Optimizing process planning, layout, and scheduling problems, in the RMS context

Isabel Barros Garcia

Par **Isabel BARROS GARCIA**

Optimizing process planning, layout, and scheduling problems, in the RMS context

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC

Soutenue le 12 juin 2024
Spécialité : Génie Industriel : Unité de recherche en Mécanique - Laboratoire Roberval (FRE UTC - CNRS 2012)
D2811

Thesis submitted to obtain the doctoral degree by

Université de Technologie de Compiègne

Optimizing process planning, layout, and scheduling problems, in the RMS context

Spécialité : Génie Industriel

Defended on June 12, 2024

By **Isabel Barros Garcia**

Jury members:

Directeurs :

Antoine Jouglet, Professor, Université de Technologie de Compiègne
Julien Le Duigou, Professor, Université de Technologie de Compiègne

Rapporteurs :

Catherine Da Cunha, Professor, École Centrale de Nantes
Corinne Lucet-Vasseur, Professor, Université de Picardie Jules Verne

Examinateurs :

Hichem Haddou Benderbal, Assistant Professor, Université d'Aix-Marseille
Dritan Nace, Professor, Université de Technologie de Compiègne David
Rivreau, Professor, Université Catholique de l'Ouest

Co-encadrant :

Joanna Daaboul, Assistant Professor, Université de Technologie de Compiègne

Acknowledgements

I am profoundly grateful to my mother, whose unwavering support and unconditional love have guided me throughout my academic journey. Despite the distance, she has been my constant source of encouragement, leading me toward the path of knowledge and empowering me to pursue my aspirations.

I extend my heartfelt appreciation to all my friends in Brazil, France, and worldwide. Your presence and companionship have brought immeasurable joy and meaningful reflections during these past years. Regardless of the geographical distances, your friendship has made my time in Compiègne more enjoyable and special.

I sincerely thank my supervisors, Antoine Jouglet, Julien Le Duigou, and my co-supervisor Joanna Daaboul, for their guidance and mentorship throughout this thesis. Their shared expertise has been instrumental in shaping my research trajectory, and I am grateful for the knowledge they have imparted to me.

Last but not least, I thank all the professors and colleagues at the UTC, with special recognition to the SIPP team, Ph.D. students, and colleagues from the two departments I had collaborated with. Your companionship and valuable contributions have been instrumental throughout my time at the university. I genuinely appreciate the assistance and the wonderful moments we have shared together.

Abstract

Reconfigurable Manufacturing Systems (RMS) have emerged as a strategic response to the dynamic and uncertain market conditions driven by the increasing demand for mass-customized products. Companies are focusing on cost-effective approaches to provide a diverse product variety efficiently. The effective production in RMS, aimed at lowering costs, is intricately connected to process planning and layout problems. These optimization problems are interdependent and might benefit from an integrated approach to increase the RMS performance, including its efficiency.

This study emphasizes the importance of integrating process planning, layout, and scheduling problems by proposing and analyzing different models, including sequential, partially sequential, and integrated approaches, to optimize resource allocation and scheduling for RMS. The objective here is to minimize the system's total costs, encompassing production, material handling, machine reconfiguration, layout reconfiguration, and tardiness.

To tackle this challenge, this research work started with developing three exact solution methods and assessing their computation time for small instances. Due to the complexity of the problem, especially for larger instances, we introduced a genetic algorithm adapted to our specific problem. Additionally, we incorporated into the genetic algorithm a local search technique also adapted to our problem to enhance results.

Finally, a case study was introduced to extend our investigation. While most of the work on Reconfigurable Manufacturing Systems (RMS) typically focuses on the production or assembly of products, we developed a case study to demonstrate that the flexibility of these systems is also well-suited for disassembly problems. This emphasis on disassembly directly influences the life cycle of products, highlighting that RMS can play a pivotal role in addressing environmental concerns. This underscores the idea that RMS can contribute to environmental sustainability, not only by optimizing gas and waste emissions (as shown in several works) but also by actively participating in a circular economy.

Résumé

Les systèmes de production reconfigurables (RMS) sont apparus comme une réponse stratégique aux marchés dynamiques et incertains, motivés par la demande croissante de produits personnalisés en masse. Les entreprises se concentrent stratégiquement sur des approches rentables pour fournir efficacement une grande variété de produits. La production dans les RMS, visant à réduire les coûts, est étroitement liée aux problèmes de planification des processus et d'implantation d'atelier.

Ce travail de thèse souligne l'importance de l'intégration des problèmes de planification des processus, d'implantation d'atelier et d'ordonnancement en proposant et en analysant différents modèles, y compris des approches séquentielle, partiellement séquentielles et intégrée, afin d'optimiser l'allocation des ressources et l'ordonnancement pour les RMS. L'objectif ici est de minimiser les coûts totaux du système, englobant les coûts de production, de transport, de reconfiguration de machines, de reconfiguration d'atelier et des pénalités de retards.

Pour relever ce défi, notre travail a commencé par le développement de trois méthodes de resolution exactes, en évaluant leur temps de calcul pour de petites instances. En raison de la complexité du problème, notamment pour les instances plus grandes, nous avons introduit un algorithme génétique adapté à notre problème. De plus, nous avons incorporé à l'algorithme génétique une technique de recherche locale également adaptée à notre problème pour améliorer encore les résultats.

Enfin, une étude de cas a été introduite pour étendre notre recherche. Alors que la majorité des travaux sur les systèmes de production reconfigurables se concentrent généralement sur la production ou l'assemblage de produits, nous avons développé une étude de cas pour démontrer que la flexibilité de ces systèmes est également bien adaptée aux problèmes de démontage de produits pour la remanufacture. Le démontage influence directement le cycle de vie des produits, soulignant que le RMS peut jouer un rôle central dans la réponse aux préoccupations environnementales. Cela souligne également l'idée selon laquelle RMS peut contribuer à la durabilité environnementale, non seulement en optimisant les émissions de gaz et de déchets (comme le montrent plusieurs travaux), mais également en participant activement à une économie circulaire.

Acronyms

AGV Automated Guided Vehicle.

AI Artificial Intelligence.

AMOSA Archived Multi-Objective Simulated Annealing.

CNC Computer Numerical Control.

CP Constraint Programming.

CPS Constraint Programming with a defined search strategy.

DMS Dedicated Manufacturing Systems.

EA Evolutionary Algorithms.

FMS Flexible Manufacturing Systems.

GA Genetic Algorithm.

ILP Integer Linear Programming.

IoT Internet of Things.

L Layout.

LS Local Search.

MH Material Handling.

MILP Mixed Integer Linear Programming.

MOPSO Multi-Objective Particle Swarm Optimization.

NSGA Non-Dominated Sorting Genetic Algorithm.

PP Process planning.

PSO Particle Swarm Optimization.

RMS Reconfigurable Manufacturing Systems.

RMT Reconfigurable Machine Tools.

S Scheduling.

SA Simulated Annealing.

Table of Contents

List of Figures

List of Tables

Introduction

Context

The demand for continuous changes in products, production technology, and markets has introduced multiple challenges within the industrial landscape, necessitating enhanced flexibility, adaptability, and responsiveness from manufacturing systems [37]. Companies worldwide face the growing demand for customized products, the requirement for high-quality products at competitive prices, shorter product life cycles, diversification of demand, and globalization. However, in recent times, additional factors have emerged, significantly impacting production processes and imposing new obstacles for industries.

The COVID-19 pandemic, geopolitical conflicts such as the Russia-Ukraine war, and economic crises demonstrated the extreme volatility of the global landscape, leading to unexpected fluctuations in product demand and market dynamics. While some products experienced a sudden and explosive increase in demand, others faced a sharp drop due to inflation and the reduced purchasing power of a significant part of the population. In addition, growing concerns about environmental sustainability have driven consumer behavior towards more conscious and environmentally friendly choices, leading companies to re-evaluate their production and materials strategies.

In response to these challenges, advanced manufacturing systems have attracted significant attention as potential solutions to meet the demands of a rapidly changing market. Reconfigurable Manufacturing Systems (RMS) have emerged as a promising among these systems.

RMS offer a high degree of flexibility and adaptability, allowing companies to respond to fluctuations in demand and product requirements rapidly. By utilizing machines and tools designed for specific product families, RMS enable agile production processes that can quickly reconfigure hardware and software at machine, control, and system levels [81, 99]. RMS's cyber-physical capabilities, advanced control mechanisms, and intelligent decision-making make them a key enabler of Industry 4.0 initiatives. In smart factories, RMS integrate data, automation, and human expertise, creating highly efficient and responsive production environments [57, 93].

Moreover, RMS play a crucial role in mass customization. They permit industries to adapt quickly, offering a wide range of customized products and maintaining the

benefits of large-scale production. Nowadays, we can observe a change in paradigm from traditional manufacturing approaches (mass production) towards flexible manufacturing approaches [21]. Implementing RMS and Industry 4.0 concepts may require significant changes to traditional production workflows, making workforce upskilling and reskilling critical components to leverage the full potential of these advanced manufacturing systems [47].

In light of these challenges, this research explores the intersection of reconfigurable manufacturing systems, Industry 4.0, mass customization, and industry challenges.

Research challenges

Industry 4.0 has brought a fundamental transformation in manufacturing, promising a new level of automation, connectivity, and intelligence. Alongside this digital revolution, Reconfigurable Manufacturing Systems (RMS) have emerged as a game-changer, allowing manufacturers to adapt and respond to changing market demands swiftly [111]. However, successfully integrating Industry 4.0 technologies into existing manufacturing environments presents a big challenge.

Assessment of capabilities, compatibilities, knowledge-driven infrastructure, data interoperability, and cybersecurity are some of the biggest obstacles the industries face while implementing Industry 4.0 technologies [111]. The coexistence of classical and modern interconnected systems demands careful planning and seamless integration to ensure a smooth transition. Additionally, providing the secure exchange of data between diverse systems and protecting manufacturing information from cyber threats have become critical priorities.

While deploying hardware and software is crucial for enhancing efficiency in industries, addressing the operational challenges that arise afterward is equally essential. The focus of this research lies in optimizing RMS for improved operability. Specifically, we concentrate on the critical areas of process planning, scheduling, and layout optimization, which play a pivotal role in reducing costs and improving overall system performance.

In any production system, classical resource allocation and scheduling optimization are essential, which also holds true for Reconfigurable Manufacturing Systems. Nevertheless, it presents some additional challenges. RMS are more complex as the number of items and components increases, and considering the capacity of flexible resources becomes harder [21]. In addition to the classic optimization, another critical question is when and how they should be reconfigured (machines and system).

To successfully implement mass customization, efficient scheduling, agile process

planning, and optimized layout, industries must exploit the potential of Industry 4.0 technologies. The implementation of algorithms can optimize resource allocation, analyze production data, and handle demand fluctuations, ensuring efficient scheduling and dynamic process planning. Moreover, advanced analytics can help optimize the layout for material flow and worker efficiency.

Addressing these challenges requires collaboration among universities, industries, and technology providers. By combining expertise and leveraging cutting-edge technologies, the potential for innovative solutions and successful implementation of RMS in the Industry 4.0 becomes tangible. Achieving efficient RMS will empower industries to create highly adaptable and competitive manufacturing systems that meet dynamic market demands and drive innovation in the digital age. Through these collective initiatives, we can pave the way for a future of manufacturing excellence, propelling the industry into new frontiers of productivity, performance, and sustainability.

Research question and objective

This work's main objective is to contribute to the successful implementation of RMS by overcoming some of the main barriers related to process planning, scheduling, and layout configuration. These three problems are essential to any production system. They get more complicated in an RMS since the layout can be reconfigured, and the planning and scheduling incorporate additional parameters and constraints, such as different machine configuration possibilities.

The main issue is that these three problems are interconnected. Hence, the question arises for integrating their solving. This research work tries to answer this question, formulated more precisely as follows: How can we best optimize process planning, scheduling, and layout configuration in an RMS for mass-customized products while reducing manufacturing costs? The costs include production, material handling, machine reconfiguration, layout reconfiguration, and tardiness costs.

Main contributions

Given the research question and objective mentioned above, the main contributions of this study are outlined as follows:

I - The proposition and validation of four mathematical models for optimizing process planning, layout, and scheduling problems in RMS. The difference between these models is in the level of integration of the three optimization problems. The first completely integrates the three optimization problems. The second does not integrate

the problems. Finally, the third and fourth offer a partial integration. For the last three models, a sequential approach is adopted. Meaning that the results of the first problem become the input for the subsequent one. To summarize, the four proposed models are as follows:

- Model 1 concerns the integrated approach. It formulates jointly the three problems.
- Model 2 concerns the sequential approach. Each problem is formulated individually and sequentially, which means the results of the first become the input for the subsequent problem. The order of the optimization problems is as follows: process planning, layout, and scheduling.
- Model 3 concerns the partially integrated approach. Here, the process planning problem is formulated individually, and the layout and scheduling problems are integrated and jointly formulated. The results of the process planning problem are the input for the joint scheduling and layout problem.
- Model 4 also concerns the partially integrated approach. It integrates the process planning and layout problems and formulates them jointly. Then, it formulates, in a sequential way, the scheduling problem. This means the results of the joint process planning and layout problem are the input to the scheduling problem.

In all models, the objective is to minimize total costs, including production, material handling, machine reconfiguration, layout reconfiguration, and tardiness costs.

II - The proposition and comparison of three exact solving approaches. The first is the resolution of an integer linear programming. The second and third ones use constraint programming approaches, whether employing a specific research strategy or not.

III - The proposition and comparison of two solution approaches based on an evolutionary algorithm. They are appropriate for the large instances found in industries.

The study's objective is to contribute significantly to research in resource allocation and machine/system reconfiguration for optimal industrial performance. It aims to integrate essential optimization problems, including process planning, layout, and scheduling, to reduce manufacturing costs for mass-customized products in modular and reconfigurable systems.

Outline of the manuscript

This research is structured into seven chapters, as depicted in Figure 1. The first chapter presents the literature review, which begins with an overview of the principal manufacturing systems, emphasizing the key distinctions between Dedicated Manufacturing Systems (DMS), Flexible Manufacturing Systems (FMS), and Reconfigurable Manufacturing Systems (RMS). Subsequently, we enter into the primary characteristics and categorization of RMS optimization problems. Furthermore, an exhaustive literature review of scheduling, process planning, and layout in RMS is conducted. Lastly, this chapter highlights the literature gaps.

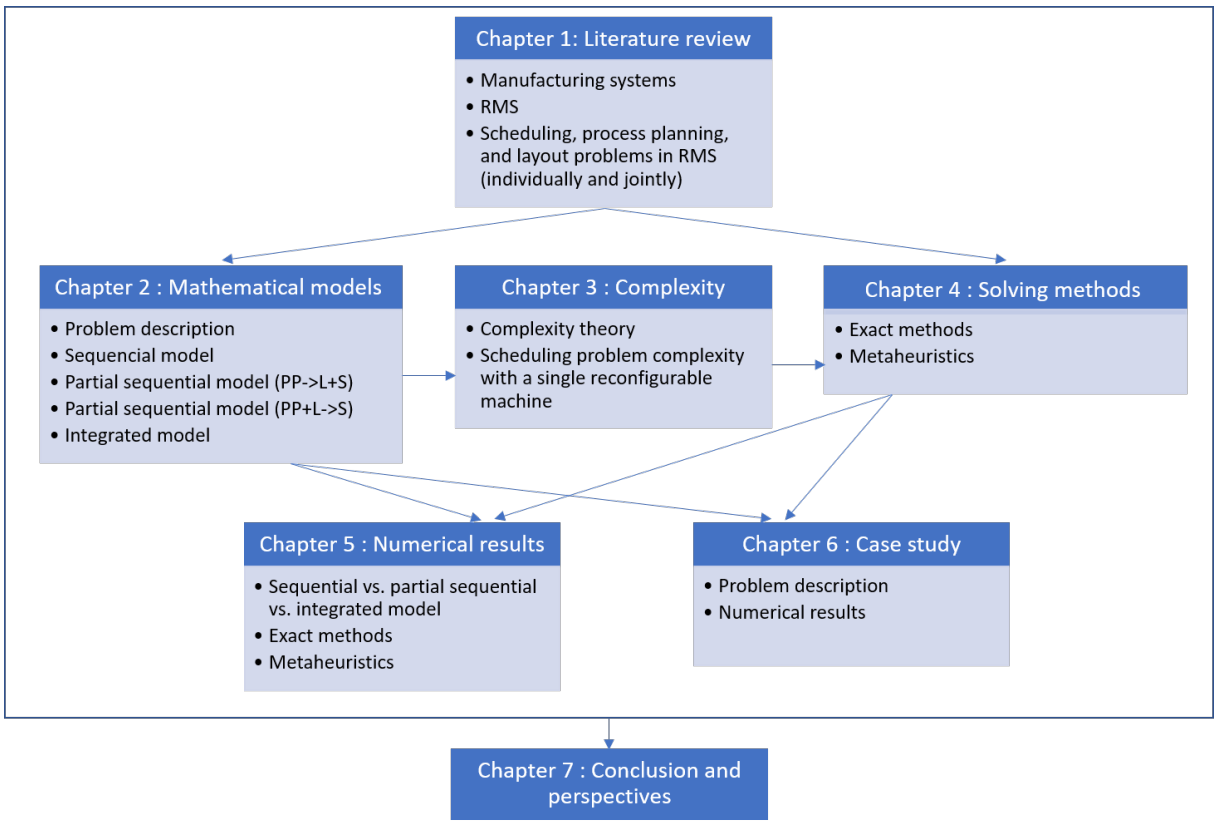

Figure 1: Summary of all chapters presented in this work.

The second chapter begins by describing the studied problem, including the industrial context, parameters, decision variables, and constraints. This chapter discusses different modeling approaches and defends the choices made regarding these methods. Further, four developed mathematical models to address process planning (PP), layout optimization (L), and scheduling (S) in RMS are detailed. The first model ($PP + L + S$) integrates all three problems and formulates them jointly. The second model ($PP \rightarrow L \rightarrow S$) follows a sequential approach, tackling each problem separately and sequentially. The third model ($PP \rightarrow L + S$) adopts a partial sequential strategy, initially formulating the process

planning problem individually and then formulating the layout and scheduling problems jointly. In this model, a sequential approach is also adopted. The outputs of the process planning problem become the inputs of the joint layout and scheduling problem. The fourth model ($PP + L \rightarrow S$) also adopts a partial integration approach, addressing process planning and layout jointly before dealing with scheduling. The chapter concludes by summarizing its findings and contributions.

The third chapter studies the problem's complexity. The chapter explains the classes identified in complexity theory: $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{NP}$-Complete, and $\mathcal{NP}$-Hard. The $\mathcal{NP}$-hard class is known for its significant difficulty. Although process planning, layout, and scheduling problems in RMS have already been proven to be $\mathcal{NP}$-hard, the primary objective of this chapter is not to reiterate the proof but rather to focus on investigating smaller sub-problems and determining their complexity. Specifically, we concentrate on scheduling problems involving a single reconfigurable machine. At the end of this chapter, we present our findings based on the analysis and exploration of problem complexity within the context of reconfigurable manufacturing systems.

The fourth chapter presents and discusses the problem-solving methods developed in this research. The chapter starts with an overview of the exact solving techniques used to ensure optimal results for small-size instances. In this research work, we studied and compared three exact methods. The first employed Integer Linear Programming (ILP) through commercial software based mainly on the simplex algorithm. Second, we explored two Constraint Programming (CP) approaches, one using the standard tools of the software and the other one integrating a specific search strategy adapted to our problem. Subsequently, this research explores non-exact methods, opting to work with two metaheuristics: Genetic Algorithm (GA) and GA with Local Search (LS). The chapter concludes with an analysis and key findings.

The fifth chapter presents numerical case studies to analyze the performance of the four mathematical formulations and models presented in Chapter 2 and the solving methods presented in Chapter 4. This chapter starts through small instances of numerical examples by comparing the performance of the four mathematical models introduced in Chapter 2 using Integer Linear Programming (ILP). The results demonstrate a significant cost superiority of the integrated model, leading us to employ it exclusively for the remainder of this chapter. Second, this chapter compares the execution time of the three exact solving methods for the integrated model on small instances. Constraint programming providing a search strategy had much shorter execution times than the other methods. Third, this work also compares the constraint programming with search strategy with the Genetic Algorithm (GA), examining their execution time and total costs for small instances. These comparisons provide valuable insights into the effectiveness of the

different methods. We analyzed the results provided by the metaheuristics, uncovering their strengths and weaknesses. Finally, in this chapter, we also explore one first industrial applicability with a large instance. Through this wide analysis, this research contributes valuable insights for practitioners and researchers seeking efficient and effective resolution methods for Reconfigurable Manufacturing Systems. This research aims to optimize manufacturing processes and enhance overall system performance by discerning the most suitable approaches.

The sixth chapter presents a complete case study for disassembling a hand light. The chapter starts with the problem description and presents the input data relevant to the disassembly process. Next, the chapter compares and analyzes the results in terms of total cost using the two proposed metaheuristics. Exact methods were not employed due to the size of the problem. The chapter concludes by summarizing the findings and outlining conclusions from the case study, highlighting the practical implications for the industries.

The final chapter discusses all the conclusions and analyses presented. In this chapter, we review the key points addressed throughout the thesis and assess the extent to which they were successfully answered and explored. In addition, we identify any aspects or issues that remain open for further investigation. This chapter provides an overview of the contributions made in this thesis and reinforces the importance of the research carried out. It also highlights the importance of ongoing research and development in the field of Reconfigurable Manufacturing Systems (RMS).

Through this general analysis, we contribute to the advancement of knowledge in the area and enrich the understanding of the optimization of RMS.

Chapter 1

Literature Review

Contents

1.1 Manufacturing systems

Throughout history, several moments have played a remarkable role in the evolution of production systems. One of them, and significantly important, was the diffusion of Dedicated Manufacturing Systems (DMS) to reduce costs and increase production capacities. DMS is still used worldwide and remains arguably the most suitable for some products, especially those with low added value and no need for customization. However, the landscape has changed significantly in many other cases.

With accelerated globalization, consumers now consider not only quality and price but also variety and product customization when making choices [22]. This scenario led to the necessity of reducing the production batch sizes and raising flexibility in production

systems. New production systems, such as Flexible Manufacturing Systems (FMS), were developed in response to these needs. The FMS have a high degree of flexibility thanks to technologies such as Computer Numerical Control (CNC) machines. On the other hand, the production rate is shallow compared to DMS, making big scalability complex and varying capacity only possible in a limited range [60].

Despite its notable flexibility, FMS can be unsuitable for many products due to its high costs, lower throughput, and complex scalability. [35]. Consequently, alternative solutions were explored, leading to the emergence of Reconfigurable Manufacturing Systems (RMS). RMS are specifically designed to rapidly change its structure. This may involve changing the layout of machines within the physical industrial space or modifying machine configurations, either at the software or hardware level. The goal is to quickly adapt the system's production capacity and functionality within a family of parts in response to sudden changes in the market [81].

RMS seek to combine the high productivity of dedicated systems with the versatility of flexible systems. It is not as flexible as FMS and produces less than DMS. While it may not offer the same level of flexibility as FMS or achieve the production output of DMS, RMS offer an intermediate solution with a higher cost-benefit ratio.

The relationship between system cost and capacity is shown in Figure 2 [80]. DMS have a constant cost until they reach their maximum capacity. When the production exceeds this limit, a significant investment is made, even if the need for the capacity increase is small. In the case of FMS, the capacity increase is in constant steps, small and costly, since it is done by adding parallel machines to the system. The RMS differential is that the production rate is adjustable at levels that are not necessarily constant and can adapt more effectively to manufacturing conditions and market changes, making this system the most cost-effective in unstable situations [80].

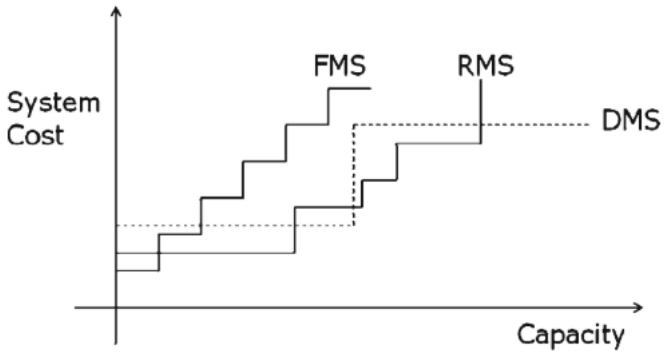

Figure 2: Comparing system cost and capacity in manufacturing systems [80].

The impact between cost and production rates is mainly explained by the machine

type used in each system. DMS are composed of dedicated lines and produce only one product. FMS use CNC machining, which has a high flexibility (more than needed many times) and a high cost. RMS use Reconfigurable Machine Tools (RMT) machines specifically designed for a family of parts. Table 1 summarizes the machines most commonly used in these systems.

System	Machine	Main feature	Scalability
DMS	Dedicated machines	Perform only one type of operation (one configuration)	x
FMS	Computer Numerical Control (CNC)	Automated and numerically controlled machines with a high degree of flexibility. They can perform several different operations just by changing the program being executed and its tools.	x
RMS	Reconfigurable Machine Tools (RMT)	Designed for a set of specific operations or product families	✓

Table 1: Machine types [36, 86].

Table 2 highlights and compares the main characteristics of the three systems discussed thus far. In general, DMS have high productivity and low costs but almost no flexibility. FMS are very flexible and very expensive. RMS end up being a balance between the previous two, as they have limited flexibility (only the needed for a family of products), moderated costs, and reasonable productivity.

	DMS	FMS	RMS
Demand	Low	Variable	Variable
Cost per part	Low	Reasonable	Medium
Initial investment	High	High	Low to medium
Productivity	Very high	Low	High
Flexibility	No	General	Customised
Machine structure	Fixed	Fixed	Adjustable
System structure	Fixed	Adjustable	Adjustable
Variety	No	Very high	High
System focus	Part	Machine	Family part

Table 2: Comparing system features.

Given the market uncertainties and the increasing demand for greater customization by customers, mass-customized products are becoming a widely adopted option for industries. Moreover, due to their balance between flexibility, cost, and productivity, RMS are well-suited to meet current needs efficiently and competitively. Therefore, this research focuses on optimizing these systems to utilize them most effectively.

1.2 Reconfigurable manufacturing systems

As explained above, RMS are most suited for mass-customized products and an unstable and highly changing market due to their advantages, such as high responsiveness to market uncertainties and the ability to meet the varied desires of customers. For these reasons, among others, RMS have been studied more deeply in recent years. In this work, the focus will be entirely on these manufacturing systems, which can be described by six main characteristics [12, 75, 80, 149]:

- Modularity: the use of hardware and software with modular properties to increase the variety/flexibility of products and systems. It is easier and cheaper to change modules than to change entire machines.
- Integrability: the ability to easily add or remove modules adhering to a specific set of system configurations and integration rules. These guidelines must be followed when designing both the systems and their machines at the software and hardware levels. This facilitates the incorporation of existing modules and allows for the integration of potential future technologies that may be introduced over time.
- Customization: the capability and flexibility of hardware and control levels to manufacture different parts of a family. This feature seeks to balance the system costs since it allows more than one part to be produced (unlike DMS) but not any part (unlike FMS).
- Convertibility: the ability to quickly adapt production functionality whenever new needs arise. Some mechanisms are required that allow quick systems to be converted, e.g., for calibrating production after change between parts manufacturing.
- Diagnosability: the self-diagnosis machine-failures and causes of unacceptable quality. Systematic measurement methods and control/statistic techniques to correct problems efficiently.
- Scalability: the possibility of rapid adaptation of the system production capacity, increasing or decreasing it. It can be done by replacing, adding, or removing tools or machines.

All characteristics are interlinked. The design of modular components of a specific family allows interfaces to be quickly integrated, with customization limited to the family part, avoiding unnecessary costs. The modularity and integrability make the system easily convertible to meet changing functionality requests. The fast changeover capability facilitates scalability changes. Additionally, diagnosability plays a critical role in a responsive system, a concept that gained prominence following the emergence of lean manufacturing,

even before the advent of FMS and RMS systems.

Table 3 provides a comparative analysis of the key characteristics of RMS in contrast to DMS and FMS.

	DMS	**FMS**	**RMS**
Modularity	Low	Low	High
Integrability	No	Limited to the system	High
Customization	No	General	Limited to part family
Convertibility	No	High	High
Diagnosability	Low	High	High
Scalability	No	Low to medium	High

Table 3: Comparing RMS main characteristics with DMS and FMS.

The field of study regarding RMS is vast. Bortolini *et al.* [12] categorized research perspectives on this topic into clusters, as illustrated in Figure 3: (Stream #1) the examination of reconfigurability, (Stream #2) the analysis of key characteristics, (Stream #3) performance analysis, (Stream #4) the application of these systems, and (Stream #5) the relationship between reconfigurability and the goals of Industry 4.0.

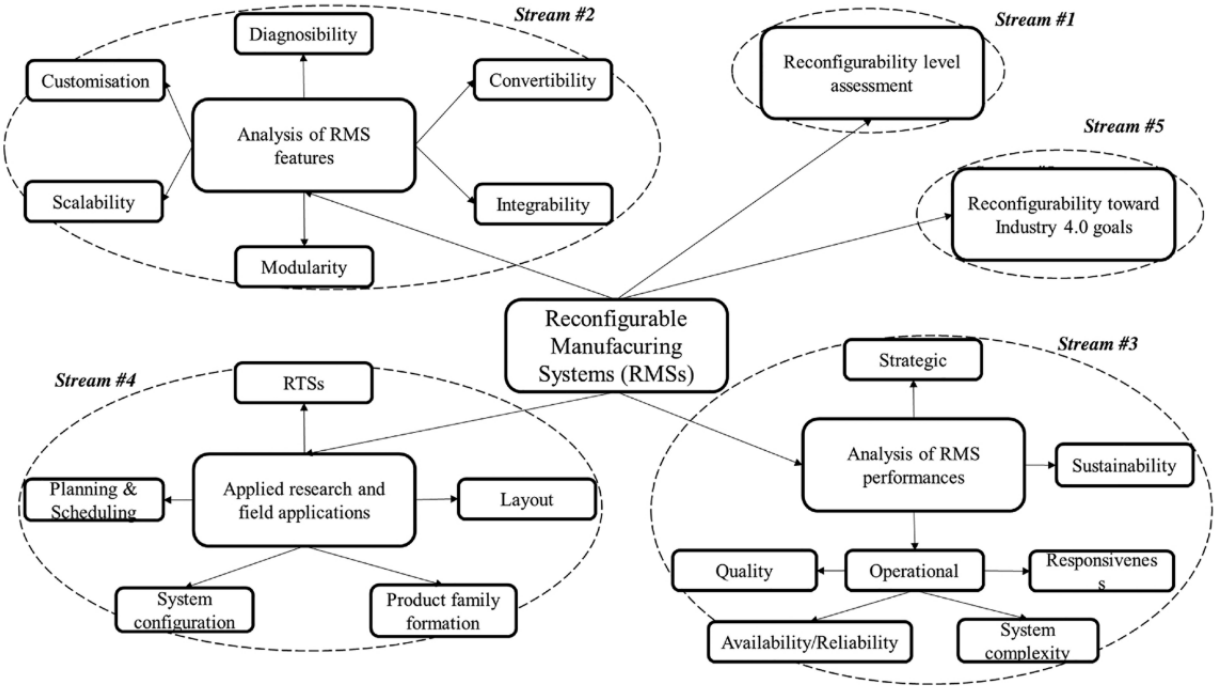

Figure 3: Schematic of RMS research perspectives [12].

This work exclusively focuses on the application of RMS in industries to optimize resources and, therefore, falls within Stream #4. Specifically, among the main tasks involved in applying RMS, this work targets planning, scheduling, and layout design – all of which are optimization problems. Yelles-Chaouche *et al.* [149] classified RMS

optimization problems into four distinct categories (Figure 4): (1) RMS design, (2) production planning and scheduling, (3) layout design, and (4) line balancing and re-balancing. Figure 4 outlines the components of each category, highlighting the problems that fall within our scope.

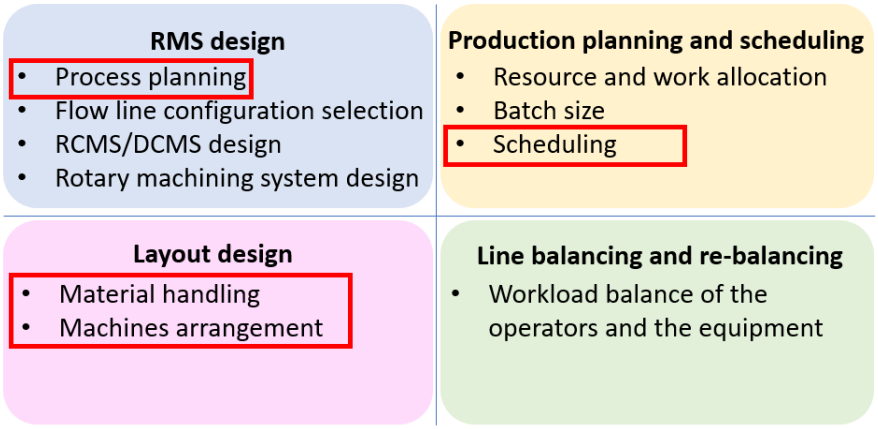

Figure 4: Categorization of optimization problems in RMS based on [149].

The problems of the first category relate to the initial optimization phase, specifically in the preliminary design. Such systems require substantial investments in technology. Thus, robust initial planning is imperative, involving stages such as process planning, flow line configuration selection, reconfigurable cellular manufacturing systems design, and rotary machining system design.

The problems related to production planning and scheduling (PPS) aim at maximizing the system's efficiency within a planning horizon while adapting to customer requirements as effectively as possible.

The problems of the third category, layout design, aim to achieve an optimized physical arrangement. What sets RMS apart is its unique ability to swiftly and cost-effectively reconfigure the layout during the production process, thereby optimizing system efficiency and responsiveness.

Lastly, the problems of the line (re)balancing category involve allocating tasks to workstations, primarily on assembly lines. Typically, these problems are associated with dedicated systems designed for extended periods. Nevertheless, they have been incorporated into RMS literature to address unforeseen issues and necessary system adaptations.

As shown in Figure 4 and stated above, this work focuses on scheduling, process planning, and layout design, including material handling and machine arrangement.

1.3 Optimization problems in RMS: scheduling, process planning, and layout

This section concentrates on the three main topics of this research work: process planning, scheduling, and layout design. It defines the three topics first, then presents a literature review of the most significant works on each topic separately, and then of the works that treated them jointly.

The definitions are as follows:

- The process planning stage is the phase where the best operations sequence to manufacture a product is chosen (respecting the precedence relations), and operations allocation to machines and configurations is determined [103].
- The scheduling stage is the phase where the sequence of operations for all products is determined. By the end of this phase, we have a clear understanding of which action should be performed at what time.[31].
- The layout stage is the phase where the decision for the plant's physical position assignment of equipment/machines is made [94]. This decision directly impacts Material Handling (MH) and system reconfiguration costs and times. It is important to note that in this work, to fit into this category, the times and costs of MH and/or layout reconfiguration must be considered.

All the articles reviewed in this section were sourced from Scopus, Science Direct, and Web of Science. Only articles were analyzed, excluding conference papers. The search process involved the use of three specific expressions: "Reconfigurable manufacturing system" AND "scheduling", "Reconfigurable manufacturing system" AND "process plan", and "Reconfigurable manufacturing system" AND "layout".

Figure 5 shows the number of articles selected for this literature review. The first dotted square shows how many were found using the three sets of keywords. At first, 298 articles were identified when adding the findings of the three databases. 184 articles have remained after removing all duplicates. The last refinement excluded conference articles that were still present, some articles in a language other than English, and articles that were not related to the studied subject after reading their abstract, i.e., in the abstract existed the terms searched but were not the article topic. In the end, 107 articles were selected.

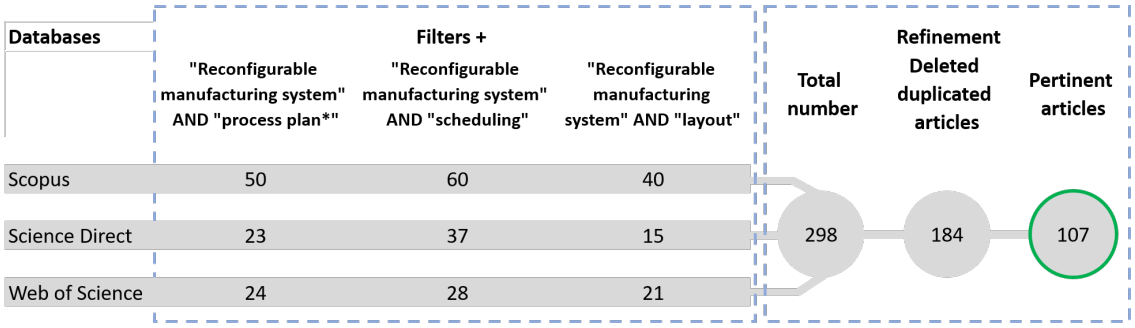

Figure 5: Articles found in the databases and selected.

This research aims to answer the following question: How can we best optimize process planning, scheduling, and layout configuration in an RMS for mass-customized products while reducing manufacturing costs? This is why the literature review focuses on how these three problems were modeled and solved individually and jointly. The objective function is analyzed for the problem modeling to identify the most considered costs and performance indicators.

The rest of this section is organized into seven sub-sections:

- Scheduling;
- Process planning;
- Layout design;
- Scheduling and process planning;
- Scheduling and layout;
- Process planning and layout;
- And finally, scheduling, process planning, and layout.

Each section discusses the objective functions and the solving methods used in the literature and ends with a conclusion.

1.3.1 Scheduling

In RMS scheduling related articles, the most common objectives are minimizing time-related metrics (e.g., tardiness, completion time, makespan) and operation costs, especially reconfiguration, which is one of the different elements compared to classical scheduling problems. Metaheuristics and heuristics are the most frequently used solving methods.

The first article reviewed, published in 1990, Liles & Huff [92] did not address a specific problem with an objective function and solving method. Instead, they introduced RMS concepts, expanding the focus from product scheduling to encompass manufacturing processes. The paper emphasized the importance of integrating process design into product design. They focused on theoretical aspects and proposed a planning and scheduling architecture for RMS. Only over a decade later, the study of specific scheduling problems for RMS was started (e.g., [13, 83]).

Below, we analyze separately the revised works' objective functions and solving methods.

Objective functions

Regarding objective functions, time-related metrics are predominant, as in scheduling for other manufacturing systems (e.g., DMS). Some authors only consider time-related metrics in their works. Hassan & Bright [58] tried to use system flexibility to reduce waiting times and increase cost-effectiveness. Mokhtari [104] improved the makespan in a flow shop, while Ruiz-Torres *et al.* [125] minimized average tardiness in a two-stage hybrid flow shop, considering worker efficiency (average or bottleneck). Yang *et al.* [145] also minimized total tardiness. Fan *et al.* [38] studied a flexible job shop with reconfigurable machine tools with limited auxiliary modules, aiming to minimize total weighted tardiness.

Some authors have included other objectives with time-related functions. Bruccoleri *et al.* [14] studied the cost-benefit of reconfiguring production in response to machine failures, analyzing costs and times of repair, reconfiguration, product transfers, and machine availability. Crawford *et al.* [26] focused on minimizing completion time in a reconfigurable printer system and rerouting operations when a problem was detected. Wang *et al.* [137] modeled a hot stamping process and analyzed the relationship between energy consumption and cycle time. Prasad & Jayswal [113] worked on scheduling product families, aiming to minimize reconfiguration efforts (at the market, system, and machine levels), prioritize close due dates, and increase profit. Vahedi-Nouri *et al.* [134] minimized workers' COVID-19 health risks, the total workers' priorities (prioritizing worker arrival time preferences for increased flexibility), and maximum completion time. Zhan *et al.* [151] studied scheduling optimization for a small, complete, and human-centered assembly unit. They proposed a multi-objective model to minimize completion time and labor costs. Yang *et al.* [144] studied an assembly line centered on optimizing production order and equipment assignment, which required reconfiguration when switching products. The objective function sought to optimize load balance, assembly/reconfiguration costs, and lead time.

Some authors focused only on cost objectives, mainly including reconfiguration costs.

Deif & Elmaraghy [29] addressed capacity scalability scheduling to balance the capacity and reconfiguration costs. Galán [39] developed a methodology to minimize functionality and capacity underutilization, along with system reconfiguration costs when selecting and sequencing product families. Abbasi & Houshmand [1] used a Mixed Integer Nonlinear Programming to maximize profit (inventory holding, production, and reconfiguration costs) by determining the production sequence, configuration, and lot size for product families. Eguia *et al.* [35] integrated cell formation and scheduling problems to minimize reconfiguration and underutilization costs.

A minority of researchers explored problems unrelated to time or reconfiguration costs. Bruccoleri *et al.* [13] managed exceptions in a system that did not allow routing flexibility and was exclusively composed of reconfigurable machines, requiring real-time scheduling decisions during machine breaks. The decisions involved determining if machines could be reconfigured for the interrupting operation and the number of parts to transfer when resuming the suspended process. Kumar *et al.* [83] proposed a model focused on information delays and optimizing job flows. Khan [71] examined product quality and system health, assessing their impact on system's performance and inventory size.

Solving methods

In terms of solving methods, heuristics are frequently employed for RMS scheduling problems and for classic (non-RMS) problems. These problems are often tackled with dispatching rule heuristics like "first come, first served", "shortest processing time", and "earliest due date". Crawford *et al.* [26] studied a reconfigurable printing system prototype, using heuristic evaluation functions for temporal planning, planning algorithms targeting wall-clock goal achievement time, and a chained best-first-search algorithm. Wang *et al.* [137] developed a model based on a finite state machine to represent a hot stamping process utilizing a multi-chamber furnace. Ruiz-Torres *et al.* [125] proposed a two-stage hybrid flow shop with several workstations, where the solution method involved multiple steps, some of which could be addressed with different scheduling rules. Khan [71] focused on scheduling, emphasizing on diagnosability. He proposed a model for a single product, analyzing the system's ability to identify and replace a unit needing substitution, such as machine modules or tools. Three heuristics were employed to solve the model, combining or omitting distinguishability and stochastic settings.

Metaheuristics are the preferred approach for tackling the scheduling problems in RMS due to their high complexity ($\mathcal{NP}$-hard problems), surpassing the popularity of heuristics. These problem-solving techniques draw inspiration from biological processes like natural evolution [64]. Among the widely used metaheuristics, Genetic Algorithm (GA) stands out. For instance, Deif & Elmaraghy [29] employed GA techniques to address

capacity scalability scheduling. They built a computer tool that utilized the GA technique to decide when and how much to scale. Abbasi & Houshmand [1] proposed a mixed-integer non-linear programming model considering stochastic parameters and developed a GA-based method. Fan *et al.* [38] tackled a flexible job shop scheduling problem by introducing a Mixed Integer Linear Programming (MILP) model and developed an improved genetic algorithm using iterated local search, applying a k-insertion approach to critical paths. Zhan *et al.* [151] developed an algorithm combining NSGA-II with Differential Evolution, a population-based evolutionary algorithm.

In addition to genetic algorithms, diverse other metaheuristic methods are employed in RMS scheduling research. Galán [39] focused on selecting and sequencing product families, utilizing product requirements (modularity, commonality, compatibility, reusability, and demand) to create similarity matrices between products and applying the Analytic Hierarchy Process and Average Linkage Clustering algorithm to group products into families. Two hybrid approaches, a nearest neighbor heuristic with Tabu Search and an Ant Colony System with local search, were compared for family selection and scheduling. Eguia *et al.* [35] presented a method addressing cell formation and scheduling, using mixed-integer linear programming for smaller instances and a tabu search algorithm for larger ones. Mokhtari [104] improved flow shop scheduling using a group search optimizer algorithm and innovative parameter calculation techniques involving the Euler method, Runge-Kutta method, quadratic interpolation, and Levy distribution. Yang *et al.* [144] developed a variant of multi-objective particle swarm optimization.

In contrast to the works mentioned above, some authors incorporated fuzzy aspects into their research. Bruccoleri *et al.* [13] employed multiple agent systems and fuzzy reasoning to handle exceptions, particularly machine breakdown. Kumar *et al.* [83] introduced an expert-enhanced colored fuzzy Petri Net model, considering customer demands as a fuzzy parameter. Prasad & Jayswal [113] presented a scheduling model for product families based on fuzzy logic rules, enabling insights into the interplay between reconfiguration effort, priority, and due date variables. A comparison with the weighted aggregate sum method revealed that divergence points primarily revolved around priority considerations.

Other authors adopted alternative solution techniques. Bruccoleri *et al.* [14] introduced an object-oriented control architecture for error handling. Hassan & Bright [58] proposed a reconfigurable computer-integrated manufacturing cell to address mass-customized product manufacturing, outlining both hardware and software aspects and presenting a simulation. Vahedi-Nouri *et al.* [134] tackled an integrated workforce planning and production scheduling problem within the context of COVID-19, utilizing MILP and Constraint Programming (CP). They identified and integrated two optimal solution

properties into both methods. Similarly, Eguia *et al.* [35] and Fan *et al.* [38] also explored exact methods, restricted to small instances. Yang *et al.* [145] employed deep reinforcement learning (deep Q-network) to handle a problem with real-time job arrivals. They had two agents: a reconfiguration agent to assess the need for system reconfiguration and a scheduling agent to determine which jobs to produce, both using priority dispatching rules.

Conclusions

Table 4 summarizes of solving methods and objectives from the papers covered in this section. Additionally, some general conclusions emerge. In most cases, the primary focus revolves around minimizing total production time, as in classical scheduling problems. However, since the RMS differential is the reconfigurability, reconfiguration time and cost can also be considered. In addition, each author adds objectives pertinent to their studies, which can vary depending on the industry's product, process, or strategic positioning. Notably, metaheuristics are the dominant methods, followed by sequencing rules heuristics and exact methods, mainly employed for model validation.

