# 1
# Introduction

## 1.1 Scheduling Problems

Scheduling is a decision-making process that deals with the allocation of resources during a period of time to perform a collection of tasks. These sets of resources and tasks can take many different forms. The resources may represent, for example, processing units in a computer environment or machines in a assembly-line. The tasks may represent, for example, the execution of computer programs or the stages of a production process.

The objective of a scheduling is to optimize one or more objective functions and can take many different forms. It may represent the minimization of the completion time of the last task or the maximization of the number of tasks completed before a due date, for example. Furthermore, tasks may have associated distinct processing times, release dates, due dates or priority factors.

Scheduling plays an important role in most information processing environments as well in manufacturing and production systems. It is also important in transportation and distribution settings. For a large number of examples that illustrate real world problems we refer to the book of Pinedo [49].

Research on scheduling is located at the intersection of theoretical computer science, operational research and industrial engineering. In particular, this multidisciplinary area can be partitioned into two fields: theory of scheduling and scheduling applications.

The theory of scheduling field comprises the study of mathematical aspects of scheduling problems. One of such aspects relates to the computational complexity characterization of scheduling problem. This characterization entails the design and analysis of algorithms for polynomially solvable cases and the *NP-hardness* proof of intractable ones. For *NP-hard* problems, there is an additional interest for approximation algorithms with performance guarantees.

The scheduling applications field comprises the study of efficient computational methods for solving a large set of real life instances, even the ones belonging to NP-hard problems. These computational methods can be divided into two main categories. The first category is composed of the so-called exact methods. These methods try to determine optimal solutions by using mathematical programming techniques, such as mixed integer programming, or enumerative methods, such as branch-and-bound and branch-and price. The second category is composed of heuristics. A heuristic is an algorithm designed to obtain good approximation solutions for a problem but for which there is no formal proof of its correctness.

Finally, scheduling research area is characterized by a virtually unlimited number of problem types. This thesis introduces some advances through a classical scheduling problem, called the `Permutation Flow Shop Scheduling Problem`. The main reference texts in scheduling are the books of Baker and Trietsch [2], Brucker [6] and Pinedo [49].



## 1.2 Permutation Flow Shop Scheduling

The permutation flow shop scheduling problem is one of the most studied problems in the area of scheduling theory and applications, known by its intractability from theoretical and computational aspects. Its definition dates from over fifty years. Since then, a large number of papers [28] were published abouts this problem and its variations.

The seminal paper from Johnson [34] was the first one to define the concept of flow shop scheduling. This work introduced the permutation flow shop scheduling problem on an environment formed by two machines with the objective of minimizing the makespan. This paper is considered by many researchers in the area of scheduling theory and applications as one of the milestones in the field. During a recent plenary talk given in the *9th Workshop on Models and Algorithms for Planning and Scheduling Problems*, Chris Potts awarded it as the most relevant work in the last fifty years of scheduling research.

We proceed now by defining the permutation flow shop scheduling problem.

## (a) Problem Definition

A **Permutation Flow Shop Scheduling** is a production planing process consisting of a set $\mathbb{J} = \{J_1, J_2, ..., J_n\}$ of $n$ **jobs** to be executed in a set $M = \{M_1, M_2, ..., M_m\}$ of $m$ **machines**. This set of machines is called a **shop**. Every job $J_j$ is composed of $m$ stages $O_{1,j}, O_{2,j}, ..., O_{m,j}$ named **operations**. Every operation $O_{i,j}$ has a non-negative **processing time** $t_{i,j}$ comprising the matrix $T \in \Re^{+}_{M \times J}$.

The following rules must be observed during a permutation flow shop scheduling:

1. All jobs must be executed in the same order by every machine, defined by a permutation $\pi : \{1, \ldots, n\} \mapsto J$, with $\pi(i)$ indicating the i-th job to be executed.

2. The job operation $O_{i,\pi(j)}$ must be only executed on machine $i$.

3. The operation $O_{i,\pi(j)}$ can be executed only after operation $O_{i-1,\pi(j)}$ has finished its execution.

4. The operation $O_{i,\pi(j)}$ can be executed only after operation $O_{i,\pi(j-1)}$ has finished its execution.

5. A machine cannot execute more than one operation per time.

6. Preemption is not allowed, i.e. once an operation is started, it must be completed without interruption.

The **completion time of an operation** $O_{i,j}$, denoted by $C(i,j)$, is defined by the recurrence:

$$C(i,\pi(j)) = \begin{cases} t_{i\pi(j)} & \text{if } i = 1 \text{ and } j = 1 \\ C(i,\pi(j-1)) + t_{i,\pi(j)} & \text{if } i = 1 \text{ and } j > 1 \\ C(i-1,\pi(j)) + t_{i,\pi(j)} & \text{if } i > 1 \text{ and } j = 1 \\ max\left\{C(i,\pi(j-1)), C(i-1,\pi(j))\right\} + t_{i,\pi(j)} & \text{if } i > 1 \text{ and } j > 1 \end{cases} \quad (1)$$

The **completion time of a job** $J_j$ is $C(m,j)$. The **makespan** of a permutation flow shop schedule is the maximum completion time of a job. The **objective** of the `Permutation Flow Shop Scheduling Problem` is to find a permutation $\pi$ that minimizes the makespan.

