describes two matheuristic algorithms that for UPMR-S problem. A heuristic algorithm is an approximate good solution, but can not guarantee the result is optimal. Metaheuristic is an advanced strategy, which directs some underlying heuristic algorithms to solve a particular problem. In other words, a metaheuristic is an interactive generation process which guides a subordinate heuristic by combining in a smart way different concepts for discovering and exploiting the search spaces, utilizing learning strategies to form information in order to find efficiently near-optimal solutions (Said et al., 2014).

The two matheuristic algorithms proposed are divided into two main parts:

1. A constructive part in which we find an initial solution, and regardless of resources, only consider the assignment of jobs and the processing sequence of jobs on machines.

2. The other part is a repairing phase, which takes into account the resource constraints, making the sequence obtained in the constructive part become feasible.

In Chapter 6, we will compare the computation times and makespans of these two heuristic algorithms (which differ in the constructive phase) and the MILP. Figure 4.1 summarizes the matheurists proposed below:

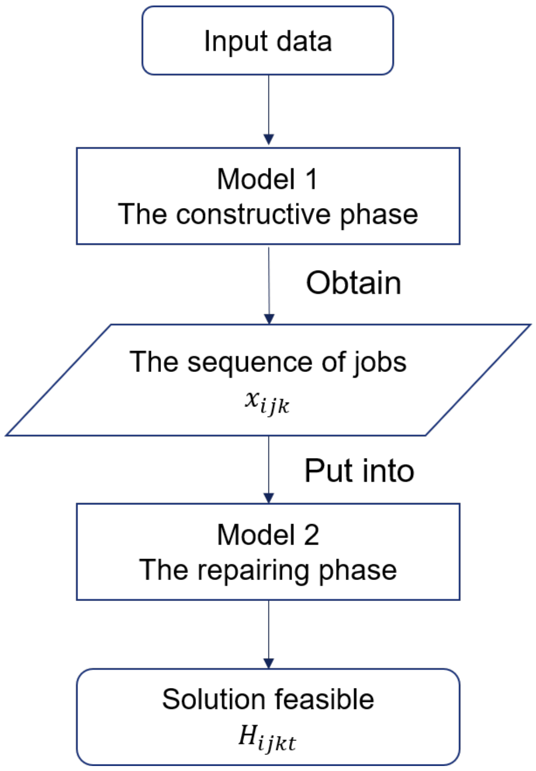

Figure 4.1: The flow chart of matheuristics algorithm.

4.1 Algorithm 1

In this section, we introduce the first algorithm, which consists of two stages. The two stages of this algorithm correspond to two models, model 1 and model 2. This algorithm works by solving an Unrelated Parallel Machines Scheduling Problem without the resources constraint, obtaining the job-machine assignment and job sequence and repairing the job sequence.

The first stage of this algorithm does not consider the resource limits required for the setup time. That is, the Constraint 3.8 of the MILP model in the previous chapter is ignored.

$$\sum_{i \in M, j \in N_0, k \in N, t' \in \left\{t, \ldots, t+s_{ijk}-1\right\}} r_{ijk} H_{ijkt'} \leq R_{max}, \forall t \leq t_{max}$$

In other words, the first stage obtains a solution to the UPMS problem associated. X_{ijk} are the sequences of jobs processed on machine i, the order of all jobs is determined and cannot be changed in the next phase. Y_{ij} are the job assignments on the machines, the job j runs on machine i. We obtain job sequences and job assignments from the first stage (Model 1), and put X^*_{ijk} and Y^*_{ij} (the optimal values of X_{ijk} and Y_{ij} in Model 1) as parameters into the second stage (Model 2). In the second phase, since we already know the assignments of jobs on each machine, what we need to consider is whether the available resources are sufficient when the setups are overlapping. If the available resources are not enough, the setup operation of certain machines will be delayed until resources are released.