	Solving Methods							**Objectives**						
Paper	**GA**	**OM**	**H**	**EM**	**CP**	**FL**	**OS**	**TF**	**RC**	**OC**	**MF**	**ENV**	**OO**	**Year**
[92]														1990
[13]						x					x			2003
[83]						x							x	2005
[14]							x	x	x		x			2006
[29]	x								x	x				2007
[39]		x							x	x				2008
[1]	x								x	x				2010
[58]							x	x						2012
[35]		x		x					x	x				2013
[26]			x					x			x			2013
[104]		x						x						2015
[137]			x					x				x		2017
[113]						x		x	x	x				2017
[125]			x					x						2021
[38]	x			x				x						2022
[71]			x							x				2022
[134]				x	x			x					x	2022
[151]	x	x						x		x				2022
[144]		x						x	x				x	2022
[145]							x	x						2023

GA: genetic algorithm; OM: other metaheuristics; H: heuristics; EM: exact methods; CP: constraint programming; FL: fuzzy logic; OS: other solving methods; TF: time functions; RC: reconfiguration costs; OC: other costs; MF: machine failure; ENV: environment; OO: other objectives.

Table 4: Summary of RMS scheduling articles.

In this review, several key points stand out. Scheduling problems typically center around two main objectives: sequencing product families [1, 35, 39, 113] or sets of

products [38, 104, 125]. Notably, the system does not need to be exclusively composed of reconfigurable machines; even non-reconfigurable systems can benefit from reconfigurable ones [14]. Genetic Algorithm (GA) emerges as a highly efficient technique, although the effectiveness of metaheuristics can be enhanced through hybridization with methods [38, 39, 104] like local search. The reconfigurability offered by RMS is a strong ally in addressing issues such as machine breakdowns [13, 14, 26]. Examining product quality aids in identifying system problems [71]. Moreover, an increase in non-conforming products can drive up demand for stock, making stock costs less significant for products with high conformity rates [71]. Identifying specific problem properties can significantly enhance the efficiency of solving methods.

1.3.2 Process planning

Among the articles considered in this review, process planning problems emerge as the most extensively studied category, representing almost 30%. They gained more attention over the past decade, in contrast to scheduling, which has been a longstanding area of investigation. For this problem, the objective functions continue to be mainly time-related functions and reconfiguration costs. In addition, a new critical factor is being considered here: material handling costs. The process planning, which involves selecting the machine for each operation, directly impacts transportation costs and times. Therefore, MH can significantly influence the efficiency of systems, underscoring its importance in this context. Regarding solving methods, metaheuristics are by far the most used, much more than in the previous section.

Objective functions

Regarding objective functions, the primary focus in most works is on minimizing total costs, including reconfiguration costs [4, 20, 22, 48, 51, 54, 70, 72, 76–78, 88, 96, 106, 107, 131, 132, 143], a significant concern in RMS. While material handling costs are also noteworthy due to their dependence on machine and system configurations, they are comparatively less present in the literature [22, 106, 107, 131, 132]. Other cost factors considered encompass machine and tool usage [11, 22, 54, 70, 78, 96, 106, 107], production [4, 20, 72, 76–78, 131, 132, 143, 147], idle cost due to unbalanced station workloads [143], scrap cost of defective operations [72] , rework [72], tolerance [4], and raw material [20, 147] costs.

Time-related functions are recurrent, especially emphasizing minimizing completion time [53, 54, 70, 76, 77] and lead time [74, 133]. Some studies also targeted minimizing production [22, 48, 78, 131, 132], reconfiguration [7, 14, 22, 78, 131, 132], material handling [14, 22, 131, 132], and machine repair [14] times.

Environmental concerns are increasingly prominent in process planning, with efforts to minimize gas emissions [76–78, 131, 147], hazardous waste [76–78, 147], and energy consumption [98, 133]. Authors have explored various other objectives as well. Shabaka & Elmaraghy [127] developed an approach for machine and configuration selection based on machine capabilities. Abellan-Nebot *et al.* [2] introduced a methodology to minimize product variability. Xie *et al.* [143] sought to balance workload and machine accuracy. Musharavati & Hamouda [107] aimed to maximize throughput. Gupta *et al.* [51] worked on minimizing under-utilization. Haddou Benderbal *et al.* [53] focused on maximizing the flexibility index. Haddou Benderbal *et al.* [54] concentrated on maximizing system modularity. Asghar *et al.* [7] aimed to maximize machine capabilities. Khan *et al.* [72] minimized non-conforming products and modularity effort. Yazdani *et al.* [147] sought to maximize social sustainability. Finally, Gonnermann *et al.* [48] wanted to enhance monitoring efficiency, among other objectives. Barrera-Diaz *et al.* [9] tackled a problem with buffers, aiming to maximize throughput and minimize total buffer capacity.

Solving methods

Metaheuristics are the dominant category of solving methods for process planning, with Genetic Algorithm (GA) and their variants once again emerging as the most prevalent choice. Some authors exclusively employed genetic algorithms. Chaube *et al.* [22] optimized the process planning for a single product with several parts using an adapted NSGA-II. Bensmaine *et al.* [11] tackled an RMS design problem for creating new production systems, opting for NSGA-II. Haddou Benderbal *et al.* [53] also worked with NSGA-II. Asghar *et al.* [7] introduced a method to optimize part family process planning via a multi-objective genetic algorithm. Ameer & Dahane [4] explored setup generation and process plans, developing a two-step hybrid genetic algorithm approach. The first step employed a heuristic for setup and fixture selection, followed by the use of a GA for process plan selection. Khettabi *et al.* [76], on the other hand, addressed their problem using four distinct genetic algorithms: NSGA-II, NSGA-III, weighted GA, and random weighted GA.

Some authors compared or mixed GA and other metaheuristics. Xie *et al.* [143] addressed their problem by employing GA and Simulated Annealing (SA). Touzout & Benyoucef [132] explored a multi-unit single-product, utilizing three hybrid metaheuristics that combined NSGA-II, single-unit process plan heuristics, and Iterated Local Search. In a other work, Touzout & Benyoucef [131] introduced an Archived Multi-Objective Simulated Annealing and NSGA-II approach. Khezri *et al.* [78] worked with a single product, employing NSGA-II and Strength Pareto Evolutionary Algorithm II. Khan *et al.* [74] studied a single-machine part problem and proposed two meta-heuristic algorithms for its resolution: GA and Cuckoo Search. GA demonstrated superior results in terms of

both outcomes and execution time. Khan *et al.* [72] also delved into the impact of product quality variation on process planning, solving two models through a hybrid metaheuristic that combined NSGA-II and Multi-Objective Particle Swarm Optimization (MOPSO). Kazemisaboor *et al.* [70] tackled a single-unit process planning (PP) problem employing three evolutionary algorithms (NSGA-II, AMOSA, and MOPSO), with the results serving as input for a multi-unit process planning scenario with unpredictable demands, solved using the Designed Periods Algorithm. Khettabi *et al.* [77] compared Pareto Evolutionary Algorithm II, NSGA-II, and NSGA-III.

A smaller subset of papers chose not to explore variations of GA. Instead, they explored other metaheuristics. For instance, Musharavati & Hamouda [107] compared four Simulated Annealing (SA) methods, some containing alternative algorithmic concepts, for a multi-part process planning problem. In another study, the same authors extended their comparison to four different SA methods [106]. Maniraj *et al.* [96] focused on the process planning problem within a single-product flow-line RMS, developing a two-part ant colony optimization approach. Haddou Benderbal *et al.* [54] created an approach to address the configuration selection problem, encompassing both system and machine levels, introducing a method based on AMOSA. Barrera-Diaz *et al.* [9] developed an improved simulation-based multi-objective optimization approach that employed customized simulation and optimization components, utilizing discrete-event for simulation and NSGA-II for optimization.

Some works employed exact methods for validating mathematical models and solving the proposed problems [72, 78, 131]. Massimi *et al.* [98] presented an exhaustive heuristic for determining the sequence of operations involving a single product unit and its associated machines and modules. Campos Sabioni *et al.* [20] proposed a model addressing the joint optimization of product configuration and process planning, employing an exact approach characterized by an exhaustive search algorithm and integer linear programming. Yazdani *et al.* [147] presented a linear multi-objective mixed-integer mathematical model, which was tackled using an Lp-metric and a Lagrangian relaxation-based heuristic algorithm. Additionally, Gonnermann *et al.* [48] developed a methodology comprising four modules to address PP and process monitoring for assembly lines. The first module generates assembly plans, the second handles monitoring processes, the third optimizes both, and the final one conducts simulation to validate collision freedom and feasibility in process planning. The case studies were still applied to small examples, so the optimization was done with a MILP.

Several authors explored alternative approaches to address and resolve their problems. Shabaka & Elmaraghy [127] developed an approach that selects machines and their configurations based on machine capabilities. Abellan-Nebot *et al.* [2] proposed a

methodology for selecting process plans for new products by analyzing existing data from a multi-station manufacturing process's prior productions. Gupta *et al.* [51] focused on part family formation, utilizing a clustering approach to construct a dendrogram (a diagram that shows the hierarchical relationship between objects) and determine the production order of family parts. Um *et al.* [133] used the STEP-NC machine control language tool to select required machine tools, factoring in machine capability, workspace, and tolerance. Although not exclusively centered on RMS, this article holds substantial value for applying such systems in industry. Leng *et al.* [88] introduced a digital twin concept centered on open-architecture machine tools, Industrial Internet of Things communication, and digital-twin structures, extending beyond our primary research focus.

Conclusions

Table 5 provides an overview of the key findings in the reviewed process planning articles. Notably, metaheuristics, particularly GA, emerged as the dominant solving method. Objective functions covered a wider range of criteria. Material handling has garnered attention as a crucial factor, given its potential to impact overall costs significantly. Furthermore, environmental concerns have emerged more strongly in recent years, becoming an increasing part of the research landscape.

The studies in this context varied, with some focusing exclusively on a single product [2, 22, 96, 98, 132], while others studying multi-part [106, 107, 127] or part family formation scenarios [51]. A majority of authors opted for multi-objective functions, necessitating the use of multi-criteria decision techniques, with the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) method being the most commonly employed [53, 54, 76, 77]. An exception was noted in Xie *et al.* [143], where a GA approach with two objectives was developed, eliminating the need for multi-objective methods like NSGA-II. The functions were normalized to address different objective dimensions. Hybrid metaheuristics demonstrated enhanced objective function results [72, 132], although efforts are required to reduce their time consumption [132]. GA and its variations yield good results, outperforming other methods [72, 74, 77]. Problem sizes tended to be smaller compared to scheduling; for example, Khan *et al.* [74] worked with one part and one machine. Likely due to this reason, it is possible to find authors who have employed exact methods not only for validation but also as their primary approaches [20, 48, 98, 147].

	Solving Methods						Objectives						
Paper	**GA**	**OM**	**H**	**EM**	**CP**	**OS**	**TF**	**RC**	**MHC**	**OC**	**ENV**	**OO**	**Year**
[127]			x									x	2007
[22]	x						x	x	x	x			2010
[2]						x						x	2011
[143]	x	x						x		x		x	2012
[107]		x						x	x	x		x	2012
[106]		x						x	x	x			2012
[11]	x						x	x		x			2013
[96]		x						x		x			2014
[51]						x		x				x	2014
[133]						x	x				x		2016
[53]	x						x					x	2017
[54]		x					x	x		x		x	2017
[7]	x						x					x	2018
[131]	x	x		x			x	x	x	x	x		2018
[132]	x	x					x	x	x	x			2019
[98]				x							x		2020
[88]						x		x					2020
[78]	x	x		x			x	x		x	x		2020
[74]	x	x					x						2021
[72]	x	x		x				x		x		x	2021
[4]	x							x		x			2021
[76]	x						x	x		x	x		2021
[70]	x	x					x	x		x			2021
[20]				x				x	x	x			2022
[147]				x						x	x	x	2022
[77]	x	x					x	x		x	x		2022
[48]				x			x	x				x	2022
[9]		x										x	2022

GA: genetic algorithm; OM: other metaheuristics; H: heuristics; EM: exact methods; CP: constraint programming; OS: other solving methods; TF: time functions; RC: reconfiguration costs; MHC: material handling costs; OC: other costs; ENV: environment; OO: other objectives.

Table 5: Summary of RMS process planning articles.

1.3.3 Layout design

The RMS layout problems aim to optimize the positioning of machines and equipment, mainly influenced by layout reconfiguration and material handling costs and time. Fewer research papers are devoted to addressing the layout problem compared to previous topics. While objective functions still encompass factors like reconfiguration costs, material handling costs, and time functions, a dominant trend in these studies is incorporating additional objectives, such as quality, production capacity, and security. In terms of solving methods, unlike the previous problems, the frequent use of metaheuristics is less common here, with most studies concentrating on specific techniques.

Objective functions

In layout problems, the objective is often to minimize reconfiguration and material handling costs. Many authors incorporate these goals with additional ones. For instance, Zhao *et al.* [153] sought to identify the optimal configuration for family changes, optimizing profit (reward earned minus reconfiguration costs). Reza Abdi [121] based their decisions on a combination of reconfiguration cost, quality, layout reconfigurability, operators, and inventory. Oke *et al.* [109] searched for a more fluid material handling flow, not necessarily aiming to optimize time and costs. Guan *et al.* [49] aimed to reduce both material handling and workstation reconfiguration costs, while Wei *et al.* [139] minimized material handling and equipment replacement costs and maximized the workshop area. Arnarson *et al.* [6] focused on minimizing the total MH done by robot platforms.

Time-related functions are not predominant in layout problems but are still present. Dang *et al.* [27] focused on determining transport paths to minimize time or energy consumption, also considering collision avoidance between pallets. Park *et al.* [112] aimed to reduce layout reconfiguration times in distributed manufacturing systems. Mo *et al.* [102] introduced a framework consisting of three reconfiguration phases. The first phase involved analyzing the capability match between the current system and the requirements of a new product. The second phase focused on optimization, including machine placement, production parameters, and sequence using the selected assets. The final phase involved configuration updates to implement the reconfiguration. In a small case study, the objective was to minimize total operating time and collision occurrences.

Additionally, some authors pursued different layout optimization objectives. Han *et al.* [55] analyzed the feasibility of storage, transport paths, processing, preparation, and tool handling. Garbie [42] aimed to increase production capacity via resource, layout, and material handling analysis. Vavrík *et al.* [135] evaluated the system's ability to meet delivery schedules, product prioritization, product family classification, and determine the optimal line configuration. Sallez *et al.* [126] focused on ensuring security, Gašpar *et al.* [45] optimized hexapod placement, and Marschall *et al.* [97] studied mobile robotic system interaction with reconfigurable modular systems, exploring scenarios where robots could manipulate platforms with mechanical arms, pallets, batteries, and other components. Their main objective was to find the number of robots needed in relation to the available platforms.

Solving methods

In contrast to previous sections where problems were predominantly addressed with metaheuristics, they played a less prominent role here. Guan *et al.* [49] tackled a Quadratic Assignment Problem where each location had pickup and delivery points, employing a

revised electromagnetism-like mechanism heuristic to solve it. Wei *et al.* [139] addressed a dynamic layout problem across multiple planning periods in a continuous shop floor with regular and unequal equipment, utilizing a chaotic genetic algorithm with an improved Tent mapping. Meanwhile, Mo *et al.* [102] explored optimization methods such as GA, SA, and PSO for their study. Arnarson *et al.* [6] utilized NSGA-II for layout optimization and validated it via digital model simulation.

Some authors opted for heuristics in their approaches. Oke *et al.* [109] focused on a scenario only with fixed machines, due to the impossibility of moving them. They introduced a U-shaped layout system with an intermediary buffer. When a new product with a different process plan was introduced, their heuristic algorithm suggested a new material handling routing. In other work, Garbie [42] proposed a three-phase methodology encompassing production, plant layout, and material handling systems. They employed a heuristic to readjust the system in response to changes in demand or catalog.

In terms of exact solving methods, Guan *et al.* [49] employed them for small instances and to validate the proposed metaheuristic. Zhao *et al.* made a sequence of four articles to optimize system configurations through a stochastic model. The first article [154] introduced a framework with three core components: optimal configurations during design, optimal selection policies during utilization, and performance metrics during improvement. Subsequent articles studied each component in depth. Here, we focus only on the first one [153]. In Zhao *et al.* [153], products were grouped into families, sharing similar products for the same system configuration. Reconfiguration occurred when there was a change in the production family, utilizing an iterative process involving exact calculations for small instances and simulation approximations for larger instances. Gašpar *et al.* [45] addressed production lines for rigidly fixed parts; they tackled the problem using nonlinear programming.

Some authors used different solving methods. Reza Abdi [121] used a fuzzy Analytical Hierarchy Process model to choose between three layout configurations (serial, parallel, and hybrid). Han *et al.* [55] proposed a simulation-based approach for manufacturing layout selection, creating a tool with a list of configuration parameters to determine feasibility. Dang *et al.* [27] explored an Electromagnetic Modular Smart Surface architecture, investigating pallet movement with path-planning algorithms like the Floyd Warshall algorithm to find the shortest path and interpolation techniques such as Cubic Hermite to refine the trajectory. Park *et al.* [112] introduced an agent communication framework featuring self-layout recognition through infrared communication, focused on multi-agent protocols beyond the scope of this work. Vavrík *et al.* [135] developed an approach to design reconfigurable manufacturing lines, combining mathematical operations for layout design with simulation, cluster analysis, and the longest common subsequence algorithm.

Sallez *et al.* [126] tackled safety concerns between robots and operators in a reconfigurable assembly system, using a two-phase methodology involving offline layout construction on a multi-agent platform and online implementation and system validation. Marschall *et al.* [97] examined interactions between a mobile robotic system and reconfigurable modular systems, analyzing different layouts and simulating diverse scenarios.

Conclusions

In summary, Table 6 contains the key information regarding layout problems, which stand out as the most diverse among the individual problems discussed in this review. Metaheuristics are not the predominant solving methods, and time-related functions are not the primary considered objective. Moreover, the modeling approaches for these problems can be quite distinct. Reza Abdi [121] focused on selecting from three layout options (serial, parallel, and hybrid). Wei *et al.* [139] addressed a continuous shop floor scenario with regular and unequal equipment. Guan *et al.* [49] had predefined locations for equipment placement. Oke *et al.* [109] worked with material handling with fixed machines. Dang *et al.* [27] concentrated on optimizing pallet transport for material handling. In a departure from these perspectives, Marschall *et al.* [97] aimed to balance the number of robots and platforms, while Gašpar *et al.* [45] concentrated on positioning a fixed structure.

	Solving Methods						Objectives						
Paper	**GA**	**OM**	**H**	**EM**	**CP**	**OS**	**TF**	**RC**	**MHC**	**OC**	**ENV**	**OO**	**Year**
[154]													2000
[153]				x		x		x					2000
[121]						x			x	x		x	2009
[109]			x						x				2011
[49]		x		x				x	x				2012
[55]						x						x	2012
[42]			x									x	2014
[27]						x	x				x		2016
[112]						x	x						2019
[139]	x							x	x			x	2019
[135]						x						x	2020
[126]						x						x	2020
[45]				x								x	2021
[97]						x						x	2022
[102]		x					x					x	2023
[6]		x							x				2023

GA: genetic algorithm; OM: other metaheuristics; H: heuristics; EM: exact methods; CP: constraint programming; OS: other solving methods; TF: time functions; RC: reconfiguration costs; MHC: material handling costs; OC: other costs; ENV: environment; OO: other objectives.

Table 6: Summary of RMS layout articles.

1.3.4 Scheduling and process planning

Moving on to the integrated problems, we notice a decrease in the number of articles in each subsequent subsection compared to the previous ones. The integration of scheduling and process planning is a widely explored area, particularly in problems without reconfiguration, where it has shown notable advantages over treating them individually [129]. Within the RMS context, the focus is on optimizing time-related functions, such as conventional scheduling problems. Furthermore, several solving methods are employed, with a predominant emphasis on heuristics and exact methods.

Objective functions

Most authors have focused on time-related objectives, with some exclusively addressing these criteria. Doh *et al.* [30] sought to minimize a range of time-related functions, encompassing makespan, total flow time, mean tardiness, the number of tardy jobs, and maximum tardiness. Yu *et al.* [150] minimized the makespan, mean flow time, and mean tardiness, while Bensmaine *et al.* [10] concentrated only on minimizing the makespan.

In other studies, time-related functions have been considered alongside additional objectives. For instance, Dou *et al.* [32, 33] addressed cost minimization (capital and reconfiguration costs) and tardiness. Ivanov *et al.* [67] sought to minimize production costs and makespan. Rahman *et al.* [115], on the other hand, exclusively targeted makespan reduction. In contrast, Choi & Xirouchakis [23] deviated from time-related functions and instead focused on minimizing energy consumption, inventory holding costs, and/or maximizing throughput.

Solving methods

To solve the joint scheduling and process planning, some studies employed sequencing heuristics. Doh *et al.* [30] tackled a job scheduling problem with multiple process plans, applying integer linear programming to instances with up to 10 jobs and 10 machines. For larger instances, they examined 36 options by comparing three rules for operation/machine selection and 12 rules for job sequencing. Yu *et al.* [150] introduced a priority rule-based heuristic for flexible job shop scheduling with multiple processes. Lastly, Bensmaine *et al.* [10] integrated process planning and scheduling problems by deploying a heuristic that assessed machine availability and then defined a selection index to determine the optimal operation sequencing and machine assignment.

Evolutionary algorithms are also used in these integrated problems. Dou *et al.* [33] introduced a mixed-integer programming model for reconfigurable flow lines, initially validating it with an exact method and subsequently employing NSGA-II using a hybrid

encoding approach to ensure chromosome feasibility. Another study by Dou *et al.* [32] employed a multi-objective particle swarm optimization, which outperformed NSGA-II in the presented examples. Rahman *et al.* [115] proposed a methodology for predictive scheduling and reactivity control, considering products with multiple process plans and using a GA in conjunction with priority dispatching rules for optimization. Before implementing the instructions in the real system, a simulation was conducted to assess feasibility and refine the schedule.

Furthermore, Choi & Xirouchakis [23] proposed a linear programming model to optimize process planning and the production quantity while balancing conflicting objectives. Ivanov *et al.* [67] developed a control approach incorporating dynamic structural-logical constraints. These constraints vary based on machine availability, representing dynamic changes. A dynamic control algorithm, using problem decomposition, was devised. Castañé *et al.* [21] introduced the ASSISTANT European project, a decision support system for agile manufacturing environments. It aims to control process planning, production planning, scheduling, and real-time operations. This ambitious project advocates for developing digital twins and using Artificial Intelligence (AI) tools, including machine learning, Internet of Things (IoT), CP, stochastic timing, and simulation. It integrates various components like data management, ethical and safety compliance, and an AI-based controller for real-time decision-making. While this article provides an overview of the project, in-depth demonstrations, use cases, architectures, algorithms, and code will be made available upon the project's completion.

Conclusions

Table 7 provides a summary of the articles jointly addressing scheduling and process planning in RMS. In general, these studies heavily rely on heuristic and metaheuristic approaches, primarily targeting time-related objectives, often coupled with other cost minimization criteria. Additionally, we observed noteworthy points. Doh *et al.* [30] studied a problem where process plans have varied types and quantities of operations, contrary to most works where the quantity of operations per product is fixed. Bensmaine *et al.* [10] proposed a heuristic approach that effectively integrated scheduling and process planning, showcasing its superiority over sequential strategies, as demonstrated in non-RMS problems. Dou *et al.* [33] focused on determining the number of stations and identical parallel machines in each station. It is essential to clarify that while some authors like Campos Sabioni *et al.,* categorize such problems as layout design, our layout category emphasizes material handling and equipment displacement costs and times, which do not apply in this case.

	Solving Methods						Objectives						
Paper	**GA**	**OM**	**H**	**EM**	**CP**	**OS**	**TF**	**RC**	**MHC**	**OC**	**ENV**	**OO**	**Year**
[30]			x	x			x						2013
[150]			x				x						2013
[10]			x				x						2014
[23]				x						x	x	x	2014
[33]	x			x			x	x		x			2020
[67]						x	x			x			2020
[32]		x		x			x	x		x			2020
[115]	x						x						2020
[21]													2020

GA: genetic algorithm; OM: other metaheuristics; H: heuristics; EM: exact methods; CP: constraint programming; OS: other solving methods; TF: time functions; RC: reconfiguration costs; MHC: material handling costs; OC: other costs; ENV: environment; OO: other objectives.

Table 7: Summary of RMS scheduling and process planning articles.

1.3.5 Scheduling and layout

Analyzing scheduling and layout together is crucial since system reconfiguration and material handling times and costs significantly influence these decisions. Therefore, optimizing equipment positioning and task sequencing can enhance production efficiency. In this section, we observe a balanced distribution of solving methods and objectives; there is no predominant solving approach or objective function.

Objective functions

Some authors have concentrated only on optimizing time-related functions. Yang & Xu [146] and Gao *et al.* [40] aimed to minimize tardiness penalties, while Naderi & Azab [108] and Manafi *et al.* [95] focused on minimizing makespan. Guo *et al.* [50] sought to minimize total waiting and setup times with additional setup quantities and tardiness analyses.

On the other hand, certain works have exclusively targeted cost-related objectives. Ye & Liang [148] minimized total costs, encompassing production, layout reconfiguration (both intracellular and extracellular), machine idle time, material handling, and work-in-process inventory costs. Renna & Ambrico [120] aimed to reduce intercell material handling, machine reallocation, and machine reconfiguration costs while maximizing profit. For Xia *et al.* [141], the goal was to minimize overall maintenance costs.

In other studies, a combined approach has been adopted, considering both time and cost factors. Prasad & Jayswal [114] developed a methodology incorporating reconfiguration effort, profit over cost ratio, and due date considerations. Han *et al.* [56] aimed to minimize unfulfilled demand within due dates, completion times, reconfiguration costs, and penalty costs for unfulfilled demands. Notably, Ghanei & Algeddawy [46] addressed sustainable

and economic aspects, striving to minimize total energy consumption (intra-day price fluctuations), system reconfiguration, and material handling costs. Khan *et al.* [73] aimed to minimize total time, maximize the line efficiency index, and enhance the customer satisfaction index.

Solving methods

As expected, some of the authors worked with metaheuristics. Ye & Liang [148] tackled these problems in reconfigurable cellular manufacturing systems, focusing on modular products with multiple parts. They initially employed an exact method to validate the model and subsequently developed a GA. Ghanei & Algeddawy [46] introduced a MILP and employed a GA. Han *et al.* [56] explored the integration of IoT sensors into RMS to enhance reconfiguration (machine and system) decision-making, employing a meta-heuristic based on a variable neighborhood search algorithm. They also compared three production scheduling dispatching rules. Gao *et al.* [40] proposed a linear mathematical model to optimize machine positioning using Cartesian coordinates and scheduling. Small instances were evaluated using an exact method in conjunction with the developed genetic algorithm. Naderi & Azab [108] presented two metaheuristics based on an artificial immune algorithm to address scheduling challenges in reconfigurable assembly systems. Manafi *et al.* [95] addressed scheduling and AGV routing while considering collision avoidance and battery-related factors. They developed a MILP model and applied a centroid opposition-based coral reefs metaheuristic for optimization.

Stepping away from metaheuristics, Renna & Ambrico [120] exclusively used exact methods. They introduced a sequential approach consisting of three distinct mathematical models for cellular manufacturing: (1) The "main model" conducted an extended analysis over one year and formulated the initial cell designs. (2) The "sub-model" used the output data from the previous model, operating over a shorter time horizon (3 months). (3) The "allocation model" took the previous results and handled weekly production scheduling without further reconfiguration.

Other authors adopted unique approaches. Xia *et al.* [141] presented a reconfigurable maintenance time window method through real-time interactive bi-level scheduling. Prasad & Jayswal [114] employed an integrated methodology combining Shannon Entropy for weight assignment and the Reference Ideal Method for multicriteria decision-making in RMS design and product scheduling. Yang & Xu [146] utilized deep reinforcement learning techniques for real-time production system optimization. Guo *et al.* [50] tackled fixed-position assembly islands, focusing on products that couldn't be moved during assembly due to their large size or fragility. The authors introduced a synchronization-oriented reconfiguration mechanism based on Industrial Internet of Things (IIoT) and digital twin

technology to manage resource organization and scheduling on the island to address this. Khan *et al.* [73] developed a heuristic employing four different assessments considering reconfiguration possibilities at both the machine and layout levels.

Conclusions

Table 8 summarizes the articles reviewed in this section. Solving methods are generally more distributed, and the goals are mainly centered around time-related functions and costs (mainly reconfiguration).

In a more specific context, it is worth analyzing how the layout problem was approached in these studies. Gao *et al.* [40] employed cartesian coordinates to determine the positioning of machines. Guo *et al.* [50] addressed the challenge of fixed assembly islands, where the product remains stationary, and only the resources required for production are mobile. Yang & Xu [146] introduced the concept of a system "mode," which involved predefined layout and production resource configurations that could be selected. Additionally, Yang & Xu [146] and Xia *et al.* [141] addressed real-time challenges in their studies, while Khan *et al.* [73] demonstrated that reconfigurability enhances system efficiency. In addressing the layout problem, there is a diversity of available approaches, as the requirements vary considerably depending on the type of process and the machinery and products involved. Therefore, selecting the most suitable approach for the particular study is crucial to avoid unnecessary complexity while ensuring that significant solutions are not overlooked.

	Solving Methods						**Objectives**						
Paper	**GA**	**OM**	**H**	**EM**	**CP**	**OS**	**TF**	**RC**	**MHC**	**OC**	**ENV**	**OO**	**Year**
[148]	x			x				x	x	x			2006
[120]				x				x	x	x			2015
[141]						x				x			2017
[114]						x	x			x		x	2017
[46]	x			x				x	x		x		2020
[56]		x	x				x	x				x	2020
[146]						x	x						2021
[40]	x			x			x						2021
[108]		x					x						2021
[50]						x	x						2021
[95]		x		x								x	2022
[73]			x				x					x	2022

GA: genetic algorithm; OM: other metaheuristics; H: heuristics; EM: exact methods; CP: constraint programming; OS: other solving methods; TF: time functions; RC: reconfiguration costs; MHC: material handling costs; OC: other costs; ENV: environment; OO: other objectives.

Table 8: Summary of RMS scheduling and layout articles.

1.3.6 Process planning and layout

The integration of process planning and layout is important due to the influence of layout configuration on the sequencing of production operations of each job. In this section, the number of articles is notably smaller than in previous sections (about 8%), with solving methods and objectives closely aligning with those of the scheduling and layout section.

Objective functions

Regarding the problems' objectives, certain works did not incorporate cost-related considerations into their analyses. Rehman [117] and Rehman & Subash Babu [118, 119] assessed simulation performance based on machine utilization, throughput time, product earliness, product lateness, and product block time. Haddou Benderbal & Benyoucef [52] aimed to maximize the average machine utilization, increase the number of machines available for replacement when needed, minimize layout evolution effort during production changeovers within product families, and reduce the penalty associated with machine-location association constraints in the system.

On the contrary, other researchers incorporated costs into their objectives. Xiaowen *et al.* [142] focused on minimizing reconfiguration costs and times. Eguia *et al.* [34] sought to minimize the non-use of machines within cells, movement between cells, and inventory holding costs. Campos Sabioni *et al.* [18, 19] minimized total costs, which included raw material costs, operation costs, layout reconfiguration, machine reconfiguration, and material handling costs.

Solving methods

As expected, metaheuristics were also applied to address these types of problems. Xiaowen *et al.* [142] investigated a multi-stage production line, constructing their model based on graph theory and utilizing a GA for solving. Haddou Benderbal & Benyoucef [52] developed a two-step method, initially proposing a layout design with an archived multi-objective simulated annealing approach based on a generated process plan, followed by an exhaustive search for optimizing non-satisfaction constraints. Campos Sabioni *et al.* conducted two studies in this domain, both aimed at selecting the optimal modules to create a product meeting customer requirements. In the first study [18], they developed a genetic algorithm, while in the second study [19], they introduced two hybrid approaches: the Modified Brute-Force Algorithm with integer linear programming and the Modified Brute-Force Algorithm with a genetic algorithm.

In contrast, some papers did not use metaheuristic methods. Rehman & Subash

Babu pioneered exploring PP and layout problems by examining a factory with nine distinct system configurations. They devised multiple scenarios, each characterized by unique input data combinations, to facilitate the comparison of simulation results across all configurations. The distinguishing factor between these articles lies in their methods for ranking these configurations. While Rehman & Subash Babu [118] relied on a benchmarking model and statistical analysis, Rehman & Subash Babu [119] used a Multiple Criteria Decision-Making methodology, specifically the Elimination and Choice-Translating Algorithm. A decade later, Rehman [117] introduced further research that shared similarities with the previous studies, particularly in terms of experimental data. Once more, the focus was on simulating a production system. The distinguishing feature of this article was its emphasis on robustness, employing the signal-to-noise ratio method to select the ultimate optimal solution. From another perspective, Eguia *et al.* [34] proposed an exact method to address the cell formation and multi-period machine loading problem within an industry exclusively equipped with CNC and RMT machines.

Conclusions

Table 9 provides a summary of the articles jointly addressing process planning and layout. Overall, metaheuristics remain a prevalent approach in this context. Unlike scheduling problems, which frequently employ heuristics based on sequencing rules, heuristics were not used here. The primary objectives centered around classical criteria associated with time and cost functions.

	Solving Methods						Objectives						
Paper	**GA**	**OM**	**H**	**EM**	**CP**	**OO**	**TF**	**RC**	**MHC**	**OC**	**ENV**	**OS**	**Year**
[118]						x	x					x	2008
[119]						x	x					x	2009
[142]	x						x	x					2013
[34]				x					x	x		x	2017
[117]						x	x					x	2019
[52]		x										x	2019
[19]	x			x				x	x	x			2020
[18]	x							x	x	x			2021

GA: genetic algorithm; OM: other metaheuristics; H: heuristics; EM: exact methods; CP: constraint programming; OS: other solving methods; TF: time functions; RC: reconfiguration costs; MHC: material handling costs; OC: other costs; ENV: environment; OO: other objectives.

Table 9: Summary of RMS process planning and layout articles.

1.3.7 Scheduling, process planning, and layout

In accordance with our initial definitions and research criteria, only two articles fit into this section. This is understandable, as integrating multiple problems makes

the overall study more complex. Integrating two problems already presents significant challenges, but integrating all three further increases the difficulty.

The first, though not recent, dates back to 2007. [152] employed a multi-agent architecture to model Dynamically Integrated Manufacturing Systems. This framework introduced an agent coordination algorithm designed to facilitate interaction among system agents for optimizing manufacturing system planning, control, and reconfigurations. Before making any changes to production planning and scheduling, the methodology involved identifying, simulating, and evaluating several proposed options. Coordination was achieved using a GA to optimize parameters within a bid distribution system for local units, leading to a collective resource plan. The algorithm allowed for the relaxation of constraints when satisfactory solutions could not be found within the existing restrictions, potentially resulting in a system reconfiguration. The work also addressed cell formation by considering material handling and buffers, which is why it is categorized as a layout problem. However, despite spanning all three classifications, the integration and solving methods employed here differ from those proposed in our research.

More recently, Gao *et al.* [41] introduced a model that integrates three objective criteria: minimizing tardiness, total costs, and reducing hazardous waste and greenhouse gas emissions. This study presented a small example involving two products and two machines. Initially, they employed a Brute-Force Search method and later compared it with the NSGA-III algorithm.

	Solving Methods						**Objectives**						
Paper	**GA**	**OM**	**H**	**EM**	**CP**	**OS**	**TF**	**RC**	**MHC**	**OC**	**ENV**	**OO**	**Year**
[152]						x	x	x	x	x			2007
[41]	x			x			x	x	x	x	x		2021
This work	x			x	x		x	x	x	x			2024

GA: genetic algorithm; OM: other metaheuristics; H: heuristics; EM: exact methods; CP: constraint programming; OS: other solving methods; TF: time functions; RC: reconfiguration costs; MHC: material handling costs; OC: other costs; ENV: environment; OO: other objectives.

Table 10: Summary of RMS scheduling, process planning, and layout articles.

1.4 Literature gaps

After reviewing the literature on optimization problems within Reconfigurable Manufacturing Systems (RMS), several key observations and gaps were identified. The main observations are:

- Increased interest in RMS optimization problems due to market demands for greater flexibility in production.

- The majority of the reviewed research work focuses on addressing individual problems or exclusively integrating two of the three: scheduling, process planning, and layout.
- In the context of RMS, the scheduling problem is the most mature, while the layout problem is the least studied.
- Metaheuristics are the most used solving methods.
- Objective functions mainly focus on time-related metrics (such as makespan and tardiness) and total costs (including production and machine reconfiguration).

The main identified gaps are as follows:

- Limited attention has been given to analyzing material handling costs and times in the context of RMS optimization, particularly outside the layout category. However, this aspect should not be neglected, as it constitutes a significant portion of production costs.
- Despite RMS's distinctive feature of reconfigurable layouts, optimization problems related to layout remain less explored compared to scheduling and process planning.
- In-depth exploration of exact methods in the literature is lacking.
- Integrated studies involving these problems have yet to present examples at an industrial scale or based on real-world scenarios.
- No existing integrated work compares a sequential approach to an integrated methodology.

To address these literature gaps, this work compares integrated and sequential models to minimize total costs, encompassing production, machine reconfiguration, layout reconfiguration, material handling, and tardiness. Additionally, we propose three exact methods and two evolutionary algorithms to tackle problems of varying scale, from small-size instances to industrial applications.

1.5 Conclusions of the chapter

This research work tries to answer the following question: How can we best optimize process planning, scheduling, and layout configuration in an RMS for mass-customized products while reducing manufacturing costs? From the literature review, it is quite clear that integrating at least two of the three optimization problems (process planning, scheduling, and layout problems) could be more effective than solving them individually. Although most researchers have traditionally addressed these issues individually, there has been a growing trend towards integrating at least pairs of these problems. This trend is

driven by the shared decision variables, which can lead to enhanced global results when considered jointly. Figure 6 illustrates that over 67% of reviewed articles deal with these problems separately, about 30% work with two problems, and only around 2% address all three simultaneously.

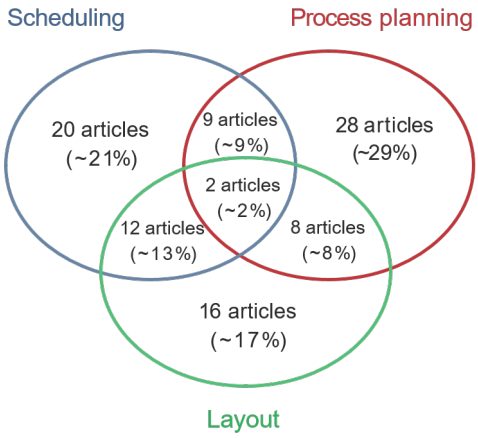

Figure 6: Category of the reviewed articles.

Integrating process planning and scheduling has been a well-established practice for non-RMS problems long before these systems emerged [90, 129], but doing so in the RMS context introduces new challenges due to reconfigurability. Moreover, RMS allows layout reconfiguration to be done by moving machines, making layout integration also crucial. All three problems share resource allocation decisions, as illustrated in Figure 7. Production and machine reconfiguration costs and times directly affect operation sequencing in process planning and scheduling. Layout reconfiguration and material handling times are most relevant to layout and scheduling, while process planning and layout primarily consider material handling costs and layout reconfiguration. Tardiness penalties are relevant across all problems.

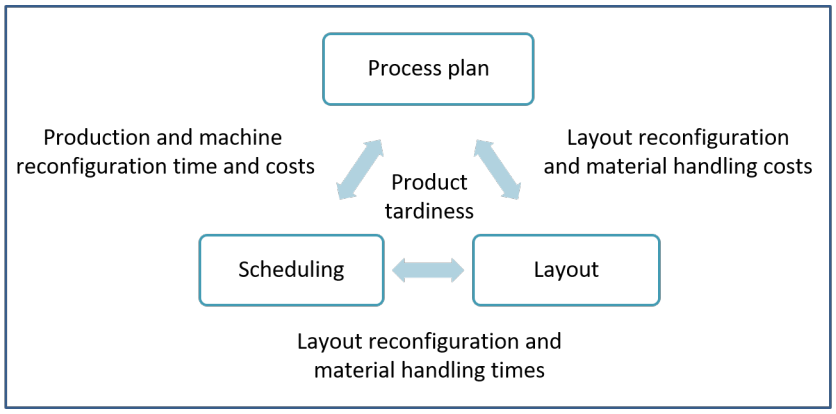

Figure 7: Connection between the process planning, scheduling, and layout design problems.

Therefore, this thesis aims to study the significance and implications of integrating

the discussed problems. Hence, several models and solving methods will be presented. The proposed mathematical models go from independent problem-solving through partial integration to finally full integration. Our solving methods were developed with complete adaptation to our proposed problem, incorporating the most effective techniques identified in the literature reviewed.

Chapter 2

Problem Description and Mathematical Models

Contents

This research proposes a way to best optimize process planning, scheduling, and layout configuration in an RMS for mass-customized products while reducing manufacturing costs. The first intuition is integrating these problems since they are interconnected and interdependent. However, it is still to be proven whether their integration will be the best way to handle them or whether integrating two of the three problems would be better. This is why we developed four new mathematical models. The first model fully integrates the three optimization problems. On the contrary, the second handles the problems individually, following a sequential approach. The sequence states the order of solving the problems and means that the outputs of a problem become the input of the subsequent one. The two final models propose a partial integration (two out of three problems are integrated) and keep a sequential approach.

These three problems, optimizing process planning, layout, and scheduling within an RMS, have been extensively studied. Unlike most authors in this field, this work studies these problems more completely, encompassing factors that are often overlooked, such as the impact of layout reconfiguration and material handling. Hence, all the proposed mathematical models aim to minimize overall costs, including production, material handling, layout reconfiguration, machine reconfiguration, and tardiness costs.

This chapter begins by describing the problem under investigation. Then, it presents the four proposed mathematical models, followed by a conclusion.

2.1 Problem description

Scheduling operations is crucial for all types of companies, including industries, commerce, and services. In this work, our focus is on industries, particularly those using RMS. Beyond resource allocation and scheduling, it is essential to address other aspects when planning production in these systems. Determining when and how systems should be reconfigured at machine and system levels becomes equally important. Using the process planning and layout flexibility that RMS provide can significantly increase results. This section describes the problem we address, and Table 11 lists all the parameters introduced here.

Customer orders are considered as jobs $J_i \in \{J_1, ..., J_n\}$, translating required products into production orders. When a customer demands two identical products, two orders will be generated, even though the same person/entity requires the same products. Each job J_i has a due date d_i before which the job is expected to be fully processed. Where a job is not completed before its due date, a job penalty cost W_i is incurred per time unit of tardiness. The tardiness T_i of job i is equal to $max\{0, C_i - d_i\}$, where C_i is the completion time of J_i. A job J_i is composed of n_i operations $J_i = \{O_{i1}, ..., O_{i,n_i}\}$. They are specific manufacturing processes (e.g., cutting, welding, assembly, or packaging), and the set of operations necessary for a job must respect a known partial order of precedence. For example, to manufacture an automobile gearbox housing (see Figure 8), some of the operations include rough milling, drilling, reaming, semi-finish milling, finish milling, tapping, semi-finish boring, and finish boring [138].

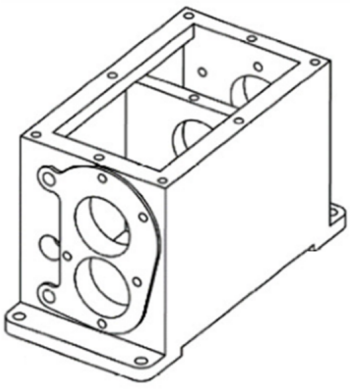

Figure 8: Automobile gearbox housing [138].

The precedence relation between operations of J_i is represented as $O_{ij} \lhd_i O_{ij'}$ where operation O_{ij} precedes $O_{ij'}$, i.e. operation $O_{ij'}$ can start only if O_{ij} has been fully processed.

Parameters

$J = \{J_1, ..., J_n\}$	the set of jobs
$O_i = \{O_{i1}, ..., O_{i,n_i}\}$	the set of operations of J_i
$M = \{M_1, ..., M_m\}$	the set of machines
$M_k = \{M_k[1], ..., M_k[q_k]\}$	the set of configurations of machine M_k
$\Lambda = \{\Lambda_1, ..., \Lambda_l\}$	the set of layouts
d_i	due date of job i
π_p	job with the due date in p^{th} position among the due dates in ascending order
W_i	penalty cost per unit of tardiness for J_i
T	time horizon, jobs are scheduled from 0 to T
$\lhd_i$	precedence relation over the set of operations of J_i, $O_{ij} \lhd_i O_{ij'}$ if O_{ij} has to be processed before $O_{ij'}$
$P_{ij}(k, e)$	O_{ij} processing time on machine M_k in configuration e
$P_k^{\chi}(e, e')$	machine reconfiguration time of M_k to change from configuration e to e'
$P^{\lambda}(u, u')$	layout reconfiguration time to change from layout u to u'
$P_u^{\tau}(k, k')$	material handling time associated with a layout u when an operation of J_i is processed on machine k and its next operation on machine k'
$K_{ij}(k, e)$	O_{ij} production cost to process O_{ij} with machine M_k in configuration e
$K_k^{\chi}(e, e')$	machine reconfiguration cost of M_k to change from configuration e to e'
$K^{\lambda}(u, u')$	layout reconfiguration cost to change from layout u to u'
K^{τ}	material handling cost per unit of time to move a J_i from a machine to another

Decision variables

$S_{ij} \in \{0, ..., T\}$	starting time of O_{ij}
$C_{ij} \in \{0, ..., T\}$	completion time of O_{ij}
$S_i = S_{i1}$, $C_i = C_{in_i}$	starting and completion time of J_i
$\sigma_{ij} \in \{1, ..., n_i\}$	position of operation j in the process sequence among all operations of J_i
$\sigma_i[p] \in \{1, ..., n_i\}$	operation executed in p^{th} position among all operations of J_i
$\mu_{ij} \in \{1, ..., m\}$	machine processing O_{ij}
$\lambda(t) \in \{1, ..., l\}$	layout used at time t
$\chi(k, t) \in \{1, ..., q_k\}$	configuration of machine k at time t

Table 11: All parameters and decision variables used in this chapter.

