INSTITUTO POLITÉCNICO DE BRAGANÇA Escola Superior de Tecnologia e Gestão

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Development of Scheduling and Routing Systems for Multiple Autonomous Robots in the Industry

João Guilherme Martins Silva - a61480

Dissertation presented to the School of Technology and Management to obtain the Master's Degree in Informatics within the scope of the double degree program with the Federal Technological University of Paraná

Supervisors:

Prof. José Lima

Prof. João Alberto Fabro

Prof. João Braun

Bragança

September 2025

INSTITUTO POLITÉCNICO DE BRAGANÇA Escola Superior de Tecnologia e Gestão

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Development of Scheduling and Routing Systems for Multiple Autonomous Robots in the Industry

João Guilherme Martins Silva - a61480

Dissertation presented to the School of Technology and Management to obtain the Master's Degree in Informatics within the scope of the double degree program with the Federal Technological University of Paraná

Supervisors:

Prof. José Lima

Prof. João Alberto Fabro

Prof. João Braun

Bragança

September 2025

Dedication

I dedicate this work to my family, especially to my parents Joel and Mariane, as this would never have been possible without their unwavering support. To my aunt Mauren, who helped me organize my thoughts and ideas.

I dedicate this work to my friends, both in Curitiba and Bragança, who suggested small improvements, showed me their ongoing work or generally helped me forget about this project for long enough for new ideas to come in. They are too many to name, and each one of their small contributions allowed me to get here.

Acknowledgment

I would like to express my gratitude to my supervisors João Fabro, José Lima and João Braun for their guidance and support during the development of this work, and for their effort and patience when correcting the final text of this dissertation. I extend this gratitude to Prof. Ana Pereira, whose key insights were fundamental to this work.

I also wish to thank the Universidade Tecnológica do Paraná (UTFPR) and the Instituto Politécnico de Bragança (IPB) for creating and accepting me into this Double Degree program, which made all of this possible in the first place.

Abstract

With transport logistics representing the majority of logistic expenses in industrial warehouses, task allocation and scheduling strategies must take into account travel times to achieve an effective optimization. This work represents the RobotAtFactory competition as a Hybrid Flow Shop Scheduling Problem with Transportation Resources, evaluates the ideal number of resources for the competition and proposes a joint scheduling and routing algorithm. The impact of the number of available vehicles was simulated while controlling for variations in machine processing time, in the initial setup of parts and in the effectiveness of the scheduling strategy. The results showed that, for the competition layout, the usage of two robots significantly reduced the makespan even when guided by simple strategies, while higher number of robots demanded a higher complexity while offering diminishing returns. The joint scheduling and routing approach did not improve upon the baseline, possibly due to a lack of complexity in the environment.
Keywords: scheduling; Hybrid Flow Shop Scheduling; path-planning; TEA*; RobotAtFactory.

Resumo

Com a logística de transporte representando a maior parte das despesas com logística em armazéns industriais, estratégias para a alocação e o escalonamento de tarefas precisam levar em conta os tempos de deslocamento para alcançar uma otimização efetiva. Este trabalho representa a competição RobotAtFactory como um Problema de Escalonamento de Flow Shop Híbrido com Recursos de Transporte, avalia o número ideal de recursos para a competição e propõe um algoritmo para o escalonamento e roteamento conjunto. O impacto do número de veículos disponíveis foi simulado controlando para variações no tempo de processamento das máquinas, no arranjo inicial de partes e na efetividade da estratégia de escalonamento. Os resultados mostram que, para o ambiente da competição, o uso de dois robôs reduz significativamente o tempo total mesmo quando guiado por estratégias simples, enquanto números maiores de robôs demandam uma complexidade maior e oferecem retornos progressivamente menores. A abordagem conjunta de escalonamento e roteamento não apresentou melhorias, possivelmente por causa de uma falta de complexidade no ambiente.
Palavras-chave: Escalonamento; Escalonamento de Flow Shop Híbrido; Planejamento de caminho; TEA*; RobotAtFactory.

Contents

1 Introduction **1**

1.1 Objectives . . . 3

2 Context and Technologies **4**

2.1 RobotAtFactory . . . 4

2.2 Scheduling . . . 7

2.2.1 Job Shop Scheduling Problem (JSSP) . . . 7

2.2.2 Incorporating Machine Allocation . . . 10

2.2.3 Incorporating Transportation . . . 11

2.2.4 Graph Representation for Scheduling Problems . . . 12

2.3 Path Planning . . . 15

2.3.1 Environment Modeling . . . 16

2.3.2 A* Algorithm . . . 17

2.3.3 Time-Extended A* Algorithm . . . 18

2.4 Genetic Algorithm Optimization . . . 19

2.4.1 Encoding . . . 21

2.4.2 Sampling . . . 21

2.4.3 Selection . . . 21

2.4.4 Mutation . . . 22

2.4.5 Crossover . . . 22

2.4.6 Repair . . . 22

2.5 Simulation . . . 23
2.5.1 Agent-Based Simulation . . . 23
2.5.2 GAMA Simulator . . . 24
2.6 Related Works . . . 25

3 Methodology **27**
3.1 Problem Definition and Research Questions . . . 27
3.2 The RobotAtFactory Competition as a Hybrid Flow Shop Scheduling Problem . . . 28
3.2.1 Including the lack of buffers in the graph . . . 30
3.3 Scenarios . . . 31
3.3.1 Scenario 1: Pure Scheduling . . . 31
3.3.2 Scenario 2: Scheduling with Complete Space Separation . . . 32
3.3.3 Scenario 3: Routing with Time-Extended A* . . . 33
3.3.4 Scenario 4: Joint Scheduling and Routing with TEA* . . . 34

4 Development **36**
4.1 A Genetic Algorithm Approach to Scheduling . . . 36
4.2 Graph Implementation . . . 43
4.2.1 Building a Base Graph from the Environment . . . 43
4.2.2 Integrating the Decision Variables . . . 44
4.2.3 Graph Weight Calculation . . . 46
4.3 Simulation Implementation . . . 49
4.3.1 The `Shop` Model . . . 51
4.3.2 The `ExperimentValidator` Model . . . 54

5 Results and Discussions **55**
5.1 Results for Research Question 1 . . . 55
5.2 Results for Research Question 2 . . . 60
5.3 Results for Research Question 3 . . . 62

5.4 Results for Research Question 4 66
5.5 Summary of the Results 67

6 Conclusions **69**

List of Figures

2.1 The Raf competition environment Source: [4] 5
2.2 Diagram of the flow of parts 5
2.3 Diagram and 3D model of a RaF machine 6
2.4 Machine constraints for input and output 7
2.5 Example of a job shop problem 8
2.6 Solution for Figure 2.5 that minimizes the time for completion of all jobs . 9
2.7 Solution for Figure 2.5 that minimizes the time for completion of one job . 9
2.8 An example of two possible transportation tasks 12
2.9 The difference between two sequences of actions 12
2.10 Disjunctive Graph representing the FJSPT 13
2.11 Environment constraints of Figure 2.10 14
2.12 Decision constraints of Figure 2.10 14
2.13 Solution for the problem case of Figure 2.10 15
2.14 Gantt chart for the solution from Figure 2.13 15
2.15 TEA* layer representation and neighborhood search. Source: [20] 19
2.16 Diagram of Genetic Algorithm execution 20
2.17 Example of repetitive permutation 21
2.18 Interface of the GAMA simulator 24
2.19 Diagram of a micro-model. Source: [29] 25

3.1 Converting the competition labels to flow shop stages 29
3.2 Spatial Graph Adapted from the Robot at Factory competition 29

3.3 Difference between graph representations for infinite and zero buffer scenarios 30
3.4 Diagram of scenario 1 31
3.5 In Scenario 1 the routes are based only in the shortest path to the next destination, creating situations where collisions may occur 32
3.6 Representation of vehicle responsibility used in scenario 2 32
3.7 Possible division of space for multiple vehicles 33
3.8 Diagram of Scenario 3 33
3.9 Situation where assumed and collision-free paths differ 34
3.10 Diagram of scenario 4 34