4.1.1 Model 1 of Algorithm 1: The Constructive Phase

Model 1 of Algorithm 1 is the MILP Model in Chapter 3 without the resource constraint. This is the construction part of the Algorithm 1, and this part finds the assignment of jobs to machines and the order in which they are proposed. This model does not consider the resource constraints required during the setup phase between the jobs. The following is Model 1 of Algorithm 1:

$$\begin{aligned}
\text{min} \quad & C_{\max} \\
\text{s.t.} \quad & \sum_{k \in M} X_{i0k} \leq 1, \quad i \in M \\
& \sum_{i \in M} Y_{ij} = 1, \quad j \in N \\
& Y_{ij} = \sum_{k \in N_0, j \neq k} X_{ijk}, \quad i \in j \in N \\
& Y_{ik} = \sum_{j \in N_0, j \neq k} X_{ijk}, \quad i \in M, k \in N \\
& \sum_{t \leq t_{max}} H_{ijkt} = X_{ijk}, \quad \forall i \in M, j \in N0, k \in N, k \neq j \\
& \sum_{t} tH_{ijkt} \geq \sum_{k' \in N_0} \sum_{t \in t_{max}} H_{ik'jt}(t + s_{ijk} + p_{ij}) - B(1 - X_{ijk}), \\
& \forall i \in M, j \in N_0, k \in N, k \neq j \\
& \sum_{t \leq t_{max}} H_{ijkt} \leq C_{max}, \quad \forall i \in M, j \in N_0, k \in N_0, k \neq j \\
& X_{ijk} \geq 0, Y_{ij} \geq 0, H_{ijkt} \in \{0, 1\}
\end{aligned}$$

This model yields a sequence given by X^*_{ijk}, the value of X_{ijk} in an optimal solution.

4.1.2 Model 2 of Algorithm 1: The Repairing Phase

Model 2 is the repairing phase of the Algorithm 1. This phase adds constraint on resources, considering that the setup phases between jobs on different machines may overlap, and the limited resources may not be able to satisfy overlapping multiple setup processes. Model 2 is designed to modify and repair the sequence X^*_{ijk} obtained in Model 1. This model is obtained from the complete model in Chapter 3 by assuming that the variables X and Y are now parameters, and we only need to find the values of variables H:

$$
\begin{aligned}
\text{min} \quad & C_{max} \\
\text{s.t.} \quad & \sum_{t \le t_{max}} H_{ijkt} = X^*_{ijk}, \quad \forall i \in M, j \in N0, k \in N, k \neq j \\
& \sum_{t} tH_{ijkt} \ge \sum_{k' \in N_0} \sum_{t \in t_{max}} H_{ik'jt}(t + s_{ijk} + p_{ij}) - B(1 - X^*_{ijk}), \forall i \in M, j \in N_0, \\
& k \in N, k \neq j \\
& \sum_{t \le t_{max}} H_{ijkt} \le C_{max}, \quad \forall i \in M, j \in N_0, k \in N_0, k \neq j \\
& \sum_{i \in M, j \in N_0, k \in N, t' \in \left\{t, \ldots, t+s_{ijk}-1\right\}} r_{ijk} H_{ijkt'} \le R_{max}, \forall t \le t_{max}
\end{aligned}
$$

- X^*_{ijk} are parameters (the sequence of jobs processing on the machines obtained in Model 1).

- $H_{ijkt} \in \{0,1\}$ are binary variables.

The solution to Model 2 is a feasible solution to the UPMR-S problem.

4.2 Algorithm 2

This section describes the second algorithm proposed in this thesis, which has two stages. Similar to the first algorithm, Algorithm 2 also ignores the resource constraints in Model 1, and repairs the job sequence in Model 2. The biggest difference between the first algorithm and the second is that the second one uses the MILP model proposed in (Fanjul-Peyro et al., in press) instead of the MILP model in Chapter 3 for finding the sequence X^*_{ijk}.