## (b) Examples

Let us consider now an example of a flow shop instance $I$ and the makespan resulting of two permutation schedules for it, denoted by $\pi$ and $\varphi$. Consider that $I$ consists of 4 jobs and 3 machines Furthermore, the processing times matrix $T$ of all operations in $I$ are given on table 1.1.

| $t_{i,j}$ | $J_1$ | $J_2$ | $J_3$ | $J_4$ |
|---|---|---|---|---|
| $M_1$ | 2 | 8 | 3 | 5 |
| $M_2$ | 3 | 4 | 1 | 7 |
| $M_3$ | 4 | 1 | 6 | 2 |

Table 1.1: Processing times matrix

Let us analyze now the effects of selecting two different permutation schedules for $I$. First, suppose that the permutation schedule $\pi$ is defined as follows: $\pi(1) = 2$, $\pi(2) = 4$, $\pi(3) = 1$ and $\pi(4) = 3$. The completion times $C(i, \pi(j))$ of each operation $O_{i,\pi(j)}$ can be calculated from equation (1):

$$\begin{cases} C\left(1, \pi(1)\right) = t_{1,\pi(1)} = 8. \\ C(1, \pi(2)) = C(1, \pi(1)) + t_{1,\pi(2)} = 8 + 5 = 13. \\ \ldots \\ C(2, \pi(1)) = C(1, \pi(1)) + t_{2,\pi(i)} = 8 + 4 = 12 \\ C(2, \pi(2)) = max\left\{C(1, \pi(2), C(2, \pi(1))\right\} + t_{2,\pi(2)} = 13 + 7 = 20. \\ \ldots \\ C(3, \pi(3)) = max\left\{C(2, \pi(3), C(3, \pi(2))\right\} + t_{3,\pi(3)} = 23 + 4 = 27. \\ C(3, \pi(4)) = max\left\{C(2, \pi(4), C(3, \pi(3))\right\} + t_{3,\pi(4)} = 27 + 6 = 33. \end{cases} \quad (2)$$

The completion times of the operations resulting from permutation $\pi$ are summarized at table 1.2. Checking over such values, it becomes clear that the makespan resulting from permutation $\pi$ is equal to 33.

| $C(i, \pi(j))$ | $\pi(1)$ | $\pi(2)$ | $\pi(3)$ | $\pi(4)$ |
|---|---|---|---|---|
| 1 | 8 | 13 | 15 | 18 |
| 2 | 12 | 20 | 23 | 24 |
| 3 | 13 | 22 | 27 | 33 |

Table 1.2: Completion times matrix for permutation $\pi$

It will be considered now that the effect of applying permutation $\varphi$ to instance $I$. Suppose that $\varphi$ is defined as follows: $\varphi(1) = 3$, $\varphi(2) = 1$, $\varphi(3) = 4$ and $\varphi(4) = 2$. The resulting completion times appears at table 1.3. Therefore, permutation $\varphi$ induces a schedule of makespan 23.

| $C(i, \varphi(j))$ | $\varphi(1)$ | $\varphi(2)$ | $\varphi(3)$ | $\varphi(4)$ |
|---|---|---|---|---|
| 1 | 3 | 5 | 10 | 18 |
| 2 | 4 | 8 | 17 | 22 |
| 3 | 10 | 14 | 19 | 23 |

Table 1.3: Completion times matrix for permutation $\pi$

### (c) Computational Complexity

The polynomial dichotomy of the permutation flow shop scheduling problem is intrinsically related to a parameter of the instance: the number of machines $m$ comprising the shop. For $m = 2$, the problem is solvable in $\Theta(n \log n)$ after a single job ordering. This polynomial time result is due to the classical paper of Johnson [34]. However, for any $m \geq 3$ the problem becomes *Strongly NP-hard* as proved by Garey, Johnson and Sethi [20]. Its *NP-hardness* proof follows by a reduction from the `3-Partition` decision problem.

## 1.3 3-Field Notation

Scheduling problems can be classified following a concise scheme introduced by Graham et al. [24]. In this classification scheme, known as the three-field notation, every scheduling problem is denoted by a triple $\alpha|\beta|\gamma$, where $\alpha$ specifies the machine environment, $\beta$ specifies the job characteristics and $\gamma$ denotes the optimality criterion.

The permutation flow shop scheduling problem, in particular, is denoted in the three-field notation as $Fm|prmu|C_{max}$, where $Fm$ specifies a flow shop environment of $m$ machines, $prmu$ represents the requirement that all machines take the jobs in the same ordering and $C_{max}$ denotes the makespan. In the case that $m = 2$, in particular,it is represented by $F2|prmu|C_{max}$.