4.1 Chromosome encoding 37
4.2 Example of chromosome encoding 38
4.3 Conversion of the job ordering and vehicle allocation vectors into the ordered task list of each vehicle 39
4.4 A range of genes is selected from the donor chromosome 40
4.5 A section from the donor chromosome is inserted in the receiver chromosome 41
4.6 The child chromosome after the execution of the Generalized Order Crossover 41
4.7 Mutation on a machine allocation array 42
4.8 Base graph variation from the environment variables 44
4.9 The edge enforcing the precedence constraint between O_{xy} and O_{kw} when they are in the same machine 45
4.10 The edge enforcing the constraint imposed by the travel time from the initial position to the first task of a vehicle 45
4.11 The edge enforcing the precedence constraint between T_{xy} and T_{kw} when they are in the same vehicle 46
4.12 Topological array of Figure 3.3b 48
4.13 Relation between optimization algorithm and simulation 49
4.14 Schema of the database used to store optimization and simulation results . 50
4.15 Screenshot of the `Shop` simulation running 51

4.16 Vehicle action flowchart 53

5.1 Average Makespan by Number of Vehicles and Production/Transport time ratio for Scenario 1 56
5.2 Increase in Efficiency when increasing the number of vehicles, for different processing times 58
5.3 Efficiency gain for each homogeneous part combination separated for each production time 59
5.4 Average Makespan by Number of Vehicles and Production Time for Scenario 2 61
5.5 Efficiency gain for multiple vehicles when compared with the same scenario for a single vehicle 61
5.6 Categorized Difference in Makespan Between Scenarios 1 and 3 as shown in Table 5.5 65
5.7 Average difference between the total makespan of scenarios 3 and 4 67

Acronyms

ESTiG Escola Superior de Tecnologia e Gestão.

FJSSP Flexible Job Shop Scheduling Problem.

FMS Flexible Manufacturing System.

FSSP Flow Shop Scheduling Problem.

GA Genetic Algorithm.

HFSSP Hybrid Flow Shop Scheduling Problem.

IPB Instituto Politécnico de Bragança.

JSSP Job Shop Scheduling Problem.

OR Operations Research.

RaF RobotAtFactory.

TEA* Time-Extended A*.

UTFPR Universidade Tecnológica Federal do Paraná.

Chapter 1

Introduction

With a world that is each day more interconnected, the current industry logistics are always pushed to the limit. It is to solve these logistical problems that the field of Operations Research (OR) is built on. Initially created as a separate and identifiable field to solve the logistical problems of the Allies during World War II, OR evolved during the following decades into a more general field tasked with designing, analyzing and improving the performance or operation of systems through the use of mathematical models [1].

The traditional class of models used under OR to deal with manufacturing environments are the optimal job scheduling problems. These problems are concerned with the optimal allocation and sequencing of a limited number of resources to tasks. Despite the name, it is usually impossible to find optimal solutions to these problems, as they have been established as NP-Hard problems [2]. This computational complexity forces most systems to rely on heuristics to provide good results in a reasonable time frame.

These traditional scheduling models, however, assume that the movement of tasks between machines can be ignored, concerning themselves only with the time each task takes to finish processing in a given machine. It has been known for a long time that pickup and deliver operations account for more than 55% of warehouse logistics expenses [3], which signals that this simplification can compromise the underlying system by returning suboptimal schedules.

To account for the additional problem of transportation, the classical formulations

where slowly incremented with transportation resources. These representations correctly estimate the time spent on movement between tasks, but do not represent the possibility of collisions between these moving agents. This was not a problem when the field of scheduling with transportation was focused on optimizing the movement of human workers in factories and industrial warehouses, but today there is a trend towards the complete automation of these places with intelligent robots. This change transforms the issue of planning paths without conflict, a task so trivial for humans, into a challenge that must be solved for the robotic workers.

Although path planning for single robots in a known environment is a largely solved problem (due to the guarantee from algorithms such as A* of an optimal path), this is definitely not the case for multi-robot systems. This poses a clear challenge when designing these systems, as a decision must be made if the possible benefits that multiple robots bring are enough to justify the added complexity needed to ensure that these systems function without defects.

As a study environment for the challenges involved in warehouse scheduling this research uses the RobotAtFactory (RaF) competition [4], a national robotics competition where one or more robots must move boxes around efficiently without external communication. These boxes (depending on their type) may need to pass through multiple machines before reaching their final destination, resembling the flow of operations inside a real industrial warehouse.

This added complexity can also be seen as an opportunity to be explored. As these two optimization problems (the scheduling with transportation resources and the multi-vehicle path planning) are now part of the same problem domain, the shared knowledge of both problems can be used to produce better solutions for these situations. This opportunity forms the basis of this research: Does the knowledge of the exact paths to be traveled by these robots enables the optimizer to produce better task schedules?

1.1 Objectives

As mentioned before, this dissertation uses as a study environment the RaF competition, where a robot must move boxes around efficiently without external communication. To that end, the optimal job scheduling representation that best represents the competition flow must be identified and its variables adjusted to match the competition environment.

In order to determine if the knowledge of the exact paths in a multi-robot system improve scheduling, the need for a multi-robot system in the first place must be clearly established.

This research proposes to verify this need with the aid of a scheduling optimizer based on a genetic algorithm and a separate multi-vehicle path-planning based on the Time-Extended A* (TEA*) algorithm. For situations where there is an advantage to multiple vehicles, a joint scheduling and routing algorithm (made of the combination of the genetic task scheduler and the TEA* algorithm) is proposed to then verify if the exact paths confer an advantage to task scheduling.
As such, this dissertation has the following objectives:

1. Describe the RaF competition as an optimal job scheduling problem, considering the peculiarities that are not accounted for in most representations.

2. Evaluate the ideal number of robots given the competition complexity.

3. Propose a joint scheduling and routing algorithm and evaluate if it brings any advantage.

The remaining work is divided into five chapters. Chapter 2 describes the context relevant to this work and the technologies used from each area. Chapter 3 describes the methodology followed and the scenarios proposed to answer this dissertation's objectives. Chapter 4 describes the implementation of the tools used and the difficulties encountered during their development. Chapter 5 describes the results that were obtained and interprets these results. Finally, Chapter 6 presents the conclusion and proposes future works.

Chapter 2

Context and Technologies

This chapter presents the technologies and algorithms relevant to scheduling, path planning and simulation. Section 2.1 presents the RobotAtFactory competition. Section 2.2 presents an evolution of scheduling problems from the job shop scheduling problem to the hybrid flow shop scheduling problem with transportation resources. Section 2.3 presents key concepts in path planning and the path planning algorithms used in this work. Section 2.4 presents the functioning of the Genetic Algorithm optimization. Section 2.5 presents the Agent-Based Simulation framework for industrial simulation and the GAMA simulator. Finally, Section 2.6 presents the approaches used in related works.

2.1 RobotAtFactory

The RobotAtFactory (RaF) competition [4] is an annual national robotics competition where participants must design one or more autonomous robots to transport materials between machines and warehouses within an environment inspired by a factory floor for Industry 4.0.

The competition's environment is composed of one incoming warehouse, one outgoing warehouse and four machines as shown in Figure 2.1.