4.2.1 Model 1 of Algorithm 2: The Constructive Phase

This section is the construction part of Algorithm 2. The following model, presented in an earlier research (Fanjul-Peyro et al., in press), finds a sequence of jobs on machines without considering resources.

$$\min \quad C_{\max} \tag{4.1}$$

$$\text{s.t.} \quad \sum_{j\in N_0, k\in N, k\neq j} s_{ijk}X_{ijk} + \sum_{j\in N} p_{ij}Y_{ij} \leq C_{max}, \quad i \in M \tag{4.2}$$

$$\sum_{k\in N} X_{i0k} \leq 1, \quad i \in M \tag{4.3}$$

$$\sum_{i\in M} Y_{ij} = 1, \quad j \in N \tag{4.4}$$

$$Y_{ij} = \sum_{k \in N_0, j \neq k} X_{ijk}, \quad i \in M, j \in N \tag{4.5}$$

$$Y_{ik} = \sum_{j \in N_0, j \neq k} X_{ijk}, \quad i \in M, k \in N \tag{4.6}$$

$$U_j - U_k + n \sum_{i \in M} X_{ijk} \leq n - 1, \quad j \in N, k \in N, j \leq k \tag{4.7}$$

$$X_{ijk} \in \{0, 1\}, Y_{ij} \geq 0$$

This model uses a new set of variables, $U_j \geq 0$, to define the number of jobs processed before job j on the machine that processes job j.

- Equation 4.1 represents the objective to be minimized (makespan).
- Constraint 4.2 defines that the sum of the setup times and the sum of all the jobs running on the machine are smaller than makespan.
- Constraint 4.3 ensures that at least one job starts running on the machine following the dummy job.
- Constraint 4.4 defines that each job must have a certain machine to execute this job. In other words, all jobs must be executed, no matter to which machine they are assigned.
- Constraint 4.5 and constraint 4.6 ensure that each job has a successor and a predecessor on the same machine. This successor and predecessor can be virtual jobs.
- Constraint 4.7 ensures that when job k is the successor of job j, $X_{ijk} = 1$, that means $U_k \geq U_j + 1$. In contrast, if job k is not a successor of job j, $X_{ijk} = 0$, that means $U_j \leq n - 1$. These constraints are used to break subtours.

4.2.2 Model 2 of Algorithm 2: The Repairing Phase

The repair phase of Algorithm 2 is the same as the repair phase of Algorithm 1.

Chapter 5

Computational Experiments

In this chapter, the results obtained by the algorithms proposed over a number of randomly generated instances will be given. The results of the algorithms proposed will also be compared with the results obtained by the MILP in Chapter 3.

5.1 Solver and Computer

For these experiments, GAMS 23.1 was the modeling language, and the solver used is CPLEX 11.2. The computer used was a laptop with Windows 10 and an Intel core i5 processor with 4GB of RAM memory.

5.2 Generation of Instances

For these experiments, a group of test instances is employed. The generation of these instances is done by varying the following parameters:

- Number of machines $n \in \{6, 8\}$
- Number of jobs $m \in \{2, 3, 4, 5\}$
- Setup times between jobs $s_{ijk} \in \{U(1, 9), U(1, 49), U(1, 99), U(1, 124)\}$. In words, the setup times are calculated randomly by means of a discrete uniform distribution between 1 and 9, between 1 and 49, between 1 and 99, and between 1 and 124.

Combining these three factors gives us 32 instances. The other input data are generated as follows:

- $R_{max} = U(3, 4)$
- $r_{ijk} = U(1, R_{max})$
- $t_{max} = min_i \sum_{j=0}^{n-1} p_{ij} + s_{i,j,j+1}$

5.3 Results

Tables 4.2 and 4.3 show the results of the MILP Model and the two matheuristics proposed. The meaning of each column is :

- cpuM1 shows the computing time of model 1, the constructive time.