For the gearbox housing, we know, for example, that the semi-finish boring operation has to be done before (precede) the finish boring operation. Moreover, it is important to emphasize that here multiple operations from the same job must be executed sequentially;

in other words, concurrent processing of two or more operations, even without precedence constraints, is not allowed.

Figure 9 gives an example involving two jobs (J_1 and J_2) and a Hasse Diagram representing their precedence relation. For job J_1, O_{12} and O_{13} can only be performed after O_{11}. To start O_{14}, O_{12} and O_{13} must already be completed. Finally, all the other operations must have been completed to start O_{15}. In job J_2, manufacturing start with O_{21}, then can go to O_{22} or O_{23}. O_{24} can only be processed after O_{22} and O_{23} are finished. The penultimate and last operations can be either O_{25} or O_{26}.

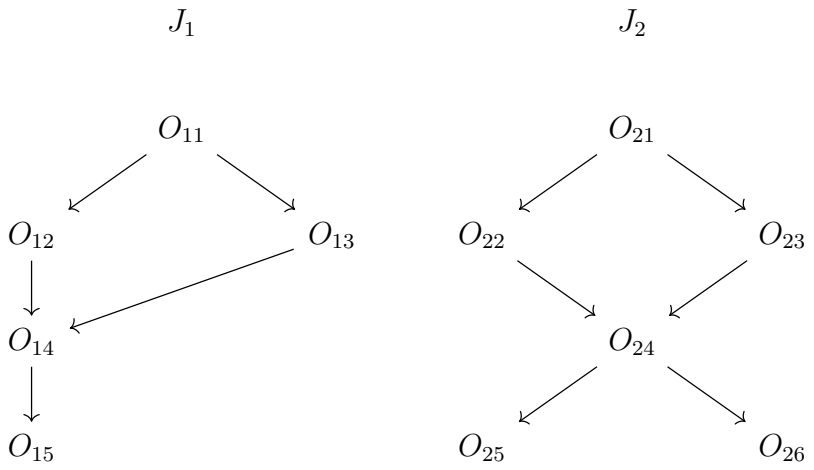

Figure 9: Production precedence example.

The potential production sequences are outlined in Table 12, with J_1 presenting two options and J_2 presenting four. Specifically, J_1 invariably starts with operation O_{11}, while J_2 starts with operation O_{21}.

	Possible production sequences
Job 1	$O_{11} - O_{12} - O_{13} - O_{14} - O_{15}$
	$O_{11} - O_{13} - O_{12} - O_{14} - O_{15}$
Job 2	$O_{21} - O_{22} - O_{23} - O_{24} - O_{25} - O_{26}$
	$O_{21} - O_{22} - O_{23} - O_{24} - O_{26} - O_{25}$
	$O_{21} - O_{23} - O_{22} - O_{24} - O_{25} - O_{26}$
	$O_{21} - O_{23} - O_{22} - O_{24} - O_{26} - O_{25}$

Table 12: Possible production sequences for the example illustrated in Figure 9.

For the gearbox housing, two possible operation production sequences are: (1) rough milling → drilling → reaming → semi-finish milling → finish milling → tapping → semi-finish boring → finish boring, or (2) rough milling → drilling → reaming → tapping → semi-finish milling → semi-finish boring → finish milling → finish boring.

The selection of a sequence in the process planning problem can differ for identical jobs, as it depends on the available resources and layout system when each job is produced. This illustrates how process planning decisions are influenced by scheduling and layout problems.

The manufacturing system has m machines $M_k(k = 1, ..., m)$ to process operations. Each machine M_k has q_k possible configurations $M_k[e](e = 1, ..., q_k)$. A machine configuration is the combination of software and hardware modules. Changing a machine configuration requires a certain amount of time and incurs reconfiguration costs. For example, the same machine can perform drilling and reaming operations, but each operation requires a different tool, involving a hardware change. Another example of reconfiguration is modifying the program (software) of a CNC machine. Reconfiguration can involve changes in both hardware and software. The duration time $P_k^{\chi}(e, e')$ and cost $K_k^{\chi}(e, e')$ needed to reconfigure the machines depends on the machine M_k in question and on the initial (e) and final (e') configurations. When a machine reconfiguration is necessary, the machine remains unavailable while reconfiguration is occurring. The same reasoning applies to costs.

Each operation O_{ij} has at least one machine/configuration combination $M_k[e]$ capable of manufacturing it. Each operation O_{ij} has an associated processing time $P_{ij}(k, e)$ and a cost $K_{ij}(k, e)$ if that operation is processed by machine k in configuration e. We have $P_{ij}(k, e) = \infty$ when a machine k in configuration e cannot produce an operation O_{ij}. Otherwise, $P_{ij}(k, e)$ is the number of time units needed for performing it. The same applies to cost, each combination having an associated cost, and where an operation cannot be performed by a given combination the cost is represented as infinite.

As highlighted in the previous chapter, machines can be separated into dedicated machines, Computer Numerical Control (CNC) systems, and Reconfigurable Machine Tools (RMT) [36, 86]. Dedicated machines perform only one type of operation (one configuration, $q_k = 1$). CNC machines are highly flexible, automated devices capable of executing various operations by simply altering the program and tools. In contrast, RMT are specifically designed for particular operations or product families, offering scalability ideal for RMS applications. Our work encompasses all three machine types: dedicated, CNC, and RMT. While RMT are theoretically well-suited for RMS, practical industrial scenarios often involve a mix of machine types. Our models reflect this diversity and can be used regardless of the type of machines used. Additionally, machines in industrial contexts can be either fixed or mobile, with full mobility being characteristic of only a few specific micro-factories. The proposed model takes into consideration fixed and mobile machines.

Concerning the machines, their allocation includes the machine's physical space, safety margins, and a buffer area. In the example illustrated in Figure 10, machine one is fixed, while the others are mobile. The mobile equipment must move avoiding the red zones.

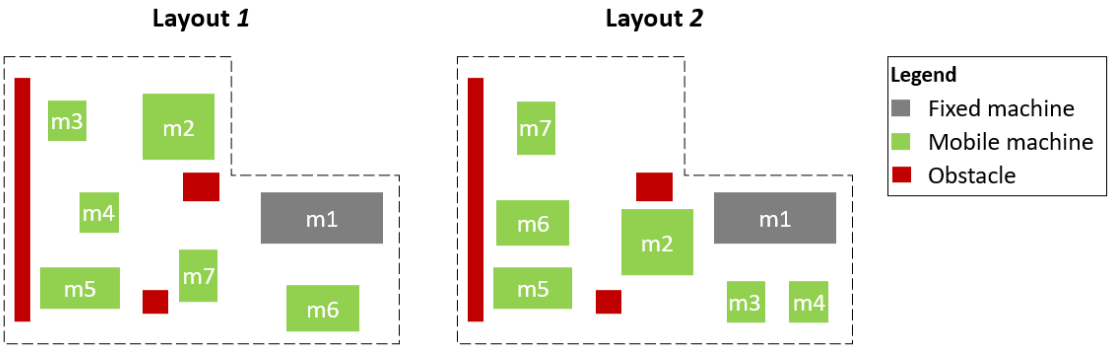

Figure 10: Layout configuration example.

Regarding physical machine location, the problem considers a set of l possible layouts $\Lambda = \{\Lambda_1, ..., \Lambda_l\}$. The displacement of mobile machines happens when an optimized system is detected in another layout. A layout reconfiguration from configuration u to configuration u' has a price $K^{\lambda}(u, u')$ and an execution time $P^{\lambda}(u, u')$. There can be no changes within the same layout, so $\forall u,\ P^{\lambda}(u, u) = 0$. The other values are equal to the reconfiguration time. The same applies to layout reconfiguration costs.

The decision to work with a predetermined set of possible layouts from practical considerations, as real-world constraints imply that machinery cannot be placed arbitrarily. Factors like electrical infrastructure, environmental conditions (e.g., humidity and temperature), inter-machine vibrations, and other considerations influence layout choices. Additionally, the physical dimensions of the industrial space and the machines themselves impose limitations on equipment displacement. Alternative approaches like the Quadratic Assignment Problem or Cartesian coordinates in open space would unnecessarily increase the complexity.

Each layout configuration u has corresponding material handling times $P_u^{\tau}(k, k')$ for displacement from machine k to machine k'. The material handling times are presented in a matrix. The leading diagonal equals zero for all layouts because there is no travel time between a machine and itself. All material handling times between two different machines are greater than zero.

Also, each layout configuration has corresponding material handling costs for displacement from machine k to machine k'. We multiply the MH times $P_u^{\tau}(k, k')$ by K^{τ}, a material handling cost parameter (constant) to obtain these costs. Longer usage durations

result in increased costs. In our analysis, we have utilized a fixed value represented as K^{τ}, but it could also be treated as a variable function depending on specific scenarios.

In summary, the expected outputs are :

- The starting times S_{ij} and completion times C_{ij} of operations, consequently, the S_i and C_i for each job;
- The manufacturing sequence σ_{ij} of operations within a job J_i;
- The machine μ_{ij} processing each operation;
- The configuration $\chi(k,t)$ of each machine k at each time;
- The layout $\lambda(t)$ used at each time t.

The problem is subject to the following assumptions:

- The demand is known;
- The due dates of the jobs are known;
- Raw materials, workforce, and material handling (transport of products between machines) are always available;
- All jobs produced have the required quality.

The main objective is to minimize the total costs, which include production costs, machine reconfiguration costs, layout reconfiguration costs, material handling costs, and tardiness costs.

The production cost comprises the sum of processing costs for each operation. Machine reconfiguration costs arise whenever a machine's configuration changes. Similarly, layout reconfiguration costs are incurred when machines are rearranged. Material handling costs encompass the expenses linked to the movement of operations between machines. Tardiness costs, on the other hand, are only accrued in cases where a job surpasses its due date.

2.2 Modeling approaches

This work aims to tackle three interconnected optimization problems: process planning, layout design, and scheduling. The primary hypothesis is that fully integrating these problems will yield superior results. However, this assertion remains unproven and needs to be compared with non-integrated or partially integrated approaches.

Therefore, we propose a total of four models: one fully integrated, one sequential,

and two partially sequential. The first model directly integrates all three problems. The second model addresses each problem individually with a sequential approach, meaning the output of a problem becomes the input of the following problem. Addressing each problem individually is not feasible without making certain assumptions due to their interdependence and mutual influence.

The third model adopts a partially sequential approach, addressing one individual problem (process planning) and two integrated problems (layout and scheduling). The final model follows a similar partially sequential approach, addressing two integrated problems (process planning and layout) and one individual problem (scheduling).

Effectively addressing the sequential model relies heavily on determining the sequence in which the problems are solved. Therefore, when modeling these three problems, it is crucial to identify the optimal sequence for solving them. The sequential modeling approach efficiency is inherently tied to the resolution sequence. Typically, scheduling is deferred to the final stage as it provides essential timing information (i.e., when each action will be taken), encompassing production operations, reconfigurations, and material handling. The order in which process planning and layout are sequenced can vary, depending on the preferences of the decision-maker.

Starting with layout can be challenging due to a lack of prior information, such as which machine will handle each operation. This approach could be more feasible in scenarios with robust historical production data. However, in our case, where this is not an initial assumption, we opt to commence with process planning. This allows us to have machine decisions as input for the layout problem. Nonetheless, as the layout has not been established when solving process planning, we can use the initial layout or an alternative method, such as calculating a weighted average of distances across all potential layouts.

Hence, our chosen approach starts with Process planning (PP), followed by Layout (L), and concludes with Scheduling (S). With the sequence established, the options outlined in Table 13 can be explored.

Methods	**Description**
PP + L + S	The three problems are handled together.
PP $\rightarrow$ L $\rightarrow$ S	Each problem is handled individually and sequentially.
PP $\rightarrow$ L + S	The process planning problem is handled individually, while the layout and scheduling problems are handled together.
PP + L $\rightarrow$ S	The process planning and layout problems are handled together, and then the scheduling problem is handled individually.

Table 13: Methods applied in this work.

The subsequent sections will describe the mathematical model for each approach.

2.3 Integrated model

The integrated model addresses all problems simultaneously. The parameters and decision variables utilized in this model are listed in Table 11. The objective function (f_c) minimizes the total costs: production (f_p), material handling (f_{MH}), layout reconfiguration (f_l), machine reconfiguration (f_m), and tardiness (f_t) costs (2.1).

$$min\ f_c = f_p + f_{MH} + f_l + f_m + f_t \tag{2.1}$$

The production cost depends on the machine and the configuration in which the operation will be processed (2.2).

$$f_p = \sum_{i=1}^{n} \sum_{j=1}^{n_i} K_{ij}(\mu_{ij}, \chi_{ij}(\mu_{ij}, S_{ij})) \tag{2.2}$$

The machine reconfiguration costs are counted when the machine changes its configuration between two units of time (2.3).

$$f_m = \sum_{t=0}^{T-1} \sum_{k=1}^{m} K_k^{\chi}(\chi(k,t), \chi(k,t+1)) \tag{2.3}$$

The layout reconfiguration costs are counted every time there is a layout change between two units of time (2.4).

$$f_l = \sum_{t=0}^{T-1} K^{\lambda}(\lambda(t), \lambda(t+1)) \tag{2.4}$$

The material handling costs depend on the displacement time of jobs between machines multiplied by a weight per unit of handling time (2.5).

$$f_{MH} = \sum_{i=1}^{n} \sum_{p=1}^{n_i-1} K^{\tau} \cdot P^{\tau}_{\lambda(S_{i\sigma_i[p]})}(\mu_{i\sigma_i[p]}, \mu_{i\sigma_i[p+1]}) \tag{2.5}$$

Tardiness costs are applied when the completion time is beyond the due date (2.6).

$$f_t = \sum_{i=1}^{n} W_i \cdot max(0, C_i - d_i) \tag{2.6}$$

The problem is subject to the following constraints. No operation is allowed during layout reconfiguration, and reconfiguration is only possible if all jobs initiated before the reconfiguration have been completed.

$$\forall t \quad \lambda(t) \neq \lambda(t+1) \quad \Rightarrow \quad \forall i \quad (C_i \leq t) \vee (S_i \geq t + P^{\lambda}(\lambda(t), \lambda(t+1))) \tag{2.7}$$

The machine-level reconfiguration can only be executed if no operation is performed, and the machine under consideration remains unavailable during the configuration change (2.8).

$$\forall t \quad \chi(\mu_{ij}, t) \neq \chi(\mu_{ij}, t+1) \Rightarrow (C_{ij} \leq t) \vee (S_{ij} \geq t + P^{\chi}_{\mu_{ij}}(\chi(\mu_{ij}, t), \chi(\mu_{ij}, t+1))) \tag{2.8}$$

Once an operation is started, it cannot be stopped and restarted later. Consequently, the operation's completion time is determined by adding its starting time and its production time, which depends on the machine and configuration where it will be processed (2.9).

$$C_{ij} = S_{ij} + P_{ij}(\mu_{ij}, \chi(\mu_{ij}, S_{ij})) \tag{2.9}$$

An operation can only start after its predecessor in the process sequence has ended and the material handling time respected (2.10).

$$\sigma_{ij} + 1 = \sigma_{ij'} \quad \Rightarrow \quad C_{ij} + P^{\tau}_{\lambda(C_{ij})}(\mu_{ij}, \mu_{ij'}) \leq S_{ij'} \tag{2.10}$$

Each machine can process only one operation at a time (2.11).

$$\mu_{ij} = \mu_{i'j'} \quad \Rightarrow \quad (C_{ij} \leq S_{i'j'}) \vee (C_{i'j'} \leq S_{ij}) \tag{2.11}$$

Constraints 2.12 and 2.13 are used to ensure model consistency. The first constraint ensures the connection between operations and their execution sequence (position) for each job, while the second constraint ensures that the precedence order is respected.

$$\sigma_{ij} = p \quad \Leftrightarrow \quad \sigma_i[p] = j \tag{2.12}$$

$$O_{ij} \lhd_i O_{ij'} \quad \Rightarrow \quad \sigma_{ij} < \sigma_{ij'} \tag{2.13}$$

2.4 Sequential model : PP → L → S

The problems are interconnected through shared variables, so we used a sequential model. It is important to highlight that the input data used for the sequential approaches (sequential model and the two partially integrated models) may vary from that of the integrated model discussed in the previous section.

As explained previously, our sequential model handles each problem individually but connects them by using the outputs of the first as the input to the second and the outputs of the first and second problems as inputs to the third one. Hence, three sub-models are proposed as presented in Figure 11: process planning, layout, and scheduling. The modeling proposed depends on the resolution sequence chosen and discussed above. This resolution is based on a proposed heuristic approach with a chosen sequence. The first sub-model focuses on process planning. Each job is individually analyzed to determine the machines and configurations for manufacturing each operation. This information serves as input parameters for the subsequent sub-model, the layout. Here, jobs are evaluated together. The layout sub-model selects the layout for processing each job and establishing the execution sequence of operations. Finally, the outputs from the first two sub-models are used to generate the final operation scheduling in the scheduling sub-model.

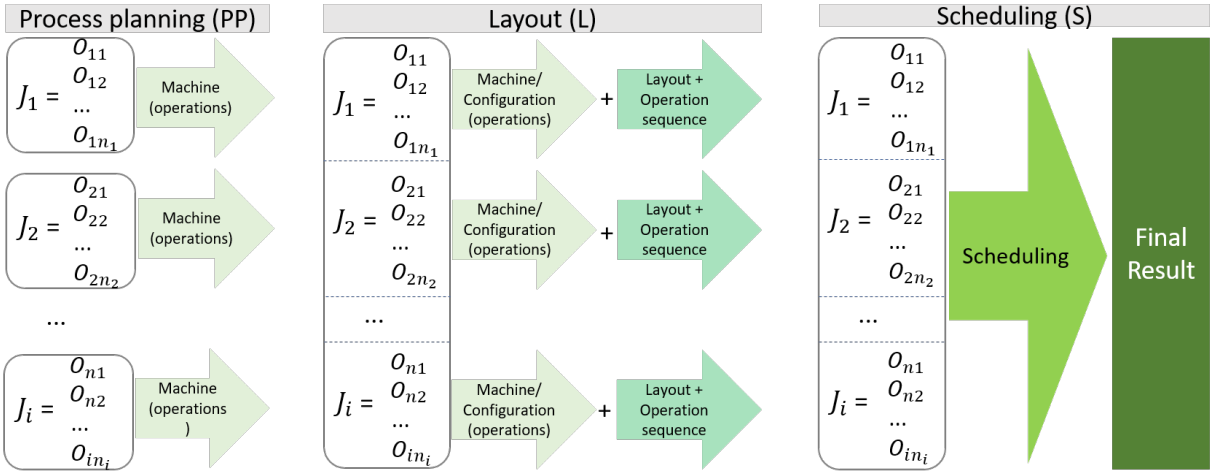

Figure 11: Sequential model.

Sub-model 1: Process Planning (PP)

In this sub-model, each job is processed individually. The variables and decision parameters utilized here are detailed in Table 14, also available in Table 11.

Parameters	
$J = \{J_1, ..., J_n\}$	the set of jobs
$O_i = \{O_{i1}, ..., O_{i,n_i}\}$	the set of operations of J_i
$M = \{M_1, ..., M_m\}$	the set of machines
$M_k = \{M_k[1], ..., M_k[q_k]\}$	the set of configurations of machine M_k
$\Lambda = \{\Lambda_1, ..., \Lambda_l\}$	the set of layouts
d_i	due date of job i
W_i	penalty cost per unit of tardiness for J_i
$\lhd_i$	precedence relation over the set of operations of J_i, $O_{ij} \lhd_i O_{ij'}$ if O_{ij} has to be processed before $O_{ij'}$
$P_{ij}(k, e)$	O_{ij} processing time on machine M_k in configuration e
$P^{\tau}_u(k, k')$	material handling time associated with a layout u when an operation of J_i is processed on machine k and its next operation on machine k'
$K_{ij}(k, e)$	O_{ij} production cost to process O_{ij} with machine M_k in configuration e
K^{τ}	material handling cost per unit of time to move a J_i from a machine to another
Decision variables	
$S_{ij} \in \{0, ..., T\}$	starting time of O_{ij}
$C_{ij} \in \{0, ..., T\}$	completion time of O_{ij}
$\sigma_{ij} \in \{1, ..., n_i\}$	position of operation j in the process sequence among all operations of J_i
$\sigma_i[p] \in \{1, ..., n_i\}$	operation executed in p^{th} position among all operations of J_i
$\mu_{ij} \in \{1, ..., m\}$	machine processing O_{ij}
$\chi(k, t) \in \{1, ..., q_k\}$	configuration of machine k at time t

Table 14: Parameters and decision variables used for the PP sub-model.

The objective function (f^{pp}_c) minimizes production (f^{pp}_p), material handling (f^{pp}_{MH}), and tardiness (f^{pp}_t) costs (2.14). The production time and cost data remain the same as the overall problem input. However, new Material Handling (MH) data is created. Since no layout decision is made in this sub-model, the MH time between two machines is considered the average time of all MH layouts ($\overline{\Lambda}$). For example, if a problem includes three layouts and the MH time between machines 1 and 2 for layout 1 is $P^{\tau}_1(1, 2) = 5$, for layout 2 is $P^{\tau}_2(1, 2) = 3$, and for layout 3 is $P^{\tau}_3(1, 2) = 10$, the MH time used here between

the same machine pair will be $P^{\tau}_{\overline{\Lambda}}(1,2) = 6$ (i.e., $\frac{5+3+10}{3}$).

$$f_c^{pp} = f_p^{pp} + f_{MH}^{pp} + f_t^{pp} \tag{2.14}$$

The due dates (d_i^{pp}) also undergo changes (2.15). They are determined as the smaller value between the original due date and the average of each job's worst and best production times. p_{ij}^{w} represents the worst production time of operation O_{ij} of job J_i and p_{ij}^{b} represents the best production time of operation O_{ij} of job J_i.To illustrate, if a job initially has a due date of 30, with a sum of worst production times equal to 25 and a sum of best production times equal to 19, the due date used for this job in this sub-model will be 22, calculated as $(min(30, \frac{25+19}{2}))$.

$$d_i^{pp} = min(d_i, \frac{\sum_{j=0}^{n_i} p_{ij}^{w} + \sum_{j=0}^{n_i} p_{ij}^{b}}{2}) \tag{2.15}$$

The production cost ($K_{ij}(k,e)$) depends on machine k and configuration e in which the operations are processed. Therefore, it is necessary to define the machine (μ_{ij}) responsible for manufacturing O_{ij} and the configuration ($\chi(k,t)$) of the machine μ_{ij} at time S_{ij} (2.16).

$$f_p^{pp} = \sum_{j=1}^{n_i} K_{ij}(\mu_{ij}, \chi_{ij}(\mu_{ij}, S_{ij})) \tag{2.16}$$

The material handling cost depends on the material handling cost per unit of time (K^{τ}), a constant parameter. This cost is then multiplied by the MH time ($P_u^{\tau}(k,k')$) between operations (2.17). To calculate the material handling time, it is necessary to consider the layout used by operations (which is the same for all jobs in this sub-model, denoted as $\overline{\Lambda}$), the machines responsible for producing operations, and the sequence of operations($\mu_{i\sigma_i[p]} \rightarrow \mu_{i\sigma_i[p+1]}$)) (2.17).

$$f_{MH}^{pp} = \sum_{p=1}^{n_i-1} K^{\tau} \cdot P^{\tau}_{\overline{\Lambda}}(\mu_{i\sigma_i[p]}, \mu_{i\sigma_i[p+1]}) \tag{2.17}$$

The tardiness cost is determined by comparing the job completion time (C_i) and the job's due date (calculated as explained earlier) (2.18). If C_i is greater than d_i, then their difference is multiplied by a penalty cost per unit of tardiness. Otherwise, it remains zero (2.18).

$$f_t^{pp} = W_i \cdot max(0, C_i - d_i^{pp}) \tag{2.18}$$

Some constraints restrict the problem. Firstly, operations are non-preemptive. Then, if the start time (S_{ij}), machine (μ_{ij}), and configuration ($\chi(k,t)$) have already been chosen,

it is possible to know the operation completion time (C_{ij}) (2.19).

$$C_{ij} = S_{ij} + P_{ij}(\mu_{ij}, \chi(\mu_{ij}, S_{ij})) \tag{2.19}$$

The production order of operations must respect the job precedence order. In cases where operation O_{ij} precedes an operation $O_{ij'}$ ($O_{ij} \lhd_i O_{ij'}$), the position of O_{ij} is lower than the position of $O_{ij'}$ ($\sigma_{ij} < \sigma_{ij'}$) (2.20).

$$O_{ij} \lhd_i O_{ij'} \quad \Rightarrow \quad \sigma_{ij} < \sigma_{ij'} \tag{2.20}$$

An operation can only start ($S_{ij'}$) after its previous one in the partial order $\lhd_i$ has ended (C_{ij}), and the material handling time ($P_u^{\tau}(k, k')$) between the machines that produce them (μ_{ij} and $\mu_{ij'}$) is respected (2.21).

$$\sigma_{ij} + 1 = \sigma_{ij'} \quad \Rightarrow \quad C_{ij} + P_{\Lambda}^{\tau}(\mu_{ij}, \mu_{ij'}) \leq S_{ij'} \tag{2.21}$$

Constraints 2.22 guarantee the link between operations and their execution sequence position.

$$\sigma_{ij} = p \quad \Leftrightarrow \quad \sigma_i[p] = j \tag{2.22}$$

Sub-model 2: Layout (L)

This sub-model focuses on the layout configuration to determine simultaneously the optimal layout for each job and define the sequence in which its operations will be executed. The output from the PP sub-model, specifically the machine responsible for each operation, serves as input for this sub-model. The variables and decision parameters utilized here are detailed in Table 15.

Parameters	
$J = \{J_1, ..., J_n\}$	the set of jobs
$O_i = \{O_{i1}, ..., O_{i,n_i}\}$	the set of operations of J_i
$M = \{M_1, ..., M_m\}$	the set of machines
$M_k = \{M_k[1], ..., M_k[q_k]\}$	the set of configurations of machine M_k
$\Lambda = \{\Lambda_1, ..., \Lambda_l\}$	the set of layouts
π_p	job with the due date in p^{th} position among the due dates in ascending order
W_i	penalty cost per unit of tardiness for J_i
$\lhd_i$	precedence relation over the set of operations of J_i, $O_{ij} \lhd_i O_{ij'}$ if O_{ij} has to be processed before $O_{ij'}$
$P^{\lambda}(u, u')$	layout reconfiguration time to change from layout u to u'
$P^{\tau}_u(k, k')$	material handling time associated with a layout u when an operation of J_i is processed on machine k and its next operation on machine k'
$K^{\lambda}(u, u')$	layout reconfiguration cost to change from layout u to u'
K^{τ}	material handling cost per unit of time to move a J_i from a machine to another
Decision variables	
$\sigma_{ij} \in \{1, ..., n_i\}$	position of operation j in the process sequence among all operations of J_i
$\sigma_i[p] \in \{1, ..., n_i\}$	operation executed in p^{th} position among all operations of J_i
$\mu_{ij} \in \{1, ..., m\}$	machine processing O_{ij}
$\lambda(t) \in \{1, ..., l\}$	layout used at time t

Table 15: Parameters and decision variables used for the L sub-model.

The objective function (f^l_c) minimizes costs, including layout reconfiguration (f^l_l), material handling (f^l_{MH}), and duration (f^l_d) costs (2.23). At this point, production times are already known due to the prior determination of machines and configurations.

$$f^l_c = f^l_l + f^l_{MH} + f^l_d \tag{2.23}$$

The layout reconfiguration costs ($K^{\lambda}(u, u')$) depend on the layout sequence and

reconfiguration between them. In this phase, the jobs' due dates determine the position sequence (π_p). Furthermore, $\lambda(\pi_0)$ is always the initial system layout, and $\lambda(\pi_p)$ is the layout of the job with the p^{th} due date in the ascending order (2.24). Considering two jobs, J_1 with a due date of 30 and J_2 with a due date of 20, the layout sequence order appears as follows: first the initial layout ($\lambda(\pi_0) =$ initial layout), followed by the layout corresponding to J_2 ($\lambda(\pi_1) = \lambda(J_2)$), and finally, the layout corresponding to J_1 ($\lambda(\pi_2) = \lambda(J_1)$.

$$f_l^l = \sum_{p=0}^{n} K^{\lambda}(\lambda(\pi_p), \lambda(\pi_{p+1})) \tag{2.24}$$

The material handling costs here are similar to the function used in PP, with the key difference being that the layout parameter (λ) is not constant ($\overline{\Lambda}$) but dependent on the job i ($\lambda(i)$) (2.25).

$$f_{MH}^l = \sum_{i=1}^{n} \sum_{p=1}^{n_i-1} K^{\tau} \cdot P_{\lambda(i)}^{\tau}(\mu_{i\sigma_i[p]}, \mu_{i\sigma_i[p+1]}) \tag{2.25}$$

The duration costs are the sum of the layout reconfiguration ($P^{\lambda}(u, u')$) and material handling ($P_u^{\tau}(k, k')$) times multiplied by a penalty cost (W_i) (2.26).

$$f_d^l = W_i \cdot \left[\sum_{p=0}^{p-1} P^{\lambda}(\lambda(\pi_p), \lambda(\pi_p + 1)) + \sum_{i=1}^{n} \sum_{p=1}^{n_i-1} P_{\lambda(i)}^{\tau}(\mu_{i\sigma_i[p]}, \mu_{i\sigma_i[p+1]}) \right] \tag{2.26}$$

Only two constraint equations, which were previously employed in the PP sub-model, are utilized here. Equation 2.27 indicates that precedence order must be respected, and Equation 2.28 ensures the coherence between variables σ_{ij} and $\sigma_i[p]$.

$$O_{ij} \lhd_i O_{ij'} \quad \Rightarrow \quad \sigma_{ij} < \sigma_{ij'} \tag{2.27}$$

$$\sigma_{ij} = p \quad \Leftrightarrow \quad \sigma_i[p] = j \tag{2.28}$$

Sub-model 3: Scheduling (S)

After obtaining the outputs from the preceding sub-models, the only information missing is when each operation starts and finishes. These decisions must respect the machine and layout reconfiguration times, as well as production capacity constraints. The variables and decision parameters utilized here are detailed in Table 16.

Parameters

$J = \{J_1, ..., J_n\}$	the set of jobs
$O_i = \{O_{i1}, ..., O_{i,n_i}\}$	the set of operations of J_i
$M = \{M_1, ..., M_m\}$	the set of machines
$M_k = \{M_k[1], ..., M_k[q_k]\}$	the set of configurations of machine M_k
$\Lambda = \{\Lambda_1, ..., \Lambda_l\}$	the set of layouts
d_i	due date of job i
W_i	penalty cost per unit of tardiness for J_i
$P_k^{\chi}(e, e')$	machine reconfiguration time of M_k to change from configuration e to e'
$P^{\lambda}(u, u')$	layout reconfiguration time to change from layout u to u'
$K_k^{\chi}(e, e')$	machine reconfiguration cost of M_k to change from configuration e to e'

Decision variables

$S_{ij} \in \{0, ..., T\}$	starting time of O_{ij}
$C_{ij} \in \{0, ..., T\}$	completion time of O_{ij}
$S_i = S_{i1}$, $C_i = C_{in_i}$	starting and completion time of J_i
$\mu_{ij} \in \{1, ..., m\}$	machine processing O_{ij}
$\lambda(t) \in \{1, ..., l\}$	layout used at time t
$\chi(k, t) \in \{1, ..., q_k\}$	configuration of machine k at time t

Table 16: Parameters and decision variables used for the S sub-model.

All the previous outputs are used as input in this sub-model. The objective (f_c^s) is to minimize machine reconfiguration (f_m^s) and tardiness (f_t^s) costs (2.29).

$$f_c^s = f_m^s + f_t^s \tag{2.29}$$

The machine reconfiguration costs ($K_k^{\chi}(e, e')$) are incurred when machine k changes from configuration $\chi(k, t)$ at time t to a different configuration $\chi(k, t+1)$ at time $t+1$ (2.30).

$$f_m^s = \sum_{t=0}^{T-1} \sum_{k=1}^{m} K_k^{\chi}(\chi(k, t), \chi(k, t+1)) \tag{2.30}$$

The tardiness of job J_i is calculated as the maximum of zero and the difference between the completion time C_i and the due date d_i, multiplied by the weight W_i (2.31).

$$f_t^s = \sum_{i=1}^{n} W_i \cdot max(0, C_i - d_i) \tag{2.31}$$

This problem is subject to some constraints. Firstly, manufacturing is not allowed during layout reconfiguration. Secondly, whenever there is a layout change between time t and $t+1$, operations must either complete before it ($C_i \leq t$) or start after the reconfiguration time ($S_i \geq t + P^{\lambda}(u, u')$) associated with the transition from layout u to layout u' (2.32).

$$\forall t \quad \lambda(t) \neq \lambda(t+1) \quad \Rightarrow \quad \forall i \quad (C_i \leq t) \vee (S_i \geq t + P^{\lambda}(\lambda(t), \lambda(t+1))) \tag{2.32}$$

When a machine configuration changes between t and $t+1$, no operations are executed on that machine during the reconfiguration (2.33). Therefore, all operations processed on machine k must either complete before the reconfiguration begins ($C_{ij} \leq t$) or start after the machine reconfiguration time ($S_{ij} \geq t + P^{\chi}_{k}(e, e')$) associated with the transition from configuration e to configuration e' (2.33).

$$\forall t \quad \chi(\mu_{ij}, t) \neq \chi(\mu_{ij}, t+1) \Rightarrow (C_{ij} \leq t) \vee (S_{ij} \geq t + P^{\chi}_{\mu_{ij}}(\chi(\mu_{ij}, t), \chi(\mu_{ij}, t+1))) \tag{2.33}$$

Machines can only process one operation at a time (2.34). As a result, operations allocated to the same machine cannot overlap in time.

$$\mu_{ij} = \mu_{i'j'} \quad \Rightarrow \quad (C_{ij} \leq S_{i'j'}) \vee (C_{i'j'} \leq S_{ij}) \tag{2.34}$$

In this model, each problem is addressed individually, but certain assumptions are necessary. We begin with the process planning stage, where material handling costs influence decision-making. Calculating these costs requires knowledge of the layouts to be used, information that is not yet available. To address this, we employ an approximation method. In our case, we calculate material handling costs using the average of the MH times across all layouts.

In the second stage, to determine the layout, we introduce the minimization of duration costs into the objective function to handle costs related to a time function. These duration costs are the sum of layout reconfiguration times and material handling costs, multiplied by a penalty.

In the scheduling solving stage, no assumptions need to be made.

2.5 Partial sequential model : PP → L + S

In this section, we address first the process planning problem, which yields information about the machine/configuration responsible for producing each operation. This output data serves as input for the integrated layout and scheduling problems (Figure 12).

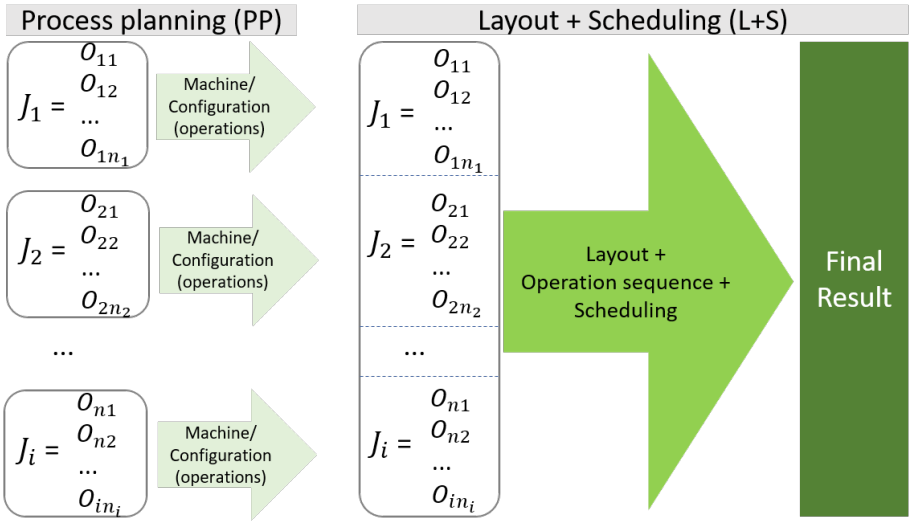

Figure 12: Partial sequential model : PP $\rightarrow$ L + S.

The mathematical model for PP remains identical to the one presented in the previous section (Sequential model). The model for Layout and Scheduling (L + S) integrates both of the aforementioned problems. This integration results in equations 2.24, 2.25, 2.30, and 2.31 composing the objective functions, and equations 2.20, 2.22, 2.32, 2.33, and 2.34 as the constraint set.

In this model, similar to the previous one, an assumption about the layout must be made for the process planning decision to calculate material handling costs. However, unlike the previous model, where layout and scheduling were considered as separate decisions, in this integrated approach, there is no need to incorporate duration costs in the objective function. The only function directly linked to time is the tardiness costs.

2.6 Partial sequential model : PP + L $\rightarrow$ S

In this section, we initially solve process planning and layout jointly, generating output data that includes the machine/configuration for each operation, the layout for each job, and the operation sequences for each job. This output data subsequently serves as input for the scheduling stage (Figure 13).

The mathematical model that integrates PP and L combines elements from the "Sequential model" with some notable modifications. In the previous model, PP dealt with individual jobs, whereas in this integrated approach, multiple jobs are addressed simultaneously. The objective function is composed of equations 2.16, 2.18, 2.24, and 2.25, where the first two must be considered for all jobs ($\forall i$). The constraints employed include 2.19, 2.20, 2.21 (utilizing $P^{\tau}_{\Lambda}(\mu_{ij}, \mu_{ij'})$), and 2.22. The mathematical model for the scheduling sub-model remains identical to the one presented in the sequential model.

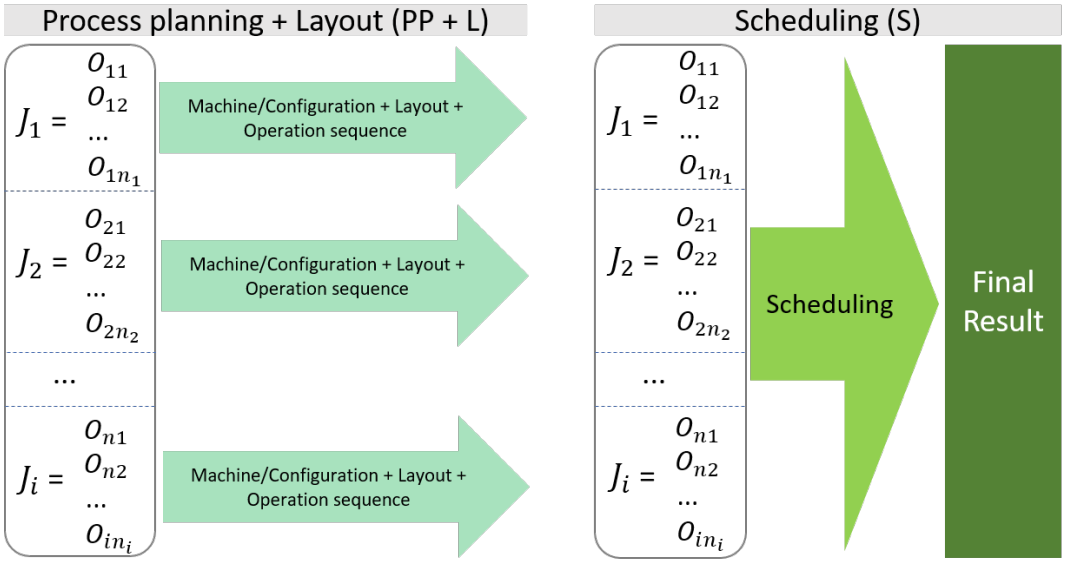

Figure 13: Partial sequential model : PP + L → S.

In this model, no assumptions are made about the layout, as it becomes a decision variable in the integrated problem involving PP and L. Additionally, Equation 2.18 is used to manage time costs; there is no need to incorporate duration costs in the objective function.

2.7 Conclusions of the chapter

In the context of RMS, it is essential to address the interconnected problems of process planning, layout, and scheduling. These problems cannot be entirely separated because they share common decision variables. For this reason, among the new models proposed to solve the stated problem, both completely and partially sequential models are heuristics. Our heuristic strategy aim to divide this complex problem into more manageable sub-problems. While three specific options have been outlined, it is important to note that there is flexibility in how these problems can be tackled. For example, we could opt to initiate the sequence order of problems with the layout, demonstrating the versatility of the heuristics approaches. Our choice made was due to the way we dealt with the problem and the inputs we considered.

Before numerically analyzing the difference between the models, it is noticeable that the sequential and partially sequential (PP → L + S) models present a limitation in terms of process plan flexibility for identical products with the same due date. This limitation arises from the fact that these models result in identical operation sequence orders. In contrast, the integrated model offers more flexibility in that the process plan for each job takes into account the overall resource situation. This enhanced adaptability

is advantageous for achieving the global optimization of the problem.

Furthermore, the approximations and assumptions made in the sequential and partially sequential models probably lead to higher total costs compared to the global optimum provided by the integrated model. However, decomposing the global problem into smaller parts may result in improved execution times. Concrete conclusions will be drawn in Chapter 5.

Chapter 3

Complexity

Contents

3.1 Computational complexity theory

The theory of computational complexity classifies and helps to understand how difficult a problem is regarding the computational execution time to solve it and the memory capacity used [28]. In this work, we will concentrate on the complexity relative to the execution time, which is mainly discussed in general.

When an algorithm is executed on a computer, the time required to reach a solution is quantified in seconds, minutes, or other time units. This duration is influenced by factors like the computer's processor and software. Therefore, measuring in time units is not the most suitable for comparing problems. In the complexity theory, the number

of elementary operations is estimated based on the instance's size, which pertains to the input data. Typically, attention is given to the dominant function, representing the most significant order of complexity, especially in worst-case scenarios [25].

The complexity of problems allows their classification into $\mathcal{P}$ (polynomial time), $\mathcal{NP}$ (non-deterministic polynomial time), $\mathcal{NP}$-complete, and $\mathcal{NP}$-hard [16]. Subsequently, the distinctions between each class will be delineated.

Moreover, problems can be categorized mainly into decision and optimization problems. Decision problems seek to determine a binary answer (yes or no) to a specific question or verify the existence of a given solution [43]. Optimization problems, on the other hand, aim to find the best solution (minimum or maximum) among the feasible solutions to the problem.

3.1.1 Class $\mathcal{P}$

Class $\mathcal{P}$ encompasses problems decidable in polynomial time by a deterministic Turing machine. This implies that the number of elementary operations required for their solution is upper bounded by a polynomial of the instance's size. For instance, a problem with complexity is expressed as $O(k^p)$, where k is a characteristic of the instance's size, and p is a fixed value independent of the input data [16].

The problems within this class are considered "tractable" [43]. They have "efficient" algorithms to solve them since the execution time is upper-bounded by a polynomial function dependent on the instance's size [25]. However, in practice, this is not always strictly true. More complex algorithms can be more efficient than polynomial ones for some specific cases. For example, the simplex algorithm (exponential) is more efficient in linear programming than the ellipsoid method (polynomial) [124]. This fact occurs because complexity analysis typically considers the worst case, but in this context, the worst case for the simplex algorithm is not representative of its overall efficiency.

$\mathcal{P}$-class primarily addresses decision problems. Nevertheless, it is feasible to convert optimization problems into decision problems. When aiming to determine the shortest path between two points (an optimization problem), it can be reformulated as the question: "Is there a path between these two points that is less than a given value x?" In this transformation, each negative response prompts an increase in x, while each positive response triggers a decrease in x.

3.1.2 Class $\mathcal{NP}$

The decision problems of class $\mathcal{NP}$ are those for which a non-deterministic Turing machine can decide it in polynomial time, and a result has a polynomial-size certificate (solution) that can be verified in polynomial time by a deterministic Turing machine. The computers used today are based on a deterministic Turing machine, i.e., when we give them an instruction, there is no doubt which operation they will perform. Conversely, in a non-deterministic machine, the machine can have a set of possibilities at each calculation step. These possibilities form a tree structure, where branches represent the different options of the machine's operation [61].

An efficient algorithm for solving all problems in this class is not known. However, it is possible to check the validity of a solution in polynomial time for all of them, placing $\mathcal{P}$ within $\mathcal{NP}$. An open question remains about whether $\mathcal{NP}$ is also contained within $\mathcal{P}$ [16]. The general belief is that $\mathcal{P} \neq \mathcal{NP}$. Figure 14 illustrates the current division between the $\mathcal{P}$ and $\mathcal{NP}$ classes.

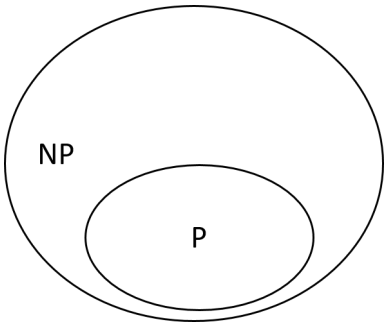

Figure 14: Classes $\mathcal{P}$ and $\mathcal{NP}$.