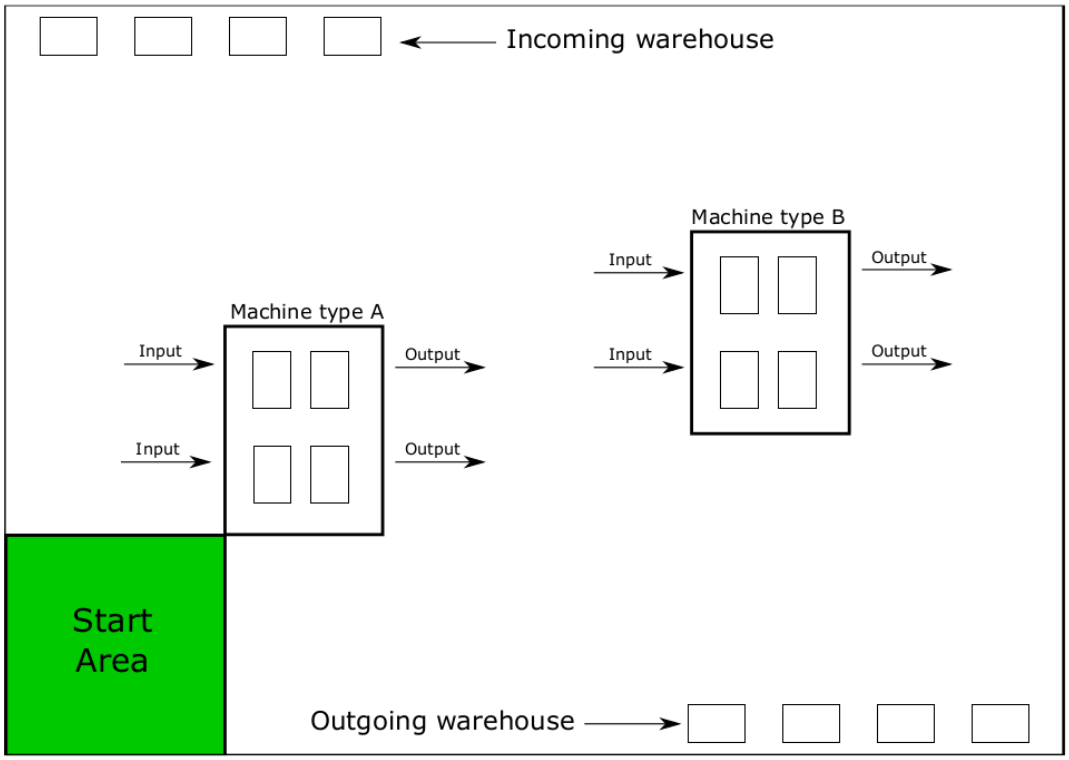


Figure 2.1: The Raf competition environment Source: [4]

The material to be transported through the various machines is refereed to as a "part" and is present in 3 variations: a "raw" part (denoted by the color red ■), an "intermediate" part (denoted by the color green ■) and a "final" part (denoted by the color blue ■). Parts must be processed from the "raw" stage to the "final" stage following the sequence in Figure 2.2.

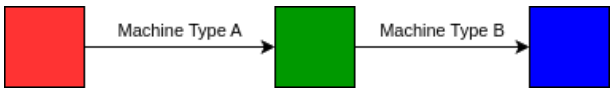
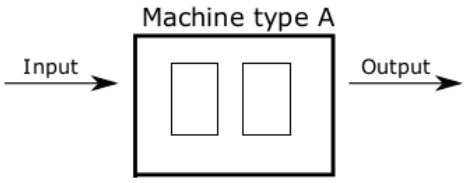


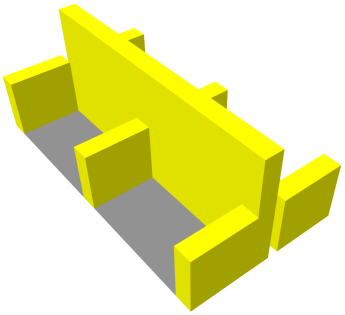
Figure 2.2: Diagram of the flow of parts

The incoming warehouse is where the parts are placed at the start of the competition. Each of the four slots holds one part and there is no restriction on which type of part must be there. The outgoing warehouse, on the other hand, is where all parts must be placed at the end of the competition and can only receive parts that are in the "final" (blue) stage.

Each machine in the competition is composed of one input warehouse (where the part should be placed) and one output warehouse (where the processed part should be picked up again), which are accessible by opposing sides as shown in Figure 2.3. Each machine is also assigned to a fixed type, Limiting the type of parts that can be processed in them. The statistical distribution of the processing time for each machine is only available to each team right before the competition.



(a) Machine diagram



(b) Machine 3D model

Figure 2.3: Diagram and 3D model of a RaF machine

As all parts placed in the outgoing warehouse must be in their final (blue) stage, any other part must pass through a processing stage before the final delivery. As shown in Figure 2.4, machines of type A are used to process raw to intermediate parts and machines of type B process intermediate to final parts.

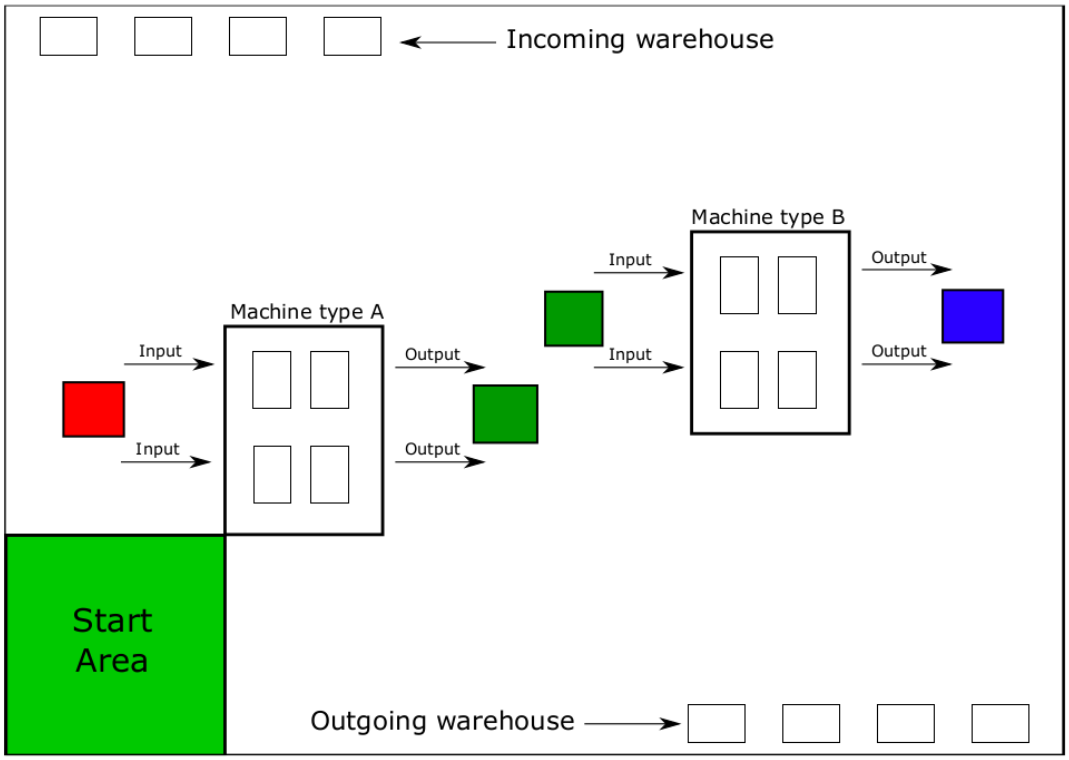


Figure 2.4: Machine constraints for input and output

2.2 Scheduling

The scheduling problem is concerned with the decision-making process for the ordering and allocation of limited resources during a set time period with the goal of maximizing a set of objectives [5]. This section presents an overview of traditional scheduling problems and traces their evolution into more complex models that integrate both machine and transportation allocation decisions. It begins by discussing the fundamental job shop and flow shop scheduling problems, then explore their flexible variants, and conclude with an examination of scheduling problems that incorporate transportation constraints.