- cpuM2 shows the computing time of model 2, the repairing time.
- cpuT shows the total computing time of each algorithm, and is equal to cpuM1+cpuM2.
- VM1 shows the value of the best solution achieved by the model 1.
- VM2 shows the value of the best solution achieved by the algorithm.
- VMILP displays the best solution found by the MILP after one hour of time.
- RPD shows the relative percent deviation of the solution found by each of our matheuristics with respect to the solution found by the MILP, and is computed with this formula:

$$RPD = 100\frac{VM2 - VMILP}{VM2} \tag{5.1}$$

- GAP shows the gap of the solution found by MILP, with respect to the MILP lower bound given by the solver.

	Algorithm 1						Algorithm 2						MILP	
Instance	cpuM1	cpuM2	cpuT	VM1	VM2	RPD1	cpuM1	cpuM2	cpuT	VM1	VM2	RPD2	VMILP	GAP
8*2	757.23	0.55	757.78	89	89	-1.12	1	0.53	1.53	88	89	-1.12	90	30.40%
8*2	1000.09	1.91	1002	168	168	-52.38	0.28	1.41	1.69	167	168	-52.38	256	80.16%
8*2	1000.09	2.11	1002.2	264	264	-11.36	0.22	2.31	2.53	263	264	-11.36	294	67.46%
8*2	1000.09	2.39	1002.48	212	212	-22.64	0.38	1.86	2.24	211	212	-22.64	260	75%
8*3	454.5	0.78	455.28	73	73	-293.15	0.53	0.77	1.3	72	73	-293.15	287	98.46%
8*3	1000.16	1.25	1001.41	104	104	-78.85	0.13	1.2	1.33	103	104	-78.85	186	71.81%
8*3	1000.17	4.14	1004.31	140	172	-183.72	0.33	3.28	3.61	126	165	-195.76	488	99.39%
8*3	1000.19	1.86	1002.05	141	141	-21.28	0.42	3.02	3.44	140	156	-9.62	171	62.79%
8*4	591.73	0.8	592.53	63	63	1.59	0.33	0.81	1.14	62	63	1.59	62	0.00%
8*4	760.02	2.61	762.63	88	88	-11.36	0.45	1.97	2.42	87	88	-11.36	98	45.45%
8*4	942.61	2.28	944.89	75	75	100.00	0.44	2.13	2.57	74	75	100.00		
8*4	1000.22	2.45	1002.67	112	112	100.00	0.5	2.39	2.89	106	107	100.00		
8*5	250.03	0.94	250.97	44	44	2.27	0.39	0.86	1.25	43	44	2.27	43	0.00%
8*5	424.94	1.23	426.17	43	43	-46.51	0.67	1.33	2	42	43	-46.51	63	96.87%
8*5	796.53	1.89	798.42	51	51	100.00	0.7	1.84	2.54	50	51	100.00		
8*5	588.02	2.86	590.88	56	60	-746.67	0.45	2.88	3.33	55	60	-746.67	508	100.00%
average	785.41	1.87	787.29	107.69	109.94	-72.82	0.45	1.78	2.23	105.56	110.13	-72.85	215.85	63.68%