3.1.3 Class $\mathcal{NP}$-Complete

$\mathcal{NP}$-complete problems represent the most difficult class within $\mathcal{NP}$. The boolean satisfiability problem (SAT) was the first to be categorized as $\mathcal{NP}$-complete in 1971 by Cook [24]. SAT is a decision problem involving boolean expressions composed of logic operators and variables, and the objective is to determine if an assignment of values exists to the variables that make the entire expression true. This problem is noteworthy because any $\mathcal{NP}$ problem can be transformed into a SAT problem with polynomial reduction. Therefore, if an $\mathcal{NP}$-complete problem can be reduced to a $\mathcal{P}$ problem in polynomial time, it implies $\mathcal{P} = \mathcal{NP}$.

A problem A is polynomially reducible to problem B if, for any instance of A, we can construct an equivalent instance of B in polynomial time such that A has a solution if and only if the equivalent instance of B also has a solution [43]. A problem is proved to be

$\mathcal{NP}$-complete using a reduction, starting from a problem already known as $\mathcal{NP}$-complete. If A belongs to $\mathcal{NP}$-complete, then B also belongs to $\mathcal{NP}$-complete. Figure 15 shows how it is believed that the $\mathcal{P}$, $\mathcal{NP}$, and $\mathcal{NP}$-complete classes are currently divided.

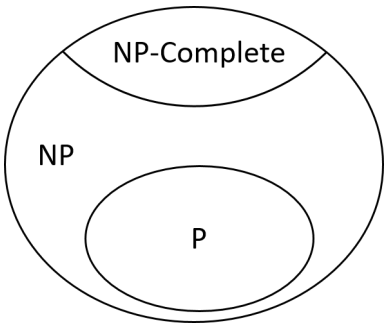

Figure 15: Classes $\mathcal{P}$, $\mathcal{NP}$, and $\mathcal{NP}$-Complete.

3.1.4 Class $\mathcal{NP}$-Hard

The class $\mathcal{NP}$-hard includes any kind of problem (e.g., decision and optimization problems). They are as hard as the hardest $\mathcal{NP}$ problems. An $\mathcal{NP}$-hard problem does not have to be in $\mathcal{NP}$ since this class covers only decision problems. Moreover, an $\mathcal{NP}$-hard that is in $\mathcal{NP}$ is an $\mathcal{NP}$-complete problem. It can also be said that an $\mathcal{NP}$-hard is generally an optimization problem whose decision problem corresponds to an $\mathcal{NP}$-complete problem [43]. Figure 16 shows how it is believed that the $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{NP}$-Complete, and $\mathcal{NP}$-Hard classes are currently divided, having $\mathcal{P} \neq \mathcal{NP}$. If one day someone proves that $\mathcal{P} = \mathcal{NP}$, the division of classes would look like in Figure 17.

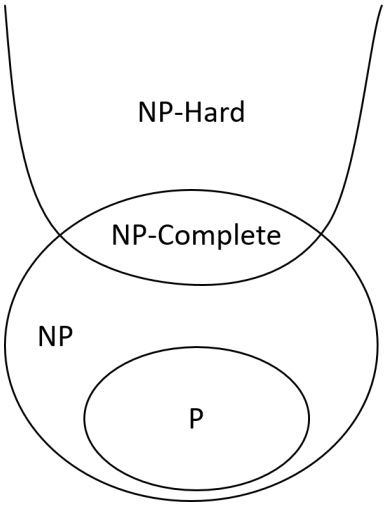

Figure 16: Classes if $\mathcal{P} \neq \mathcal{NP}$.

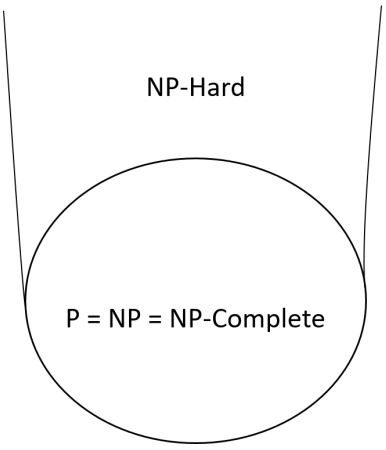

Figure 17: Classes if $\mathcal{P} \neq \mathcal{NP}$.

The three problems studied in this work are $\mathcal{NP}$-hard: process planning, layout, and scheduling [49, 78, 104]. However, small sub-problems of our main problem can

belong to class $\mathcal{P}$. This chapter will deepen the complexity analysis for some scheduling sub-problems.

3.2 Scheduling problem complexity with reconfigurable machines

To the best of our knowledge, there are no studies on the complexity involving reconfigurable machines and layouts. This motivated us to investigate the behavior of the complexity for sub-problems of our main problem. Using multiple machines and configurations, each of our problems (process planning, layout configuration, and scheduling) is known to be $\mathcal{NP}$-hard [49, 78, 104]. However, what remains uncertain is the classification of much smaller sub-problems.

The goal is to find some boundaries where a sub-problem is Polynomial and at what point, by adding components or constraints, it becomes $\mathcal{NP}$-hard. Our exploration focuses explicitly on machine reconfiguration, which represents a simplified version of the broader problem defined in Chapter 2 and will be detailed in Section 3.2.1.

Numerous scheduling problems without machine reconfiguration have been proven to be $\mathcal{NP}$-hard. Notably, this is true for $1||\sum T_i$ [87], $P||C_{max}$ [44] (even in the case of 2 machines [89]), $P||\sum w_i C_i$ [15], or $1|r_i, pmtn|\sum w_i C_i$ [84]. Considering these problems as special cases of scheduling problems where machines have only one configuration, it seems unlikely that polynomial algorithms exist to solve versions of these problems when machine reconfiguration environments are considered.

In contrast, the problem $1||\sum w_i C_i$ is polynomially solvable using the Smith's rule [128]: jobs are scheduled as soon as possible by sequencing them in non-decreasing order of the $\frac{p_i}{w_i}$ values. Similarly, $1||C_{max}$ can be polynomially solved by scheduling jobs as soon as possible in any order. Polynomial algorithms also exist for solving $P||\sum C_i$ (see, for example, [15, 62]) or $1|r_i, ptmtn|\sum C_i$ (see [8]). The interesting question arises: can we still achieve polynomial solutions for these problems when considering machine reconfiguration environments? In this work, we do not provide a complete answer to this question, but we aim to make progress on the subject.

3.2.1 Problem description

We focus on the single-machine scheduling problem with machine reconfiguration, aiming to minimize the total weighted completion time or the makespan. In this problem, a set $J = \{1, \ldots, n\}$ comprising n jobs needs to be scheduled on a machine that can

be configured in one of the m configurations from the set $\chi = \{1, \ldots, m\}$. Each job is considered available from time 0 and is associated with a weight w_i.

In configuration $k \in \chi$, the processing time of a job $i \in J$ is denoted as p_{ik}. A transition from configuration j to configuration k, i.e., a reconfiguration from j to k, is represented as r_{jk}. The duration of this reconfiguration is noted as Δ_{jk}.

At the beginning and end of a valid schedule, the machine is assumed to be in a resting configuration 0, where no job can be processed. We establish that $\forall i \in J, p_{i0} = \infty$. The durations Δ_{0k} and Δ_{k0} can be set to zero or any other relevant value from an industrial perspective.

A schedule can be represented by a sequence σ comprising jobs and reconfigurations. It is important to note that this sequence must start with a reconfiguration r_{0k} where $k \in \chi$, must end with a reconfiguration r_{l0} where $l \in \chi$, and must include each job of J exactly once.

For each schedule/sequence σ, we can associate its cost, representing either its total weighted completion time $f(\sigma) = \sum_{i=1}^{n} w_i C_i$ or its makespan $g(\sigma) = \max_{i=1}^{n} C_i$, where C_i is the completion time of job i in the schedule. Given that all jobs are available from time 0, the jobs are scheduled as soon as possible in the order of the sequence, as only active schedules are optimal [8]. We start the process at time $t = 0$, with configuration $c = 0$, and a cost $f = 0$. We iteratively consider each element $\sigma(j)$ in the sequence order. If $\sigma(j)$ is a reconfiguration r_{kl}, we increment t by Δ_{kl}, and replace configuration k by l. If $\sigma(j)$ is a job i, we increase t by p_{ic}, set $C_i = t$ as the completion time of i, and increase f by $w_i C_i$. If we are analyzing the makespan, g is the completion time of the last job in the sequence.

The problem is to find the sequence σ of jobs and reconfigurations minimizing either $f(\sigma) = \sum_{i=1}^{n} w_i C_i$ or $g(\sigma) = \max_{i=1}^{n} C_i$.

It is worth noting, without loss of generality, that $\forall x, y, z \in \chi$, we consider $\Delta_{xy} + \Delta_{yz} \geq \Delta_{xz}$. In the other case (if $\Delta_{xy} + \Delta_{yz} < \Delta_{xz}$), it is always better to use the two consecutive transitions r_{xy} and r_{yz} to move from configuration x to configuration z, rather than the reconfiguration r_{xz} alone. Consequently, we can formulate an equivalent problem in which the reconfiguration r_{xz} implies the combination of the two reconfigurations r_{xy} and r_{yz}.

In the remainder of this chapter, when we refer to sequences, we specifically mean a sequence of jobs and reconfigurations as defined earlier (i.e., starting with a reconfiguration r_{0k} where $k \in \chi$, ending with a reconfiguration r_{l0} where $l \in \chi$, and including each job in J exactly once).

Consider σ as a given sequence. When referring to a subsequence of σ starting at

position k and ending at position j, we employ the notation $\sigma[k;j]$. It is important to note that if the first element of a subsequence is a job, we keep in memory the configuration in which this job and the subsequent jobs are processed until the next reconfiguration is processed.

In the following, we frequently use the notation σ to refer either to a sequence or a subsequence. The number of elements in a sequence or subsequence σ is noted as $|\sigma|$. Note that for a sequence σ, we observe $|\sigma| \geq n+2$ since it encompasses at least one reconfiguration r_{0k}, one reconfiguration r_{k0}, and the n jobs.

When referring to the cost $f(\sigma)$ of a specific subsequence σ, this cost is calculated by scheduling jobs of the subsequence from time 0. The notation $f(\sigma,t)$ is employed when discussing the cost of the sequence computed by scheduling jobs from time t.

Finally, we use the notation $\sigma \cdot \delta$ to denote the subsequence obtained by concatenating subsequences σ and δ.

3.2.2 Properties

After describing all the necessary notations, some properties can be demonstrated before starting to analyze the complexity of specific problems.

Proposition 1. The set of sequences in which at least one job is always sequenced between two reconfigurations is dominant.

Proof. Suppose an optimal sequence σ contains a subsequence $r_{xy} \cdot r_{yz}$, i.e., two consecutive reconfigurations without a job sequenced between them. Recall that $\Delta_{xy} + \Delta_{yz} \geq \Delta_{xz}$. Consequently, if we construct the sequence σ' by replacing the subsequence $r_{xy} \cdot r_{yz}$ with r_{xz}, we obtain $f(\sigma') \leq f(\sigma)$ and $g(\sigma') \leq g(\sigma)$. This is because all jobs sequenced after reconfiguration r_{xz} will start at the same time or earlier in the sequence σ' compared to the sequence σ. □

Consequently, in the remainder of the chapter, we will restrain the research of an optimal solution to the sequences where there is always at least one job between two reconfigurations.

In this case, observe that we have $|\sigma| \leq 2 + n + (n-1) = 2n+1$ since there are at most $n-1$ reconfigurations between jobs in an optimal solution (one between each job of the sequence). Consequently, the costs $f(\sigma)$ and $g(\sigma)$ of such a sequence σ can be evaluated in $O(n)$ time.

Proposition 2. Let σ be a subsequence and consider two times t and t'. We have $f(\sigma,t') = f(\sigma,t) + (t'-t)\sum_{i\in\sigma} w_i$

Proof. The completion C_i of each job $i \in \sigma$ when the jobs are scheduled from time t is changed by $(t' - t)$ (this value being negative or positive) when the jobs are scheduled from time t'. Consequently, the associated cost with i increases by $(t' - t)w_i$. □

Proposition 3. Every subsequence τ within an optimal sequence σ, which minimizes the total weighted completion time and consists exclusively of jobs (with no reconfiguration inside the subsequence), follows the Smith's rule.[128].

Proof. Each job within a subsequence τ is scheduled in the same configuration k (as there is no reconfiguration between jobs). Suppose there exists a subsequence $i \cdot j$ in τ with $\frac{p_{ik}}{w_i} > \frac{p_{jk}}{w_j}$, meaning two consecutive jobs in the subsequence τ where Smith's rule is not satisfied. In such a case, we can construct a new sequence σ' from σ in which jobs i and j are interchanged. This leads to $f(\sigma') < f(\sigma)$, contradicting the optimality of σ. □

Thus, an optimal solution for the total weighted completion time criterion is necessarily a sequence of subsequences of jobs (scheduled in the same configuration) following the Smith's rule, with reconfigurations occurring between these subsequences. It is worth noting that the Smith's rule can be applied between all jobs scheduled in the same configuration, even if these jobs are part of two different subsequences, as shown in the following proposition.

Proposition 4. Let σ be an optimal sequence minimizing the total weighted completion time, represented as $\sigma = A \cdot \tau_1 \cdot B \cdot \tau_2 \cdot C$, where A, B, and C are subsequences of jobs and reconfigurations, and τ_1 and τ_2 are two subsequences of jobs scheduled in the same configuration k. In such a case, $\tau_1 \cdot \tau_2$ must follow the Smith's rule.

Proof. Note that both τ_1 and τ_2 follow the Smith's rule. In another case, it contradicts the fact that σ is optimal because we can build a better solution by simply resequencing τ_1 or τ_2 according to the Smith's rule.

Let i be the last job of subsequence τ_1 and let j be the first job of subsequence τ_2. Suppose that $\tau_1 \cdot \tau_2$ does not follow the Smith's rule. Therefore, we have $\frac{p_{ik}}{w_i} > \frac{p_{jk}}{w_j}$. However, since both τ_1 and τ_2 follow the Smith's rule, it implies that i has the highest value of $\frac{p_{xk}}{w_x}$ in τ_1, while j has the smallest value of $\frac{p_{xk}}{w_x}$ in τ_2.

Let P_B be the sum of all processing times and reconfiguration durations of jobs in subsequence B, and let W_B be the sum of all weights of jobs belonging to subsequence B. Suppose $\frac{P_B}{W_B} > \frac{p_{jk}}{w_j}$. We can construct a new sequence σ' by removing job j from sequence τ_2 and inserting it at the end of sequence τ_1, just before subsequence B (see Figure 18). During this process, the cost of j decreases by $P_B w_j$, while the cost of sequence B increases by $p_{jk}W_B$ (see Proposition 2). The cost of any other job in σ does not change since their

completion time in σ' is the same as in σ. Thus, $f(\sigma') = f(\sigma) - P_B w_j + p_{jk} W_B$, and then $f(\sigma') - f(\sigma) < 0$ (since $P_B w_j > p_{jk} W_B$), contradicting the optimality of sequence σ.

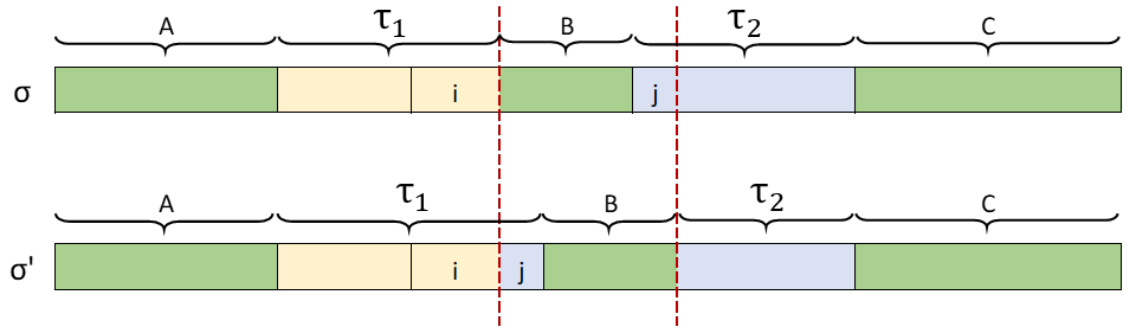

Figure 18: Changing job position when $\frac{P_B}{W_B} > \frac{p_{jk}}{w_j}$.

Therefore, we can assume from now that $\frac{P_B}{W_B} \leq \frac{p_{jk}}{w_j} < \frac{p_{ik}}{w_i}$. We construct sequence σ'' from σ by removing job i from τ_1 and inserting it at the beginning of τ_2, just after subsequence B (see Figure 19). During this insertion, the cost of subsequence B decreases by $p_{ik} W_B$, while the cost of job i increases by $P_B w_i$. The cost of any other job in σ remains unchanged since their completion times in σ'' are the same as in σ. Consequently, $f(\sigma'') = f(\sigma) - p_{ik} W_B + P_B w_i$, leading to $f(\sigma'') - f(\sigma) < 0$ (as $p_{ik} W_B > P_B w_i$), contradicting the optimality of σ.

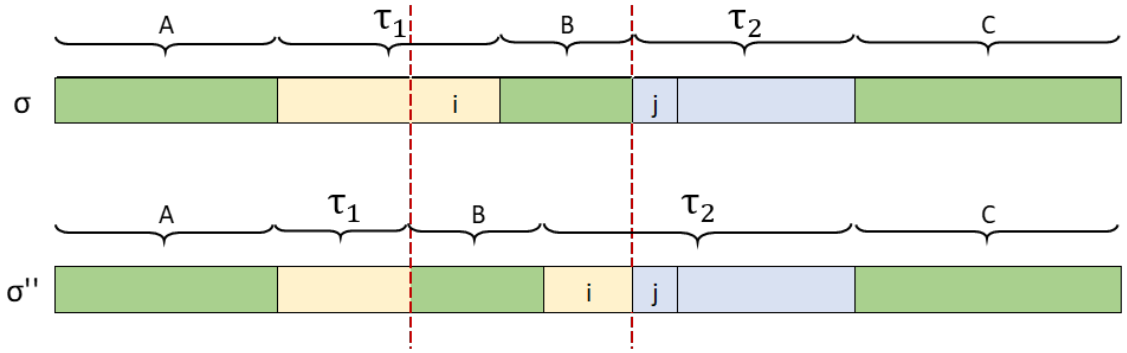

Figure 19: Changing job position when $\frac{P_B}{W_B} \leq \frac{p_{jk}}{w_j} < \frac{p_{ik}}{w_i}$.

We can that conclude that $\tau_1 \cdot \tau_2$ must follow the Smith's rule. □

In the following, we designate a specific subsequence σ as optimal if there is no other subsequence with a lower associated cost (either the total weighted completion time or the makespan), with exactly the same jobs, and such that the initial job of the sequence starts in the same configuration.

Proposition 5. A sequence σ is considered optimal if and only if every subsequence $\sigma[k; |\sigma|]$, where $k \in \{1, \ldots, |\sigma|\}$, is optimal.

Proof. Assuming that σ is an optimal sequence, if there exists $k \in \{1, \ldots, |\sigma|\}$ such that $\sigma[k; |\sigma|]$ is not optimal, it implies the existence of a subsequence δ with the same jobs as $\sigma[k; |\sigma|]$, starting in the same configuration, and having $f(\delta) < f(\sigma[k; |\sigma|])$ (respectively $g(\delta) < g(\sigma[k; |\sigma|])$). Consequently, the sequence $\sigma[1; k-1] \cdot \delta$ would have $f(\sigma[1; k-1] \cdot \delta) < f(\sigma)$ (respectively $g(\sigma[1; k-1] \cdot \delta) < g(\sigma)$), contradicting the optimality of σ.

□

3.2.3 The special case of null reconfiguration duration times

In this subsection, we consider a scheduling problem where, for all $j, k \in \chi$, $\Delta_{jk} = 0$ to minimize a regular criterion [8]. A regular criterion is an increasing function of the completion times of the jobs. Examples of such criteria include C_{max} and $\sum w_i C_i$. The following result holds:

Proposition 6. In scheduling problems with machine reconfigurations aiming to minimize a regular criterion, when the reconfiguration duration times are null, it is always optimal to schedule each job in a configuration where its processing time is the shortest.

Proof. Suppose we have an optimal schedule for such a problem and a job i is not scheduled in a configuration with the shortest processing time. We can create a new schedule from the original one by changing the configuration of job i to one with the shortest processing time. This modification can result in job i and all subsequent jobs potentially completed earlier, with no delaying of any other job. Therefore, the revised schedule is optimal, as the completion times of the jobs can only decrease. □

After determining the configuration in which each job has its shortest processing time, a process achievable in $\mathcal{O}(mn)$, we can apply an algorithm designed for solving the problem without reconfiguration. Since polynomial solutions exist for $1||C_{max}$ and $1||\sum w_i C_i$, the same is applied to the version of the problem with machine reconfiguration and null reconfiguration duration times. Note that the result also holds for the versions of the $P||\sum C_i$ and $1|r_i, pmtn|\sum C_i$ with machine reconfiguration and null reconfiguration duration times.

3.2.4 Solving the total weighted completion time problem

In this section, we present an algorithm to solve the total weighted completion time problem. Algorithm 1 systematically explores all possible assignments of jobs in J to configurations in χ. For it, we build $v \in \chi^n$, where $v[i]$ represents the configuration in which job $i \in J$ has to be processed. The algorithm constructs an array τ consisting of m lists of jobs. The list $\tau[k]$ contains the list of jobs i that need to be processed in

configuration k, i.e., $\{i \mid v[i] = k\}$, sorted in non-increasing order of $\frac{p_{ik}}{w_i}$, corresponding to the reverse order of the Smith's rule. This sorting process can be done in $O(n \ln n)$ time. The notation $|\tau[k]|$ denotes the number of jobs in list $\tau[k]$. The array τ is then passed to the SOLVINGWITHFIXEDCONFIGURATION subroutine, detailed in Algorithm 2, which is designed to solve the problem when the configuration of each job is fixed.

Algorithm 1 Solving the single-machine total weighted completion scheduling problem with machine reconfigurations

Data: $J = \{1, \ldots, n\}$, the set of jobs
Data: $\forall i \in J, w_i$, the weights of jobs
Data: $\chi = \{1, \ldots, m\}$, the set configurations
Data: $\forall i \in J, \forall k \in \chi, p_{ik}$, the duration of jobs in each configuration
Data: $\forall k, l \in \chi, \Delta_{kl}$, the duration of each possible reconfiguration
$\sigma \leftarrow \epsilon$, $f \leftarrow \infty$
//enumerate all possible assignments of jobs with configurations
for all $v \in \chi^n$ **do**
//preparation of the data associated with v
    for $k \leftarrow 1$ **to** m **do**
        build the list $\tau[k]$ of the jobs which have to be processed in configuration k sorted in non-increasing order of $\frac{p_{ik}}{w_i}$
    $(\sigma', f') \leftarrow$ SOLVINGWITHFIXEDCONFIGURATION(τ)
    if $f' < f$ **then** //a better solution has been found
        $\sigma \leftarrow \sigma'$, $f \leftarrow f'$
return (σ, f)

Algorithm 2 is a dynamic programming algorithm that iteratively generates the set of non-dominated subsequences, verifying Propositions 1, 3, 4, and 5. The subsequences constructed all along the algorithm represent the end of the sequences obtained by the algorithm and always add a job or a reconfiguration at the beginning of the sequence. During the algorithm, when considering a subsequence σ in a current configuration c, the insertion of a new job at the beginning of the sequence can occur by processing it in the same configuration c or by processing it in another configuration $k \neq c$, followed immediately by a reconfiguration r_{kc}.

A state of the dynamic programming algorithm is defined by a tuple $s = (c, u, \sigma, f, w)$ representing the sequences, where:

- c is the current configuration of the machine,
- $u(s)$ is an array of m indices, denoted as $u[k]$, where $k \in \chi$, indicating how many jobs have already been sequenced in list $\tau[k]$,
- σ is one optimal subsequence associated with s, i.e., the cost of an optimal subsequence

scheduling all the $u[k]$ first jobs of each list $\tau[k]$,

- f is the cost of such an optimal subsequence,
- $w = \sum_{i \in J \cap \sigma} w_i$ is the sum of the weights of already sequenced jobs in σ.

The algorithm begins with the set of states $S_0 = \{(k, u, \sigma, f, w) \mid k \in \chi\}$ where the current configuration of the machine is k, $\forall j \in \chi, u[j] = 0$ (no job has been sequenced), $\sigma = r_{k0}$ (the last operation of the sequence is a reconfiguration from k to 0), $w = 0$ (no job has been sequenced), and $f = 0$ (the cost is zero).

Algorithm 2 SOLVINGWITHFIXEDCONFIGURATION: solving the when the configurations of each job has been fixed

Data: τ: an array of m lists where $\tau[k]$ is the list of the jobs which have to be processed in configuration k sorted in non-increasing order of $\frac{p_{ik}}{w_i}$
//Initialisation of the first states
$S_0 \leftarrow \emptyset$
for $k \leftarrow 1$ **to** m **do**
    $u[k] \leftarrow 0$
for $k \leftarrow 1$ **to** m **do**
    $u[k] \leftarrow 0$
    insert $(k, u, r_{k0}, 0, 0)$ in S_0
for $i \leftarrow 1$ **to** n **do**
    // Computing the set of states S_i from set S_{i-1}
    $S_i \leftarrow \emptyset$
    for all $s = (c, u, \sigma, f, w) \in S_{i-1}$ **do**
        for $k \leftarrow 1$ m **do**
            //We try to process a new job in configuration k
            if $u[k] < |\tau[k]|$ **then**
                //at least one job to process in configuration k
                $j \leftarrow \tau[k][u[k]+1]$ //the next job to sequence
                $u' \leftarrow u$, $u'[k] \leftarrow u[k]+1$
                if $k = c$ **then** //process j in the same configuration
                    $\sigma' \leftarrow j \cdot \sigma$
                    $f' \leftarrow f + p_{jk}(w_j + w)$
                else//process j in another configuration
                    $\sigma' \leftarrow j \cdot r_{kc} \cdot \sigma$ // insert a reconfiguration before σ
                    $f' \leftarrow f + p_{jk}w_j + (p_{jk} + \Delta_{kc})w$
                $w' \leftarrow w + w_j$
                $s' \leftarrow (k, u', \sigma', f', w')$
                search another state $s'' = (k, u', \sigma'', f'', w'')$ in S_i
                if $s'' = NIL$ or $f' < f''$ **then**
                    if $s'' \neq NIL$ **then**
                        remove s'' from S_i // s'' is dominated by s'
                    insert s' in S_i
$\sigma \leftarrow \epsilon$, $f \leftarrow \infty$
for all $s = (k, u, \sigma, f, w) \in S_n$ **do**
    $\sigma' \leftarrow r_{0k} \cdot \sigma$
    $f' \leftarrow f + \Delta_{0k}w$
    if $f' < f$ **then**
        $\sigma \leftarrow \sigma'$, $f \leftarrow f'$ //a better solution is found
return (σ, f)

At each step i of the main algorithm loop, we generate the set S_i of states, where i jobs in J have been sequenced. These states are derived from the set S_{i-1}, which contains

states generated in the previous step.

Each state $s = (c, u, \sigma, f, w) \in S_{i-1}$ is considered. For each $k \in \chi$, we examine the first not-sequenced job in $\tau[k]$, denoted as $j = \tau[k][u[k] + 1]$. This job is then scheduled at the beginning of σ, creating a new state $s' = (k, u', \sigma', f', w')$. Note that this is only possible if there are remaining jobs in list $\tau[k]$, i.e. if $u[k] < |\tau[k]|$. Two scenarios can happen:

1. If $c = k$, then the first job j in the new sequence $\sigma' = j \cdot \sigma$ is processed in the same configuration as the first job in σ.
2. If $c \neq k$, a reconfiguration from k to c is inserted between the first job j of the new sequence and σ, resulting in $\sigma' = j \cdot r_{kc} \cdot \sigma$.

In all cases, the cost of σ' can be then computed from f in $O(1)$ times by using Proposition 2. In case 1, the cost of jobs in σ is increasing from $p_{jk}(w_j + w)$ due to the insertion of j, while in case 2 the cost of these jobs is increasing from $p_{jk}w_j + (p_{jk} + \Delta_{kc})w$ because of the reconfiguration duration time.

Note that each built subsequence verifies Propositions 3 and 4 since the jobs are considered in the order of the lists in τ. The sequences kept in each state S_i at the end of step i are only the optimal ones to verify Proposition 5. To achieve this, the dominated sequences are discarded. Thus, if two states $s' = (k, u, \sigma', f', w)$ and $s'' = (k, u, \sigma', f', w)$ are built during the same step i, i.e., two states starting in the same configuration k and having exactly the same already sequenced jobs, only the best one is kept.

After scheduling all n jobs, the algorithm adds the reconfiguration r_{0k} at the beginning of each sequence σ within every state $s = (k, u, \sigma, f, w) \in S_n$. Subsequently, the associated cost is updated accordingly. At the end, the algorithm identifies the best sequence among the generated m sequences and returns it with its corresponding cost.

During the algorithm, since each value $u[k]$ with $k \in \chi$ in a state $s = (c, u, \sigma, f, w)$ can take a value in $\{0, \ldots, |\tau[k]|\}$ and since $|\tau[k]| \leq n$, the cardinal of S_i is upper-bounded by $m \times n^m$. States of S_i can be recorded in an AVL-tree. Thus, adding or researching a state runs in $\mathcal{O}(m \ln(n) + \ln(m))$. Finally, Algorithm 2 runs in $\mathcal{O}(m \times n^{m+1} \times (m \ln(n) + \ln(m)))$. It leads to the following result:

Proposition 7. If a constant x bounds the number of configurations m, and the configuration for each job is fixed, then the problem with fixed configurations becomes polynomial with a complexity of $\mathcal{O}(n^{x+1} \ln(n))$.

Since there are $\mathcal{O}(m^n)$ possible assignments v enumerated during Algorithm 1, its complexity is $\mathcal{O}(m^{n+1} \times n^{m+1} \times (m \ln(n) + \ln(m)))$. Unfortunately, it is not polynomial

in n, even if m is considered fixed.

3.2.5 Solving the makespan problem

When analyzing the same problem with a different objective function, specifically the minimization of makespan, other properties are found:

Proposition 8. The set of sequences in which there is no reconfiguration between jobs scheduling in the same configuration is dominant.

Proof. Consider an optimal sequence σ minimizing the makespan. Suppose σ can be expressed as $\sigma = A \cdot \tau_1 \cdot B \cdot \tau_2 \cdot C$, where A, B, and C are subsequences of jobs and reconfigurations. Where τ_1 and τ_2 are subsequences of jobs scheduled in the same configuration k, and where A, B, and C begin and end with reconfigurations. We can construct a sequence $\sigma' = A \cdot \tau_1 \cdot \tau_2 \cdot B \cdot C$ by grouping all jobs processed in configuration k together without increasing the makespan of the entire sequence. Since B ends with a reconfiguration r_{xk} and C begins with a reconfiguration r_{ky}, the subsequence $r_{xk} \cdot r_{yk}$ can be replaced by the single reconfiguration r_{xy} without increasing the makespan, given that $\Delta_{xy} \leq \Delta_{xk} + \Delta_{ky}$. □

Proposition 9. Consider a subset $\chi' \subset \chi$ representing the configurations used by an optimal sequence σ. Then, each job $j \in J$ is processed in a configuration belonging to χ' where its processing time is the shortest.

Proof. Consider an optimal sequence σ that minimizes the makespan, and let χ' be the subset of configurations used by this solution. Suppose there exists a job i processed in configuration $x \in \chi'$ in σ, while there exists $y \in \chi'$ such that $p_{iy} < p_{ix}$. In this case, we can rearrange job i within a subsequence of σ where job i is processed in configuration y. The resulting sequence σ' has a makespan $g(\sigma') < g(\sigma)$, contradicting the optimality of σ. □

Using the above properties, the problem can be solved as follows:

- Listing all possible partial permutations c of χ, denoted as $\sum_{k=1}^{m} A_m^k = \sum_{k=1}^{m} \frac{m!}{(m-k)!}$, where each configuration is considered at most once.
- For every partial permutation c, assign the configuration from c to each job where it has the shortest processing time. This process can be executed in $O(nm)$ time. Subsequently, sequence the jobs following the order defined by c.

Note that if m is considered upper-bounded by a constant, the number of partial enumerations is a constant. Hence, the subsequent conclusion is valid:

Proposition 10. When the number of machine configurations, denoted as m, is considered upper-bounded by a constant x, the problem is polynomial in $\mathcal{O}(n)$.

3.2.6 Complexity of the general one-machine problem

The single-machine scheduling problem with machine reconfiguration shares similarities with the scheduling problem involving families of jobs with setup times [91].

This problem categorizes jobs into m families denoted by $\chi = \{1, \ldots, m\}$. During the processing of jobs on a machine, a setup time s_{lk} is required whenever there is a transition from processing a job in family l to a job in family k. The problem is represented as $1|s_{lk}, p_{kj}| \sum_{k,j} w_i C_i$ in [91], where p_{kj} represents the processing time of the j^{th} job in family k.

The problem of scheduling families of jobs with setup times can be seen as a particular case of the one-machine scheduling problem with machine reconfiguration, where the configuration in which each job will be processed is predetermined.

When the number m of families is considered to be lower than a given constant, the problem can be solved in polynomial time [105]. However, for an arbitrary number of families, Rinnooy Kan has proven that the $1|s_{lk}, p_{kj}| \sum_{k,j} w_i C_i$ problem is $\mathcal{NP}$-hard [122], even when each family contains only one job. His proof holds even in cases where all jobs have the same processing time [91] or if all weights are equal to 1. Thus, $1|s_{lk}, p| \sum_{kj} C_{kj}$ is also $\mathcal{NP}$-hard. It leads to the following result:

Proposition 11. The single-machine scheduling problem with machine reconfiguration to minimize the total (weighted) completion time is $\mathcal{NP}$-hard. This holds even in cases where all processing times are equal.

Proof. Given an instance of the problem $1|s_{lk}, p| \sum_{kj} C_{kj}$, we can build an instance of the single-machine scheduling problem with machine reconfiguration aiming to minimize the total completion time. Each family of jobs in $1|s_{lk}, p| \sum_{kj} C_{kj}$ is associated with a configuration of the machine. The j^{th} job in family k has a processing time $p_{kj} = p$ in configuration k and an infinite processing time in the other configurations. This reduction is polynomial. Consequently, the single-machine scheduling problem with machine reconfiguration to minimize the total (weighted) completion time is $\mathcal{NP}$-hard, even when all processing times are equal.

□

The $1|s_{lk}, p|C_{max}$ problem is also $\mathcal{NP}$-hard, even when each family contains only one job [91]. Similarly, we can demonstrate that the problem to minimize C_{max} is also $\mathcal{NP}$-hard:

Proposition 12. The single-machine scheduling problem with machine reconfiguration to minimize the makespan is $\mathcal{NP}$-hard. This holds even in cases where all processing times

are equal.

Proof. Given an instance of the problem $1|s_{lk}, p|C_{max}$, we can build an instance of the single-machine scheduling problem with machine reconfiguration aiming to minimize the makespan. Similar to the previous proposition, each family of jobs in $1|s_{lk}, p|C_{max}$ is associated with a configuration of the machine. The j^{th} job in family k has a processing time $p_{kj} = p$ in configuration k and an infinite processing time in the other configurations. This reduction is polynomial. Therefore, the single-machine scheduling problem with machine reconfiguration to minimize the makespan is $\mathcal{NP}$-hard, even when all processing times are equal.

□

3.2.7 Parallel machines

The parallel machines scheduling problem with machine reconfiguration to minimize the makespan is $\mathcal{NP}$-hard. This conclusion stems from the fact that the $P2||C_{max}$ problem, which is a sub-problem of the considered one with only two machines and one configuration, is $\mathcal{NP}$-hard [44]. Hence, even with a constant m, the problem remains $\mathcal{NP}$-hard.

For the problem of minimizing the sum of weighted completion time, it remains $\mathcal{NP}$-hard irrespective of whether m is constant or not. This is due to the fact that the sub-problem with only one configuration, $P2||\sum w_i C_i$ [17], is $\mathcal{NP}$-hard. The complexity of the problem, where the number of configurations is constant and the goal is to minimize the sum of completion time, is still an open question. If we can establish that the problem with only one machine and a fixed configuration is $\mathcal{NP}$-hard, this directly implies the same complexity for parallel machines. Conversely, if it can be proven to be polynomial for one machine, further analysis would be needed for parallel machines.

3.3 Conclusions of the chapter

The scheduling, process planning, and layout problems in Reconfigurable Manufacturing Systems (RMS) are known to be $\mathcal{NP}$-hard. However, to the authors' best knowledge, existing studies have not thoroughly analyzed the complexity of sub-problems considering machine and system reconfiguration. To address this gap, our analysis focuses on smaller sub-problems where machine reconfiguration is possible. We primarily investigate the scheduling problem for a single reconfigurable machine, aiming to minimize the makespan or the sum of completion times. Additionally, we extend our analysis to problems involving parallel machines.

The decision to exclusively analyze these objective functions is driven by the potential to draw conclusions about other problems through reductions, particularly if they are $\mathcal{NP}$-hard. Figure 20 shows the reduction graph of scheduling problems according to the objective function. They are divided into the following categories [16]:

- C_{max} : maximum completion time;
- L_{max} : maximum lateness ($C_i - d_i$, where d_i is the due date);
- $\sum C_i$: total completion time;
- $\sum w_i C_i$: total weighted completion time;
- $\sum T_i$: total tardiness ($max\{0, C_i - d_i\}$);
- $\sum w_i T_i$: total weighted tardiness;
- $\sum U_i$: total unit penalty (0 if $C_i \leq d_i$, otherwise equal to 1);
- $\sum w_i U_i$: total weighted unit penalty;

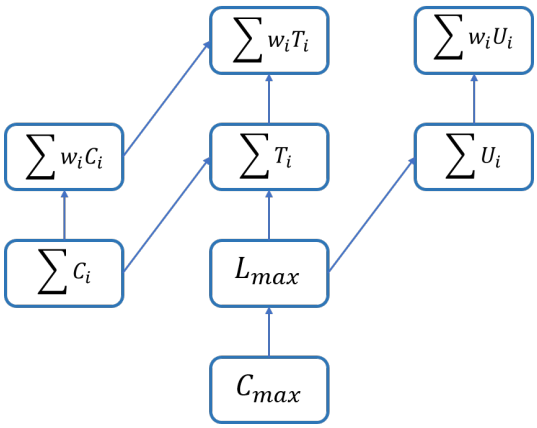

Figure 20: Reductions between objective functions [16].

Table 17 summarizes our findings. The first row indicates the objective function under consideration. The second column denotes whether the number of configurations is treated as constant. The third column specifies whether it is a single-machine problem or a parallel problem (limited to two machines, the smallest possible number). The last column indicates the complexity.

Objective function	m **constant**	**Machine type**	**Complexity**
C_{max}	yes	one-machine	Polynomial
$\sum C_i$	yes	one-machine	Open
$\sum w_i C_i$	yes	one-machine	Open
C_{max}	no	one-machine	$\mathcal{NP}$-hard
$\sum C_i$	no	one-machine	$\mathcal{NP}$-hard
$\sum w_i C_i$	no	one-machine	$\mathcal{NP}$-hard
C_{max}	yes	P2	$\mathcal{NP}$-hard
$\sum C_i$	yes	P2	Open
$\sum w_i C_i$	yes	P2	$\mathcal{NP}$-hard
C_{max}	no	P2	$\mathcal{NP}$-hard
$\sum C_i$	no	P2	$\mathcal{NP}$-hard
$\sum w_i C_i$	no	P2	$\mathcal{NP}$-hard

Table 17: Main results found in this chapter.

In this chapter, we introduced an algorithm to address the problem of a single machine minimizing the sum of weighted completion times. However, it is important to note that the algorithm currently operates at an exponential time complexity. This does not conclusively indicate that the problem is non-polynomial; it simply means that we have not demonstrated a polynomial solution or proven the problem to be $\mathcal{NP}$-hard. Therefore, the question of the problem's complexity remains open. If it is proven to be $\mathcal{NP}$-hard, it would directly imply that the open problem for parallel machines is also $\mathcal{NP}$-hard.

Chapter 4

Solving methods

Contents

In Chapter 3, we studied the complexity of a simplified problem. This chapter returns to the general problem outlined in Chapter 2 and presents the solving methods utilized and applied in this work. The first part is dedicated to three exact methods: integer linear programming, constraint programming, and constraint programming with a defined search strategy. After that, two metaheuristics are presented: a genetic algorithm and a genetic algorithm with local search. Finally, a conclusion summarizes the main findings of this chapter.

4.1 Exact Methods

One way to get an optimal solution is to go through all the possible combinations of the variable sets, which is often not feasible in practice because the time required for this is too long. Therefore, some techniques can reduce the search space and, consequently, the search time. This section presents the exact methods used to solve the studied problem without doing an exhaustive search.

The methods described are: Integer Linear Programming (ILP), Constraint Programming (CP), and Constraint Programming with a defined search strategy (CPS). This section mainly explains the strategy used in CPS since ILP and CP will be solved by the standard IBM ILOG CPLEX Optimizer and IBM ILOG CP Optimizer software tools, respectively. On the other hand, CPS will use the same software as CP but we define the required search strategy. In the "Numerical results" chapter, we will compare the execution times using these different methods over a set of instances.

4.1.1 Integer linear programming

Integer linear programming (ILP) stands as one of the classical methods for mathematically modeling and solving optimization problems, where all variables belong to the set of integers, and all equations (objective function and constraints) are linear [25].

As presented in Chapter 2, our model is inherently non-linear due to using the max function in constraint 2.6. However, it can be easily linearized. For a constraint $X = max(x_1, x_2)$, we introduce a binary variable y, defined as 1 if $x_1 \geq x_2$ and 0 if $x_1 < x_2$. Additionally, we introduce a constant M that is greater than both x_1 and x_2. The following constraints enforce the definition of y:

- $x_1 - x_2 \leq My$
- $x_2 - x_1 \leq M(1 - y)$

Consequently, the following constraints enforce $X = max(x_1, x_2)$:

- $X \geq x_1$
- $X \geq x_2$
- $X \leq x_1 + M(1 - y)$
- $X \leq x_2 + My$

After the linearization, IBM ILOG CPLEX Optimizer standard tool was used to solve the ILP, which applies different algorithms based mainly on the simplex algorithm [66].

4.1.2 Constraint programming

To improve the execution time required to obtain the best solution, it was decided to explore other exact method techniques, in this case, constraint programming.

Constraint Programming (CP) is an approach employed to address combinatorial search problems by integrating techniques from artificial intelligence, operations research,

algorithms, and graph theory. The main components and principles associated with Constraint Programming are [123]:

- Variables: a set of decision variables, each with values from a predefined domain.
- Domains: the range of potential values a variable may adopt. Variables receive values from their respective domains during the problem-solving process.
- Constraints: relationships or limitations between variables, expressing conditions that must be met for a solution to be considered valid.
- Objective Function (Optional): in optimization scenarios, an objective function minimizes or maximizes some objective. Constraint Programming encompasses both constraint satisfaction problems (seeking any valid solution) and optimization problems (identifying the optimal solution based on specific criteria).
- Search: CP typically involves a systematic search process exploring the solution space. The algorithm seeks to assess various assignments of values to variables, guided by the constraints, until a valid or optimal solution is attained.
- Propagation: a crucial step that utilizes constraints to enforce local consistency. This process involves removing inconsistencies within the search space, thereby eliminating values from variable domains that cannot contribute to a valid solution. This pruning increases search efficiency.
- Solver: specialized solvers are commonly employed to solve Constraint Programming problems. These solvers implement diverse algorithms for constraint propagation, search, and optimization.

Therefore, CP involves exploring the domains of variables, where constraints are utilized to establish connections between variables and identify solutions that meet all specified constraints. Efficiently narrowing down potential variable values is crucial for the effectiveness of CP. While CP is good at finding feasible solutions, it can face challenges when searching for optimal solutions, especially when refining or confirming them, which may take considerable time [136].

One of the central CP characteristics is flexibility in the constraints modeling that can be linear, nonlinear, or use logical operators. That is a significant advantage considering that certain combinatorial optimization problems are not easily linearized and solved with traditional mathematical programming methods. A tool like ILOG CP Optimizer provides a large set of arithmetic and logical constraints and a robust optimizer that brings all the benefits of a model-and-run development process to these problems [65].

The algorithms used in CPLEX and CP Optimizer are different, so experimentally

comparing them is interesting to test which is better adapted to our problem.

4.1.3 Constraint programming with a defined search strategy

The constraint programming method with a defined search strategy CPS uses the same model as in CP. The difference lies in the used search strategy. In CP, we use the IBM ILOG CP Optimizer standard tool, and in CPS, we specify how the search should be done and in what order. This method was applied only for the integrated model (PP + L +S).

To get a result with CPS, it is necessary to fix six decision variables:

- μ_{ij} : the **machine** processing operation O_{ij};
- χ_{ij} : the **configuration** of the machine used to process operation O_{ij};
- λ_{ij} : the **layout** used to process operation O_{ij};
- σ_{ij} : the **position** of operation O_{ij} in the process sequence (in terms of starting time) among the operations of J_i, $\sigma_{ij} \in \{1, ..., n_i\}$;
- ϕ_{ij} : the **total position** of operation O_{ij} in the process sequence (in terms of starting time) among all the operations, $\phi_{ij} \in \{1, ..., \sum_{i=0}^{n} n_i\}$;
- S_{ij} : the starting time of operation O_{ij}.

When these decision variables are fixed with CPS we can arrive at a result. For this, the strategy developed was the following:

1. We simultaneously fix the machines and configurations (μ_{ij} and χ_{ij}) used for each operation, starting with the shortest processing time to the longest.
2. Then, we fix the layout (λ_{ij}) of each job, starting with their initial layout, which is always the first index ($l = 1$), and then in the increasing order of the indices.
3. Next, we fix each operation's position (σ_{ij}) concerning the same job, starting with the smallest possible position for each.
4. After, we fix the positions concerning all operations (ϕ_{ij}), respecting the previous step and fixing first the operations with the lowest due date.
5. Finally, we fix the start time to the minimum possible time, respecting the total position.