2.2.1 Job Shop Scheduling Problem (JSSP)

The traditional Job Shop Scheduling Problem is, as the name implies, concerned with the scheduling of elements inside a job shop. Here the term job shop refers to an environment

(such as a factory) where a set of "jobs" must be processed until all are considered to be in a "completed" state. A job is nothing more than a set of atomic tasks (here called operations) that must be completed in an order specific to each job. To complete these operations is to "process" them for a set amount of time in one of the machines that are available inside the job shop.

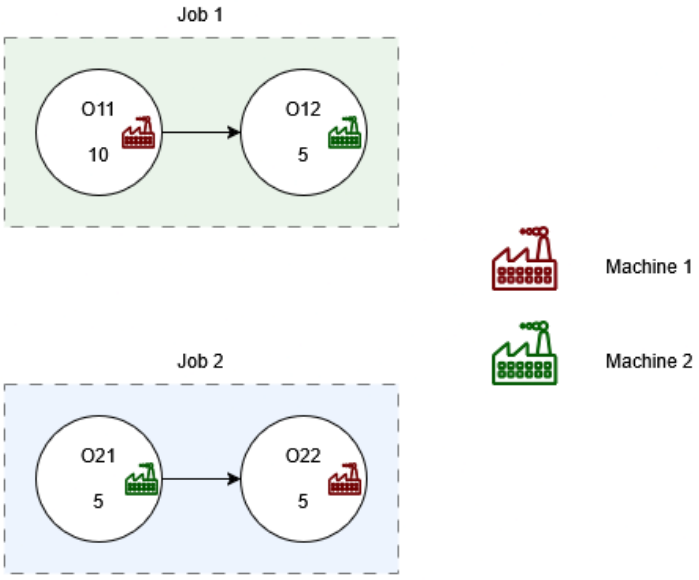


Figure 2.5: Example of a job shop problem

In the instance shown in Figure 2.5, O11, O12, O21 and O22 represent operations, with their respective expected times: 10, 5, 5 and 5 minutes. Thus, "Job 1" indicates that a piece must be processed first by Machine 1 (Red) for 10 minutes, and then be subsequently processed in Machine 2 (Green) for 5 minutes. "Job 2" also needs to use these same machines, but in a different order.

Each operation is pre-assigned to a specific machine and is given a fixed processing time. This leaves the order of operations on each machine as a decision variable that impacts the production time. For example, if the jobs in Figure 2.5 are parallelized as much as possible it will result in the solution displayed in the Gantt chart of Figure 2.6. On the other hand it might be best to finish one job as soon as possible, in which case machine 1 would be idling until job 2 could be processed there, as seen in the Gantt chart

of Figure 2.7.

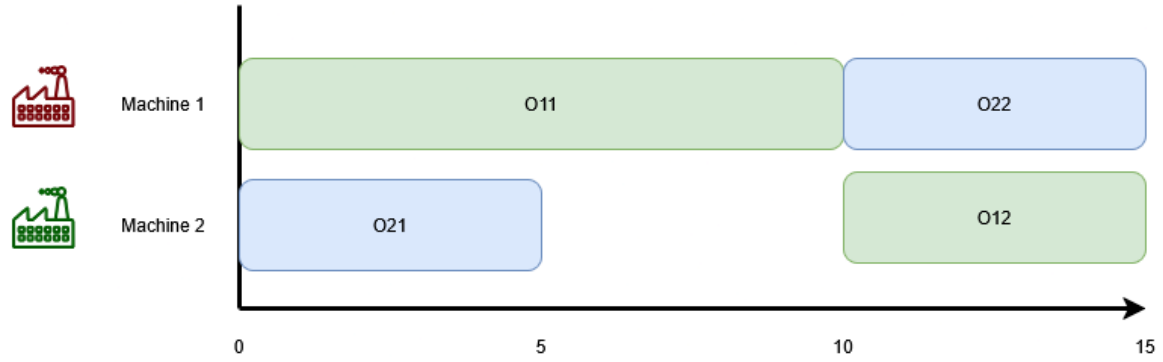


Figure 2.6: Solution for Figure 2.5 that minimizes the time for completion of all jobs

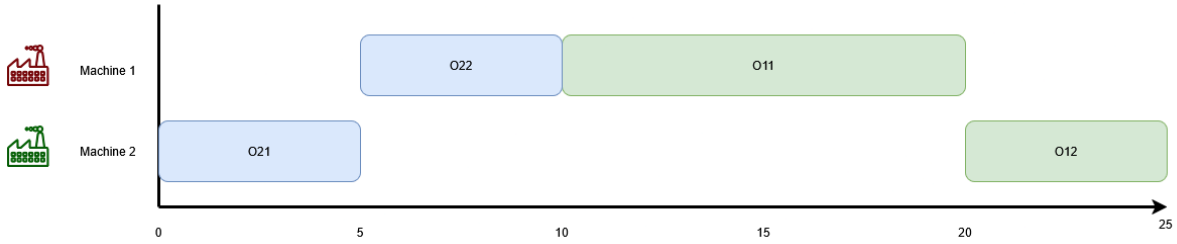


Figure 2.7: Solution for Figure 2.5 that minimizes the time for completion of one job

More formally it is said that the Job Shop Scheduling Problem is composed of a job shop containing a set of machines $M = \{M_1, M_2, ..., M_m\}$ and a set of jobs $J = \{J_1, J_2, ..., J_j\}$ where each job j contains an ordered set of operations $O_j = \{O_{j1}, O_{j2}, ..., O_{jo}\}$, with each operation being assigned to one machine in M with a fixed production time p_{jn} before the scheduling takes place. The end goal of this problem is to find the combined order of execution in all machines that optimizes the final makespan (the end time of the last operation) of the overall system [6]. This problem has been established as NP-Hard for any setup containing more than two machines [2].

While the JSSP does not enforce any relation between the operation position in the sequence and the machine to which it was assigned (so the same machine that processed the first operation of Job 1 could process the second operation of Job 2 and vice-versa in

Figure 2.5), this freedom does not make sense in many manufacturing systems. Industrial machines are complex equipments with very specific inputs and outputs, and most industrial processes are planned to be executed in a very specific sequence. From this restriction is defined the Flow Shop Scheduling Problem (FSSP), where each machine can only process operations belonging to a specific stage. The FSSP with n stages has a set of jobs $J = \{J_1, J_2, ..., J_j\}$ and an **ordered** set of machines $M = \{M_1, M_2, ..., M_n\}$, where each machine represents a stage in the production line. In the standard version of the FSSP every job has a set of operations $O_j = \{O_1, O_2, ..., O_n\}$, as they all must pass through all stages. If *skipping* is allowed then some jobs may skip a number m_j of the initial stages as long as they pass through the remaining stages from $m_j + 1$ to n [7].

2.2.2 Incorporating Machine Allocation

One of the greatest strengths of production lines lies in its ability to parallelize processes when necessary. This choice of parallelization must be a part of the optimization process, leading to the introduction of the machine allocation as a decision variable in the JSSP and the birth of the Flexible Job Shop Scheduling Problem (FJSSP), which extends the original problem by providing a set of eligible machines M_{ij} for each operation O_{ij}[8]. This can be seen as removing the pre-allocated machines in Figure 2.5, with the expectation that the optimizer will decide upon each operation assignment and then with the order of their execution.

Hybrid Flow Shop Scheduling

When this machine flexibility is applied to the FSSP, it results in the Hybrid Flow Shop Scheduling Problem [9], which increments the amount of machines available on each stage. This hybrid flow shop view not only captures the order of operations but also integrates the unique challenges of resource allocation, making the problem more representative of real-world manufacturing and logistics systems. Many real world complex problems can be modeled as hybrid flow shops, such as semiconductor manufacturing.