In this work, for the sake of simplicity, the $Fm|prmu|C_{max}$ problem will be denoted as the `PFS` problem. For the same reasons, the notation `2-PFS` will be used for the $F2|prmu|C_{max}$ problem.

## 1.4 Gannt Charts

The Gannt chart [18] is one of the most fundamental models used to aid decision making in scheduling. It is a 2-dimensional graphical representation consisting of two axis. In the permutation flows shop scheduling problem, the horizontal axis represents the time elapsed during the scheduling, while the vertical axis represents the machines comprising the shop. Figures 1.1 and 1.2 shows two examples of Gannt charts corresponding to the permutation schedules $\pi$ and $\varphi$, respectively, given at section 1.2.

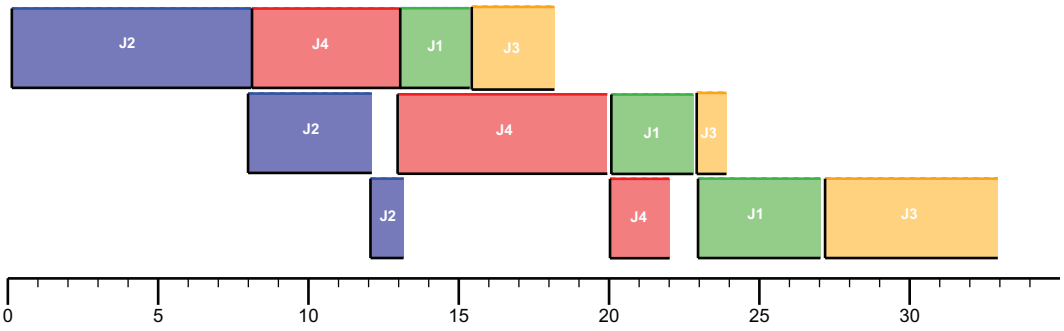


Figure 1.1: Gannt chart of permutation schedule $\pi$

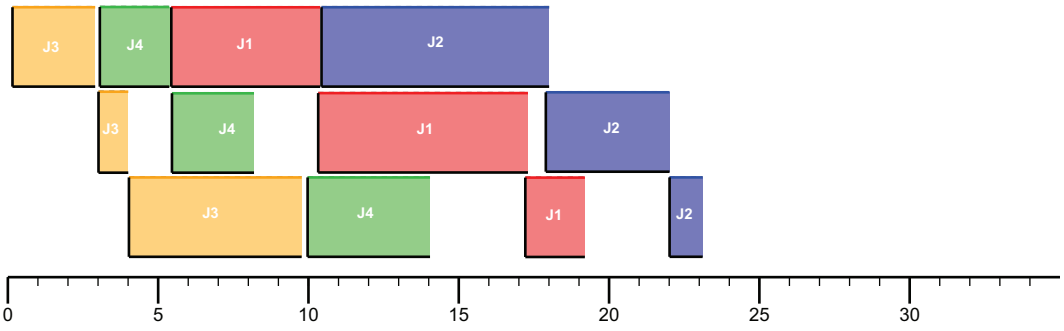


Figure 1.2: Gannt chart of permutation schedule $\varphi$

## 1.5 Objective of this work

The objective of this thesis is to introduce three major contributions for the PFS problem, comprising theoretical and practical aspects.

The first contribution is an approximation algorithm for the PFS problem with $n$ jobs and $m$ machines. This algorithm achieves an approximation guarantee of

$O(\sqrt{n+m})$ and runs in linear time. This is the best performance ratio already obtained for the PFS problem in the case of $n = \Theta(m)$. Furthermore, a novel connection between PFS and monotone subsequence problems is established, resulting on an extension of the Erdös-Szekeres theorem to weighted monotone subsequences.

The second result is a faster algorithm for the PFS with $n$ jobs and two machines (2-PFS). We give an $O(n \log k)$ algorithm that determines optimal solutions for the 2-PFS problem, where $k \leq n$ is the minimum number of cliques necessary to cover the nodes of an underlying interval graph. From the best of our knowledge, this is the first improvement upon the $O(n \log n)$ time complexity of the classical algorithm from Johnson.

The third contribution of this work is a new family of competitive deterministic heuristic for the PFS problem. Four new heuristics are introduced as extensions of the classical NEH heuristic. Such heuristics are based on pruning techniques on the implicit enumeration tree of the PFS problem. Computational results attest that the new proposed methods stand among the most effectives for the PFS problem.

## 1.6 Outline

This thesis is organized as follows. The objective of chapter 1 was to introduce the PFS problem, providing an overview about the work contained in this thesis. Chapter 2 presents a new approximation algorithm for the PFS problem in the case that the number of jobs $n$ and the number of machines $m$ of the schedule are not fixed. Chapter 3 introduces a faster polynomial time algorithm for the PFS problem with $n$ jobs and two machines (2-PFS). Chapter 4 comprises a new a family of of competitive deterministic heuristic for the PFS problem. Final conclusions and future work are considered at chapter 5.