The following small example illustrates how it works. Consider two jobs with two operations each, two machines with two configurations each, and two layouts. Table 18

shows the processing times of the operations on each machine/configuration, and Table 19 shows the due date of J_1 and J_2.

		$\mathbf{M_1}$		$\mathbf{M_2}$	
		$\mathbf{M_1[1]}$	$\mathbf{M_1[2]}$	$\mathbf{M_2[1]}$	$\mathbf{M_2[2]}$
$\mathbf{J_1}$	$\mathbf{O_{11}}$	10	∞	7	5
	$\mathbf{O_{12}}$	∞	5	∞	6
$\mathbf{J_2}$	$\mathbf{O_{21}}$	∞	∞	∞	12
	$\mathbf{O_{22}}$	∞	∞	∞	∞

Table 18: Operation time (time unit).

	Due date
$\mathbf{J_1}$	43
$\mathbf{J_2}$	24

Table 19: Job's due date.

First, the machine/configuration of each operation (O_{11}, O_{12}, O_{21}, and O_{22}) are fixed. At each decision point of the decision tree, two options are possible: to fix the variable to a specific machine/configuration or to reject it (excluding the machine/configuration from the set of possibilities). Figure 21 shows a decision point where these two choices are illustrated, and where the branching continues for both. In the machine/configuration case, the value taken is the shortest processing time in the set of possibilities. For example, if we analyze O_{11}, the initial set of possibilities contains $M_1[1]$, $M_2[1]$, and $M_2[2]$, with the shortest processing time in $M_2[2]$. The search order choice is made by trying to find the most promising possibilities from the beginning. Of course, a machine/configuration with a shorter processing time has more chances to be chosen in the final result. In addition, the selection of the machine and the configuration are made together because this choice is interconnected according to the processing time.

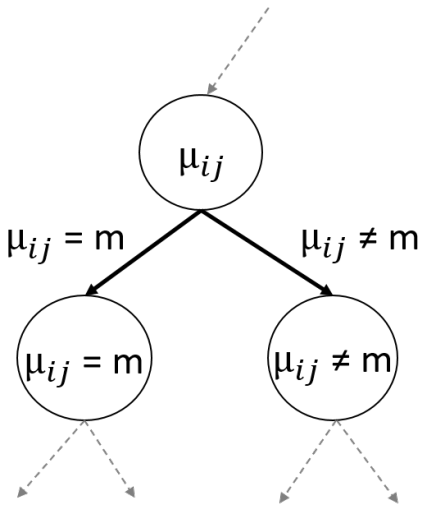

Figure 21: Decision point representation.

Then, the layout of each job (J_1 and J_2) is fixed. At this decision point, we fix the variable to a specific value or we reject it (excluding the value from the set of possibilities). The first tried value is the job's initial layout ($l = 1$). When the initial layout is not in the set of possibilities, we take the one with the lowest index. For our example, we set $\lambda_{11} = 1$; otherwise, $\lambda_{11} = 2$. Choosing the initial layout first seems to be the best strategy because the layout reconfiguration incurs high costs that are not interesting in most cases. Then, it is probably better to continue on the same layout.

Next, the operation position among the operations of each job is fixed for each operation (O_{11}, O_{12}, O_{21}, and O_{22} in the example). At the decision point, we fix the variable to a specific value or we reject it (excluding the value from the set of possibilities). We chose the position with the smallest index. For example, if no precedence constraint is associated with job 1, O_{11} can be sequenced in position 1 or 2 (even for O_{12}). Then, we will first fix $\sigma_{11} = 1$ and then $\sigma_{11} = 2$. This time, the choice is arbitrary because we have no element to decide the most suitable or promising order.

Then, the positions of each operation, among all operations, are fixed. We sequence first the operations with the smallest due date. In our example, the sequence order would be O_{21}, O_{22}, O_{11}, and O_{12}. At the decision point, we fix the variable to a specific value or we reject it (excluding the value from the set of possibilities). Once again, we chose to sequence the operation with the lowest possible index of the set of possibilities for each operation. In the previous step, we have already defined the position in relation to the operations of the same job. Now, we must set it in relation to all the operations. There is no precedence restriction between the operations of different jobs, but it must respect what was stipulated previously. For example, if the first operation to be fixed (O_{21}) has a $\sigma_{21} = 2$, then in our example, ϕ_{21} will first be tested as $\phi_{21} = 2$, then $\phi_{21} = 3$, and finally $\phi_{21} = 4$. This choice was made because there is a higher probability that jobs with a lower due date will have their operations processed earlier than those with a higher one.

Finally, we fix the starting times of each operation. The order in which the decisions are taken is based on the total position. For example, if we are in a node with a total position equal to [O_{22}, O_{11}, O_{12}, O_{21}], then it is the order in which we will fix the starting time of the operations. Since all the other variables have already been set, and only the starting times need to be fixed, semi-active scheduling must lead us to an optimal solution. A semi-active scheduling is when operations are scheduled the earliest, avoiding any idle time [68]. We have a dominance rule that reduces the solution space [68], reducing our variable domains to the lowest value (earliest time). Consequently, we set the shortest starting time without needing a decision point.

The chosen strategy seemed the most promising. While adjustments, such as altering

the sequence in which decision variables are set, are possible. However, our approach finds optimal solutions respecting dominance rules. Consequently, the starting time is always the last variable to be fixed (no decision point). Otherwise, the starting time should be seen as a decision point, which significantly increases computational time.

4.1.4 Conclusions

In this work, three exact solving methods are employed. The first is the ILP, where we need to linearize the model from Chapter 2. We chose to solve it using commercial software, which uses various algorithms, mainly the simplex algorithm. While widely used in combinatorial optimization problems, this method tends to be time-consuming for large combinatorial problems like the one at hand. For this reason, we decided to explore alternative exact methods. This is particularly noteworthy as many works in this domain often skip exact methods, going directly to non-exact approaches.

The second method is Constraint Programming (CP), which explores variable domains using constraints to identify solutions that satisfy all conditions. Although traditionally employed to find possible solutions, CP can also be adapted for optimization. However, this objective may raise time execution due to the difficulty of proving the optimality of a solution.

To enhance the efficiency of CP, we conducted a detailed study, developing a search strategy explicitly adapted to our problem. This strategy is designed to be more efficient than the general approaches used by the software used for CP. The third method is the most promising. More concrete conclusions will be drawn in the next chapter.

4.2 Metaheuristics

With exact methods, we are sure of the optimal answer to a problem. However, they are not adapted for solving real instances- of $\mathcal{NP}$-hard problems since they are very time-consuming. Therefore, other alternatives are needed, such as metaheuristics.

Metaheuristics are solution methods that coordinate local improvement procedures with higher-level strategies to create a strategy capable of escaping local optima and performing a robust search in the solution space of a problem [116]. They can deal with large instances, delivering satisfactory solutions in a reasonable time.

In this work, we study a huge combinatorial problem, so using approaches other than the exact ones is necessary. There are a wide variety of possibilities within this category of methods. We decided to work with the genetic algorithm and a version of it with local

search. This choice was made because its efficiency was proven in several works in our field, e.g., in [1, 11, 38, 148].

Although the genetic algorithm is a generic method, it can be adapted to a specific problem. In the remainder, we explain in detail how we used GA and GA with local search to solve our integrated problem (PP + L + S).

4.2.1 Genetic algorithm

GA is a particle class of Evolutionary Algorithms (EA) based on Darwin's theory of evolution, which states that a population of individuals in a given environment will always seek to adapt in the best possible way. Thus, more adapted individuals are generated in each new generation via inheritance, mutation, selection, and crossover [82].

In the development of the genetic algorithm, an initial population is created, comprising individuals (also known as chromosomes) that represent potential solutions. Each individual is composed of several genes representing the decision variables. The fitness function assesses the quality of the solution, enabling the evaluation of the most promising individuals. After this, the algorithm makes a partial or complete selection (depending on the reproduction rate) for reproduction. Pairs of chromosomes are selected to reproduce and create offspring. Reproduction can happen by applying genetic operators, such as crossover, mutation, and/or selection. At the end of reproduction, a new generation is formed, and population control is usually done so that the populations will always have the same size. Creating new generations continues until a stopping criterion is reached [85, 100].

Figure 22 resumes our GA steps. First, an initial population is created. After, the individuals are evaluated using the fitness function. Then, we verify if the population is bigger than the initial population. If the answer is positive, the algorithm makes a population control, keeping only the n best individuals, n being the number of individuals in the initial population. After this control, it is verified if the stop criterion (the maximum number of iterations) is reached. If the answer is positive, the algorithm stops. Otherwise, after the stop criterion verification, we apply the genetic operators. Alternatively, we can arrive at this same point directly after the size population verification if the population is not bigger than the initial population. We apply the crossover operator first and then the mutation operator. After this, the number of iterations is incremented, and we loop back to the fitness function evaluation of the individuals.

In the following, some concepts will be discussed in more detail, explaining the genetic algorithm.

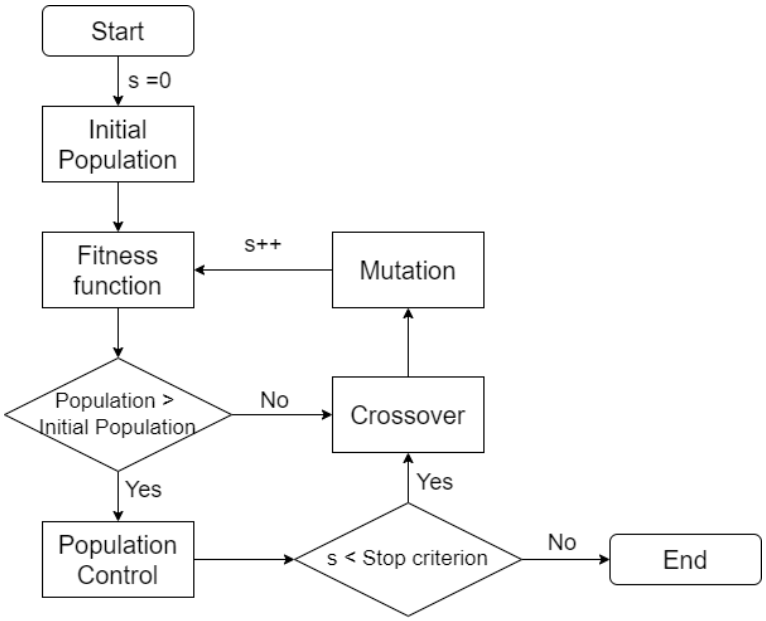

Figure 22: Genetic algorithm flowchart.

Encoding

The chromosome is a sequence of values that, when decoded, represents a solution to one problem. The size, the numerical representation, and the other characteristics of the chromosomes depend directly on the problem studied. In this work, a chromosome contains four fields for each job operation. Figure 23 shows that the first part is dedicated to the information regarding the machines used for each operation. The second part is dedicated to the machines' configurations to process operations. The third part is dedicated to the layouts. Finally, the fourth part is dedicated to the position of each operation in the schedule. All these variables were introduced previously in the subsection "Constraint programming with a defined search strategy".

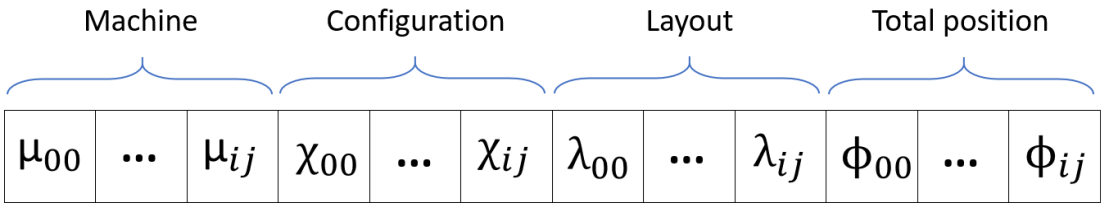

Figure 23: Chromosome representation.

Therefore, the number of chromosome genes (n_g) is equal to 4 multiplied by the total number of operations (4.1). For example, in a problem with two jobs, where the first job has three operations and the second one has two operations, the chromosome will be a vector of 20 values (4*(3+2)).

$$n_g = 4 \times \sum_{i=0}^{n} n_i \tag{4.1}$$

While in the section "Constraint programming with a defined search strategy", we used six variables, now in this method, we are only using four, not having to consider the operation's position among operations in the same job and the starting time variables. The position within the job is not required because we already have the total position among all operations (of all jobs) that gives the necessary information. The starting time is calculated to be the smallest possible, considering the information during the decoding processes. More details are provided in the following subsections.

All our chromosomes represent feasible solutions because the machine and configuration can only be chosen from the possible options for each operation, all jobs can be processed in all layouts, and the operations' position respects the precedence constraints.

Initialization

The initial population is often randomly generated. However, in some cases, it can be oriented to probable optimal solutions or randomly generated using techniques to increase the diversification in the search space [82, 100]. In this work, the initial population is entirely randomly generated.

In addition, a crucial parameter is the number of individuals belonging to the initial population. If this number is too small, the search space coverage may be small, which may cause low-quality solutions. On the other hand, if the population is too large, the algorithm may take too long in its execution time [69]. The number also depends on the size of the instance being treated. We utilized a factorial design of experiments technique to determine the initial population size and other GA input parameters like the number of generations, crossover rate, and mutation rate. The details of this technique and results will be explained in the next chapter.

Fitness function

A fitness function allows the performance of each individual to be evaluated. In our case, we generate a valid schedule from the chromosome information by fixing the start times like in the exact method using a greedy algorithm (Algorithm 3). The fitness is then equal to the objective function of this schedule. Since our problem aims to minimize total costs, the smaller the fitness function, the better the solution.

Algorithm 3 Greedy algorithm for our scheduling

1: **while** $a < sequence.size()$ **do**
2: **if** $a = 0$ **then**
3: $op \leftarrow sequence[a]$
4: **if** $layout[l_o] \neq layout[op]$ **then**
5: **for** all machines **do**
6: $time_m[machine] \leftarrow RL_{time}$
7: **if** $configuration[machine[c_o]] \neq configuration[machine[op]]$ **then**
8: $time_m[machine[op]] \leftarrow time_m[machine[op]] + RC_{time}[machine[op]]$
9: $Set\_time(op, op)$
10: **else if** $a > 0$ **then**
11: $op_m \leftarrow$ last operation in $machine[sequence[a]]$
12: $op_j \leftarrow$ last operation in $job[sequence[a]]$
13: $op_p \leftarrow sequence[a-1]$ $op_c \leftarrow sequence[a]$
14: $RL = RM = MH \leftarrow 0$
15: $Check\_actions(op_c, op_p, op_m, op_j)$
16: **switch** (RL, RM, MH) **do**
17: **case 1** $= (0,0,0)$ $Set\_time(op_c, op_p, op_m)$
18: **case 2** $= (1,0,0)$ $Layout\_reconfiguration(op_c, op_p, op_m)$
19: **case 3** $= (0,1,0)$ $Machine\_reconfiguration(op_c, op_p, op_m)$
20: **case 4** $= (0,0,1)$ $Material\_handling(op_c, op_p, op_m, op_j)$
21: **case 5** $= (1,1,0)$
22: **if** RM cannot be done between op_m and op_c **then**
23: $time_m[machine[op_c]] \leftarrow time_m[machine[op_c]] + RM_{time}[machine[op_c]]$
24: $Layout\_reconfiguration(op_c, op_p, op_m)$
25: **case 6** $= (1,0,1)$
26: $Layout\_reconfiguration(op_c, op_p, op_m)$
27: $Material\_handling(op_c, op_p, op_m, op_j)$
28: **case 7** $= (0,1,1)$
29: $Machine\_reconfiguration(op_c, op_p, op_m)$
30: $Material\_handling(op_c, op_p, op_m, op_j)$
31: **case 8** $= (1,1,1)$
32: **if** RM cannot be done between op_m and op_c **then**
33: $time_m[machine[op_c]] \leftarrow time_m[machine[op_c]] + RM_{time}[machine[op_c]]$
34: $Layout\_reconfiguration(op_c, op_p, op_m)$
35: $Material\_handling(op_c, op_p, op_m, op_j)$
36: **end switch**
37: **end while**

As input data, which machines/configuration, layout, and production sequence (total position) of operations are provided. The sequence vector orders the operations according to their start time, i.e., if O_{23} comes before O_{11} in the sequence, then $s[O_{23}] \leq s[O_{11}]$. All operations must be sequenced. The first operation of the sequence (Algorithm 3: lines

2-10) is fixed as follows:

- Check the layout reconfiguration: if the layout of the operation differs from the initial layout (Algorithm 3: line 4), the reconfiguration time is counted before any operation is started.
- Check the machine reconfiguration: if the configuration used for the operation differs from the initial configuration of the machine in question (Algorithm 3: line 9), the machine reconfiguration time is counted before the operation is started.

After these two examinations, the start and completion times of the operations can be set (Function Set_time). There is no possibility of material handling for the first operation since there is surely no preceding operation of the same job that has started to be produced. For the other operations, it is important to retain the current operation (op_c), the preceding operation in the sequence (op_p), the last operation that was done on the same machine as the current operation (op_m), and the last operation produced from the same job as the current operation (op_j) (Algorithm 3: lines 14-16). Three auxiliary variables are used to check if it is necessary layout reconfiguration (RL), machine reconfiguration (RM), and/or material handling (MH) between the current operation and operations already fixed. For this, function $Check\_actions$ is used. We have eight possibilities once we know which actions must be taken to sequence the current operation. The first (Algorithm 3: line 20) is in the case where there is no reconfiguration or material handling. The starting time is directly set to the first availability of the machine processing the operation using the Set_time function. The second to fourth cases are when only one of the actions is required (Algorithm 3: lines 21-23). In case 2, there is a layout reconfiguration, so the $Layout\_reconfiguration$ function will first set the availability of all the machines (after the layout reconfiguration is finished) and then set the operation using the Set_time function. In case 3, the reconfiguration of the machine processing the operation in question must be considered. Using the $Machine\_reconfiguration$ function, we first check whether there is enough time for reconfiguration between the last operation processed by the machine (op_m) and the lowest time that the current operation can start (op_c). We can set the position directly if there is enough time for RM. Otherwise, we need to count the RM time and then set the position. In case 4, the only action to be considered is material handling. As in the previous case, we check whether there is enough time for the MH between the last operation of the same job that has already been scheduled and the current operation; if so, we fix the operation directly. If not, we first count the MH and then fix the operation. The other cases are a combination of RL, RM, and MH (Algorithm 3: lines 24-40). Each of them will first consider the necessary actions and then set up the operation using the functions already mentioned.

Algorithm 4 $Set\_time(op_c, op_p)$

1: $s[op_c] \leftarrow time_m[machine[op_c]]$
2: **if** $s[op_c] < s[op_p]$ **then**
3: $\quad s[op_c] \leftarrow s[op_p]$
4: $c[op_c] \leftarrow s[op_c] + Production_{time}[op_c]$
5: $time_m[machine[op_c]] \leftarrow c[op_c]$

Algorithm 5 $Check\_actions(op_c, op_p, op_m, op_j)$

1: **if** $layout[op_p] \neq layout[op_c]$ **then**
2: $\quad RL \leftarrow 1$
3: **if** $configuration[op_m] \neq configuration[op_c]$ **then**
4: $\quad RM \leftarrow 1$
5: **if** $op_c \neq$ fist operation of job$[op_c]$ **then**
6: $\quad$ **if** machine$[op_j] \neq machine[op_c]$ **then**
7: $\quad\quad MH \leftarrow 1$

Algorithm 6 $Layout\_reconfiguration(op_c, op_p, op_m)$

1: $c_{max} \leftarrow \max(time_m[machine])$
2: **for** all machines **do**
3: $\quad time_m[machine] \leftarrow c_{max} + RL_{time}$
4: $Set\_time(op_c, op_p)$

Algorithm 7 $Machine\_reconfiguration(op_c, op_p, op_m)$

1: **if** RM can be done between op_m and op_c **then**
2: $\quad Set\_time(op_c, op_p)$
3: **else if** RM cannot be done between op_m and op_c **then**
4: $\quad s[op_c] \leftarrow time_m[machine[op_c]] + RM_{time}[machine[op_c]]$
5: $\quad c[op_c] \leftarrow s[op_c] + Production_{time}[op_c]$
6: $\quad time_m[machine[op_c]] = c[op_c]$

Algorithm 8 $Material\_handling(op_c, op_p, op_m, op_j)$

1: **if** $time_m[machine[op_c]] - c[op_j] \geq MH_{time}[machine[op_j] \rightarrow machine[op_c]]$ **then**
2: $\quad Set\_time(op_c, op_p, op_m)$
3: **else if** RM cannot be done between op_m and op_c **then**
4: $\quad s[op_c] \leftarrow c[op_j] + MH_{time}[machine[op_j] \rightarrow machine[op_c]]$
5: $\quad c[op_c] \leftarrow s[op_c] + Production_{time}[op_c]$
6: $\quad time_m[machine[op_c]] \leftarrow c[op_c]$

Population control

The total population increases each time a new generation is formed after the crossover and mutation. Without population control, the algorithm can become slow. Therefore, all individuals are sorted in non-decreasing order according to their fitness after the offspring are generated. The best chromosomes stay in the population, and the worst ones are excluded. The amount of remaining individuals is the same as the initial population parameter.

Genetic operators

The genetic operators are the tools to generate new populations (reproduction). In the implemented GA algorithm, we use crossover and mutation operators.

Crossover

In this operation, two individuals are chosen to give part of their genes (representation of each decision variable) to their offspring. The crossover can be done in several manners. The single-point crossover is the most common, as shown in Figure 24. One point in both chromosomes is selected to divide the chromosome into two parts [100]. The parents' genetic material is combined: the first part of the first parent is combined with the second part of the second parent, and the second part of the first parent with the first part of the second parent, generating the offspring (two children).

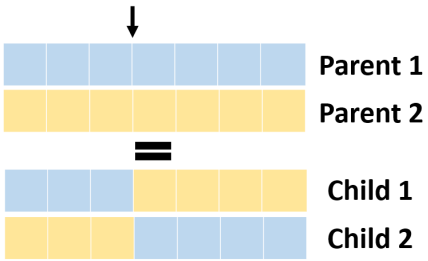

Figure 24: Single-point crossover.

After a population generation, chromosomes are sorted by the fitness function, from the best solution (lowest fitness) to the worst (highest fitness). A crossover rate must be defined, i.e., the percentage of best chromosomes that will crossover. If the crossover rate is, for example, 80%, then the best 80% of individuals will be used as parents. The parents' pair choice is made by the roulette wheel method [101]. Consequently, the lower the fitness (better result), the higher the probability that an individual will be selected first. Whenever a chromosome is chosen, the probabilities used for the roulette wheel method are recalculated without counting the already chosen chromosomes.

The single-point crossover was used as inspiration for our crossover method. Some adaptations were necessary since our goal is to generate only individuals with feasible solutions. As the "encoding" section explains, our chromosome comprises four parts for each operation, corresponding to machine (μ_{ij}), configuration (χ_{ij}), layout (λ_{ij}), and total position (ϕ_y). In Figure 25, we represent a possible chromosome with the abovementioned parts.

After the pairs of chromosomes to be used for the crossover have been chosen, we randomly choose a value $r \in \{0, ..., y\}$, where y is the number total of operations. r is the point where the machine, configuration, and layout parts will be cut for the permutation. Nevertheless, this logic does not work for the total position since it can generate infeasible solutions because of the operations' precedence order.

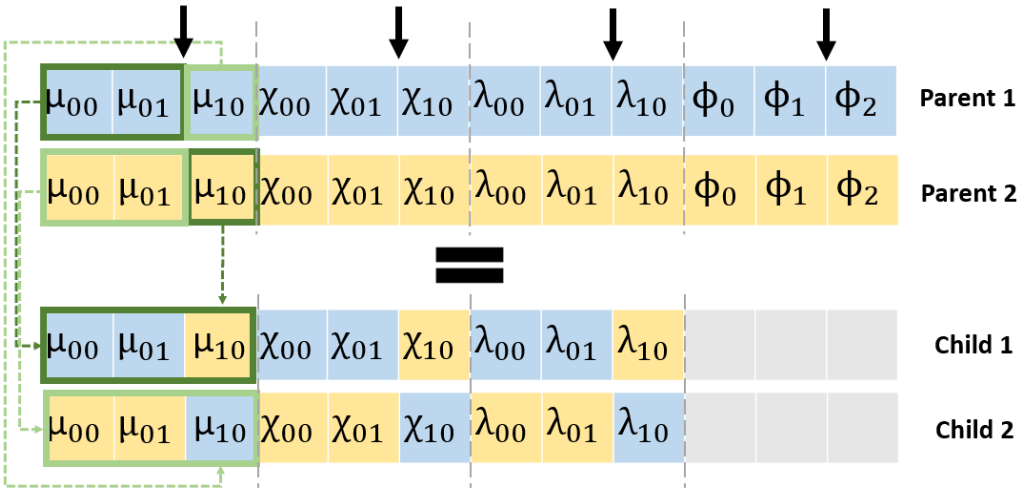

Figure 25: Crossover.

A conversion is done for the total position part to make it clearer. The term "total sequence" (Ψ_y) is introduced, where $y \in \{1, ..., \sum_{i=0}^{n} n_i\}$. For the total position, we have the position p of the operation O_{ij} among all the operations ($\phi_{ij} = p$), and for the total sequence, we have operation O_{ij} that is in the position p ($\Psi_p = O_{ij}$). We use the total sequence and take the same point chosen for the other variables. After, we copy the first part of Parent 1 in Child 1 and the first part of Parent 2 in Child 2. However, the second part of the parents cannot be directly copied to avoid non-feasible solutions caused by a possible non-respect of the precedence constraints or duplicate operations. Then, Child 1 will follow the same order as Parent 2, excluding the operations already sequenced (the first part coming from Parent 1). The same occurs for the second part of Child 2, which follows the same order as Parent 1, excluding the already sequenced operations. Figure 26 shows this, using an example with two jobs (each with three operations). If the crossover point is $r = 2$, the first child will have the first two operations from Parent 1 (operations

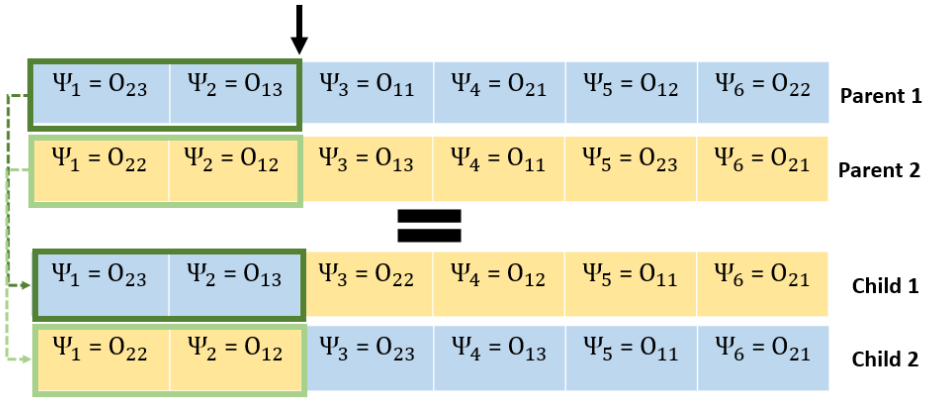

Figure 26: Crossover sequence.

O_{23} and O_{13}) and the rest from Parent 2, following the original sequence excluding the operations already sequenced (operations O_{22}, O_{12}, O_{11}, and O_{21}). The same happens with Child 2. This permutation ensures that no operation is repeated and that the precedence constraints are necessarily respected since the parents are feasible solutions.

Mutation

A mutation modifies one (or more) chromosome genes to increase the population's diversity and avoid the converging of the algorithm8 too early to a local optimum. Nevertheless, the mutation rate can not be too high, so the search does not become random [100]. In our case, the mutation rate indicates how many percent of the population will be mutated. Chromosomes can be mutated in four different ways, and this choice is made randomly:

- Mutation 1: a random gene is selected to change the machine for another one randomly chosen. The configuration remains the same if it is possible (feasible solution). Otherwise, a possible configuration is also randomly chosen.
- Mutation 2: a random gene is selected to change only its configuration for another one randomly chosen.
- Mutation 3: a random gene is selected to change only its layout for another one randomly chosen.
- Mutation 4: a random gene selected to invert its sequence value with the subsequent gene. If the precedence order does not allow it (generating an unfeasible solution), the algorithm continues searching for a subsequent gene that allows the inversion.

Stopping criterion

The stopping criterion can be defined by the number of generations created, execution time, or solution quality. Since evaluating the quality of solutions is not possible, there are only the other two options. Stop the algorithm according to the number of generations was the one chosen. This information is given as a problem input parameter.

Thus, all steps of the genetic algorithm were covered. The following section will analyze numerical examples using all the resolution methods presented in this work.

4.2.2 Genetic algorithm with local search

Local search is a strategic algorithm for optimizing problems, focusing on refining a single candidate solution by iteratively making small changes to it. Designing a local search algorithm necessitates defining a structure among feasible solutions, often established by considering solutions as neighbors if they can be transformed into each other through small changes known as local transformations [63].

This technique is frequently employed with metaheuristics to enhance the results of complex problems. In our case, we integrated local search into the genetic algorithm developed in the previous section.

During the algorithm execution, some individuals are chosen from the population to undergo the local search process. This search occurs within a small part of the chromosome, the size of which is specified as input to the problem. The local search can be conducted in three distinct parts for each individual: machine and configuration, sequence, and layout. The following sub-sections explain each of them.

Machine and configuration

Initially, we must choose a chromosome and determine the part size for the local search, which is an input to the problem. Subsequently, an operation is randomly selected as the starting point of the part for the local search. The ending point is calculated by the starting point plus the part size. For each of these operations within the specified part, we systematically explore all possible machine/configuration combinations.

For instance, let us consider a chromosome comprising 10 jobs, each job having 5 operations, and we set the input size to 5. We randomly designate the starting operation for analysis as Op_{34}. Consequently, we examine it and its consecutive operations Op_{35}, Op_{41}, Op_{42}, and Op_{43}. Figure 27 shows the machine and configuration of each of the operations where the local search will be carried out.

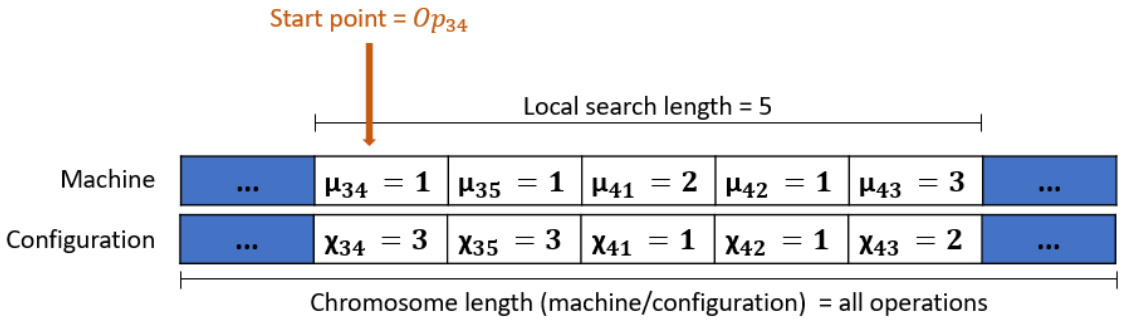

Figure 27: Machine/configuration local search.

Assuming that each of these operations has the following machine/configuration possibilities:

- Op_{34} : $M_1[3]$ and $M_2[1]$;
- Op_{35} : $M_1[3]$, $M_2[2]$, and $M_3[2]$;
- Op_{41} : $M_2[1]$ and $M_3[3]$;
- Op_{42} : $M_1[1]$, $M_2[1]$, $M_3[1]$, and $M_4[2]$;
- Op_{43} : $M_3[2]$, $M_4[1]$;

In this scenario, there are 96 possibilities resulting from the small change in the machine/configuration of these 5 operations. We evaluated the fitness function for each of them and retained the best one.

Sequence

The way we have chosen to conduct the local search for the sequence involves randomly selecting a point on the chromosome and extracting a segment from that point. The size of the segment where the local search will be executed is determined by an input parameter. Figure 28 provides an illustration of an example where the chromosome segment contains the sequence information, indicating the starting point (position 14) for the local search and the size of this segment (5). Unaltered sections where the local search will not be applied are not specified since they will remain unchanged. The values within the local search segment contain the existing values before the local search.

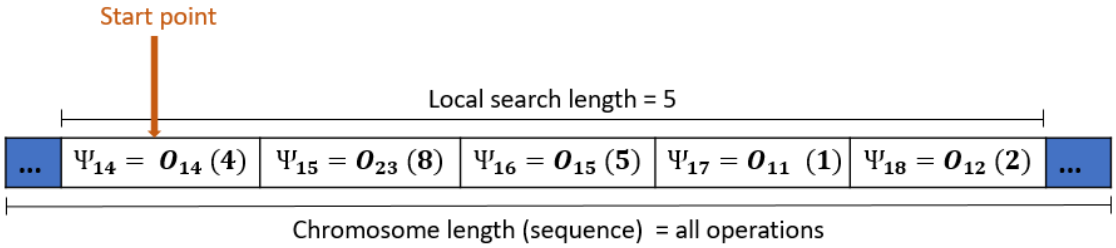

Figure 28: Sequence local search.

Taking the values of the part selected for the local search, a tree can be constructed with all possible combinations of sequences. Each path from the root to a leaf in the tree represents a potential sequence for the selected part. Only the feasible paths, i.e., respecting precedence constraints, are generated. In other words, when constructing the tree, if the order of an operation violates precedence constraints, it is not included, and consequently, all possible children are not added to the tree. Returning to the initial example, assuming that 1 precedes 2 and 8 precedes 5, we end up with Figure 29. In this figure, all paths that are not feasible have been excluded. Therefore, the root cannot have 2 and 5 as children, and there is no path that allows 1 after 2 or 8 after 5.

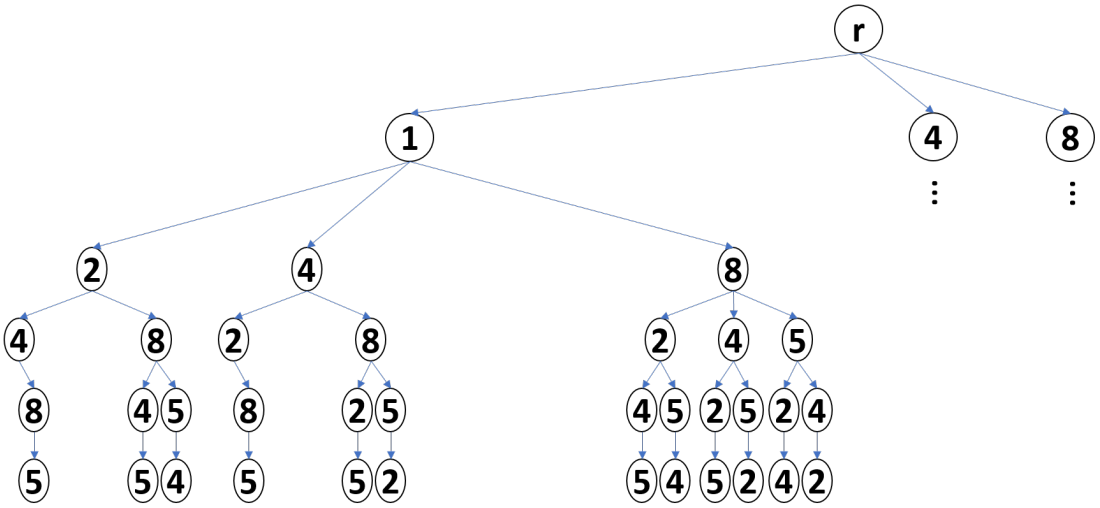

Figure 29: Generated sequences for the LS.

Layout

In the context of layout considerations, we observed that for the majority of the problems we addressed, the most impactful reconfiguration occurs at the beginning of the scheduling when reconfiguring from the initial layout to another. While other layout reconfigurations may occur during scheduling in some instances, they typically do not yield significant improvements in the results.

Therefore, we opted to restrict the local layout search only to the same layout for all jobs. This involves testing all jobs within the same layout for the selected chromosome, and we repeat this process for all possible layouts within the predefined set.

In this case, the part size is not used as the local search encompasses the entire layout. The fitness functions are analyzed, and the best result is retained.

Integrating LS and GA

Local search can be integrated into GA in various ways and at different stages. We have chosen to incorporate it at the beginning of the algorithm, during the execution of generations, and at the end. Figure 30 illustrates how the GA algorithm appears with LS. The distinction from GA alone is that immediately after creating the initial population, we apply local search to enhance the initial population. During the algorithm's progress, local search can be utilized; in our case, we incorporate it at specific intervals, such as every 50 or 100 generations. Finally, local search is applied at the end to try to further refine the solution.

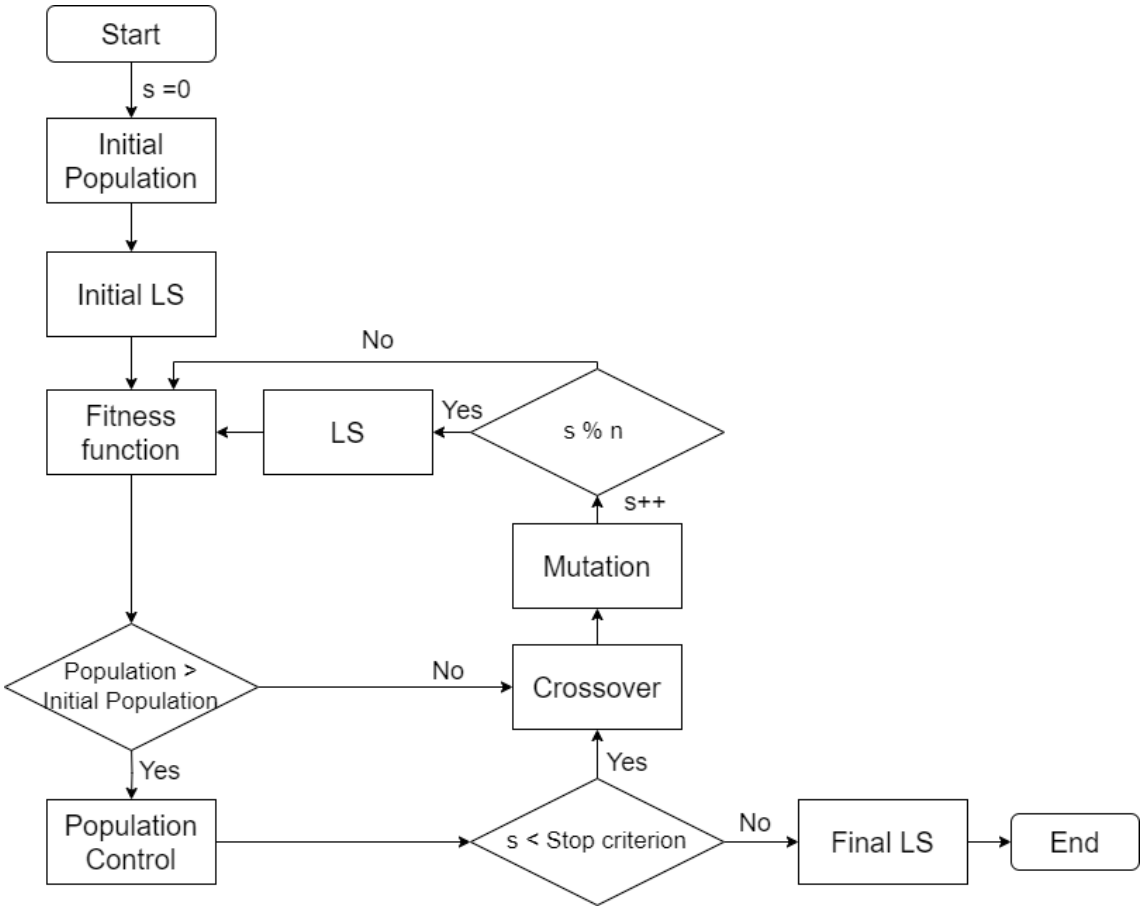

Figure 30: Genetic algorithm with local search flowchart.

The decision for an initial local search was made because initial populations generally

have extremely high fitness, which takes a few generations to reach more reasonable values. To begin with a more adapted population, we applied all three types of LS (machine/configuration, sequence, and layout). For machine/configuration and sequence, it is applied to a part of the operations of each individual (the size of this part being an input). For layout, it is applied to all operations, as explained above, also for each individual.

The local search that takes place during the evolution of generations is applied only to the best individual, and another one is chosen at random. All three types of LS are applied. This decision was made to try to enhance the best individual and introduce more variability (applying LS to a random chromosome). The latter is because, as explained in the upcoming chapters, the mutation rate without local search is high, likely used to boost variability and avoid getting trapped in local optimal solutions.

The final local search is applied only to the best individual aiming to improve it further. In this case, we decided to apply it multiple times for machine/configuration and sequence. Generally, it is applied a number of times equal to the number of jobs, but this can be considered as another input. The starting point is still random but is distributed over the chromosome to cover an even larger part. In other words, if we apply it x times, the starting point for each LS iteration will be within an interval between $[i \times total_{op}, i \times total_{op} + x]$, where $0 \leq i < x$, and $total_{op}$ is the total number of operations.

4.2.3 Conclusions

In the search for more robust methods to address our problem, it became imperative to explore non-exact methods. We opted to develop a Genetic Algorithm (GA) developed to our specific needs, ensuring that all generated individuals are always feasible. This adaptation focused particularly on the crossover method, adjusting the part that considers the sequence of operations to find viable solutions. Additionally, to calculate the fitness function, we developed a greedy algorithm.

In pursuit of an even more promising approach, we introduced a second method integrating local search into the GA. Once again, special attention was given to the sequence of operations to ensure that local search efficiently explores feasible solutions without wasting time. A proposed algorithm outlines how our local search is executed. This proposal involves applying the LS in specific generations on both the best individual and one randomly chosen individual. This choice aims to enhance the current best individual and potentially escape local optima by exploring new possibilities in further generations.

4.3 Conclusions of the chapter

This chapter presented the solving methods developed and applied in this work. Using at least one exact method is essential to examine and validate the model's behavior. Moreover, it helps in the first analysis of non-exact methods since it serves as a comparison parameter for small instances. We decided to go beyond a single exact method and look for other alternatives with the potential to improve its performance. This choice was motivated by three main reasons. First, our combinatorial problem is huge, so even for small instances, the ILP is very time-consuming to get good solutions (execution time analysis made in the next chapter). Besides, constraint programming has been considered an efficient way to solve scheduling problems, so it would be interesting to see its performance in our problem. Finally, they may have other applications, such as evaluating the metaheuristics' fitness function or using these techniques in hybrid methods that could be developed (which was not the case, but it was a possible clue at the beginning of the thesis).

As seen in the literature review, metaheuristics are frequently used to solve problems in RMS, mainly for process planning and scheduling. Despite the wide variety of metaheuristics, most follow the same logic: to generate an initial random population, then create new generations from the combination of the previous ones and, during the process, add some variability to avoid local optimums. For this reason, it was considered more important to focus on the form of implementation, adapting it to the problem being solved, than the metaheuristic itself. We decided to use the genetic algorithm because its quality has been demonstrated in several works. An essential point of the implementation of GA, shown in this chapter, is that only feasible individuals are generated, which for example, was a difficulty for Gao *et al.* [41] to deal with the conflicts in the solutions.

Although metaheuristics are efficient, there is place for improvement. One way to achieve this is hybridizing with other techniques. In our case, we sought to enhance our GA by incorporating local searches. It is important to note that local search is a "generic" term for exploring a part of the solution space, and, in our strategy, we have detailed a specific approach for our algorithm.

The most significant difference between what has been presented in this chapter and what we find in the literature is the fact that we do not limit ourselves to an exact solution method, and especially when we use CP, we explore more than the standard tools that the chosen solving software offers, creating a strategy specially adapted to our problem. Similarly, while Genetic Algorithms (GA) and Local Search (LS) are commonly employed methods, we have developed custom-fit algorithms for the nuances of the problem under study. Nevertheless, it is worth noting that there is the possibility of refining even more these algorithms in the future.

In the following two chapters, we will examine the performance of all the methods proposed in this chapter, highlighting their respective advantages and disadvantages.

Chapter 5

Numerical Results

Contents

In this chapter, the models and methods presented in this thesis are numerically tested. First, the four models introduced in Chapter 2 are compared. The integrated one will always obtain the optimal value (using an exact method), and the others can be as good in the best case. With the examples shown below, it is possible to highlight the superiority of the integrated version. For this reason, only the integrated model was considered for the following. In the second part, the three exact methods are compared. Also, the metaheuristics are compared (with the optimal value and between them). Finally, a general conclusion is drawn from all the analyses.

A set of 20 small instances are used throughout this chapter. They were created with jobs, operations, machines, configurations, and layouts between 2 and 3 units. The other data, such as times, costs, and precedence order, were randomly generated. Table 20 presents their main characteristics. The instances are in the first column (Instance). The number of jobs (NJ) for each instance is in the second column. Then, the number

of operations per job (NO). Next is the number of machines (NM). Then, the number of configurations per machine (NC). The last column includes the number of possible layouts (layout set) (NL). All small instances were solved using a laptop computer (16 GB RAM, Intel(R) Core(TM) i7-8665U CPU @ 1.90GHz 2.11 GHz). The big instances used the university's server (80 core, 512 Gb RAM, Linux CentOS) since they require a much larger memory allocation than a laptop offers.