2.2.3 Incorporating Transportation

The extension of transportation resources merits a subsection of its own, considering the complexity added to the overall problem. There are two main ways of considering transportation in scheduling problems with escalating levels of complexity:

Transportation Constraints

The simplest way of representing movement inside a job shop environment is with transportation constraints, were the transport vehicles are not considered as fully-fledged resources, but represent instead a time-constraint between the end of one task and the start of another. There is no concern over vehicle allocation or collision, and these details are usually not sequence-dependent, varying only with the distance between machines. It is a useful abstraction to represent the usage of conveyor belts in real-world factories, or any other continuous method of transportation. This approach corresponds to about 46% of all papers in Hosseini et al. review [10], demonstrating its popularity as a method of abstracting away the complexities of transportation.

Transportation Resources

Transportation resources represent an evolution in complexity over transportation constraints, as the use of a limited number of resources to deal with transportation brings additional associated constraints. The most fundamental constraint associated with this is the need for the so-called free-carry rides, periods where the vehicle does not carry anything but must proceed to a starting point in order to begin the next operation. These transportation resources affect the problem in a substantially different way compared to machine resources due to their dependence on sequence-dependent setup times. This hidden cost has nothing to do with the actual transportation task, but it is actually specific to the pair composed by the previous task and the current task. Consider the example

from Figure 2.8, where there are two transportation tasks of equal length, one from point A to point B and the other from point C to point D.

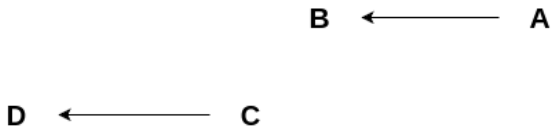
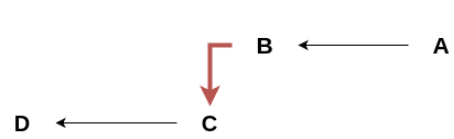
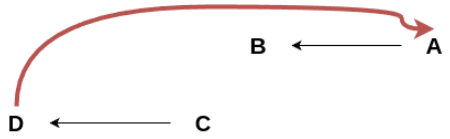


Figure 2.8: An example of two possible transportation tasks

If these were processing tasks on a machine or simply represented as transportation constraints their order would not be relevant to the final makespan, as they both will take the same time to be completed. However when dealing with vehicles it must be taken into account that each task realized by them also changes their position in space, influencing their next task. In this case if task $A \rightarrow B$ is executed before $C \rightarrow D$(Figure 2.9a) the vehicle will have a much easier time going from the end of one task to the beginning of another than if it had executed $C \rightarrow D$ before $A \rightarrow B$(Figure 2.9b). The time that the vehicle takes to complete this *setup* to be able to advance to the next task in its *sequence* is the sequence-dependent setup time.



(a) Movement sequence when executing $A \rightarrow B$ before $C \rightarrow D$



(b) Movement sequence when executing $C \rightarrow D$ before $A \rightarrow B$

Figure 2.9: The difference between two sequences of actions

2.2.4 Graph Representation for Scheduling Problems

In the literature scheduling problems are usually represented with Gantt charts, however there is evidence that the knowledge in scheduling problems is better represented by graph structures [11]. Different approaches have been proposed to represent different variations of scheduling problems over the years [12]–[14], with the most relevant iteration for this work being the representation proposed by Berterottière et Al. to the Flexible Job Shop

Scheduling Problem with Transportation Resources [15].

Their representation utilizes a disjunctive graph $G = (V, A, E)$ to represent the problem. In this graph there is one node for each operation and two additional dummy nodes to capture the start and end of the overall system. The arcs capture precedence constraints and resource allocations, with the set of conjunctive arcs (directed arrows) A representing the fixed precedence relations between operations and the set of disjunctive arcs (undirected edges) E representing possible assignments of operations to any shared resource (vehicle or machine). An example of a flexible job shop with 2 jobs, 2 machines and 2 vehicles is present in Figure 2.10.

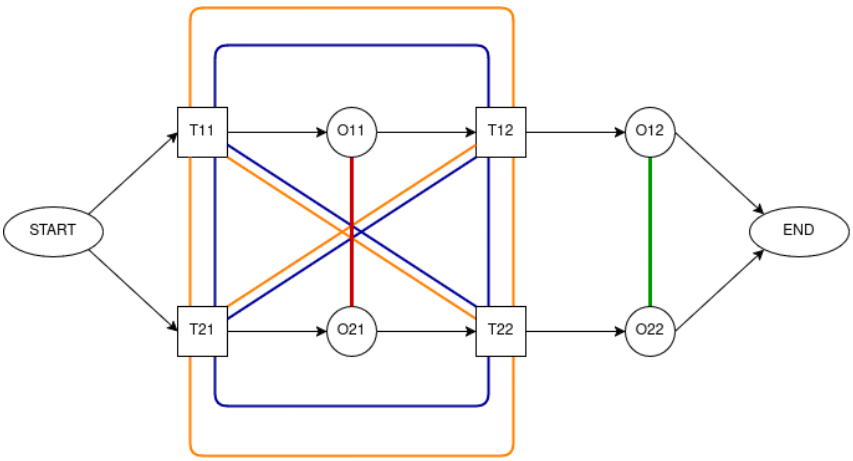


Figure 2.10: Disjunctive Graph representing the FJSPT

To better understand, consider the division between constraints (i.e. edges) based on their origin and malleability: environment-derived constraints and decision-derived constraints. Environment constraints are those inherent to the problem itself and cannot be changed by optimization decisions. They represent essential characteristics of the system, such as the necessary order of flow through all the different stages. In Figure 2.11 they represent the necessary order of events for Job 1 and Job 2. The outgoing edges from the START node indicate that both jobs need to first undergo a transport operation before their first processing operation, that is, they must be transported from their start positions (wherever they may be) to their respective machines (which can be

same machine or separate ones). After they must be again transported to where they will pass through a second processing operation, to only then reach the end of the scenario.

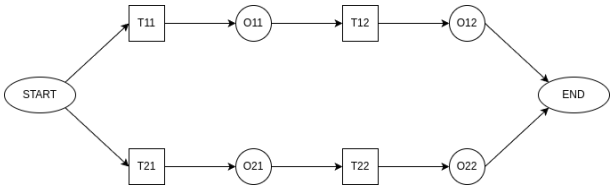


Figure 2.11: Environment constraints of Figure 2.10

Decision constraints, on the other hand, are those derived from the proposed allocation and order of resources (both vehicles and machines). They are the pieces that can moved to alter the solution, and are the objects to be optimized. In the case of Figure 2.12 operations O_{11} and O_{21} share an undirected edge, indicating that they must be processed in the same machine but the order in which they will be executed has not yet been defined (the same is true for operations O_{12} and O_{22}). For the transport operations there are more undirected edges as they represent all the possible combinations of paths for vehicle 1 (represented by the orange edges) and vehicle 2 (represented by the blue edges).

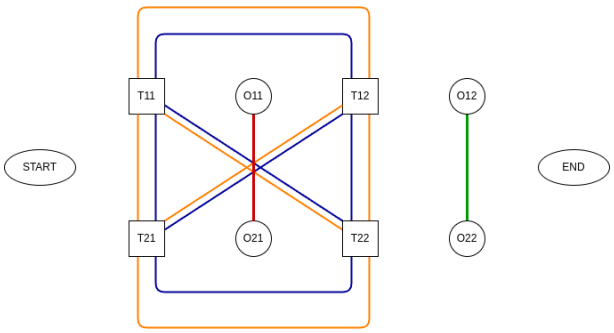


Figure 2.12: Decision constraints of Figure 2.10

A solution is found when all disjunctive arcs are either removed or replaced with conjunctive arcs, representing the assignment and sequencing of operations among the available resources. The resulting directed graph must be acyclic, as any cycle represents a deadlock situation and consequently an infeasible solution.