Table 5.1: Results of 8 machines

	Algorithm 1						Algorithm 2						MILP	
instance	cpuM1	cpuM2	cpuT	VM1	VM2	RPD1	cpuM1	cpuM2	cpuT	VM1	M2	RPD2	VMILP	GAP
6*2	12.91	0.20	13.11	75.00	75.00	0.00	0.23	0.30	0.53	74.00	75.00	0.00	75	0.00%
6*2	22.94	0.67	23.61	96.00	102.00	2.94	0.13	0.84	0.97	95.00	102.00	2.94	99	
6*2	58.39	1.91	60.30	188.00	200.00	-2.00	0.05	1.63	1.68	187.00	188.00	-8.51	204	66.67%
6*2	52.99	1.98	54.97	165.00	171.00	-135.67	0.09	1.98	2.07	164.00	171.00	-135.67	403	86.00%
6*3	20.97	0.36	21.33	53.00	53.00	0.00	0.17	0.44	0.61	52.00	53.00	0.00	53	0.00%
6*3	75.44	0.59	76.03	96.00	96.00	0.00	0.09	0.55	0.64	95.00	96.00	0.00	96	0.00%
6*3	17.69	0.47	18.16	59.00	59.00	0.00	0.17	0.55	0.72	58.00	59.00	0.00	59	0.00%
6*3	66.38	1.53	67.91	120.00	130.00	-204.62	0.34	1.69	2.03	119.00	120.00	-230	396	92.40%
6*4	23.77	0.44	24.21	43.00	43.00	0.00	0.23	0.28	0.51	42.00	43.00	0.00	43	0.00%
6*4	75.80	0.94	76.74	76.00	76.00	0.00	0.34	1.02	1.36	75.00	76.00	0.00	76	0.00%
6*4	26.53	1.00	27.53	27.00	27.00	0.00	0.05	0.80	0.85	26.00	27.00	0.00	27	0.00%
6*4	62.86	1.19	64.05	78.00	78.00		0.19	1.36	1.55	77.00	78.00			
6*5	36.30	0.34	36.64	53.00	53.00	0.00	0.05	0.42	0.47	52.00	53.00	0.00	53	0.00%
6*5	47.89	0.55	48.44	48.00	48.00	0.00	0.14	0.72	0.86	47.00	48.00	0.00	48	0.00%
6*5	43.94	1.16	45.10	70.00	70.00		0.06	1.13	1.19	69.00	70.00			
6*5	42.93	1.16	44.09	53.00	53.00	0.00	0.55	1.44	1.99	52.00	53.00	0.00	53	0.00%
average	42.98	0.91	43.89	81.25	83.38	-24.24	0.18	0.95	1.13	80.25	82.00	-26.52	120.36	0.19

Table 5.2: Results of 6 machines

According to Table 5.1 and Table 5.2 above, it can be clearly seen that the amount of computing times varied considerably across the two algorithms. Comparing the computing times of the first stage (cpuM1) of the two algorithms: in Table 5.1, the average of cpuM1 of Algorithm 1 is 1745 times more than the average of cpuM1 of Algorithm 2; in Table 5.2, the average of cpuM1 of Algorithm 1 is 240 times more than the average of cpuM1 of Algorithm 2. The average of computing times of the first phase of Algorithm 2 is much faster than Algorithm 1. Comparing the computing times of the second stage (cpuM2) of the two algorithms: in these two tables, the average of cpuM2 of the two algorithms are not much different, they are almost the same. The average of total computing times of both phases (cpuT) of Algorithm 2 is much faster than Algorithm 1.

In these two tables, the value of the best solution achieved by the model 1 (VM1) of both algorithms are almost same. However, the average value of VM1 of Algorithm 2 is a little lower than the average of VM1 of Algorithm 1. Similarly, the average value of VM2 of Algorithm 2 is slightly lower than the average of VM2 of Algorithm 1. In addition, the average of VM2 of Algorithm 1 and the average of VM2 of Algorithm 2 are both less than the average of the best solution found by the MILP (VMILP). Moreover, the RPD1 is larger than the RPD2, and the difference is very small.

The computing times of the two algorithms are greatly improved compared to the computing times of the MILP, and the average of computing times of the Algorithm 2 is the shortest. We use the best solution obtained by two matheuristic algorithms and MILP Model to generate RPD index (Equation 5.1), as we can see, RPD1 is slightly larger than RPD2.

In short, the performances of both matheuristic algorithms are superior to MILP in terms of computing times and the optimal value obtained. Furthermore, the performance of Algorithm 2 is better than that of Algorithm 1.

Chapter 6

Conclusion

In this thesis, a complex Unrelated Parallel Machines Scheduling problem with resources in setups (UPMR-S) is addressed. Two two-stage matheuristics have been proposed for this problem.

Table 5.1 and Table 5.2 in Chapter 5 show that the two two-stage matheuristic algorithms have a better performance for small-sized instances than medium-sized instances. Both algorithms are capable of obtaining an optimal solution for some instances of small size.