Instance	NJ	NO	NM	NC	NL
1	2	2,2	2	2,2	2
2	2	2,2	2	2,2	2
3	2	3,2	2	2,2	2
4	2	3,2	2	2,2	2
5	2	2,2	3	2,2,2	2
6	2	2,2	2	2,2	3
7	2	2,2	3	2,2,2	2
8	2	2,2	2	2,2	3
9	3	2,2,2	3	2,2,2	2
10	3	2,2,2	3	2,2,2	2
11	3	3,2,2	3	2,2,2	2
12	3	3,2,2	3	2,2,2	2
13	3	3,2,2	3	3,2,2	2
14	3	3,2,2	3	3,2,2	2
15	3	3,3,3	3	3,3,3	2
16	3	3,3,3	3	3,3,3	2
17	2	3,2	3	3,3,2	3
18	2	3,2	3	3,3,2	3
19	2	3,2	2	3,2	3
20	2	3,2	2	3,2	3

NJ: number of jobs; NO: number of operations per job; NM: number of machines; NC: number of configuration per machine; NL: number of layouts.

Table 20: Main characteristics of random instances.

5.1 Comparing the models: sequential, partial sequential, and integrated

In this subsection, the results of the four models presented above are compared, using Constraint Programming with a defined search strategy (CPS), for the instances presented in Table 20. Even before analyzing the results, it was already possible to know that the integrated model would be the best regarding costs since it treats all parameters together and does not use assumptions. The central question becomes how far the other models' results are from the optima and how many seconds they consume for the execution time.

Table 21 shows the results found. The first column lists the instances. The second

column to the fifth indicates the costs found using each model, in the following order: sequential, partially sequential integrating layout and scheduling, partially sequential integrating process planning and layout, and finally, the integrated (which is marked in green for always having the best results). The sixth to ninth columns show the execution times for each model, in the same order as the costs. The last three columns quantify the difference in percentage between the optimal cost and each sequential or partially sequential model ($\% = \frac{analyzed\ model - optimal}{optimal}$). The values marked in bold are the closest to the optimum. In some cases, the values for the three models are marked because they have the same percentage, for example, Instances 2 to 9.

	Cost				**Time(s)**				**% distance(optimum)**		
Instance	**S**	**PS1**	**PS2**	**I**	**S**	**PS1**	**PS2**	**I**	**S**	**PS1**	**PS2**
1	42	42	36	36	0	0	0	1	17%	17%	**0%**
2	41	41	41	39	0	0	0	1	**5%**	**5%**	**5%**
3	42	42	42	36	0	0	0	1	**17%**	**17%**	**17%**
4	67	67	67	57	0	0	0	1	**18%**	**18%**	**18%**
5	18	18	18	18	0	0	0	1	**0%**	**0%**	**0%**
6	52	52	52	38	0	0	0	1	**37%**	**37%**	**37%**
7	35	35	35	31	0	0	0	1	**13%**	**13%**	**13%**
8	61	61	61	57	0	0	0	1	**7%**	**7%**	**7%**
9	39	39	39	37	0	0	0	2	**5%**	**5%**	**5%**
10	44	44	49	42	0	0	1	10	**5%**	**5%**	17%
11	59	59	59	56	0	0	7	6	**5%**	**5%**	**5%**
12	58	58	56	52	0	1	5	19	12%	12%	**8%**
13	148	140	148	79	0	2	7	35	87%	**77%**	87%
14	76	76	91	70	0	0	2	8	**9%**	**9%**	30%
15	58	56	56	52	1	2	75	164	12%	**8%**	**8%**
16	95	90	125	73	0	0	166	3000+	30%	**23%**	71%
17	54	54	54	33	0	0	0	4	**64%**	**64%**	**64%**
18	35	35	36	35	0	0	0	2	**0%**	**0%**	3%
19	35	35	37	35	0	0	0	1	**0%**	**0%**	6%
20	88	84	80	62	0	0	0	1	42%	35%	**29%**

S: sequential model (PP - L - S); PS1: partial sequential model 1 (PP - L + S); PS2: partial sequential model 2 (PP + L - S); I: integrated model (PP + L + S).

Table 21: Comparison of models.

Before having the results, we expected that the partially sequential models would always present the same cost result or better than the sequential one since they integrate two problems, but that was not the case. The sequential model showed superior results for five instances (10, 14, 16, 18, and 19) compared to PS2. This discrepancy arises because, when integrating PP and L, we do not consider the constraint that the machines can only process one operation at a time; this constraint only appears in S. Consequently, for some instances, as observed in these five cases, certain machines end up more overloaded

with PS2 than when using the sequential model. On the other hand, as anticipated, the sequential model will never outperform PS1. Both share the same PP outputs, and PS1 integrates L and S, implying that its costs are equal to or better than those of the sequential model.

Even if the results were not so good compared to the others in some cases, PS2 was the best for Instances 1, 12, and 20. Compared to the others, the model with the most satisfactory results (frequency) was PS1. Generally, it is possible to see that for a few instances, the three models presented a difference of up to 5%. These were Instances 2, 5, 9, 11, and 18.

The models occasionally present good cost results but can easily deviate from the optimum (e.g., Instances 6, 13, 17, and 20). PS1 is considered the best because it is more consistent than the others. PS2 is the most variable, sometimes much worse or better than the others. However, this analysis was done for small instances. For larger, we believe the difference in costs can be even bigger due to the number of variables and bigger search space. Good cost results for each problem individually may induce bad results globally. The integrated model presents great superiority since it simultaneously considers all the shared variables and does not need any assumptions.

On the other hand, when analyzing the execution times, the initial expectation was that the sequential model would perform best, followed by the partially sequential models, and lastly, the integrated model. This expectation aligned with the observed results. Specifically, focusing on instances 9 to 18 where the integrated model takes more than 1 second, the superiority of the sequential model concerning computation time becomes evident. PS1 closely follows, outperforming expressively PS2 and the integrated model in instances 15 and 16. PS2, despite being surpassed by PS1, still exhibits significantly better results than the integrated model in these particular instances.

5.1.1 Conclusions

When comparing the models, the integral model exhibited significant cost superiority over the others, which was already predicted from the beginning. Surprisingly, when analyzing costs, the sequential model outperformed PS2 in some instances. However, PS1 consistently outperformed the sequential model due to their shared beginning (PP problem) and the integration of L and S in PS1's final stage. This integration allows PS1 to achieve results equal to or better than the sequential model. In addition, the integrated model is the most time-consuming, followed by the PS2, PS1, and sequential. In conclusion, the integrated model emerged as the most efficient, followed by PS1, then PS2, and finally the sequential model. PS1 was considered better than PS2 due to showing

lower cost variability.

5.2 Exact methods

As stated at the beginning of this chapter, due to its superior performance, only the integrated model is considered for comparing the exact methods. The same 20 instances are used in this section. Here, only the execution times of the three exact methods presented in Chapter 4 are compared. The costs are not analyzed because they will always be the same since all methods find the optimal value.

Table 22 shows the results. When the execution times are longer than 3000 seconds to reach the optimal result, we stopped the execution in 3000 seconds. These cases are represented with "3000+". Furthermore, Instances 15 and 16 show considerably longer execution times than the others due to their larger size, with their main characteristics detailed in Table 20. In the case of the exact methods, any increase in the input parameters of instances leads to a substantial rise in execution time. Although Instances 15 and 16 share the same main characteristics, the latter requires more time. This discrepancy can be attributed to the heightened flexibility of the machines, providing more possibilities (machine/configuration) to produce an operation and resulting in a larger search space.

Instance	**CPLEX**	**CP**	**CPS**
1	61	44	1
2	26	13	1
3	115	90	1
4	114	102	1
5	105	376	1
6	223	82	1
7	388	297	1
8	159	29	1
9	1630	68	2
10	3000+	426	10
11	3000+	518	6
12	3000+	464	19
13	3000+	1430	35
14	3000+	3000+	8
15	3000+	3000+	164
16	3000+	3000+	3000+
17	3000+	3000+	4
18	1054	312	2
19	439	96	1
20	3000+	328	1

Table 22: Execution time of the exact methods.

When comparing CPLEX and CP, CP shows significant superiority in execution time in almost all instances. Only in Instance 5 CPLEX was more efficient. The difference between the computation times is evident when comparing CPS with the other two methods. Most instances are solved in less than 10 seconds. There was only one instance not solved in less than 3000 seconds, Instance 16. This fact occurs because, in constraint programming, sometimes the search gets "blocked" in some search spaces, making it difficult to improve or prove the optimality of the solution.

5.2.1 Conclusions

We just compared the execution times, given that all three models are exact and yielded identical results. Constraint programming demonstrated significant superiority, particularly when incorporating our search strategy (CPS). However, we can note that as the instance size increases, as observed in Instances 15 and 16, constraint programming alone (CP and CPS) may not be the most suitable method for real-world cases.

5.3 Metaheuristics

In this subsection, we analyze the metaheuristics (GA and GA with local search) by first comparing the results found by these methods with the optimal values of the 20 instances used previously. Then, we proceed to test a larger instance.

5.3.1 Genetic algorithm

The genetic algorithm does not necessarily provide the optimal solution to a problem, but it is expected to get a solution as close to an optimal one as possible. As this method is not exact, we will analyze the execution time for a given number of iterations and the quality of solutions according to the cost.

Tuning parameters for small instances

The first step before running GA is to choose an appropriate set of parameters for the problem since it directly impacts the results' quality. This work considered four parameters: the size of the initial population, the number of generations, the crossover rate, and the mutation rate. Since our 20 instances are of similar sizes, the same parameters were employed for all. We analyzed the parameter impacts on the two most time-consuming instances in CP, Instances 15 and 16.

For this, a factorial design of experiments technique was applied, having three levels for each parameter, resulting in a combination of 81 experiments (3^4) that were run five

times, totaling 405 runs for each instance (81 × 5). Table 23 shows the parameters in the first column, the nomenclature given to each one in the second column, and then the tested levels.

Parameters	**Notation**	**Levels**		
		1	**2**	**3**
Initial population	pop	10	50	100
Number of generations	gen	10	50	100
Crossover rate	cross	0.5	0.8	1
Mutation rate	mut	0.1	0.2	0.5

Table 23: GA parameters and their levels.

The data were treated by the Minitab software, and the results are presented in Figures 31 to 34. Figures 31 and 32 show the main effects of each parameter level on cost minimization. The vertical axis represents the costs, and the horizontal axis represents the levels of each parameter. Both figures show that a very low initial population and number of generations considerably decrease the performance of the GA. The mutation rate also significantly impacts the values, improving the results when it increases. The crossover rate is the one that presents the most negligible impact, obtaining an almost horizontal line. Moreover, it is important to analyze the interaction of the parameters with each other. Figures 33 and 34 show the best set of parameters found, which was the same for both instances:

- Initial population: 100 individuals;
- Number of generations (stop criterion): 100 generations;
- Crossover rate: 50%;
- Mutation rate: 50%.

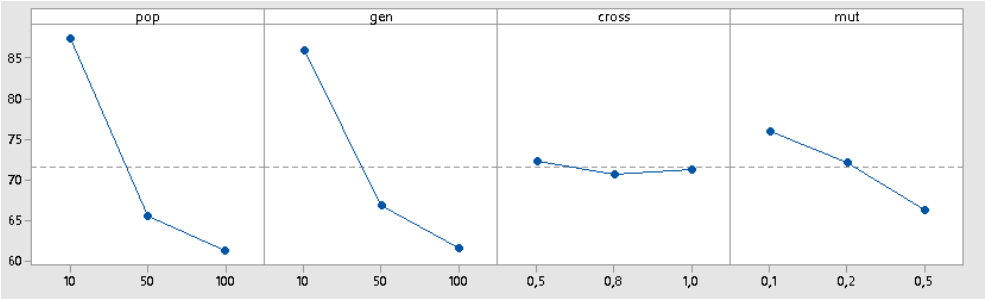

Figure 31: Main effects on cost minimization for Instance 15 (adjusted averages).

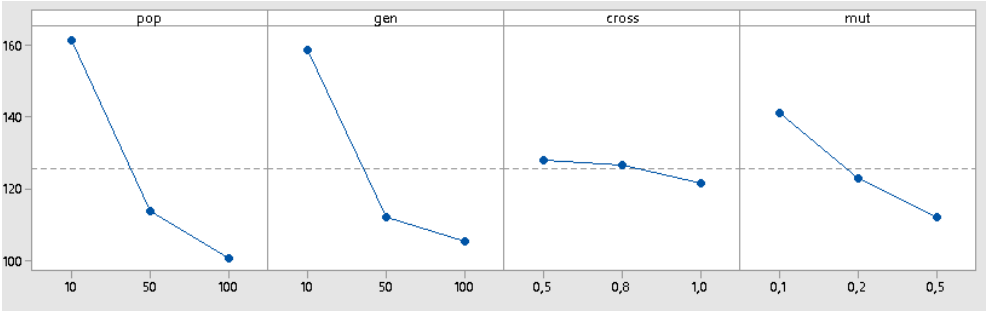

Figure 32: Main effects on cost minimization for Instance 16 (adjusted averages).

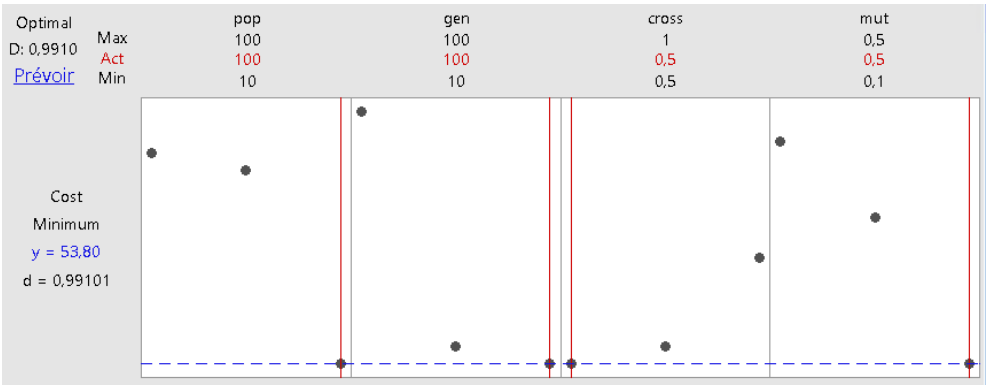

Figure 33: Optimization diagram for Instance 15.

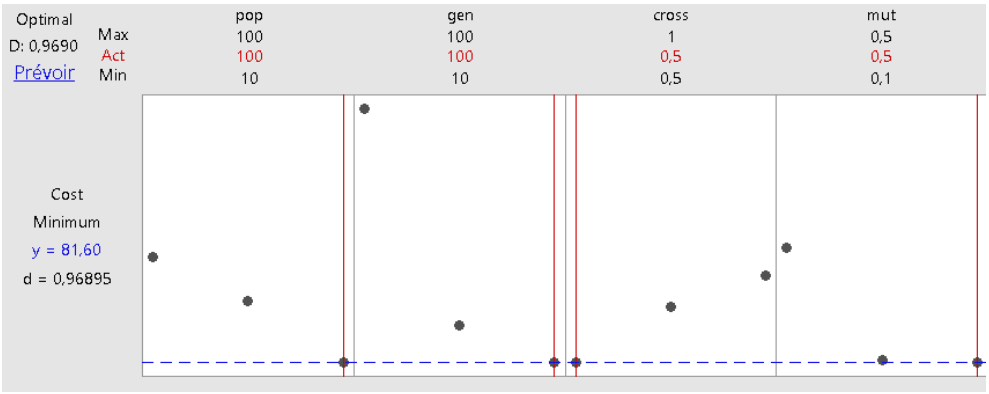

Figure 34: Optimization diagram for Instance 16.

After analyzing the graphs, we considered the pertinence of conducting new tests to observe the behavior of parameters such as 'pop,' 'gen,' and 'mut' when set higher. We evaluated it unnecessarily because 'pop' and 'gen' tend to stabilize quickly, and the current calibration suffices for method comparison since we are still analyzing small instances. Nevertheless, this information will not be neglected for larger instances, particularly the effect of considering higher mutation rates.

For the analysis, we also examined the results when generating only random individuals, which is equivalent to creating the initial population in GA without executing any operations afterward. This approach allows us to check the relevance of GA. To do this, we analyzed the case of the creation of ten thousand random individuals (R10) and one hundred thousand random individuals (R100). Table 24 presents the results. In the first column are the instances; in the second column is the optimal cost of each instance; in the third column is the average cost of R10; in the fourth column, the average cost of R100; followed by the average cost of GA; then, the average cost of GA with local search (GA + LS). For all methods, all mean costs are over ten runs. The last four columns represent the average distance between the optimal cost and the cost obtained by each method (average cost - optimal cost) / optimal cost.

	Cost					**% distance**			
Instance	**Optimal**	**R10**	**R100**	**GA**	**GA+LS**	**R10**	**R100**	**GA**	**GA+LS**
1	36	36	36	36	36	0%	0%	0%	0%
2	39	39	39	39	39	0%	0%	0%	0%
3	36	36	36	36	36	1%	0%	0%	0%
4	57	57	57	57	57	0%	0%	0%	0%
5	18	19	18	18	18	7%	0%	0%	0%
6	38	38	38	40	38	0%	0%	5%	0%
7	31	31	31	31	31	0%	0%	0%	0%
8	57	57	57	57	57	0%	0%	0%	0%
9	37	42	38	38	37	14%	2%	3%	0%
10	42	45	43	42	42	8%	2%	0%	0%
11	56	73	68	57	57	30%	22%	2%	2%
12	52	63	60	56	53	21%	15%	8%	2%
13	79	103	94	81	81	31%	19%	3%	3%
14	70	104	88	82	74	48%	26%	17%	6%
15	52	105	90	57	54	101%	73%	10%	4%
16	73	205	169	91	79	181%	132%	25%	8%
17	33	36	36	33	33	10%	8%	0%	0%
18	35	48	37	35	35	38%	5%	0%	0%
19	35	36	35	35	35	2%	0%	0%	0%
20	62	66	62	62	62	6%	0%	0%	0%

Table 24: Costs of random method, the GA and GA with local search.

The costs found by the random methods are only good for very small instances. As the instances grow, even marginally, the results deteriorate significantly, exemplified by Instance 16, where both methods show over a 100% difference from the optimum. The costs obtained by GA are either optimal or very close to the optimum in most instances. Instances 14, 15, and 16 are exceptions, with a distance equal to or greater than 10% from the optimal.

Table 25 examines only the execution times of the methods. Regarding GA, all instances yield results in less than 1 second, showcasing significant superiority in execution time compared to the exact methods. As the instance data increases, the exact methods

struggle to solve it in a reasonable time frame (considered here as less than one day) while the genetic algorithm remains effective.

	Time(s)			
Instance	**R10**	**R100**	**GA**	**GA+LS**
1	2	406	1	2
2	3	408	1	1
3	2	373	1	2
4	3	370	1	2
5	3	394	1	2
6	3	411	1	1
7	3	403	1	1
8	3	398	1	1
9	3	458	1	2
10	3	469	1	2
11	3	463	1	2
12	3	480	1	2
13	3	465	1	2
14	3	451	1	2
15	3	511	1	2
16	3	521	1	3
17	2	386	1	2
18	2	378	1	2
19	2	372	1	1
20	3	374	1	1

Table 25: Execution times of random method, the GA and GA with local search.

5.3.2 Genetic algorithm with local search

To evaluate the GA with local search, we maintained the previously determined GA parameters and set a local search size of only 1, i.e., each time the local search is applied to just one operation. We did not conduct a statistical test to determine the size of the LS because the instance is very small, consisting of only four operations. If we increase the size, it would encompass at least half of the instance, going against the purpose of the LS, which is to explore small portions. Additionally, we applied the three local searches (initial, during the generations, and final). The LS during the generations was set to occur every 10 generations.

Table 24 shows that this method achieves values very close to the optimum, with none of the average costs differing by more than 8% from the optimum. When compared with GA, this method consistently yields better or equal cost results. Notably, instances 14, 15, and 16 show significant improvement, as these were the instances where GA had the poorest performance.

When analyzing the execution times, this method remains highly efficient for small instances. However, as anticipated, the computational time is longer compared to GA using the same input parameters, attributed to the additional steps incorporated into the algorithm.

5.3.3 Conclusions

The first important point highlighted is that generating random solutions is inefficient, particularly as the size of instances increases. Consequently, employing strategies becomes imperative. In our case, we opted for the Genetic Algorithm (GA). When applying GA, it is essential to calibrate parameters such as the initial population, number of generations, crossover rate, and mutation rate. Calibration is not required for each instance but for a set with similar characteristics. Given that our test set comprises instances of similar sizes with values varying within a comparable range, we conducted calibration specifically for Instances 15 and 16, the largest and with the longest calculation times, using exact methods.

After the initial calibration, we compared the values obtained by the genetic algorithm with and without local search. The execution times for both were extremely low using the parameters chosen in the calibration. Notably, GA tended to deviate further from the optimum in the larger instances than GA with LS. This initial analysis suggests that LS can be a valuable ally in enhancing cost results. Regarding the difference in execution time, we cannot draw more concrete conclusions at this stage, as the instances are very small.

5.4 Industrial applicability

The model presented in this work can be applied in diverse industries, e.g., assembly or disassembly lines. Moreover, it is applicable when considering a system composed only of machines or of machines and humans. In any process where process planning, layout, and scheduling problems are relevant, their integration will generate at least equivalent results than when they are treated separately, with prominent chances of being better. This is a consequence of the sharing of variables between them. The integration of process planning and scheduling has already been studied. The importance of integrating layout has not been emphasized yet, probably because when considering production lines that use large machines, it is not intuitive to consider that moving them can bring any benefit.

The model's applicability can be extremely important in micro-factories and areas with equipment that can easily be placed on vehicles or having a vehicle attached, such as Automated Guided Vehicle (AGV). The previous cases require a high industrial automation

degree. However, the applicability can go beyond this. It can be extended to cases where the workstations are fixed, and the operators and tools are not. In these cases, the operators are represented by machines in the model, and the reconfiguration times are associated with the operator and tool displacements.

In this section, an industrial-scale example is presented. Real data inspired it, despite the times, costs, and precedence graphs were randomly generated for confidentiality reasons.

In an industry that fabricates large kitchen utensils, two of the products in the catalog are produced in an area with 20 operators (considered as the machines in the model presented in section 4). The products have variants, such as color and type of finishing, but this does not interfere with the fact that Product 1 comprises 12 operations and Product 2 comprises 8. Each operator can perform different tasks, between 2 and 4, according to his technical skills. There are five different layout possibilities. A time period where five products must be manufactured is considered, being two Products 1 and three Products 2. The main information of the problem can be listed as follows:

- Number of jobs: 5.
- Number of operations per job: 12, 8, 12, 8, and 8.
- Number of machines: 20.
- Number of configuration per machine: 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 4, 4, 2, 3, 3, 4, 4, 2, and 2.
- Number of layouts: 5.

A new factorial design of experiments was made to define the best parameters for the genetic algorithm in this example. As the problem is bigger this time, new values were assigned to the levels. Table 26 shows the values used.

Parameters	**Notation**	**Levels**		
		1	**2**	**3**
Population size	pop	100	250	500
Number of generations	gen	100	250	500
Crossover rate	cross	0.5	0.8	1
Mutation rate	mut	0.2	0.5	0.7

Table 26: Industrial case GA parameters and their levels.

The results led us to the following parameters:

- Population size: 500 individuals;
- Number of generations (stop criterion): 500 generations;

- Crossover rate: 80%;
- Mutation rate: 70%.

The mutation rate is extremely high, especially according to some authors, like [5], who say this rate should be between 0.01 and 0.1. This high percentage is probably justified by the relatively small initial population (to avoid increasing execution time) and the absence of new individuals being inserted during generations. To facilitate escaping local optima, the factorial design of experiments indicates that higher mutation levels lead to improved results.

GA was run ten times on the problem using the university's (UTC) server (80 core, 512 Gb RAM, linux CentOS), obtaining a cost average of 271.4 with an average run time of 15 seconds. Table 27 indicates the cost and the run time for each run. The standard deviation of the total cost was less than 5% compared to the average total cost. In the case of the run time, the standard deviation was about 5%. For this example, GA is stable in cost and time results.

	Total cost	**Run time(s)**
Run 1	273	16
Run 2	261	16
Run 3	273	14
Run 4	266	15
Run 5	262	15
Run 6	292	14
Run 7	253	15
Run 8	267	16
Run 10	280	14
Mean	**271.4**	**15**
Standard deviation	**12.16**	**0.82**

Table 27: Results of the GA for the industrial scale example.

Another analysis that can be made is comparing with CPS. CPS is not able to reach the optimum in a reasonable time. Regardless, it is possible to let it run and stop to see the best answer until a stipulated time. When CPS runs for 15 seconds, GA average time, it cannot find any solution. When CPS ran for one hour (3600 seconds), the average cost found was 518.

We can also observe the impact on the results when applying GA with LS. For a first analysis, we maintained the same parameters previously determined for GA. We employed the three local searches, all with a size of 3. The results from 10 runs are presented in Table 28. The average cost experienced a reduction of a little over 5%, which, while not exceptionally high, reflects a positive trend.

	Total cost	Run time(s)
Run 1	248	421
Run 2	276	423
Run 3	266	424
Run 4	277	417
Run 5	260	420
Run 6	254	425
Run 7	263	422
Run 8	242	420
Run 9	258	426
Run 10	255	419
Mean	**259.9**	**421.7**
Standard deviation	**11.17**	**2.83**

Table 28: Results of the GA with LS for the industrial scale example.

However, for this specific instance, a more detailed analysis reveals that the gain cannot be significantly higher due to two specific jobs consistently being late, primarily because of their small due dates. It was observed that local search reduces their tardiness but does not eliminate it entirely, as production times and material handling already surpass the due dates. We will not analyze this example in more detail here because, in the next chapter, we detail a case study and its analysis.

The execution time surpassed 400 seconds. Although this is notably longer than GA, it is still a reasonable duration, particularly when considering the scheduling of an entire shift or production day, being less than 8 minutes of execution.

After the analysis conducted thus far, some questions have been raised. Firstly, we observed that GA with LS requires significantly more execution time compared to GA alone. Would run GA for the same duration as GA with LS yield better results? Secondly, does the proximity of job due dates influence the efficiency of GA with LS compared to GA alone? For instance, if jobs are distant from their due dates or are sure to be delayed, can this variability affect the effectiveness of local search?

Six new instances were generated to derive conclusions on these two points, and their main characteristics are presented in Table 29. The machines and configurations remain the same as those of the instance introduced at the beginning of this section. This time, we limited our analysis to three layout possibilities. Furthermore, some input values vary from instance to instance, including due dates and the number of jobs. For the initial two instances, job due dates are considered 'normal,' implying they are neither overly restrictive nor excessively distant. Instances 3 and 4 have very distant due dates, ensuring no job will be late. The due dates for the last two instances are close, and we anticipate that all jobs will be late.

Instance	NJ	NO	NM	NC	NL
1	4	12,8,12,8	20	2,2,2,2,2,2,4,2,2,2,2,4,4,2,3,3,4,4,2,2	3
2	6	12,8,12,12,8,12	20	2,2,2,2,2,2,4,2,2,2,2,4,4,2,3,3,4,4,2,2	3
3	6	12,8,12,8,12,12	20	2,2,2,2,2,2,4,2,2,2,2,4,4,2,3,3,4,4,2,2	3
4	6	12,8,12,8,12,12	20	2,2,2,2,2,2,4,2,2,2,2,4,4,2,3,3,4,4,2,2	3
5	6	12,8,12,8,12,12	20	2,2,2,2,2,2,4,2,2,2,2,4,4,2,3,3,4,4,2,2	3
6	5	12,8,12,8,8	20	2,2,2,2,2,2,4,2,2,2,2,4,4,2,3,3,4,4,2,2	3

NJ: number of jobs; NO: number of operations per job; NM: number of machines; NC: number of configuration per machine; NL: number of layouts.

Table 29: Main characteristics of the industrial instances.

The results, averaged over ten runs, are presented in Table 1. Each row corresponds to an instance. The initial columns indicate costs and times obtained using the genetic algorithm, followed by results from the genetic algorithm with local search (GA+LS), and finally, results from the genetic algorithm without local search (GA 450) running for 450 seconds. The duration of 450 seconds was chosen as it denotes the average time between the executions of GA+LS. Given that GA with LS has a considerably longer execution time than GA alone, we aim to assess whether integrating local search is truly beneficial or if extending the run time of GA alone could yield similar or better results.

The findings in Table 30 demonstrate the consistent superiority of GA+LS over GA, even when the latter is allowed a 450-second runtime. Table 31 provides the distance between GA (alone and running for 450 seconds) and GA+LS. In the first column, the largest observed difference within the given intervals is 15%, with the smallest being 4%. Moving to the second column, we note that the largest difference within the intervals is 8%, while the smallest is 2%. Even if the difference decreases when GA runs for 450 seconds, it is still not superior to using LS. Addressing the second question about the impact of due dates on the efficacy of LS, the conclusion is that LS remains relevant in all scenarios. However, for the instances analyzed, the impact of LS is lower when we know that the jobs will be late. While a more thorough analysis is a perspective for the future, our current findings confirm the significance of LS in its existing form.

	GA		**GA + LS**		**GA 450**	
	cost	**time(s)**	**cost**	**time(s)**	**cost**	**time(s)**
1	352.6	11.2	320.5	327.8	339.6	450
2	655.4	20	570	505.7	615.8	450
3	511.3	16.8	481	502.1	500.5	450
4	327,7	16.7	315.1	500.5	321.5	450
5	1174.1	20.2	1105.5	496.3	1167.5	450
6	816.5	14.5	752.8	375.8	800.4	450

Table 30: Results of the GA and GA with local search for big instances.

	% distance to GA+LS	
	GA	**GA 450**
1	10%	6%
2	15%	8%
3	6%	4%
4	4%	2%
5	6%	6%
6	8%	6%

Table 31: Distance from GA+LS results to GA for big instances.

5.4.1 Conclusions

A new calibration was performed for the larger instances, considering that the inputs to the problem are entirely different from the set of 20 small instances. This time, a larger population size, number of generations, and mutation rate were considered. In the case of large instances, it became evident that incorporating local search consumes significantly more time than using GA alone. However, it was also demonstrated that even when equalizing the execution time of the two methods, including local search remains beneficial. Moreover, this holds irrespective of whether the jobs are close to their due dates or not.

5.5 Conclusions of the chapter

This work presents some modeling versions of process planning, layout, and scheduling problems, which are sequential, partially sequential, and in an integrated way. Sequential and partially sequential models can give good results in some cases, but in others, they are completely far from optimum. This occurs because the problems have shared variables treated separately in these cases. Also, some hypotheses are taken to manage the three problems sequentially. Therefore, a good result for an individual problem does not guarantee a good global result.

After proving the superiority of the integrated model, the following studies were made only with it. Our combinatorial problem is huge, so the ILP method is time-consuming, even for small instances. In the search for performance improvement, three exact methods were compared: ILP, CP, and CPS. CPS was the best among them. The resolution times were much lower than the others for the presented examples.

In any case, exact methods cannot solve real-size problems in a reasonable time. Therefore the development of metaheuristics adapted to our problem was proposed. The metaheuristics results for the small instances compared with the optimal values were satisfactory.

For the industrial case, GA rapidly converges to a result with low variability. On the other hand, GA with LS takes considerably more time, although it remains more efficient. We can summarize the findings of this chapter as follows:

- The integrated model significantly outperforms the others when considering costs exclusively.
- For small instances, regarding costs and distance from the optimum, the rankings are as follows: integrated model, PS1 ($PP \rightarrow L + S$), PS2 ($PP + L \rightarrow S$), and sequential model.
- The execution time of the integrated model is longer than that of the others using exact methods, but this behavior would most likely not be repeated with metaheuristics.
- Constraint programming with a strategy adapted to our problem is the most efficient among the three exact methods analyzed.
- Exact methods are sensitive to increases in instance size, as expected.
- Metaheuristics found good results.
- Adding local search to the GA is beneficial in terms of cost and execution time.

Chapter 6

Case Study - Disassembly

Contents

In this chapter, we present a detailed case study to show the applicability of our integrated model solved by the proposed genetic algorithm. Many studies in this field focus on assembly and manufacturing issues. However, RMS are not limited to these areas; they can also handle disassembly problems. This aspect is crucial for the circular economy, for instance.

In Figure 35, the difference between the classical linear economy and the circular economy can be seen. The dark blue line shows the linear model, including resource extraction, manufacturing, use, and disposal. The circular model, represented by the light blue arrows, aims to minimize initial extraction and keep products in circulation through recycling, remanufacturing, or reusing. The goal is to extend, close, and regenerate resource cycles, reducing the need for new raw materials and reducing waste production. This approach emphasizes the use of materials that are durable, recyclable, and renewable [140].

The disassembly process can be used at different stages of the circular economy based on the product and its condition. Depending on their materials, disassembled product parts are primarily reused if they are in good condition or recycled.

Many products, especially electronics, are disassembled for recycling at the end of their life. Effectively managing the reuse or disposal of components has a significant positive impact on environmental concerns. This fact has prompted global discussions on introducing new laws and regulations to force reverse logistics in certain sectors. Even without such laws, it is highly advisable for environmental and economic reasons.

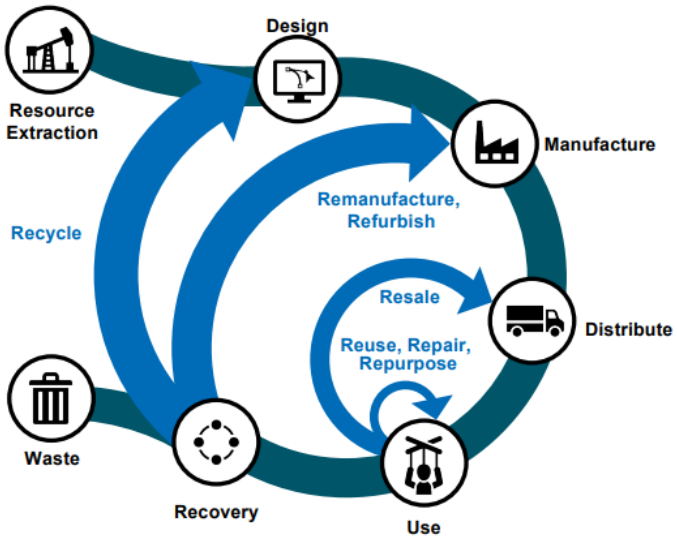

Figure 35: Circular economy system diagram [3].

Disassembly operations require careful planning and cost-effective optimization, similar to assembly problems [110]. Depending on the product and available technology, it can be executed by human operators, robots, or a combination of both. It is comprehensible that the responsibility for this process lies with the company, as products from different companies have varying components and processes. This diversity makes it challenging for a third party to handle the variability of the disassembly process of many enterprises simultaneously. A company's in-depth knowledge of its assembly process facilitates the knowledge of its disassembly process. Additionally, the flexibility of systems can be a powerful ally, as Reconfigurable Manufacturing Systems (RMS) can be implemented for both assembly and disassembly purposes.

The initial section of this chapter focuses on defining the problem. Subsequently, we will present all the data used and a numerical analysis of the example.

6.1 Problem description

In this section, we examine the disassembly process of a hand light, dividing it into seven parts. Various process plans for this process are explored based on three different precedence graphs. This process involves three operators and two robots, all of which can be reconfigured. Both humans and machines have three possible configurations. These will be explained hereafter. Additionally, the layout itself is reconfigurable, offering five different possibilities.

Finding real-world data in the existing literature is challenging, especially regarding

detailed aspects like time and cost associated with reconfiguring layouts and material handling. To address this gap, our analysis combines real and made-up data.

Figure 36 shows the hand light before any operation. The hand light's structure and potential disassembly sequences were outlined in a study by Tang *et al.* [130]. More recently, He [59] utilized this example in their research. This product comprises seven parts: battery, bulb, cover, glass, head housing, main housing, and spring, numbered from 1 to 7, as illustrated in Figure 37.

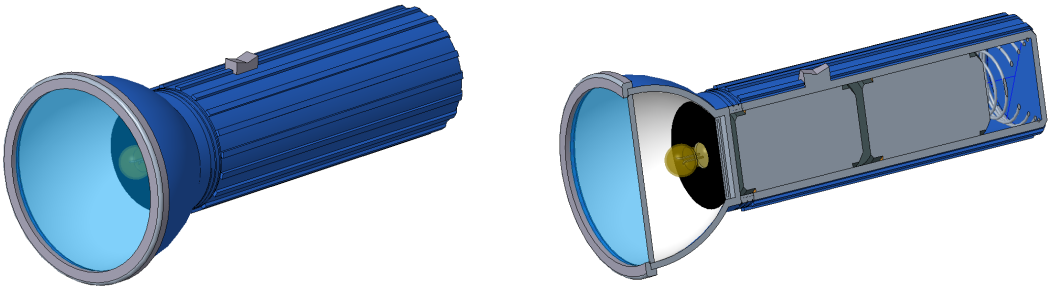

Figure 36: Assembled hand light (normal and section view).

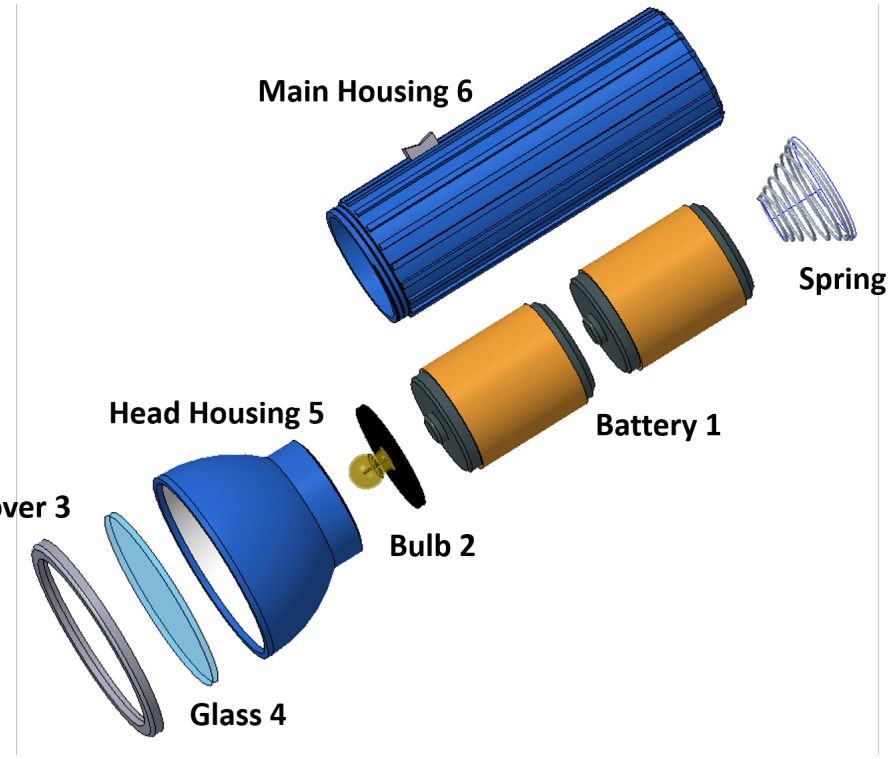

Figure 37: Hand light's parts.

Figure 38 illustrates the directed graph of feasible disassembly sequences as presented in Tang *et al.* [130]. In the original study, the parts of the hand light were labeled with letters; here, they are enumerated according to Figure 37. The first rectangle in Figure 38 represents the initial state, where the product has not yet been disassembled. Following

this, there are two possibilities. In the first scenario, the parts are divided into two sets: one consisting of the cover, glass, bulb, and head housing, and the other with the battery, spring, and main housing. The second possibility also separates the parts into two sets: the first set contains only the cover and glass, while the second set contains the remaining components. These disassembly options continue until all parts are disassembled, resulting in seven sets, each containing a single part. It is important to note that the disassembly needs to follow an order of precedence, e.g., disassembling the bulb before the cover/glass is not allowed.

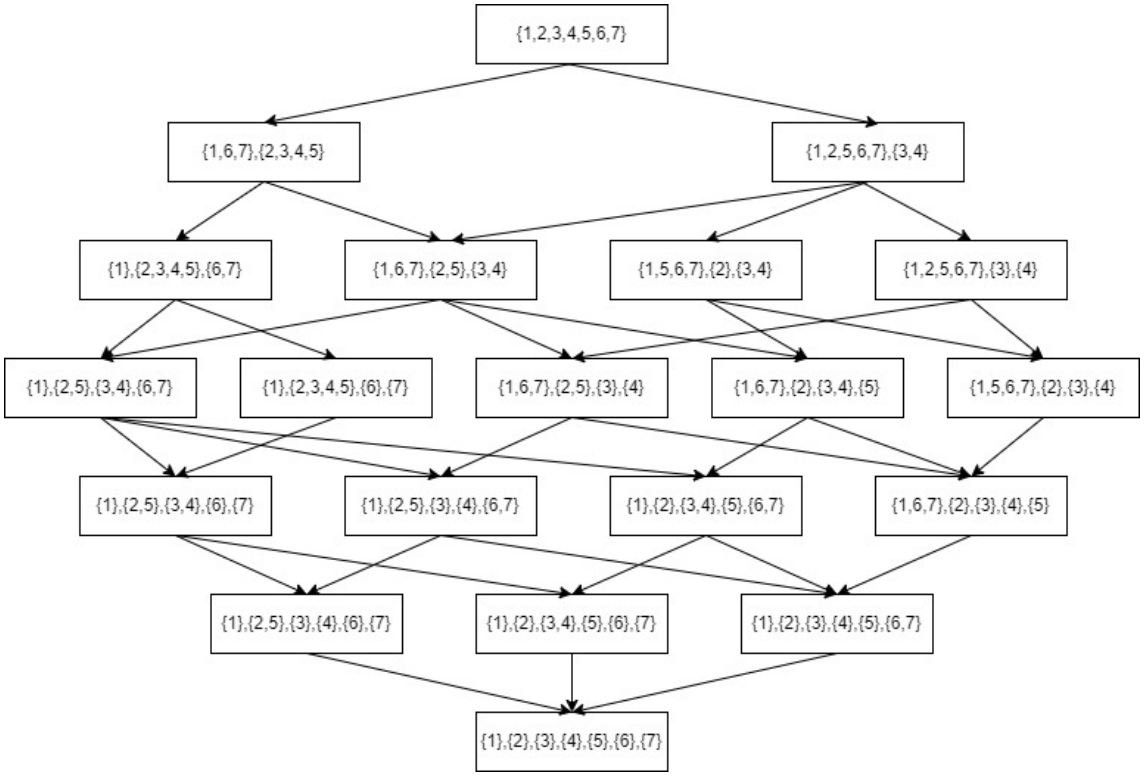

Figure 38: Possible disassembly sequences.

In his study, He [59] introduced a decomposition color graph, an alternative representation of the directed graph illustrating feasible disassembly sequences, building upon the AND/OR graph outlined in Tang *et al.* [130]. He [59] identified three potential disassembly schemes, having ten possible operations. Table 32 outlines the disassembly outcomes for each operation. The first column lists the operations, the second column specifies the sets to be disassembled, and the last column shows the result of the disassembly operation.

In the second column, there is always only one set, leading to two sets as each operation represents a single disassembly step. For instance, Op_1 and Op_2 are applicable only when the product has not yet been disassembled. After operation Op_1, we have two sets: one with parts 1, 6, and 7, and another with parts 2, 3, 4, and 5. After operation Op_2, we have one set containing parts 1, 2, 5, 6, and 7, and another set with parts 3 and

4. The same reasoning for the others.

Operation	**Before operation**	**After operation**
$Op1$	{1,2,3,4,5,6,7}	{1,6,7},{2,3,4,5}
Op_2	{1,2,3,4,5,6,7}	{1,2,5,6,7},{3,4}
Op_3	{2,3,4,5}	{2,5},{3,4}
Op_4	{1,2,5,6,7}	{1,6,7},{2,5}
Op_5	{1,2,5,6,7}	{1,5,6,7},{2}
Op_6	{3,4}	{3},{4}
Op_7	{2,5}	{2},{5}
Op_8	{1,5,6,7}	{1,6,7},{5}
Op_9	{1,6,7}	{1},{6,7}
Op_{10}	{6,7}	{6},{7}

Table 32: Disassembly resulting from each operation.

To align the problem with our model, we converted the three potential schemes from the decomposition color graph in [59] into three precedence graphs, as depicted in Figure 39. Each graph comprises six operations, corresponding to the edges in Figure 38. It is necessary to traverse all six operations, progressively disassembling the product until it is completely taken apart. Regardless of the chosen graph, the process always begins with either operations 1 or 2, which are the initial disassembly possible steps (refer to Table 32).

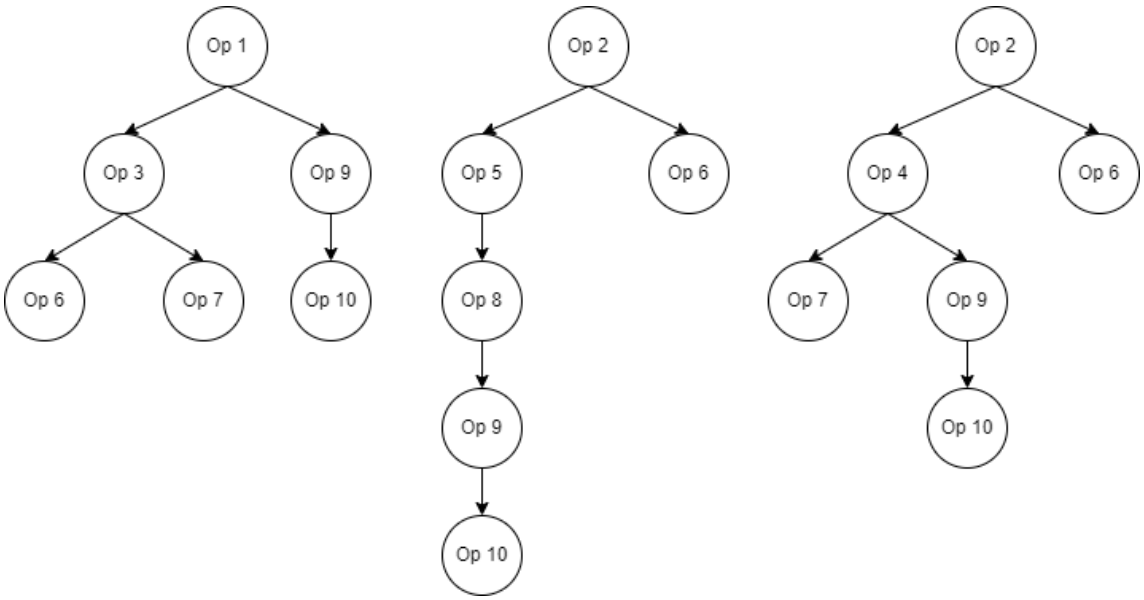

Figure 39: Possible precedence graphs.