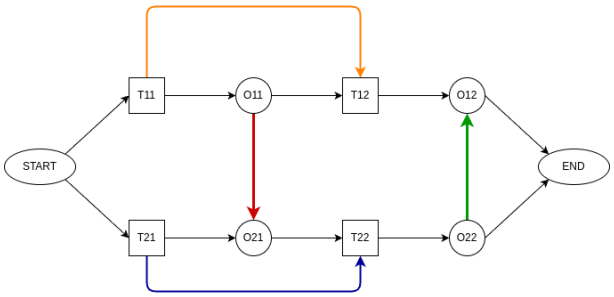


Figure 2.13: Solution for the problem case of Figure 2.10

To visualize a possible solution for this graph, the decision variables are fixed: vehicle 1 will be responsible for transporting all operations of job 1 (T_{11} and T_{12}) and vehicle 2 will be responsible for transporting all operations of job 2 (T_{21} and T_{22}); Operation O_{11} will be executed before operation O_{21} and operation O_{12} will be executed after operation O_{22}. This results in the graph shown in Figure 2.13, with the respective Gantt chart shown in Figure 2.14.

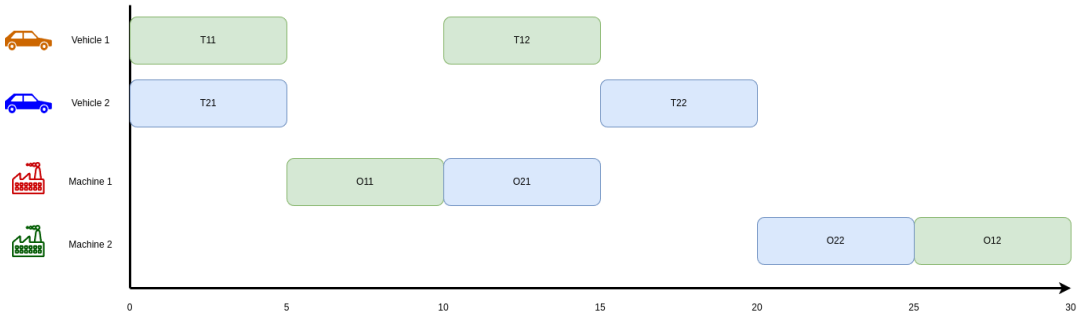


Figure 2.14: Gantt chart for the solution from Figure 2.13

2.3 Path Planning

To find the best schedule for a system with transportation tasks it is essential to know how much time is used by each transportation task. This is where the need for path planning arises, as it refers to the problem of finding a collision-free route from one place to another. In practice this usually means finding the fastest or shortest route, but there are scenarios where researchers can focus on reducing computation time or

prioritizing a smooth trajectory [16]. The field is divided between global and local path planning based on how much information is available about the environment. Global path planning refers to situations where the agent has a complete knowledge of the operational environment and is able to plan the entire route before execution, which is why it is sometimes called offline or static path planning. Local path planning is needed when the agent has only partial knowledge of the environment, usually only of its immediate surroundings. These scenarios require real-time mapping and planning of the environment, and are usually called online or dynamic path planning [17]. In [18] a distinction is made between classical path planning approaches and heuristic-based ones. The following subsections will describe the main classical algorithms still in use while heuristic methods will be described in a separate section, as they are also relevant to scheduling problems.

2.3.1 Environment Modeling

Before the question of which algorithm will be used is ever asked, the environment must be discretized in order for path planning to be possible. Here there are two possibilities: Cell Decomposition and the Roadmap Approach.

Cell Decomposition

Cell decomposition involves dividing the environment in a finite number of cells in such a way that the actual path planning problem is reduced to a grid search problem. To be a valid cell decomposition, three points must be satisfied [5]:

1. Computing a path from one point to another inside a cell should be trivial
2. Cell adjacency information should be easily available
3. The cell containing a certain point p should be efficiently determined

Exact cell decomposition involves drawing parallel lines from the vertex of each obstacle, forming a grid of unoccupied cells.

Roadmap Approach

The roadmap approach seeks to reduce the environment to a graph with nodes and edges, producing an image that looks similar to a roadmap. Three popular techniques for it are the *Visibility Graph*, the *Voronoi Diagram* and the *Probabilistic Roadmap*. The *Visibility Graph* treats the obstacles as polygons, considering each vertex a node on the network. These nodes are then all connected to all other visible nodes and a graph search algorithm is used to find the optimal path. The *Voronoi Diagram* uses the edges of the obstacles to determine the perpendicular bisector between any two edges. This enables the agent to traverse in the fastest possible path between any two objects, avoiding collisions. The *Probabilistic Roadmap* works by generating many nodes in the environment. If the nodes lies in a free space it is added to the graph. If any two nodes can be connected by an edge also in free space, it is also added to the graph. This causes an effect where the longer it runs the better the resulting path will be, but also demands a great increase in computation power.

2.3.2 A* Algorithm

The A* algorithm is a heuristic approach to the classical Dijkstra search algorithm [19]. A* is guaranteed to find optimal solutions for shortest-path problems in known discrete environments when its heuristic function satisfies two conditions: it must be admissible (never overestimates the true cost to target) and consistent, which demands that $h(u) \leq cost(u, v) + h(v)$ must be true for every edge (u, v). The classical Dijkstra algorithm works by:

1. Initializing the distance to the source node as 0 and the distance to all the other nodes as ∞
2. Adding all nodes to a priority queue ordered by smallest distance.
3. While the queue isn't empty:
   (a) Pop the node with the smallest distance.

(b) For each neighbor of the current node:

i. Calculate the distance to the neighbor node by adding the edge weight to the distance of the current node.

ii. Assign this new distance to the neighbor node if it is shorter than the neighbor's recorded distance.

(c) Mark the current node as visited

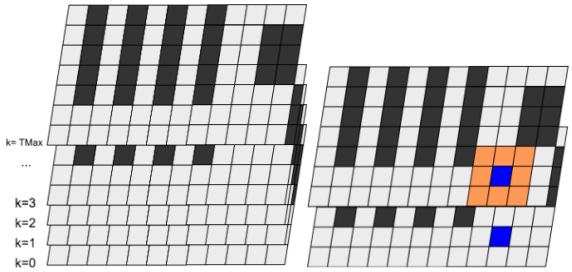
4. Stop when the target node is found or after all possible nodes have been visited.

The A* algorithm incorporates an heuristic in the calculation of the smallest path in the queue, representing the best guess to where the shortest path should be. This means that while Dijkstra uses the distance $g(X)$ from the current node to the source node, A* sums this distance with an heuristic value $h(n)$ of the distance (usually the euclidean or manhattan distance) from the current node to the target node.

$$f(X) = g(X) + h(X) \tag{2.1}$$

2.3.3 Time-Extended A* Algorithm

The Time-Extended A* (TEA*) algorithm is an extension of the A* algorithm for online path-planning of multiple vehicles [20]. TEA* enables multi-vehicle path-planning by incorporating a time dimension t to the search, allowing the algorithm to store the period under which each position is occupied. This effectively turns the 2D environment into a 3D one seen in Figure 2.15, where every position is defined by an x, y and t coordinates.



(a) Input Map over the time. (b) Analysed Neighbor Cells

Figure 2.15: TEA* layer representation and neighborhood search. Source: [20]

The practical effect on the A* algorithm is seen mainly on the neighborhood exploration: a TEA* algorithm will always search in the neighborhood of the next time layer. While true obstacles block their occupying grid across all time layers, the path of other vehicles in the same grid are transformed into temporary obstacles to block the search.

The TEA* avoids deadlocks by considering the current position of other vehicles as obstacles for a limited number of time layers, after which it is the responsibility of these idle vehicles to move out of the way.

2.4 Genetic Algorithm Optimization

First proposed by John Holland as a theory of adaptive systems [21], Genetic Algorithms (GAs) have become the best known algorithms in the field of Evolutionary Computation [22] and the most popular method for solving Hybrid Flow Shop scheduling problems [23]. Inspired by the theory of natural evolution proposed by Charles Darwin and its implication on genomes, GA aims to optimize the values (genes) of a given array (chromosome) from an initial random population of individuals.