Comparing the computation time of the two matheuristics and the relative percent deviation of the solution found by each of our two matheuristic algorithms with respect to the solution found by the MILP (RPD), we conclude that Algorithm 2 uses less computation times. Furthermore, Algorithm 2 has smaller RPD indicators than Algorithm 1, which means that Algorithm 2 is capable of obtaining solutions closer to optimal. Overall, the performance of Algorithm 2 is deemed better considering the computing times and the best value obtained, as well as the large number of constraints and variables required.

There are still some issues to be resolved in this thesis: First of all, this thesis does not use large-sized instances in experiments. Second, the number of instances is not large enough, so the results may not be significant. Finally, for further research, it would be useful to apply the proposed algorithms to other parallel machine scheduling problems that are more complex. It would also be interesting to extend this work by improving Model 1 (the constructive phase) and investigate more new changes of Model 2 (the repairing phase) for other problems.

During this thesis I have improved my skills in the following scientific tools: modelling language GAMS, data analysis with Excel, and scientific writing with latex. Besides, I have faced a serious optimization problem for the first time in my life, which I think will be of great help in my future career.

References

Blazewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., & Weglarz, J. (2007). *Handbook on scheduling: from theory to applications*. Springer Science & Business Media.

Edis, E. B., & Oguz, C. (2011). Parallel machine scheduling with additional resources: a lagrangian-based constraint programming approach. In *International conference on ai and or techniques in constriant programming for combinatorial optimization problems* (pp. 92–98).

Fanjul-Peyro, L., & Ruiz, R. (2010). Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research*, *207*(1), 55–69.

Fanjul-Peyro, L., Ruiz, R., & Perea, F. (in press). Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. *Computers and Operations Research*.

Floudas, C. A., & Lin, X. (2005). Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications. *Annals of Operations Research*, *139*(1), 131–162.

German, Y., Badi, I., Bakir, A., & Shetwan, A. (2016). Scheduling to minimize makespan on identical parallel machines. *International Journal of Scientific, Engineering Research*, *7*, Issue 3.

Kayvanfar, V., Aalaei, A., Hosseininia, M., & Rajabi, M. (2014). Unrelated parallel machine scheduling problem with sequence dependent setup times. In *International conference on industrial engineering and operations management, bali* (pp. 7–9).

Lenstra, J. K., Shmoys, D. B., & Tardos, E. (1990). Approximation algorithms for scheduling unrelated parallel machines. *Mathematical programming*, *46*(1-3), 259–271.

Leung, J. Y. (2004). *Handbook of scheduling: algorithms, models, and performance analysis.* CRC Press.

Lin, S.-W., & Ying, K.-C. (2017). Uniform parallel-machine scheduling for minimizing total resource consumption with a bounded makespan. *IEEE Access*, *5*, 15791–15799.

Ouazene, Y., Yalaoui, F., Chehade, H., & Yalaoui, A. (2014). Workload balancing in identical parallel machine scheduling using a mathematical programming method. *International Journal of Computational Intelligence Systems*, *7*, 58–67.

Ruiz, R., & Andrés-Romano, C. (2011). Scheduling unrelated parallel machines with resource-assignable sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology*, *57*(5-8), 777–794.

Said, G. A. E.-N. A., Mahmoud, A. M., & El-Horbaty, E.-S. M. (2014). A comparative study of meta-heuristic algorithms for solving quadratic assignment problem. *arXiv preprint arXiv:1407.4863*.

T'kindt, V., & Billaut, J.-C. (2006). *Multicriteria scheduling: theory, models and algorithms.* Springer Science & Business Media.

Toksarı, M. D., Oron, D., Aydoğan, E. K., & Barbosa, J. G. (2014). New developments in scheduling applications. *The Scientific World Journal*, *2014*.

Vallada, E., & Ruiz, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, *211*(3), 612–622.

Yepes, J. C. (2017). *Una metaheurística para un problema de secuenciación con necesidad de recursos en los ajustes* (Unpublished master's thesis). Valencia/Universidad Politécnica de Valencia.