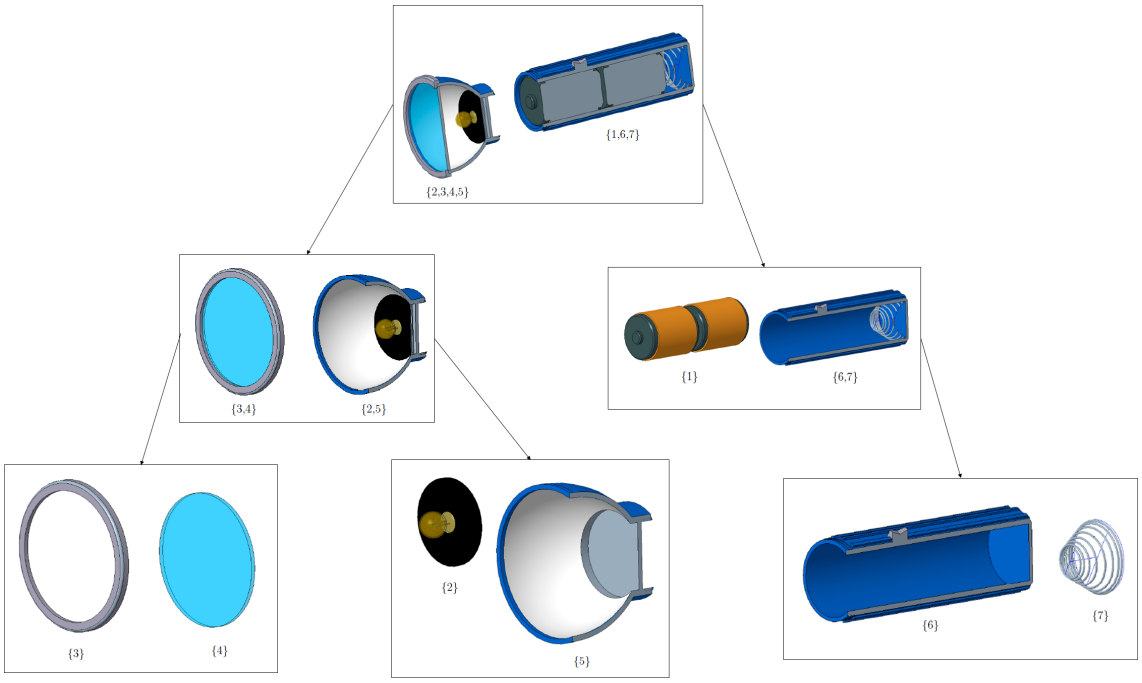

Figure 40: Illustrated precedence graph 1.

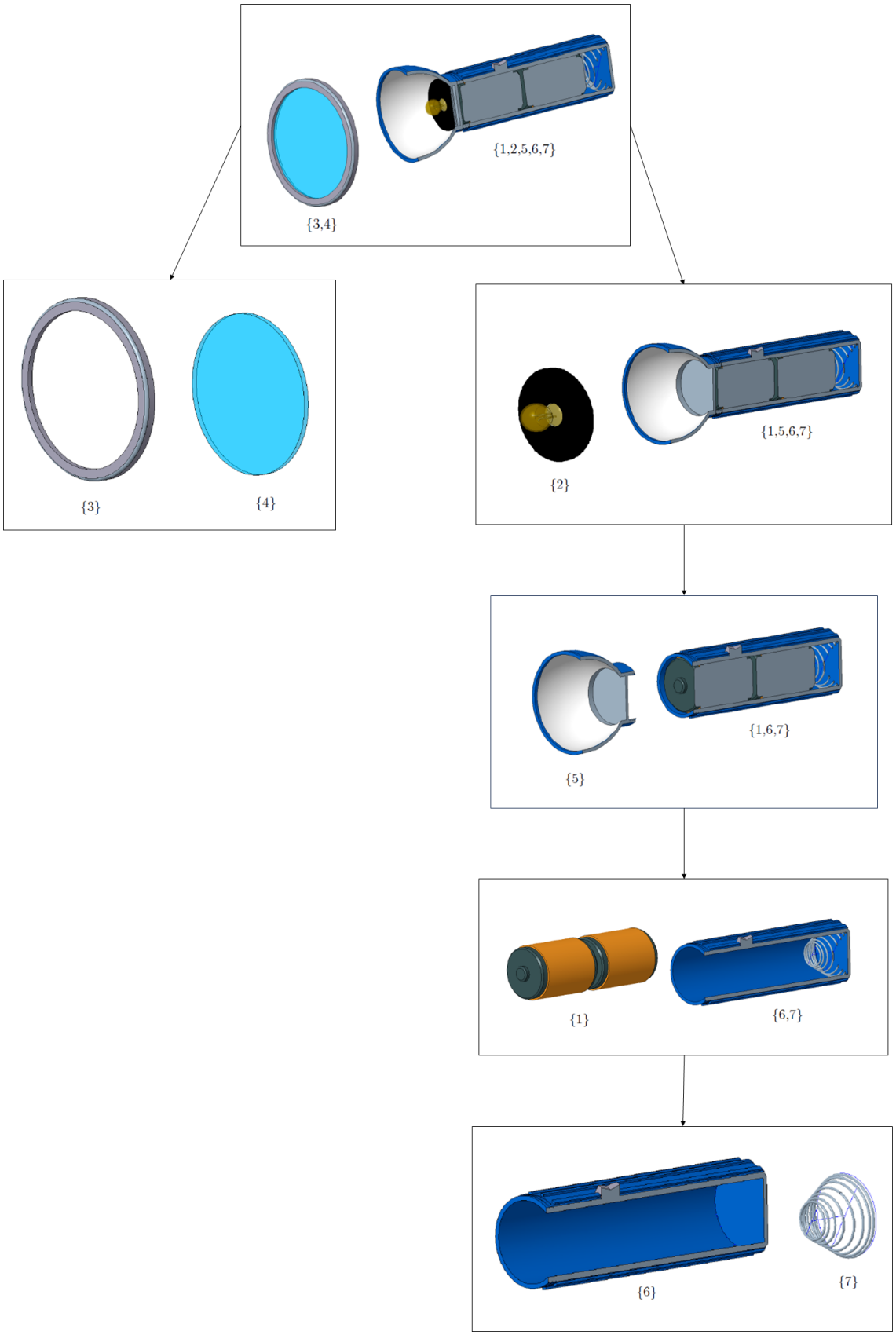

Figure 41: Illustrated precedence graph 2.

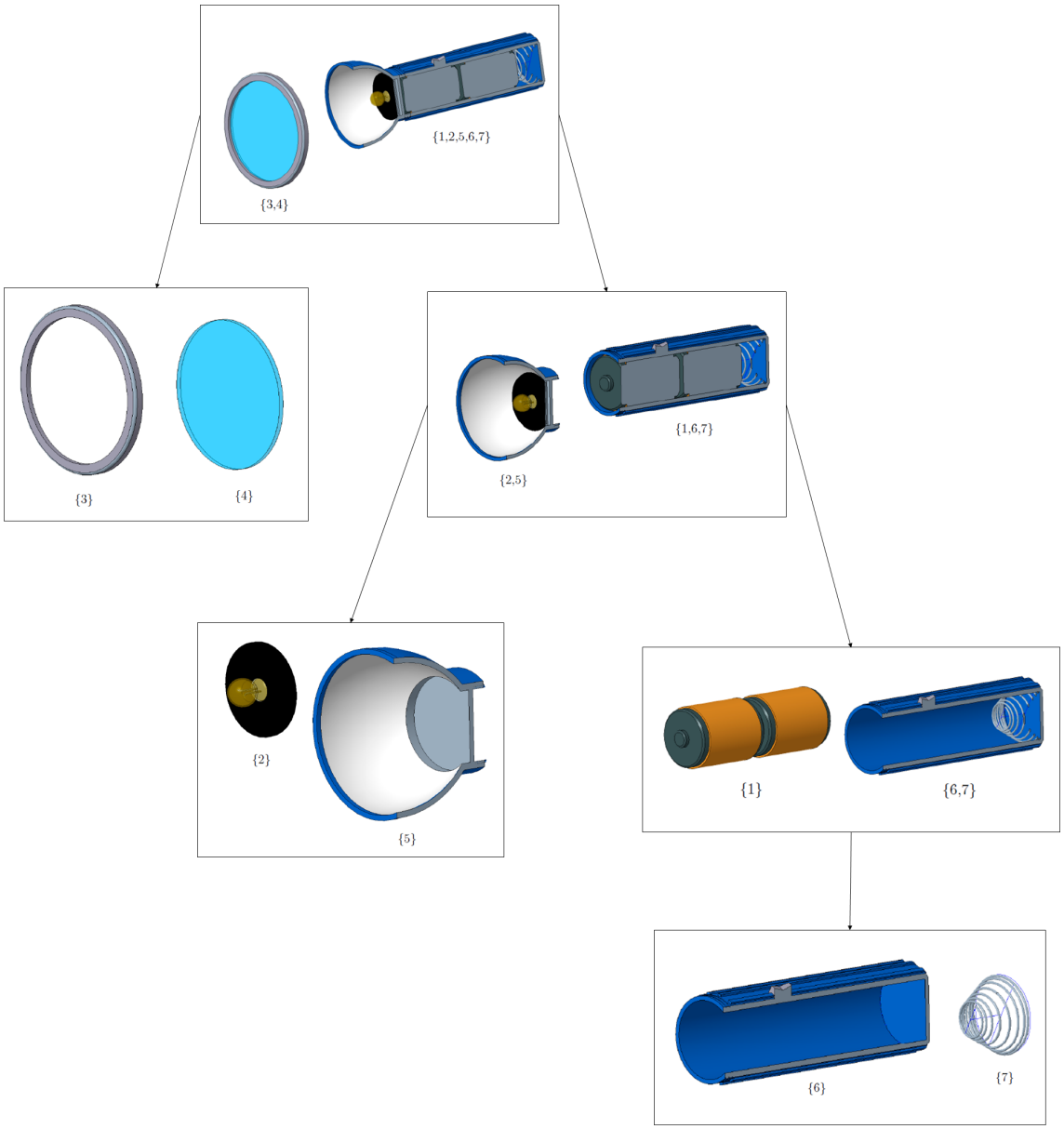

Figure 42: Illustrated precedence graph 3.

6.2 Problem input data

This section details all the data used in the case study. There is only one type of hand light, but it can follow three different precedence graphs (Figure 39). For this reason, we will assume three job types: A, B, and C, one for each graph. They comprise six operations each.

In the context of this study, the term "machine" encompasses both human operators and robots. Specifically, our study has three human operators and two robots. The operators can perform all the required operations using the three distinct configurations.

The first configuration is used for operations 2, 3, 5, 6, and 7, during which operators are required to wear protective gloves due to the involvement of glass (parts 4 and 2 of the hand light). The second configuration is exclusive to operation 10, necessitating a special tool to separate the spring from the main housing. The third configuration is employed for the remaining operations: 1, 4, and 8, which separate the head housing from the main housing, and operation 9, which removes the batteries.

The two robots used in this study are limited to executing basic operations, specifically operations 2, 3, 6, and 9. They cannot unscrew components (operations 1, 4, and 8) and are unsuitable for operations 5 and 7, which involve the light bulb. These operations require careful inspection to determine if the bulbs are still functional. Additionally, the robots are not equipped for operation 10.

The time taken by human operators to complete a task can vary, while the time for the robots remains constant as we consider them to have identical capabilities. Table 33 shows the possible operations in each configuration for each machine. Machines M_1, M_2, and M_3 are humans, and machines M_4 and M_5 are robots. The first column indicates the machine, the second column shows the operations that can be performed on configuration 1, then the operations for configuration 2, and finally the operations for configuration 3.

Machine	**Configuration 1**	**Configuration 2**	**Configuration 3**
M_1	Op_2 Op_3 Op_5 Op_6 Op_7	Op_{10}	Op_1 Op_4 Op_8 Op_9
M_2	Op_2 Op_3 Op_5 Op_6 Op_7	Op_{10}	Op_1 Op_4 Op_8 Op_9
M_3	Op_2 Op_3 Op_5 Op_6 Op_7	Op_{10}	Op_1 Op_4 Op_8 Op_9
M_4	Op_2 Op_3	Op_6	Op_9
M_5	Op_2 Op_3	Op_6	Op_9

Table 33: Operations that can be executed by each machine/configuration.

Consider disassembling 60 hand lights within a single shift. Due to logistical requirements, 15 products must be disassembled every hour. Therefore, the due date of the first 15 products is in the first hour, the next 15 in the second hour, and so forth until the end of the morning. Failure to meet the due date results in a tardiness penalty of 0.001 monetary units per unit of time, as the parts of these products will need to be reassigned to other transports.

Each machine has a specific duration for completing each operation. Operators take longer than robots, but they can be reconfigured much faster than robots. Table 34 provides details about the time each machine takes to perform an operation in different configurations. The first column lists the ten operations, and the subsequent columns show the corresponding times. For example, the second column presents the time taken by machine M_1 in configuration 1 for each operation. If a machine or configuration cannot

perform a certain operation, we indicate the time as infinite. The processing times for operators are derived from the data in Kim & Lee [79], while the processing times for robots are based on He [59].

	M_1			M_2			M_3			M_4			M_5		
	$M_1[1]$	$M_1[2]$	$M_1[3]$	$M_2[1]$	$M_2[2]$	$M_2[3]$	$M_3[1]$	$M_3[2]$	$M_3[3]$	$M_4[1]$	$M_4[2]$	$M_4[3]$	$M_5[1]$	$M_5[2]$	$M_5[3]$
Op1	∞	∞	82	∞	∞	70	∞	∞	68	∞	∞	∞	∞	∞	∞
Op2	90	∞	∞	92	∞	∞	85	∞	∞	16	∞	∞	16	∞	∞
Op3	78	∞	∞	89	∞	∞	84	∞	∞	16	∞	∞	16	∞	∞
Op4	∞	∞	90	∞	∞	82	∞	∞	72	∞	∞	∞	∞	∞	∞
Op5	67	∞	∞	93	∞	∞	78	∞	∞	∞	∞	∞	∞	∞	∞
Op6	97	∞	∞	82	∞	∞	86	∞	∞	∞	31	∞	∞	31	∞
Op7	74	∞	∞	55	∞	∞	54	∞	∞	∞	∞	∞	∞	∞	∞
Op8	∞	∞	65	∞	∞	55	∞	∞	74	∞	∞	∞	∞	∞	∞
Op9	∞	∞	52	∞	∞	69	∞	∞	72	∞	∞	25	∞	∞	25
Op10	∞	95	∞	∞	88	∞	∞	55	∞	∞	∞	∞	∞	∞	∞

Table 34: Operation times (time unit).

The processing times are not the only factors to consider; each operation also carries a specific cost determined by the machine and configuration used. Details of these costs can be found in Table 35.

	M_1			M_2			M_3			M_4			M_5		
	$M_1[1]$	$M_1[2]$	$M_1[3]$	$M_2[1]$	$M_2[2]$	$M_2[3]$	$M_3[1]$	$M_3[2]$	$M_3[3]$	$M_4[1]$	$M_4[2]$	$M_4[3]$	$M_5[1]$	$M_5[2]$	$M_5[3]$
Op1	∞	∞	25	∞	∞	21	∞	∞	20	∞	∞	∞	∞	∞	∞
Op2	27	∞	∞	28	∞	∞	26	∞	∞	48	∞	∞	48	∞	∞
Op3	23	∞	∞	27	∞	∞	25	∞	∞	48	∞	∞	48	∞	∞
Op4	∞	∞	27	∞	∞	25	∞	∞	22	∞	∞	∞	∞	∞	∞
Op5	20	∞	∞	28	∞	∞	23	∞	∞	∞	∞	∞	∞	∞	∞
Op6	29	∞	∞	25	∞	∞	26	∞	∞	∞	93	∞	∞	93	∞
Op7	22	∞	∞	17	∞	∞	16	∞	∞	∞	∞	∞	∞	∞	∞
Op8	∞	∞	20	∞	∞	17	∞	∞	22	∞	∞	∞	∞	∞	∞
Op9	∞	∞	16	∞	∞	21	∞	∞	22	∞	∞	65	∞	∞	65
Op10	∞	29	∞	∞	26	∞	∞	17	∞	∞	∞	∞	∞	∞	∞

Table 35: Operation cost (cost unit).

The reconfiguration of machines involves tool changes. For operators, reconfiguration times and costs are small, encompassing tasks such as putting on a glove. In contrast, for robots, reconfiguration is more time-consuming and costly. It entails adjusting the tools used by the robots or even replacing them. In both cases, this process can necessitate some tests and adjustments. Table 36 shows the reconfiguration times and costs.

	M_1	M_2	M_3	M_4	M_5
Cost	5	5	5	30	30
Time	10	10	10	60	60

Table 36: Machine reconfiguration.

Regarding the layout, there are five available possibilities, as illustrated in Figure 43. Reconfiguration occurs when a new layout reduces costs by altering the material handling

flow. This effortless adjustment is facilitated by the mobility of workstations and robots, often utilizing Automated Guided Vehicle (AGV), enabling reconfiguration even midway through the disassembly process. However, reconfiguring the layout involves additional costs and time, as outlined in Table 37 and 38.

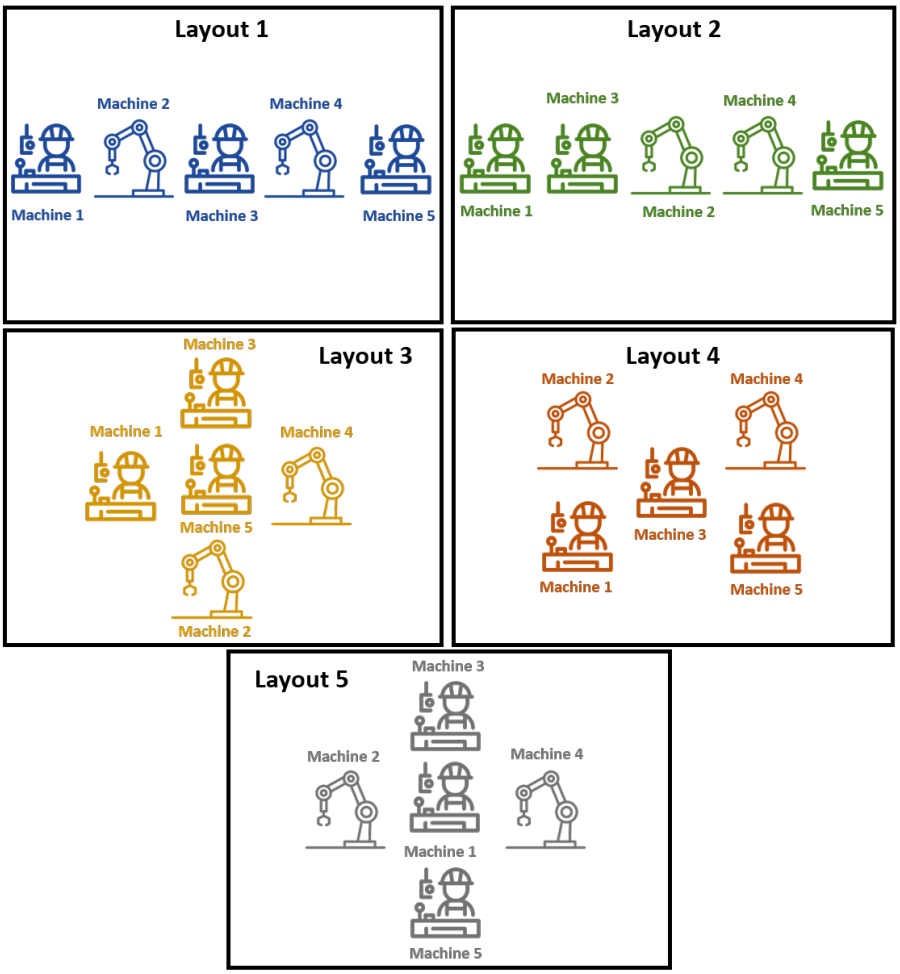

Figure 43: Layout set

	Λ_1	Λ_2	Λ_3	Λ_4	Λ_5
Λ_1	0	17	30	21	24
Λ_2	18	0	24	21	32
Λ_3	23	23	0	25	19
Λ_4	19	20	26	0	23
Λ_5	20	23	19	31	0

Table 37: Layout reconfiguration costs.

	Λ_1	Λ_2	Λ_3	Λ_4	Λ_5
Λ_1	0	35	67	51	59
Λ_2	45	0	58	52	65
Λ_3	54	54	0	60	48
Λ_4	47	51	66	0	55
Λ_5	52	57	47	63	0

Table 38: Layout reconfiguration times.

As a result, the material handling times fluctuate based on the chosen layout. Specific material handling times are detailed in Table 39. The monetary cost of material handling is determined by multiplying the distance between machines by a constant of 1 in this case.

Layout 1

	M_1	M_2	M_3	M_4	M_5
M_1	0	6	12	18	24
M_2	6	0	6	12	18
M_3	12	6	0	6	12
M_4	18	12	6	0	6
M_5	24	18	12	6	0

Layout 2

	M_1	M_2	M_3	M_4	M_5
M_1	0	12	6	18	24
M_2	18	0	6	6	18
M_3	6	6	0	12	18
M_4	18	6	12	0	18
M_5	24	12	18	6	0

Layout 3

	M_1	M_2	M_3	M_4	M_5
M_1	0	9	9	18	6
M_2	9	0	21	9	6
M_3	9	18	0	9	6
M_4	15	9	9	0	6
M_5	6	6	6	6	0

Layout 4

	M_1	M_2	M_3	M_4	M_5
M_1	0	6	9	21	12
M_2	6	0	9	12	24
M_3	6	6	0	6	6
M_4	18	12	9	0	6
M_5	15	21	9	6	0

Layout 5

	M_1	M_2	M_3	M_4	M_5
M_1	0	6	6	6	6
M_2	6	0	9	18	9
M_3	6	9	0	9	18
M_4	6	15	9	0	9
M_5	6	9	21	9	0

Table 39: Material handling times.

All the information provided in this section will serve as input data for solving the problem.

6.3 Numerical results

In this section, the established case study is solved first using the GA. Similar to the approach taken in the previous chapter, the initial step involves tuning the parameters of the genetic algorithm for this specific problem. Once again, we employ a factorial design of experiments technique with three levels per parameter. This results in a total of 81 experiments (3^4 combinations) conducted ten times, leading to 810 runs for this instance (81×10). Table 40 presents the parameters in the first column, followed by their corresponding nomenclature in the second column, and the tested levels.

Parameters	Notation	Levels		
		1	2	3
Population size	pop	250	500	1000
Number of generations	gen	250	500	1000
Crossover rate	cross	0.5	0.8	1
Mutation rate	mut	0.2	0.6	0.8

Table 40: Hand light GA parameters and their levels.

Figure 44 shows the main effects of each parameter on the cost when varied individually. Mainly, larger population sizes, numbers of generations, and mutation rates lead to decreased costs, aligning with the objective function. The most substantial impact is observed in the number of generations; when this parameter is set to 250, the results are the poorest. For population and mutation parameters, the costs worsen at the last level. However, it is important to note that this does not imply that higher values inherently lead to worse outcomes. Based on the conducted experiments, the average cost was lower with a population size of 500 and a mutation rate of 0,6 (analyzing each parameter individually). This suggests that the variation between a population of 500 or 1000 or a mutation rate of 0.6 or 0.8 does not significantly influence the costs observed. Crossover exhibited the least variability but also demonstrated that increasing this rate enhances the results.

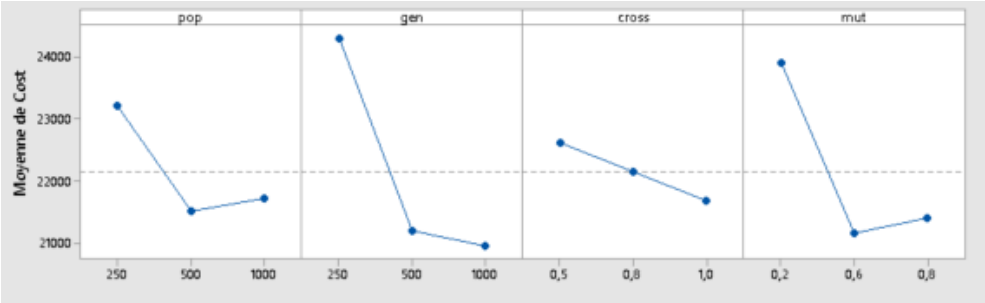

Figure 44: Graph of the main effects on cost minimization (adjusted averages).

However, considering the overall picture is more important than analyzing each parameter individually. Figure 45 indicates the optimal parameterization. When focusing only on the cost and analyzing the data obtained, the best choice is:

- Population size: 500 individuals;
- Number of generations (stop criterion): 1000 generations;
- Crossover rate: 100%;
- Mutation rate: 80%.

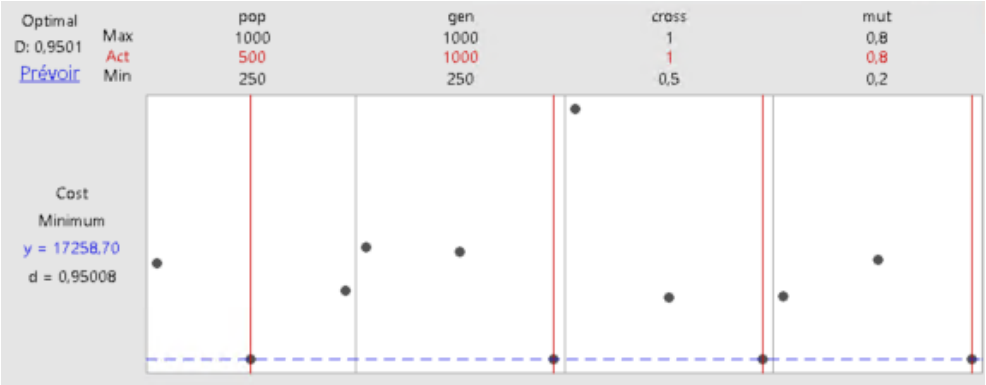

Figure 45: Cost optimized diagram.

In the previous chapter, the application of the design of experiments focused exclusively on cost. In this section, we extend the evaluation to include execution time, ranging from approximately 20 to 6000 seconds.

Figure 46 illustrates the main effects on the execution time when each parameter is varied individually. As anticipated, the execution time increases with the increase in the value of each parameter. The most significant impact is observed with the increase in population size, followed by the number of generations.

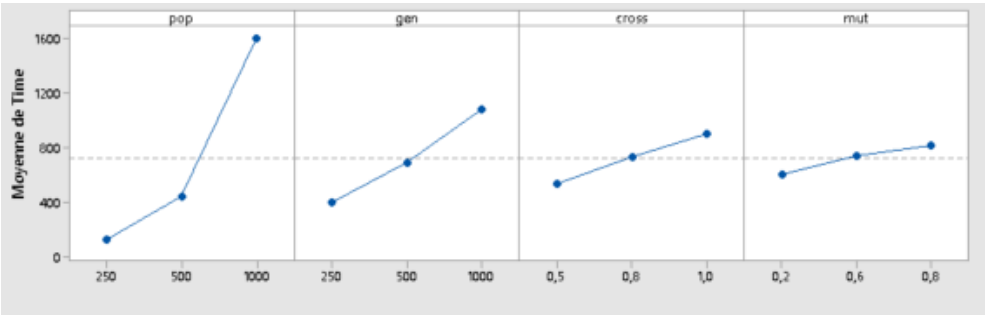

Figure 46: Graph of the main effects on time minimization (adjusted averages).

Undoubtedly, the best scenario involves using the lowest possible values for all parameters to optimize time. Time assumes a secondary role in this context since the primary goal is to minimize costs. Nevertheless, we can search for the best balance between costs and execution time.

In the cost-optimized configuration, a population of 500 is favored over 1000, offering an advantageous compromise. The number of generations stands as the second parameter with the most significant impact. If the decision maker deems the calculation time excessively high, reducing the number of generations can notably enhance execution time without substantially degrading the cost. In the studied example, when considering the

simultaneous minimization of costs and execution times, the optimal parameters are (as shown in Figure 47):

- Population size: 250 individuals;
- Number of generations (stop criterion): 500 generations;
- Crossover rate: 80%;
- Mutation rate: 60%.

To determine these parameters, we applied a weight of 2 to cost minimization and a weight of 1 to execution time minimization.

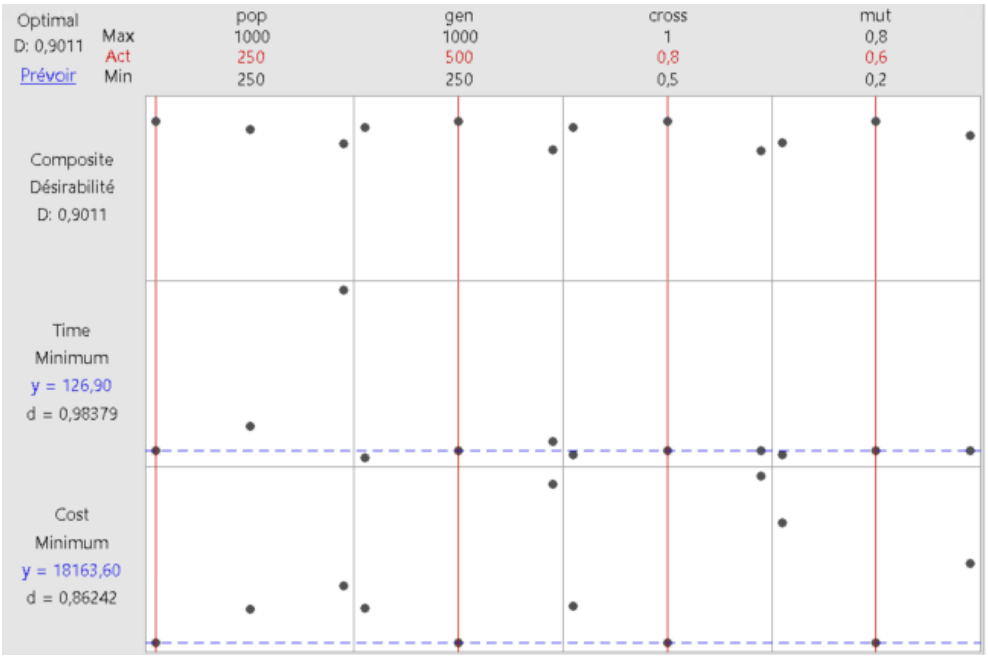

Figure 47: Cost and runtime optimized diagram.

Parameter calibration is done only once for the same production process. It only becomes necessary when significant variations occur, such as changing machines, introducing a new product, or designing a new assembly/disassembly line. The chosen parameters heavily depend on the company's specific needs, particularly the product family under consideration. In some cases, calculation time is a critical factor that must be minimized. In others, longer computation times are acceptable, especially for scheduling a bigger time window (e.g., a monthly schedule). In our example, we selected parameters that balance time and cost, considering the short planning period and a process with low added value.

Table 41 presents the results of 10 runs. The second column shows the costs found in each run, and the last column shows the times. The first ten rows are each dedicated to a run. The penultimate row shows the average result of the ten runs, and the last row

shows the standard deviation. We used the parameters set that balances cost and time considerations (population = 250, number of generations = 500, crossover rate = 80%, and mutation rate = 60%). The costs exhibit a standard deviation of around 5%. The execution time has a standard deviation of approximately 2%. We can conclude that both standard deviations are low, meaning that we have costs that do not vary much from the average (no outlier either), and the time using the same parameters does not vary significantly either.

	Total cost	**Run time(s)**
Run 1	17036	86
Run 2	17307	86
Run 3	18533	88
Run 4	17302	87
Run 5	16663	88
Run 6	16849	87
Run 7	16769	85
Run 8	18429	89
Run 9	18333	83
Run 10	19312	86
Mean	**17653.3**	**86**
Standard deviation	**920.31**	**1.7**

Table 41: Results of the GA for the example with 60 hand lights.

When we apply GA with LS search using the same parameters obtained above (population = 250, generation = 500, crossover = 1, and mutation = 0.8) and incorporating the initial local search, the LS during the execution of generations (every 100 generations), and the final one, we obtain the values shown in Table 42. The values obtained demonstrate an improvement in the average cost by approximately 10%. Although the execution time increases by more than 20%, it remains extremely low, at less than two minutes. While the 10% reduction in costs may not seem substantial, it holds significance for the industry.

	Total cost	Run time(s)
Run 1	15625	107
Run 2	16503	110
Run 3	15723	109
Run 4	16207	111
Run 5	15735	108
Run 6	15076	111
Run 7	16089	108
Run 8	16009	109
Run 9	15489	108
Run 10	16494	103
Mean	**15895**	**108.4**
Standard deviation	**452.17**	**2.32**

Table 42: Results of the GA with LS for the example with 60 hand lights.

As the example involving 60 hand lights is too large for graphical representation and to present all the results, we have chosen to illustrate a smaller example with 12 hand lights. The input values for production times and costs, machine reconfiguration, layout reconfiguration, due date, material handling, and tardiness have been adjusted for this more compact example. The updated data is provided in Appendix A.

Utilizing the parameters identified in the previous example with 60 hand lights (population = 250, generation = 500, crossover = 0.8, and mutation = 0.6), Table 43 shows the results of 10 runs.

	Total cost	Run time(s)
Run 1	972	7
Run 2	996	7
Run 3	940	7
Run 4	891	7
Run 5	898	7
Run 6	1016	6
Run 7	954	8
Run 8	979	7
Run 9	948	7
Run 10	1030	7
Mean	**962.4**	**7**

Table 43: Results of the GA for the example with 12 hand lights.

The resulting scheduling of Run 1 is illustrated in Figure 48 in diagram format. The figure presents when each operation is manufactured and on each machine/configuration, with operations from the same job represented in the same color. Table 53, in Appendix

B, summarizes the key outputs of each operation, including the machine, configuration, and layout used, along with their respective start and completion times.

For the GA with local search, we incorporate the three types of local search, and the segment size is set to 5. Table 44 presents the results for 10 runs. In comparison to the GA results, there is an improvement of more than 8% in costs, and the average execution time increases by only 1 second. The resulting scheduling of Run 1 is illustrated in Figure 49 in diagram format. Table 54, in Appendix C, summarizes the key outputs of each operation, including the machine, configuration, and layout used, along with their respective start and completion times.

	Total cost	**Run time(s)**
Run 1	818	9
Run 2	851	9
Run 3	920	8
Run 4	880	8
Run 5	849	8
Run 6	860	8
Run 7	898	8
Run 8	890	8
Run 9	966	8
Run 10	921	9
Mean	**885.3**	**8.2**

Table 44: Results of the GA with LS for the example with 12 hand lights.

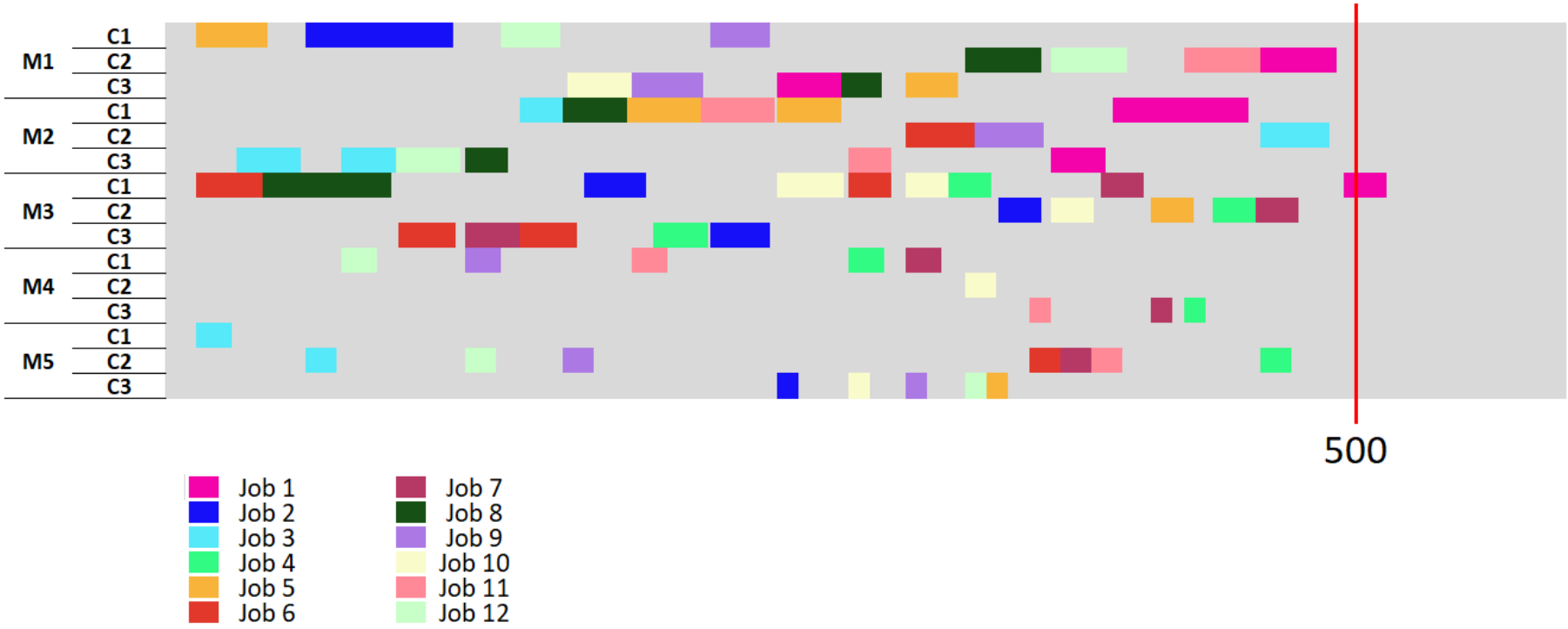

Figure 48: Graphical representation of the scheduling found using GA for 12 hand lights.

[H]

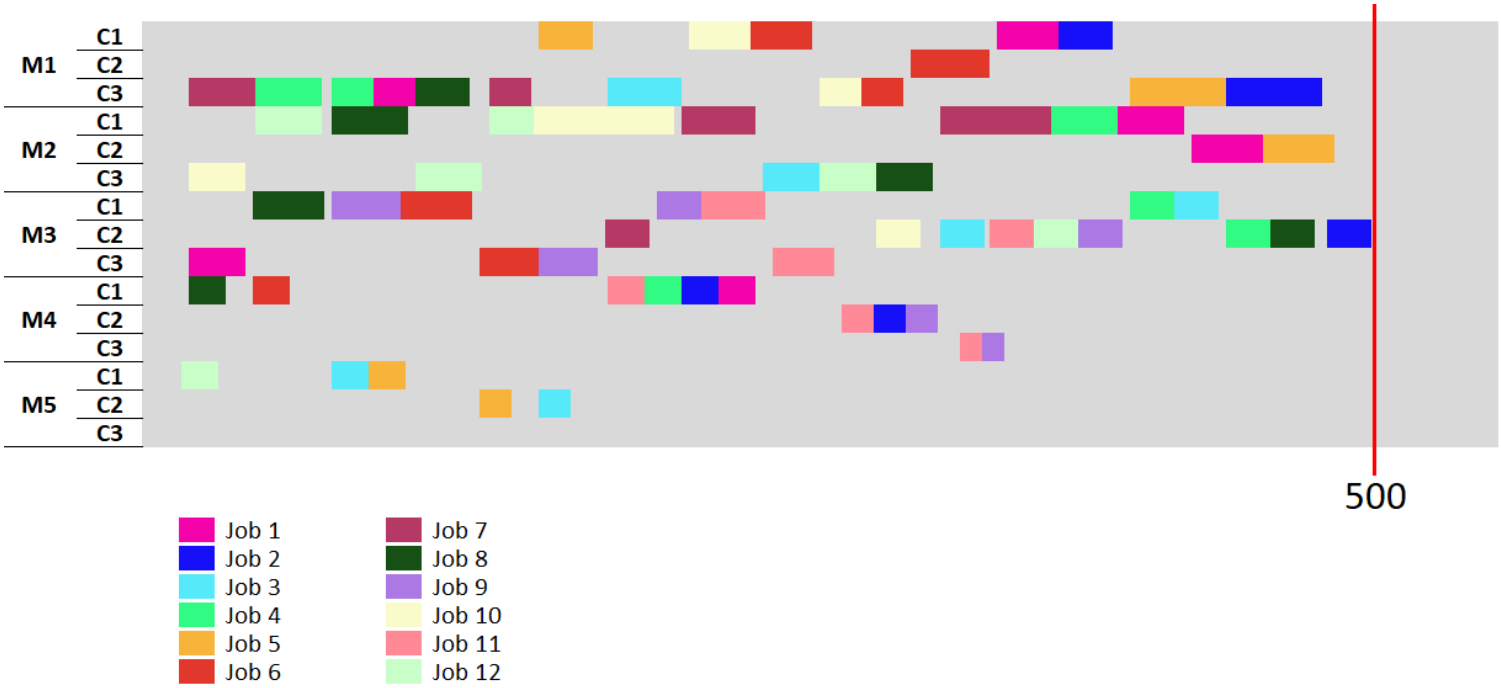

Figure 49: Graphical representation of the scheduling found using GA with LS for 12 hand lights.

[H]

6.4 Conclusions of the chapter

In this chapter, a complete case study is analyzed. The first interesting point is that the case study is not about producing a part family or the product assembly. Here, an example of disassembly was presented, which makes it adapted to demonstrate that the applicability of the RMS is vast and goes beyond what is usually found in the literature. Moreover, it is not necessary to work with a 100% automated system (e.g., micro-factories). It is possible to use RMS concepts in processes done by operators, making it feasible to model manual and automatic processes in the same problem, which is common in industries.

Another critical point is that the genetic algorithm can solve large problems in a reasonable amount of time. The most time-consuming aspect is the initial calibration of parameters, which is typically a one-time process. Recalibration becomes necessary only with significant changes, such as adding or removing machines or introducing new products. The chosen parameters directly align with the priorities of the studied production line. For instance, in the automotive sector, cost reduction is the most important in line with a weekly planning schedule, even if it requires more calculation time. This is due to the sector's high value-added nature, where minimizing costs, even by small percentages, holds a significant economic impact.

Incorporating LS can enhance the obtained solutions, we observe a slightly over 8% improvement. In an industrial context, this percentage can translate into substantial cost savings annually. Furthermore, adding LS did not lead to a substantial increase in execution time.

Chapter 7

Conclusion and Perspectives

7.1 Conclusion

In today's dynamic industrial landscape, marked by ever-evolving market demands and heightened customer expectations, companies must adapt swiftly to sudden changes. This adaptability is particularly crucial in light of unforeseen events that can disrupt operations or fluctuations in customer demand. The search for a diverse range of products at competitive prices, coupled with a growing demand for customized solutions and shorter product life cycles, contributes significantly to the increasing variability in demand. Industries must be proactive and responsive to these dynamics, employing strategies that allow for rapid adjustments to production processes and resource allocation. Adapting and thriving in a dynamic environment is crucial for staying competitive and meeting customer expectations.

To adapt to this context, companies are exploring ways to enhance their responsiveness and flexibility while maintaining cost-effectiveness. Adopting Reconfigurable Manufacturing Systems (RMS) has emerged as a response to existing needs. These systems are designed to quickly change their structure, allowing rapid adjustments to their capacity and functionality. The changes involve reconfiguring both machines and the system as a whole. One notable difference compared to other systems is the ease of rearranging the layout.

RMS have been extensively studied over the last two decades, primarily focusing on optimizing production when utilizing these systems. In research on these systems, common areas of study include planning, scheduling, system configuration, layout, product family formation, and material handling [12]. Among these topics, product family formation differs by specifically focusing on designing the system to ensure its suitability for a particular family. The remaining subjects are interconnected. However, few studies consider these problems in an integrated way.

Our work proposes an integration of all these interconnected elements, also giving attention to the less commonly addressed aspects of layout and material handling. The aim is to optimize resource allocation, considering several factors crucial for effective scheduling

in such systems.

Hence, this research aims to underscore the importance of addressing these challenges globally, optimizing process planning, scheduling, and layout configuration in an RMS for mass-customized products while reducing manufacturing costs. Additionally, we aim to demonstrate the development of efficient methods for tackling these intricately connected issues.

The three problems studied in this work are $\mathcal{NP}$-hard [49, 78, 104]. However, we observed a gap in the literature regarding the complexity analysis of RMS optimization problems. To address this gap, we explored the complexity of scheduling problems involving single and parallel machines, with objectives including minimizing makespan and total (weighted) completion time. Our findings revealed that the majority of these problems were $\mathcal{NP}$-hard. However, the problems concerning a single machine minimizing total (weighted) completion time and parallel machines minimizing total weighted completion time remain open for further analysis.