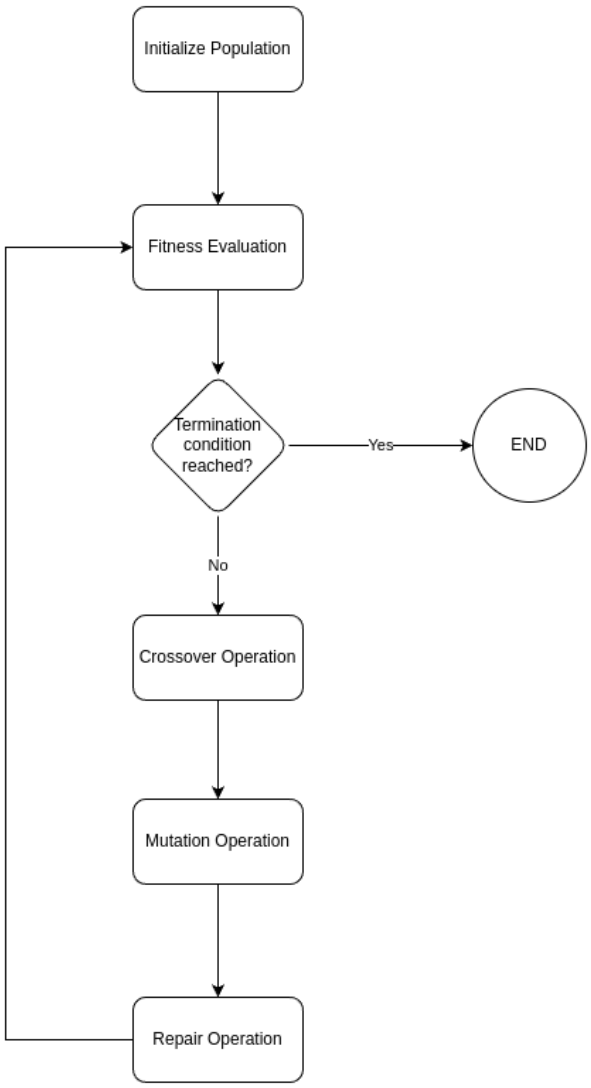


Figure 2.16: Diagram of Genetic Algorithm execution

The algorithm works in an iterative manner over many generations of individuals, with its flow described in Figure 2.16. Initially, a sample of individuals is generated, from where the best individuals are extracted and combined with each other in a crossover operation. After, random mutations occur in the genes of these newly generated individuals to explore solutions that exist outside the scope of the initial population. If necessary these individuals are repaired and sent again to the fitness evaluation, now as part of the second generation of chromosomes.

2.4.1 Encoding

The choice of encoding is the fundamental decision to take when optimizing a problem with GA. The encoding must codify a single solution to the problem into an array, preferably in such a way that it is impossible to generate invalid solutions.

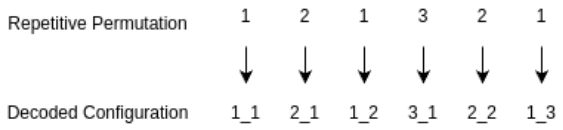


Figure 2.17: Example of repetitive permutation

In scheduling problems it is common practice to use repetitive permutation (such as the one in Figure 2.17) to represent invariable precedence constraints, as the repeated numbers can be swapped around without invalidating the solution and can be trivially converted to the original sequence.

2.4.2 Sampling

Sampling refers to the choice of individuals that will compose the first generation of the algorithm. When dealing with large problem spaces, the initial sampling can be crucial to ensure a diverse genetic pool, as that is essential for the algorithm to travel across as much of the solution space as possible.

2.4.3 Selection

The fitness of an individual compared to the rest of the population is the core of the algorithm and is tightly coupled to the chosen encoding and the problem at hand. Some good considerations when building a fitness function include low computational complexity (as they are executed thousands of times per optimization) and deterministic behavior (as the fitness of identical individuals should ideally be identical).

2.4.4 Mutation

The mutation operation is essential for exploring the space outside the initial genetic pool. Through a randomized method, different parts of an array can be modified to values previously inexistent in the sampled population but still valid in the space of possible solutions. The simplest possible mutation is the bitflip mutation, where binary genes are conditionally flipped between 1 and 0 depending on a predetermined probability. The bitflip mutation extended to categorical values (such as a bounded integer) is called the Random Resseting mutation.

2.4.5 Crossover

The crossover aims to preserve and combine well performing chromosomes for the next generations, and it is applied between two chromosomes with one selected as donor and the other as receiver. A popular crossover operator for problems using repetitive permutation encoding is the Generalized Order Crossover [24], a variation of the standard Order Crossover that is capable of dealing with repetitions in permutations without generating invalid sequences. The operator works by annotating the genes of both parent chromosomes with their respective position in the job order.

2.4.6 Repair

When it is not possible to use an encoding scheme to guarantee valid solutions then repair mechanisms must be utilized. They work by making small adjustments after each operation that produces chromosomes (Sampling, Mutation and Crossover) to ensure that they represent valid solution. While not a strictly necessary operation, it is useful to not waste computation time exploring families of invalid solutions.

2.5 Simulation

Simulation models are essential tools for experimenting and validating complex system designs, as well as offering predictions on future system performance. This can aid the decision-making process by allowing for potential failures and bottlenecks to be identified before being implemented in the real world. Due to the increasing dominance of Internet of Things (IOT) approaches in the industry, there has been an explosion in the number of papers published in recent years about simulations in industrial contexts [25].

Among the several possible simulation methods available, the established framework has been Discrete Event Simulation, but the increase in computational power has allowed other methods such as Agent Based Simulation and System Dynamics to experience a rapid growth in the number of publications with each year [26]. After considering the frameworks Discrete Event Simulation and Agent-Based Simulation for this work, it was decided that the Agent-Based approach provided a more natural way to express the studied system.

2.5.1 Agent-Based Simulation

Agent based simulation is built around the idea of constructing the various agents that will interact in the environment and allowing emergent behavior to appear during the simulation. It is the best approach on complex systems where environments are dynamic and hard to model using a discrete event approach. It is more resource intensive than DES as it uses fixed time steps.

There are well-established commercial in the simulation market, and most of them support more than one type of simulation. Among the best-known and most used are Flexsim, Simio and Anylogic [27]. Although their use was considered, the need for a special license for research prompted a search for open-source alternatives that could meet the needs of this work, which led to the choice of the GAMA simulator.

2.5.2 GAMA Simulator

GAMA [28] is an open-source modeling and simulation platform to develop spatially explicit agent-based simulation. The biggest advantages that GAMA offers in comparison with alternative are its easy integration with external databases and its comodel capabilities. Models in GAMA are defined as code written in its GAML language, as shown in Figure 2.18.

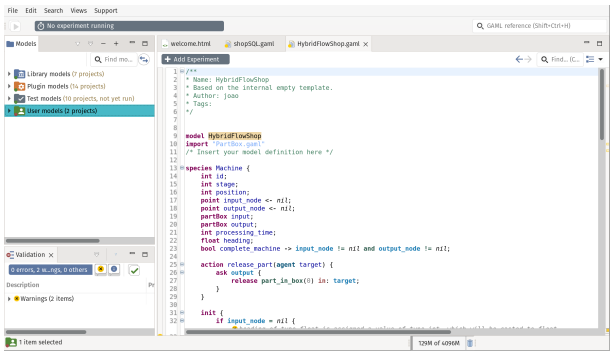


Figure 2.18: Interface of the GAMA simulator

Database Access

Gama supports the execution of SQL queries to MySQL, PostgreSQL and SQLite databases. By inheriting the AgentDB multiple SQL queries can be made through a single connection, enabling easy recording of the results of many experiments in a way that is easily accessible afterwards.