To achieve these research goals, the initial phase involved formalizing the problem of cost minimization, encompassing production costs, layout reconfiguration, machine reconfiguration, material handling, and tardiness. Subsequently, we developed four new models to optimize these RMS optimization problems (scheduling, process planning, and layout). The first model is fully integrated. The second is sequential, with no integration between the three problems. The last two are partially sequential and, hence, with some integration. After the model formulation, the next step was studying the performance of the four models and identifying whether the integrated model performed better than the sequential model, as expected. We demonstrated the clear superiority of the integrated model in terms of costs, while the other models were superior in terms of execution time for small instances. It is important to note that the comparison was made using an exact solution method, which does not guarantee a similar discrepancy in execution time if we use other methods, such as metaheuristics. Using an exact method, we can observe that the integrated model achieves a global optimum while solving the problems simultaneously. On the other hand, the other models only reach individual optimums for each sub-problem, which then serve as input for the following sub-problem. As a result, the costs are the same or frequently worse than those of the integrated method. For this reason, we opted to employ only the integrated model for the subsequent work, given its better results.

After the problem formulation and choosing the model, the next phase is proposing efficient solving methods. We proceeded to evaluate some solving methods, beginning with a comparison of three exact methods: Integer Linear Programming (ILP), Constraint Programming (CP), and Constraint Programming with a defined search strategy (CPS).

Since these methods guarantee optimal solutions, the analysis focused on their computation times. Notably, CPS demonstrated a marked superiority over the others, highlighting the potential of employing a well-designed strategy to significantly reduce the search space. However, despite this advantage, CPS is not the most suitable method for solving large instances. At the outset of this research, we considered hybridizing CPS with other methods. For instance, we initially considered incorporating it into the genetic algorithm to evaluate the fitness function. As the study progressed, we realized it may not yield good execution times. Instead, we opted to employ a greedy algorithm for this purpose. Nonetheless, the possibility of integrating CPS with other methods remains open for future work.

Since the three problems are $\mathcal{NP}$-hard, we created, for large instances, a metaheuristic specifically designed for our problem. Given the similar population evolution logic shared by many evolutionary algorithms, we chose to concentrate on Genetic Algorithm (GA), which has proven effective in various studies within the field. Initially, we developed the GA, and subsequently, we incorporated Local Search (LS) techniques to enhance the results. GA exhibited good outcomes in terms of both costs and times, the GA combined with LS yielding even better results. The improvements achieved with this final method were approximately 10% of total cost savings for the presented examples. This percentage can translate into a significant competitive advantage within an industrial context.

To demonstrate the applicability of our algorithms, we created a case study involving hand light disassembly. This case study is relevant for many reasons. Firstly, it underscores the capability of RMS and our model to handle the coexistence of human operators and machines. Secondly, while most authors focus only on parts production or product assembly problems, we showed our model's versatility by applying it to disassembly. Thirdly, we expanded the discussions addressing environmental concerns, which usually center around optimizing gas and waste emissions. We outlined the importance of RMS in the scope of product life cycles and the circular economy. Lastly, we have demonstrated the efficiency of our algorithms in managing large-scale instances.

7.2 Open Issues and Perspectives

In terms of future research, there are several potential avenues to explore. We can further study the open problems from Chapter 3: the single-machine scheduling problem with machine reconfiguration to minimize the total (weighted) completion time and the two parallel machines scheduling problem with machine reconfiguration to minimize the total completion time.

Regarding the model, new variables can be incorporated, mainly to address the

specific needs of each industry. For example, product quality is a critical consideration for some industries, while others prioritize environmental concerns. Additionally, production balancing may be a key aspect in specific sectors. The model proposed in this work is general, allowing adaptations to meet the specific needs of individual industrial applications.

Regarding solving methods, alternative approaches could be developed. One possibility is to explore hybrid methods, such as combining CPS with other techniques. Depending on the application, employing new metaheuristics or considering stochastic methods could also be valuable.

Concerning the undertaken work, certain aspects warrant further development and examination. In all presented examples, we applied the three proposed types of local search simultaneously, each with the same size. Similar to how we tuned the parameters of the genetic algorithm, conducting a statistical study to determine the optimal parameterization of LS could enhance our performance.

Another consideration is that we retained the parameters determined for the genetic algorithm when applying it in combination with local search. Local search introduces more significant variability during the algorithm's execution, potentially allowing for a reduction in population size and mutation rate parameters. If such adjustments can be made without compromising quality and cost, the execution times of the genetic algorithm with local search could be reduced.

In the case study, adding the local search to the GA did not significantly increase execution times, which can be attributed to two factors. Firstly, we set the local search size to a value with a limited impact (the growth in size exponentially increases the execution time of the local search). Secondly, the system's flexibility in our example was not excessively high. Each machine could execute only a few operations, which limits the search space during the local search. When the system is too flexible, our developed local search may not be the most suitable and potentially very time-consuming.

Furthermore, an important aspect would be implementing the study in a real-world system, evaluating its performance, and identifying necessary adaptations, encompassing both the model and the resolution methods.

Bibliography

1. Abbasi, M. & Houshmand, M. Production planning and performance optimization of reconfigurable manufacturing systems using genetic algorithm. *International Journal of Advanced Manufacturing Technology* **54,** 373–392 (2010).
2. Abellan-Nebot, J. V., Liu, J. & Romero Subiron, F. Design of multi-station manufacturing processes by integrating the stream-of-variation model and shop-floor data. *Journal of Manufacturing Systems* **30,** 70–82 (2011).
3. Agarwal, R. *Discover the Advantages of Asset Management in enabling Circular Economy for Boosting Resource Efficiency and Catalyzing Change* 2023. blogs.sap.com/2023/06/08/discover-the-advantages-of-asset-management-in-enabling-circular-economy-for-boosting-resource-efficiency-and-catalyzing-change/.
4. Ameer, M. & Dahane, M. Reconfigurability improvement in Industry 4.0: a hybrid genetic algorithm-based heuristic approach for a co-generation of setup and process plans in a reconfigurable environment. *Journal of Intelligent Manufacturing* **34,** 1445–1467 (2021).
5. Angelova, M. & Pencheva, T. Tuning genetic algorithm parameters to improve convergence time. *International Journal of Chemical Engineering* **2011** (2011).
6. Arnarson, H., Yu, H., Olavsbråten, M. M., Bremdal, B. A. & Solvang, B. Towards smart layout design for a reconfigurable manufacturing system. *Journal of Manufacturing Systems* **68,** 354–367 (2023).
7. Asghar, E., Zaman, U. K. U., Baqai, A. A. & Homri, L. Optimum machine capabilities for reconfigurable manufacturing systems. *International Journal of Advanced Manufacturing Technology* **95,** 4397–4417 (2018).
8. Baker, K. *Introduction to sequencing and scheduling* (1974).
9. Barrera-Diaz, C. A., Nourmohammadi, A., Smedberg, H., Aslam, T. & Ng, A. H. C. An Enhanced Simulation-Based Multi-Objective Optimization Approach with Knowledge Discovery for Reconfigurable Manufacturing Systems. *Mathematics* **11** (2023).
10. Bensmaine, A., Dahane, M. & Benyoucef, L. A new heuristic for integrated process planning and scheduling in reconfigurable manufacturing systems. *International Journal of Production Research* **52,** 3583–3594 (2014).
11. Bensmaine, A., Dahane, M. & Benyoucef, L. A non-dominated sorting genetic algorithm based approach for optimal machines selection in reconfigurable manufacturing environment. *Computers and Industrial Engineering* **66,** 519–524 (2012).

12. Bortolini, M., Galizia, F. G. & Mora, C. Reconfigurable manufacturing systems: Literature review and research trend. *Journal of Manufacturing Systems* **49,** 93–106 (2018).
13. Bruccoleri, M., Amico, M. & Perrone, G. Distributed intelligent control of exceptions in reconfigurable manufacturing systems. *International Journal of Production Research* **41,** 1393–1412 (2003).
14. Bruccoleri, M., Pasek, Z. J. & Koren, Y. Operation management in reconfigurable manufacturing systems: Reconfiguration for error handling. *International Journal of Production Economics* **100,** 87–100 (2006).
15. Brucker, P. *Scheduling Algorithms* ISBN: 9783540205241. https://books.google.fr/books?id=XO8fRT1yr58C (Springer Berlin Heidelberg, 2004).
16. Brucker, P. *Scheduling Algorithms* ISBN: 9783540695158 (2004).
17. Bruno, J. & Coffman, E. G. Scheduling Independent Tasks To Reduce Mean Finishing Time. *Communications of the ACM* **17,** 382–387. ISSN: 15577317 (1974).
18. Campos Sabioni, R., Daaboul, J. & Le Duigou, J. An integrated approach to optimize the configuration of mass-customized products and reconfigurable manufacturing systems. *International Journal of Advanced Manufacturing Technology,* 141–163 (2021).
19. Campos Sabioni, R., Daaboul, J. & Le Duigou, J. Concurrent optimisation of modular product and Reconfigurable Manufacturing System configuration: a customer-oriented offer for mass customisation. *International Journal of Production Research* **60,** 2275–2291 (2020).
20. Campos Sabioni, R., Daaboul, J. & Le Duigou, J. Joint optimization of product configuration and process planning in Reconfigurable Manufacturing Systems. *International Journal of Industrial Engineering and Management* **13,** 58–75 (2022).
21. Castañé, G. *et al.* The ASSISTANT project: AI for high level decisions in manufacturing. *International Journal of Production Research* (2022).
22. Chaube, A., Benyoucef, L. & Tiwari, M. K. An adapted NSGA-2 algorithm based dynamic process plan generation for a reconfigurable manufacturing system. *Journal of Intelligent Manufacturing* **23,** 1141–1155 (2010).
23. Choi, Y. C. & Xirouchakis, P. A holistic production planning approach in a reconfigurable manufacturing system with energy consumption and environmental effects. *International Journal of Computer Integrated Manufacturing* **28,** 379–394 (2014).
24. Cook, S. A. The complexity of theorem-proving procedures. *Proceedings of the Annual ACM Symposium on Theory of Computing,* 151–158 (1971).
25. Cormen, T. H., Leiserson, C. E., Revest, R. L. & Stein, C. *Introduction to algorithms* ISBN: 9783540241669 (2022).

26. Crawford, L. S. *et al.* Online reconfigurable machines. *AI Magazine* **34,** 73–88 (2013).
27. Dang, T. A. T., Bosch-Mauchand, M., Arora, N., Prelle, C. & Daaboul, J. Electromagnetic modular Smart Surface architecture and control in a microfactory context. *Computers in Industry* **81,** 152–170 (2016).
28. Dean, W. *Computational complexity theory* (2015).
29. Deif, A. M. & Elmaraghy, W. Investigating optimal capacity scalability scheduling in a reconfigurable manufacturing system. *International Journal of Advanced Manufacturing Technology* **32,** 557–562 (2007).
30. Doh, H. H., Yu, J. M., Kim, J. S., Lee, D. H. & Nam, S. H. A priority scheduling approach for flexible job shops with multiple process plans. *International Journal of Production Research* **51,** 3748–3764 (2013).
31. Dou, J., Li, J. & Su, C. Bi-objective optimization of integrating configuration generation and scheduling for reconfigurable flow lines using NSGA-II. *International Journal of Advanced Manufacturing Technology* **86,** 1945–1962 (2015).
32. Dou, J., Li, J., Xia, D. & Zhao, X. A multi-objective particle swarm optimisation for integrated configuration design and scheduling in reconfigurable manufacturing system. *International Journal of Production Research* **59,** 3975–3995 (2021).
33. Dou, J., Su, C. & Zhao, X. Mixed integer programming models for concurrent configuration design and scheduling in a reconfigurable manufacturing system. *Concurrent Engineering Research and Applications* **28,** 32–46 (2020).
34. Eguia, I., Molina, J. C., Lozano, S. & Racero, J. Cell design and multi-period machine loading in cellular reconfigurable manufacturing systems with alternative routing. *International Journal of Production Research* **55,** 2775–2790 (2016).
35. Eguia, I., Racero, J., Guerrero, F. & Lozano, S. Cell formation and scheduling of part families for reconfigurable cellular manufacturing systems using Tabu search. *Simulation* **89,** 1056–1072 (2013).
36. Eguía, I., Villa, G. & Lozano, S. Efficiency Assessment of Reconfigurable Manufacturing Systems. *Procedia Manufacturing* **11,** 1027–1034 (2017).
37. ElMaraghy, H. A. Flexible and reconfigurable manufacturing systems paradigms. *Flexible Services and Manufacturing Journal* **17,** 261–276 (2006).
38. Fan, J., Zhang, C., Liu, Q., Shen, W. & Gao, L. An improved genetic algorithm for flexible job shop scheduling problem considering reconfigurable machine tools with limited auxiliary modules. *Journal of Manufacturing Systems* **62,** 650–667 (2022).
39. Galán, R. Hybrid heuristic approaches for scheduling in reconfigurable manufacturing systems. *Studies in Computational Intelligence* **128,** 211–253 (2008).

40. Gao, S., Daaboul, J. & Le Duigou, J. Layout and scheduling optimization problem for a reconfigurable manufacturing system. *International Journal of Industrial Engineering and Management* **12,** 174–186 (2021).
41. Gao, S., Daaboul, J. & Le Duigou, J. Process Planning , Scheduling , and Layout Optimization for Multi-Unit Mass-Customized Products in Sustainable Reconfigurable Manufacturing System. *sustainability* **13,** 13323 (2021).
42. Garbie, I. H. A methodology for the reconfiguration process in manufacturing systems. *Journal of Manufacturing Technology Management* **25,** 891 –915 (2014).
43. Garey, M. R. & Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness* 1979.
44. Garey, M. & Johnson, D. Strong NP-completeness results: motivation, examples, and implications. *J. Assoc. Comput. Mach.* **25,** 499–508 (1978).
45. Gašpar, T., Kovač, I. & Ude, A. Optimal layout and reconfiguration of a fixturing system constructed from passive Stewart platforms. *Journal of Manufacturing Systems* **60,** 226–238 (2021).
46. Ghanei, S. & Algeddawy, T. An Integrated Multi-Period Layout Planning and Scheduling Model for Sustainable Reconfigurable Manufacturing Systems. *Journal of Advanced Manufacturing Systems* **19,** 31–64 (2020).
47. Ghobakhloo, M. *et al.* Identifying industry 5.0 contributions to sustainable development: A strategy roadmap for delivering sustainability values. *Sustainable Production and Consumption* **33,** 716–737 (2022).
48. Gonnermann, C., Hashemi-Petroodi, S. E., Thevenin, S., Dolgui, A. & Daub, R. A skill- and feature-based approach to planning process monitoring in assembly planning. *International Journal of Advanced Manufacturing Technology* **122,** 2645–2670 (2022).
49. Guan, X., Dai, X., Qiu, B. & Li, J. A revised electromagnetism-like mechanism for layout design of reconfigurable manufacturing system. *Computers and Industrial Engineering* **63,** 98–108 (2012).
50. Guo, D. *et al.* Synchronization-oriented reconfiguration of FPAI under graduation intelligent manufacturing system in the COVID-19 pandemic and beyond. *Journal of Manufacturing Systems* **60,** 893–902 (2021).
51. Gupta, A., Jain, P. K. & Kumar, D. Part family formation for reconfigurable manufacturing system using K-means algorithm. *International Journal of Internet Manufacturing and Services* **3,** 244–262 (2014).
52. Haddou Benderbal, H. & Benyoucef, L. Machine layout design problem under product family evolution in reconfigurable manufacturing environment: a two-phase-based AMOSA approach. *International Journal of Advanced Manufacturing Technology* **104,** 375–389 (2019).

53. Haddou Benderbal, H., Dahane, M. & Benyoucef, L. Flexibility-based multi-objective approach for machines selection in reconfigurable manufacturing system (RMS) design under unavailability constraints. *International Journal of Production Research* **55,** 6033–6051 (2017).
54. Haddou Benderbal, H., Dahane, M. & Benyoucef, L. Modularity assessment in reconfigurable manufacturing system (RMS) design: an Archived Multi-Objective Simulated Annealing-based approach. *International Journal of Advanced Manufacturing Technology* **94,** 729–749 (2017).
55. Han, K. H., Bae, S. M., Choi, S. H. & Lee, G. Parameter-driven rapid virtual prototyping of flexible manufacturing system. *International Journal of Mathematics and Computers in Simulation* **6,** 387–396 (2012).
56. Han, S., Chang, T. W., Hong, Y. S. & Park, J. Reconfiguration decision-making of IoT based reconfigurable manufacturing systems. *Applied Sciences (Switzerland)* **10,** 4807 (2020).
57. Harrison, R. & Colombo, A. W. *Collaborative automation from rigid coupling towards dynamic reconfigurable production systems* **1,** 184–192 (IFAC, 2005).
58. Hassan, N. & Bright, G. A Hybrid reconfigurable computer-integrated manufacturing cell for the production of mass customised parts. *South African Journal of Industrial Engineering* **23,** 139–150 (2012).
59. He, J. *Effective models and methods for stochastic disassembly line problems Thèse de doctorat* PhD thesis (Université Paris-Saclay, 2020).
60. Hees, A. & Reinhart, G. Approach for production planning in reconfigurable manufacturing systems. *Procedia CIRP* **33,** 70–75 (2015).
61. Hopcroft, J. E., Motwani, R. & Ullman, J. D. Introduction to automata theory, languages, and computation. *PEARSON Addison Wesley* (2006).
62. Horn, W. A. Technical Note—Minimizing Average Flow Time with Parallel Machines. *Operations Research* **21,** 846–847. ISSN: 0030-364X (1973).
63. Hromkovič, J. *Algorithms for hard problems: Introduction to combinatorial optimization, randomization, approximation, heuristics* (2013).
64. Hussain, K., Mohd Salleh, M. N., Cheng, S. & Shi, Y. Metaheuristic research: a comprehensive survey. *Artificial Intelligence Review* **52,** 2191–2233 (2019).
65. IBM. *ILOG CP Optimizer - Solve detailed scheduling problems using CP Optimizer* https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-cp-optimizer.
66. IBM. *What does CPLEX Optimization Studio do?* https://www.ibm.com/docs/en/icos/20.1.0?topic=cplex-what-does-do.
67. Ivanov, D. *et al.* A control approach to scheduling flexibly configurable jobs with dynamic structural-logical constraints. *IISE Transactions* **53,** 21–38 (2020).

68. Jouglet, A. & Carlier, J. Dominance rules in combinatorial optimization problems. *European Journal of Operational Research* **212,** 433–444 (2011).
69. Katoch, S., Chauhan, S. S. & Kumar, V. *A review on genetic algorithm: past, present, and future* **5,** 8091–8126 (2021).
70. Kazemisaboor, A., Aghaie, A. & Salmanzadeh, H. A simulation-based optimisation framework for process plan generation in reconfigurable manufacturing systems (RMSs) in an uncertain environment. *International Journal of Production Research* **60,** 2067–2085 (2021).
71. Khan, A. S. Problem-Specific Heuristics for Diagnosability and Inventory Analysis in a Reconfigurable Manufacturing System. *IEEE Access* **10,** 70032–70052 (2022).
72. Khan, A. S., Homri, L., Dantan, J. Y. & Siadat, A. Modularity-based quality assessment of a disruptive reconfigurable manufacturing system-A hybrid meta-heuristic approach. *International Journal of Advanced Manufacturing Technology* **115,** 1421–1444 (2021).
73. Khan, A. S., Khan, R., Saleem, W., Salah, B. & Alkhatib, S. Modeling and Optimization of Assembly Line Balancing Type 2 and E (SLBP-2E) for a Reconfigurable Manufacturing System. *Processes* **10** (2022).
74. Khan, I. S., Ghafoor, U. & Zahid, T. Meta-heuristic approach for the development of alternative process plans in a reconfigurable production environment. *IEEE Access* **9,** 113508–113520 (2021).
75. Khanna, K. & Kumar, R. Reconfigurable manufacturing system: a state-of-the-art review. *Benchmarking* **26,** 2608–2635 (2019).
76. Khettabi, I., Benyoucef, L. & Amine, M. Sustainable reconfigurable manufacturing system design using adapted multi-objective evolutionary-based approaches. *The International Journal of Advanced Manufacturing Technology* **115,** 3741–3759 (2021).
77. Khettabi, I., Benyoucef, L. & Amine Boutiche, M. Sustainable multi-objective process planning in reconfigurable manufacturing environment: adapted new dynamic NSGA-II vs New NSGA-III. *International Journal of Production Research* **60,** 6329–6349 (2022).
78. Khezri, A., Haddou Benderbal, H. & Benyoucef, L. Towards a sustainable reconfigurable manufacturing system (SRMS): multi-objective based approaches for process plan generation problem. *International Journal of Production Research* **59,** 4533–4558 (2020).
79. Kim, H.-W. & Lee, D.-H. A sample average approximation algorithm for selective disassembly sequencing with abnormal disassembly operations and random operation times. *The International Journal of Advanced Manufacturing Technology* **96,** 1341–1354 (2018).

80. Koren, Y. General RMS Characteristics. Comparison with Dedicated and Flexible Systems. *Reconfigurable manufacturing systems and transformable factories,* 27–45 (2006).
81. Koren, Y. *et al.* Reconfigurable Manufacturing Systems: Introduction. *CIRP annals* **48,** 527–540 (1999).
82. Kumar, M., Husain, M., Upreti, N. & Gupta, D. Genetic Algorithm: Review and Application. *International Journal of Information Technology and Knowledge Management* **2,** 451–454 (2020).
83. Kumar, R., Kumar, S. & Tiwari, M. K. An expert enhanced coloured fuzzy Petri net approach to reconfigurable manufacturing systems involving information delays. *International Journal of Advanced Manufacturing Technology* **26,** 922–933 (2005).
84. Labetoulle, J., Lawler, E., Lenstra, J. & Rinnooy Kan, A. Preemptive scheduling of uniform machines subject to release dates. *Progress in combinatorial optimization (Waterloo Ont., 1982), Academic Press,* 245–261 (1984).
85. Lambora, A., Gupta, K. & Chopra, K. Genetic Algorithm - A Literature Review. *International conference on machine learning, big data, cloud and parallel computing (COMITCon),* 380–384 (2019).
86. Landers, R. G., Min, B. K. & Koren, Y. Reconfigurable machine tools. *CIRP Annals - Manufacturing Technology* **50,** 269–274 (2001).
87. Lawler, E. A pseudo-polynomial algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics* **1,** 331–342 (1977).
88. Leng, J. *et al.* Digital twin-driven rapid reconfiguration of the automated manufacturing system via an open architecture model. *Robotics and Computer-Integrated Manufacturing* **63,** 101895 (2020).
89. Lenstra, J., Rinnooy Kan, A. & Brucker, P. Complexity of machine scheduling problems. *Annals of Discrete Mathematics* **1,** 343–362 (1977).
90. Li, J. W. Simulation study of coordinating layout change and quality improvement for adapting job shop manufacturing to CONWIP control. *International Journal of Production Research* **48,** 879–900 (2010).
91. Liaee, M. M. & Emmons, H. Scheduling families of jobs with setup times. *International Journal of Production Economics* **51,** 165–176. ISSN: 0925-5273 (1997).
92. Liles, D. H. & Huff, B. L. A computer based production scheduling architecture suitable for driving a reconfigurable manufacturing system. *Computers and Industrial Engineering* **19,** 1–5 (1990).
93. Mabkhot, M. M., Al-Ahmari, A. M., Salah, B. & Alkhalefah, H. Requirements of the smart factory system: A survey and perspective. *Machines* **6.** ISSN: 20751702 (2018).

94. Maganha, I., Silva, C. & Ferreira, L. The layout design in reconfigurable manufacturing systems: a literature review. *International Journal of Advanced Manufacturing Technology* **105,** 683–700 (2019).
95. Manafi, E., Tavakkoli-Moghaddam, R. & Mahmoodjanloo, M. A centroid opposition-based coral reefs algorithm for solving an automated guided vehicle routing problem with a recharging constraint. *Applied Soft Computing* **128,** 109504 (2022).
96. Maniraj, M., Pakkirisamy, V. & Parthiban, P. Optimisation of process plans in reconfigurable manufacturing systems using ant colony technique. *International Journal of Enterprise Network Management* **6,** 125–138 (2014).
97. Marschall, M. *et al.* Defining the Number of Mobile Robotic Systems Needed for Reconfiguration of Modular Manufacturing Systems via Simulation. *Machines* **10,** 316 (2022).
98. Massimi, E., Khezri, A., Haddou Benderbal, H. & Benyoucef, L. A heuristic-based non-linear mixed integer approach for optimizing modularity and integrability in a sustainable reconfigurable manufacturing environment. *International Journal of Advanced Manufacturing Technology* **108,** 1997–2020 (2020).
99. Mehrabi, M. G., Ulsoy, A. G. & Koren, Y. Reconfigurable manufacturing systems: key to future manufacturing. *Journal of Intelligent Manufacturing* **11,** 403–419. ISSN: 09565515 (2000).
100. Mirjalili, S. Genetic Algorithms. *Evolutionary Algorithms and Neural Networks,* 43–55. ISSN: 21971773 (2019).
101. Mitchell, M. An introduction to genetic algorithms. *MIT press* (1998).
102. Mo, F. *et al.* A framework for manufacturing system reconfiguration and optimisation utilising digital twins and modular artificial intelligence. *Robotics and Computer-Integrated Manufacturing* **82,** 102524 (2023).
103. Mohapatra, P., Benyoucef, L. & Tiwari, M. K. Integration of process planning and scheduling through adaptive setup planning: A multi-objective approach. *International Journal of Production Research* **51,** 7190–7208. ISSN: 00207543 (2013).
104. Mokhtari, H. Research on group search optimizers for a reconfigurable flow shop sequencing problem. *Neural Computing and Applications* **27,** 1657–1667 (2015).
105. Monma, C. L. & Potts, C. N. On the Complexity of Scheduling with Batch Setup Times. *Operations Research* **37,** 798–804. ISSN: 0030364X, 15265463. http://www.jstor.org/stable/171025 (2022) (1989).
106. Musharavati, F. & Hamouda, A. M. S. Simulated annealing with auxiliary knowledge for process planning optimization in reconfigurable manufacturing. *Robotics and Computer-Integrated Manufacturing* **28,** 113–131 (2012).

107. Musharavati, F. & Hamouda, A. S. M. Enhanced simulated-annealing-based algorithms and their applications to process planning in reconfigurable manufacturing systems. *Advances in Engineering Software* **45,** 80–90 (2012).
108. Naderi, B. & Azab, A. Production scheduling for reconfigurable assembly systems: Mathematical modeling and algorithms. *Computers and Industrial Engineering* **162,** 107741 (2021).
109. Oke, A. O., Abou-El-Hossein, K. & Theron, N. J. The design and development of a reconfigurable manufacturing system. *South African Journal of Industrial Engineering* **22,** 121–132. ISSN: 2224-7890 (2011).
110. Ong, S. K., Chang, M. M. L. & Nee, A. Y. C. Product disassembly sequence planning: state-of-the-art, challenges, opportunities and future directions. *International Journal of Production Research* **59,** 3493–3508 (2021).
111. Oztemel, E. & Gursev, S. Literature review of Industry 4.0 and related technologies. *Journal of Intelligent Manufacturing* **31,** 127–182 (2020).
112. Park, J. W., Shin, M. & Kim, D. Y. An Extended Agent Communication Framework for Rapid Reconfiguration of Distributed Manufacturing Systems. *IEEE Transactions on Industrial Informatics* **15,** 3845–3855 (2019).
113. Prasad, D. & Jayswal, S. C. Fuzzy logic based scheduling of the product families in reconfigurable manufacturing systems. *Songklanakarin Journal of Science and Technology* **43,** 1071–1077 (2021).
114. Prasad, D. & Jayswal, S. C. Reconfigurability consideration and scheduling of products in a manufacturing industry. *International Journal of Production Research* **56,** 6430–6449 (2017).
115. Rahman, A. A. A., Adeboye, O. J., Tan, J. Y., Salleh, M. R. & Rahman, M. A. A. An Optimization Approach for Predictive-Reactive Job Shop Scheduling of Reconfigurable Manufacturing Systems. *Jordan Journal of Mechanical and Industrial Engineering* **16,** 793–809 (2022).
116. Raidl, G. R., Puchinger, J. & Blum, C. *Handbook of Metaheuristics* (2010).
117. Rehman, A. U. An approach to assess robustness of a reconfigurable manufacturing system. *International Journal of Industrial and Systems Engineering* **33,** 542–557 (2019).
118. Rehman, A. U. & Subash Babu, A. Manufacturing competitiveness assessment for business excellence. *International Journal of Business Excellence* **1,** 231–261 (2008).
119. Rehman, A. U. & Subash Babu, A. The evaluation of manufacturing systems using concordance and disconcordance properties. *International Journal of Services and Operations Management* **5,** 326–349 (2009).

120. Renna, P. & Ambrico, M. Design and reconfiguration models for dynamic cellular manufacturing to handle market changes. *International Journal of Computer Integrated Manufacturing* **28,** 170–186 (2015).
121. Reza Abdi, M. Layout configuration selection for reconfigurable manufacturing systems using the fuzzy AHP. *International Journal of Manufacturing Technology and Management* **17,** 149–165 (2009).
122. Rinnooy Kan, A. *Machine sequencing problem: classification, complexity and computation* (The Hague, 1976).
123. Rossi, F., Van Beek, P. & Walsh, T. Constraint programming. *Foundations of Artificial Intelligence* **3,** 181–211 (2008).
124. Roughgarden, T. Beyond worst-case analysis. *Communications of the ACM* **62,** 88–96. ISSN: 15577317 (2019).
125. Ruiz-Torres, A. J., Paletta, G. & Adenso-Díaz, B. Hybrid two stage flowshop scheduling with secondary resources based on time buckets. *International Journal of Production Research* **60,** 1954–1972 (2021).
126. Sallez, Y., Berger, T. & Bonte, T. The concept of "safety bubble" for reconfigurable assembly systems. *Manufacturing Letters* **24,** 77–81 (2020).
127. Shabaka, A. I. & Elmaraghy, H. A. Generation of machine configurations based on product features. *International Journal of Computer Integrated Manufacturing* **20,** 355–369 (2007).
128. Smith, W. E. Various optimizers for single-stage production. *Naval Research Logistics Quarterly* **3,** 59–66 (1956).
129. Tan, W. & Khoshnevis, B. Integration of process planning and scheduling - a review. *Journal of Intelligent Manufacturing* **11,** 51–63 (1998).
130. Tang, Y., Zhou, M. C., Zussman, E. & Caudill, R. Disassembly modeling, planning, and application. *Journal of Manufacturing Systems* **21,** 200–217 (2002).
131. Touzout, F. A. & Benyoucef, L. Multi-objective sustainable process plan generation in a reconfigurable manufacturing environment: exact and adapted evolutionary approaches. *International Journal of Production Research* **57,** 2531–2547 (2018).
132. Touzout, F. & Benyoucef, L. Multi-objective multi-unit process plan generation in a reconfigurable manufacturing environment: a comparative study of three hybrid metaheuristics. *International Journal of Production Research* **57,** 7520–7535 (2019).
133. Um, J, Suh, S.-H. & Stroud, I. STEP-NC machine tool data model and its applications. *International Journal of Computer Integrated Manufacturing* **29,** 1058–1074 (2016).
134. Vahedi-Nouri, B., Tavakkoli-Moghaddam, R., Hanzálek, Z. & Dolgui, A. Workforce planning and production scheduling in a reconfigurable manufacturing system facing the COVID-19 pandemic. *Journal of Manufacturing Systems* **63,** 563–574 (2022).

135. Vavrík, V. *et al.* Design of manufacturing lines using the reconfigurability principle. *Mathematics* **8** (2020).
136. Wallace, M. Practical applications of constraint programming. *Constraints* **1,** 139–168. ISSN: 13837133 (1996).
137. Wang, L., Zhu, B., Wang, Q. & Zhang, Y. Modeling of hot stamping process procedure based on finite state machine (FSM). *International Journal of Advanced Manufacturing Technology* **89,** 857–868 (2017).
138. Wang, Y., Zhang, G., Wang, J., Liu, P. & Wang, N. Reconfigurable Machine Tool Design for Box-Type Part Families (2021).
139. Wei, X., Yuan, S. & Ye, Y. Optimizing facility layout planning for reconfigurable manufacturing system based on chaos genetic algorithm. *Production and Manufacturing Research* **7,** 109–124 (2019).
140. Williams, I. D. & Shittu, O. S. Development of Sustainable Electronic Products, Business Models and Designs Using Circular Economy Thinking. *Detritus* **21,** 45–54. ISSN: 26114135 (2022).
141. Xia, T., Xi, L., Pan, E. & Ni, J. Reconfiguration-oriented opportunistic maintenance policy for reconfigurable manufacturing systems. *Reliability Engineering and System Safety* **166,** 87–98 (2017).
142. Xiaowen, X., Beirong, Z. & Wei, X. Configuration optimization method of reconfigurable manufacturing systems. *Research Journal of Applied Sciences, Engineering and Technology* **6,** 1389–1393 (2013).
143. Xie, N., Li, A. & Xue, W. Cooperative optimization of reconfigurable machine tool configurations and production process plan. *Chinese Journal of Mechanical Engineering* **25,** 982–989 (2012).
144. Yang, J., Liu, F., Dong, Y., Cao, Y. & Cao, Y. Multiple-objective optimization of a reconfigurable assembly system via equipment selection and sequence planning. *Computers and Industrial Engineering* **172,** 108519 (2022).
145. Yang, S., Wang, J., Xin, L. & Xu, Z. Real-time and concurrent optimization of scheduling and reconfiguration for dynamic reconfigurable flow shop using deep reinforcement learning. *Journal of Manufacturing Science and Technology* **40,** 243–252 (2023).
146. Yang, S. & Xu, Z. Intelligent scheduling and reconfiguration via deep reinforcement learning in smart manufacturing. *International Journal of Production Research* **60,** 373–392 (2021).
147. Yazdani, M. A., Khezri, A. & Benyoucef, L. Process and production planning for sustainable reconfigurable manufacturing systems (SRMSs): multi-objective exact and heuristic-based approaches. *International Journal of Advanced Manufacturing Technology* **119,** 4519–4540 (2022).

148. Ye, H. & Liang, M. Simultaneous modular product scheduling and manufacturing cell reconfiguration using a genetic algorithm. *Journal of Manufacturing Science and Engineering, Transactions of the ASME* **128,** 984–995 (2006).
149. Yelles-Chaouche, A. R., Gurevsky, E., Brahimi, N. & Dolgui, A. Reconfigurable manufacturing systems from an optimisation perspective: a focused review of literature. *International Journal of Production Research* **59,** 6500–6418 (2020).
150. Yu, J. M. *et al.* Input sequencing and scheduling for a reconfigurable manufacturing system with a limited number of fixtures. *International Journal of Advanced Manufacturing Technology* **67,** 157–169 (2013).
151. Zhan, R. *et al.* Configuring a seru production system to match supply with volatile demand. *Applied Intelligence,* 12–20 (2022).
152. Zhang, D. Z., Anosike, A. & Lim, M. K. Dynamically integrated manufacturing systems (DIMS) - A multiagent approach. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans* **37,** 824–850 (2007).
153. Zhao, X., Wang, J. & Luo, Z. A stochastic model of a reconfigurable manufacturing system - Part 2: Optimal configurations. *International Journal of Production Research* **38,** 2829–2842 (2000).
154. Zhao, X., Wang, J. & Luo, Z. A stochastic model of a reconfigurable manufacturing system Part 1: A framework. *International Journal of Production Research* **38,** 2273–2285 (2000).

Appendices

7.2.1 Appendice A

	M_1			M_2			M_3			M_4			M_5		
	M_11	M_12	M_13	M_21	M_22	M_23	M_31	M_32	M_33	M_41	M_42	M_43	M_51	M_52	M_53
Op1	∞	∞	27	∞	∞	23	∞	∞	23	∞	∞	∞	∞	∞	∞
Op2	30	∞	∞	31	∞	∞	28	∞	∞	15	∞	∞	15	∞	∞
Op3	26	∞	∞	30	∞	∞	28	∞	∞	15	∞	∞	15	∞	∞
Op4	∞	∞	30	∞	∞	27	∞	∞	24	∞	∞	∞	∞	∞	∞
Op5	22	∞	∞	31	∞	∞	26	∞	∞	∞	∞	∞	∞	∞	∞
Op6	32	∞	∞	27	∞	∞	29	∞	∞	∞	13	∞	∞	13	∞
Op7	25	∞	∞	18	∞	∞	18	∞	∞	∞	∞	∞	∞	∞	∞
Op8	∞	∞	22	∞	∞	18	∞	∞	25	∞	∞	∞	∞	∞	∞
Op9	∞	∞	17	∞	∞	23	∞	∞	24	∞	∞	9	∞	∞	9
Op10	∞	32	∞	∞	29	∞	∞	18	∞	∞	∞	∞	∞	∞	∞

Table 45: Operation time (time unit).

	M_1			M_2			M_3			M_4			M_5		
	M_11	M_12	M_13	M_21	M_22	M_23	M_31	M_32	M_33	M_41	M_42	M_43	M_51	M_52	M_53
Op1	∞	∞	8	∞	∞	7	∞	∞	7	∞	∞	∞	∞	∞	∞
Op2	9	∞	∞	9	∞	∞	9	∞	∞	11	∞	∞	11	∞	∞
Op3	8	∞	∞	8	∞	∞	8	∞	∞	11	∞	∞	11	∞	∞
Op4	∞	∞	9	∞	∞	8	∞	∞	7	∞	∞	∞	∞	∞	∞
Op5	7	∞	∞	9	∞	∞	8	∞	∞	∞	∞	∞	∞	∞	∞
Op6	10	∞	∞	8	∞	∞	9	∞	∞	∞	15	∞	∞	15	∞
Op7	7	∞	∞	6	∞	∞	5	∞	∞	∞	∞	∞	∞	∞	∞
Op8	∞	∞	7	∞	∞	6	∞	∞	7	∞	∞	∞	∞	∞	∞
Op9	∞	∞	5	∞	∞	7	∞	∞	7	∞	∞	11	∞	∞	11
Op10	∞	10	∞	∞	9	∞	∞	6	∞	∞	∞	∞	∞	∞	∞

Table 46: Operation cost (cost unit).

	M_1	M_2	M_3	M_4	M_5
Cost	2	2	2	5	5
Time	3	3	3	9	9

Table 47: Machine reconfiguration.

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}	J_{11}	J_{12}
Due date	500	500	500	500	500	500	500	500	500	500	500	500

Table 48: Due dates.

Layout 1

	M_1	M_2	M_3	M_4	M_5
M_1	0	2	4	6	8
M_2	2	0	2	4	6
M_3	4	2	0	2	4
M_4	6	4	2	0	2
M_5	8	6	4	2	0

Layout 2

	M_1	M_2	M_3	M_4	M_5
M_1	0	4	2	6	8
M_2	6	0	2	2	6
M_3	2	2	0	4	6
M_4	6	2	4	0	6
M_5	8	4	6	2	0

Layout 3

	M_1	M_2	M_3	M_4	M_5
M_1	0	3	3	6	2
M_2	3	0	7	3	2
M_3	3	6	0	3	2
M_4	5	3	3	0	2
M_5	2	2	2	2	0

Layout 4

	M_1	M_2	M_3	M_4	M_5
M_1	0	2	3	7	4
M_2	2	0	3	4	8
M_3	2	2	0	2	2
M_4	6	4	3	0	2
M_5	5	7	3	2	0

Layout 5

	M_1	M_2	M_3	M_4	M_5
M_1	0	2	2	2	2
M_2	2	0	3	6	3
M_3	2	3	0	3	6
M_4	2	5	3	0	3
M_5	2	3	7	3	0

Table 49: Material handling times.

	Λ_1	Λ_2	Λ_3	Λ_4	Λ_5
Λ_1	0	6	10	7	8
Λ_2	6	0	8	7	11
Λ_3	8	8	0	8	6
Λ_4	6	7	9	0	8
Λ_5	7	8	6	10	0

Table 50: Layout reconfiguration cost.

	Λ_1	Λ_2	Λ_3	Λ_4	Λ_5
Λ_1	0	7	13	13	16
Λ_2	9	0	14	12	16
Λ_3	14	14	0	16	10
Λ_4	11	13	18	0	14
Λ_5	13	15	9	18	0

Table 51: Layout reconfiguration time.

MH cost	2
Tardiness penalty	1

Table 52: Material handling cost to be multiplied by MH time and tardiness penalty.

7.2.2 Appendice B

		Machine	Configuration	Sequence	Layout	Stat time	Completion Time
J_1	O_1	1	3	31	2	257	284
	O_2	2	1	13		398	428
	O_3	2	1	25		428	455
	O_4	3	1	14		495	513
	O_5	2	3	43		372	395
	O_6	1	2	15		460	492
J_2	O_1	1	1	7	2	59	89
	O_2	3	1	44		176	202
	O_3	1	1	17		89	121
	O_4	3	3	67		229	254
	O_5	5	3	9		257	266
	O_6	3	2	68		350	368
J_3	O_1	5	1	32	2	13	28
	O_2	2	3	69		30	57
	O_3	5	2	49		59	72
	O_4	2	1	37		149	167
	O_5	2	3	46		74	97
	O_6	2	2	70		460	489
J_4	O_1	3	3	35	2	205	228
	O_2	4	1	16		287	302
	O_3	5	2	45		460	473
	O_4	3	1	51		329	347
	O_5	4	3	55		428	437
	O_6	3	2	8		440	458
J_5	O_1	1	1	26	2	13	43
	O_2	2	1	50		194	225
	O_3	2	1	61		257	284
	O_4	1	3	19		311	333
	O_5	5	3	62		345	354
	O_6	3	2	52		414	432
J_6	O_1	3	1	10	2	13	41
	O_2	3	3	1		98	122
	O_3	5	2	11		363	376
	O_4	3	1	27		287	305
	O_5	3	3	56		149	173
	O_6	2	2	47		311	340
J_7	O_1	3	3	64	2	126	149
	O_2	4	1	20		311	326
	O_3	5	2	59		376	389
	O_4	3	1	34		393	411
	O_5	4	3	36		414	423
	O_6	3	2	58		458	476
J_8	O_1	3	1	53	2	41	69
	O_2	3	1	38		69	95
	O_3	2	1	28		167	194
	O_4	2	3	22		126	144
	O_5	1	3	48		284	301
	O_6	1	2	57		336	368
J_9	O_1	4	1	71	2	126	141
	O_2	1	3	54		196	226
	O_3	5	2	29		167	180
	O_4	1	1	12		229	254
	O_5	5	3	33		311	320
	O_6	2	2	65		340	369
J_10	O_1	1	3	5	2	169	196
	O_2	3	1	72		257	285
	O_3	4	2	60		336	349
	O_4	3	1	39		311	329
	O_5	5	3	63		287	296
	O_6	3	2	40		372	390
J_11	O_1	4	1	2	2	196	211
	O_2	2	1	41		225	256
	O_3	5	2	30		389	402
	O_4	2	3	3		287	305
	O_5	4	3	23		363	372
	O_6	1	2	66		428	460
J_12	O_1	4	1	24	2	74	89
	O_2	2	3	42		97	124
	O_3	5	2	21		126	139
	O_4	1	1	6		141	166
	O_5	5	3	18		336	345
	O_6	1	2	4		372	404

Table 53: Outputs of example 1.

7.2.3 Appendice C

		Machine	Configuration	Sequence	Layout	Stat time	Completion Time
J_1	O_1	3	3	67	4	19	42
	O_2	4	1	37		234	249
	O_3	2	1	43		396	423
	O_4	1	1	55		347	372
	O_5	1	3	1		94	111
	O_6	2	2	45		426	455
J_2	O_1	4	1	31	4	219	234
	O_2	1	1	19		372	394
	O_3	4	2	69		297	310
	O_4	1	3	44		440	462
	O_5	1	3	23		462	479
	O_6	3	2	13		481	499
J_3	O_1	5	1	49	4	77	92
	O_2	1	3	25		189	219
	O_3	5	2	5		161	174
	O_4	3	1	33		419	437
	O_5	2	3	68		252	275
	O_6	3	2	46		324	342
J_4	O_1	1	3	32	4	46	73
	O_2	4	1	27		204	219
	O_3	2	1	70		369	396
	O_4	3	1	41		401	419
	O_5	1	3	56		77	94
	O_6	3	2	26		440	458
J_5	O_1	5	1	15	4	92	107
	O_2	1	1	50		161	183
	O_3	5	2	42		137	150
	O_4	1	3	57		401	423
	O_5	1	3	61		423	440
	O_6	2	2	14		455	484
J_6	O_1	4	1	20	4	45	60
	O_2	3	3	52		137	161
	O_3	3	1	7		105	134
	O_4	1	1	38		247	272
	O_5	1	3	58		292	309
	O_6	1	2	62		312	344
J_7	O_1	1	3	2	4	19	46
	O_2	2	1	34		219	249
	O_3	2	1	17		324	351
	O_4	2	1	64		351	369
	O_5	1	3	71		141	158
	O_6	3	2	59		188	206
J_8	O_1	4	1	63	4	19	34
	O_2	2	1	35		77	108
	O_3	3	1	9		45	74
	O_4	1	3	47		111	133
	O_5	2	3	60		298	321
	O_6	3	2	51		458	476
J_9	O_1	3	1	36	4	77	105
	O_2	3	3	39		161	185
	O_3	4	2	18		310	323
	O_4	3	1	65		209	227
	O_5	4	3	53		341	350
	O_6	3	2	66		380	398
J_10	O_1	2	3	4	4	19	42
	O_2	2	1	40		159	189
	O_3	2	1	72		189	216
	O_4	1	1	21		222	247
	O_5	1	3	8		275	292
	O_6	3	2	54		298	316
J_11	O_1	4	1	3	4	189	204
	O_2	3	1	22		227	253
	O_3	4	2	28		284	297
	O_4	3	3	16		256	281
	O_5	4	3	29		332	341
	O_6	3	2	6		344	362
J_12	O_1	5	1	24	4	16	31
	O_2	2	3	10		111	138
	O_3	2	1	30		46	73
	O_4	2	1	48		141	159
	O_5	2	3	11		275	298
	O_6	3	2	12		362	380

Table 54: Outputs of example 2.