Encapsulating logic with Comodels

Another extremely helpful tool provided by the GAMA simulator is the ability to transform simulations into comodels. With the aim of supporting complex multi-disciplinary projects, GAMA introduces the concept of comodels [29] as a way of encapsulating logic in several micro-models inside a large macro-model.

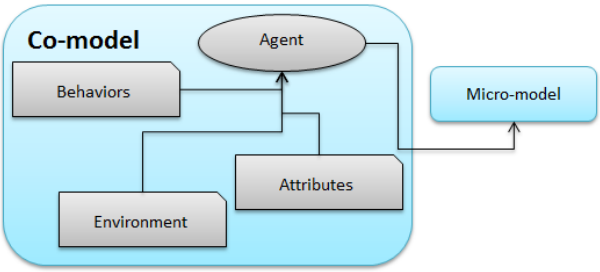


Figure 2.19: Diagram of a micro-model. Source: [29]

Since each micro-model is essentially its own simulation (with a separate global state, as shown in Figure 2.19), they are very useful when running multiple simulations in batch, as each separate case is its own simulation with a loop controlled by a macro-simulation. This solves the problem of validating all cases without depending on an external script to initialize each simulation with different variables or on implementing this functionality inside the model (which would make it impossible to ensure reproducible results from the simulation).

2.6 Related Works

While this research exists in an intersection between multiple domains - production scheduling, multi-robot path planning and simulation-based optimization - this work is hardly the first to make the association between these domains. Hosseini et al. describe 144 papers in their Literature Review of scheduling in manufacturing with transportation [10]. What is uncommon, according to the authors, is the consideration of collision-prone routes, as only 6 papers out of the 144 include the possibility of collisions in their considerations.

Among the 6 works that *do* integrate scheduling with collision-aware path-planning, 2 of them deal with vehicles bound to a straight rail with only lateral movements [30], [31], which do not apply to this work. Among the remaining four there are two works dealing with Flexible Manufacturing Systems [32], [33], one work with a Job Shop environment [34] and one work with a Flexible Job Shop [35]. To our knowledge there are no works

that deal with routing aware of collisions in a hybrid flow shop environment. Furthermore, all machines in related works assume that the delivery and pickup of jobs are done in the same location and all machines are assumed to have infinite buffers.

Table 2.1: Comparison of Features in Related Works

No.	Author	Shop Model				Path Topology		Machine Buffer		Machine Topology	
		FMS	JSSP	FJSSP	HFSSP	Rail-Bound	Network	Infinite	Zero	Pickup and Delivery at the same location	Pickup and Delivery at different locations
[30]	Bürgy and Gröflin		✓			✓			✓	✓	
[31]	Zhou and Lei	✓				✓		✓		✓	
[32]	Lyu et Al.	✓					✓	✓		✓	
[33]	Nishi et Al.		✓				✓	✓		✓	
[34]	Saidi-Mehrabad et Al.		✓				✓	✓		✓	
[35]	Liu et Al.			✓			✓	✓		✓	
	This Work				✓		✓		✓		✓

Chapter 3

Methodology

This chapter is divided in 3 sections, explaining the problem, the research questions and the proposed solutions. Section 3.1 defines the problem and the research questions that guided this research. Section 3.2 defines the competition from the point of view of Operations Research and the associated graph representation used to represent this problem and its associated solutions. Section 3.3 defines the different scenarios proposed for answering each research question.

3.1 Problem Definition and Research Questions

To guide this investigation, a series of research questions were formulated to address both the scheduling efficiency and the integration of realistic constraints into the system. These questions contribute to an understanding of both the theoretical limits of the scheduling model and the practical impact of routing constraints and resource allocation on overall system performance.

1. When applying a hybrid flow shop scheduling model to the RobotAtFactory environment — ignoring collision constraints — how significant is the impact of the number of vehicles on the total completion time for different combinations of parts and different machine processing times?

2. Given that the analysis conducted in the previous question indicates a potential reduction in completion time with increased vehicle numbers, does enforcing fixed spatial constraints (to prevent collisions, making each robot move on mutually exclusive sectors of the environment) maintain this efficiency advantage?

3. Considering that the ideal scheduling scenario (from Question 1) does not account for collision avoidance, the TEA* algorithm is used to plan conflict-free paths for all vehicles. How close do these TEA*-derived paths come to achieving the ideal completion time?

4. The GA's fitness value is computed using an approximated time (based on shortest paths) but may differ from the true execution time when more realistic routing (using TEA*) is applied. As the evaluation becomes increasingly realistic, is there a significant difference in optimization efficiency?

3.2 The RobotAtFactory Competition as a Hybrid Flow Shop Scheduling Problem

The RobotAtFactory competition, as explained in Section 2.1, is composed of multiple tasks which can be scheduled in different ways to minimize completion time. This allows the competition to be examined from the perspective of Operation Research, from where the competition extends naturally into a Hybrid Flow Shop model, described in Section 2.2.2.

To better understand how this relation takes place, let's understand how the competition's tasks are organized. Departing from the base job shop model, the parts on RaF clearly follow a fixed sequence of stages (Red → Green → Blue) and the machines are grouped based on the kind of transformation that they are capable of (either Red → Green for machines of type A or Green → Blue for machines of Type B). This introduces flow constraints into the model - transforming it into a flow shop model - and the presence of multiple machines per stage evolves the problem into a hybrid flow shop model. The

machines also can only accommodate one part at a time, independently if the part is in process or not, which implies a lack of input and output buffers in the model (refereed here as a no-buffer problem).

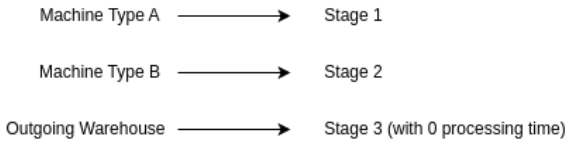


Figure 3.1: Converting the competition labels to flow shop stages

This results in a no-buffer hybrid flow shop problem with 3 stages, where the first stage has 2 available machines, the second stage has 2 available machines and the third stage has 4 available machines, as shown in Figure 3.1. Since it acts only as a delivering point, the third stage is considered a special machine with a processing time of 0.
The robots responsible for transport start in the green area (Figure 3.2 - left) and must travel within the boundaries established by a spatial graph connecting all input and output nodes of each machine. Unless otherwise stated every algorithm uses the spatial graph derived from the original AruCo markers on the competition floor presented in Figure 3.2 (right).

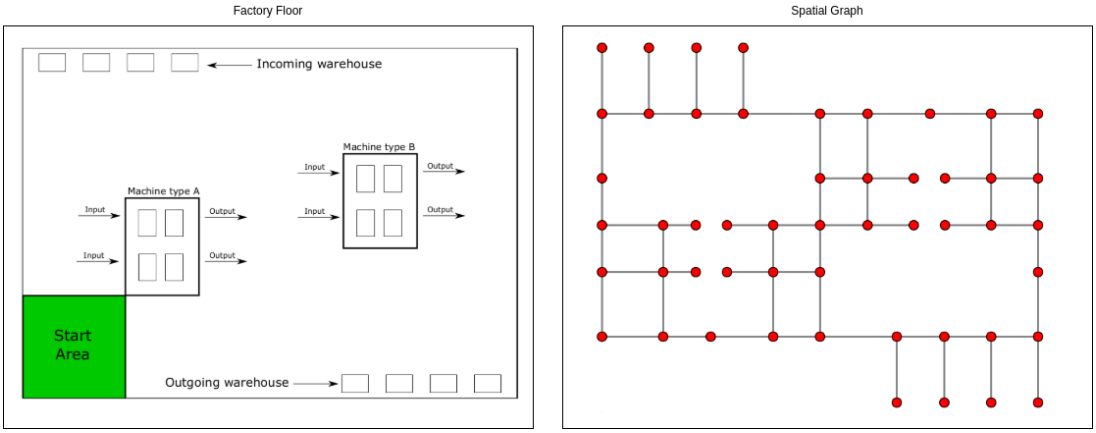


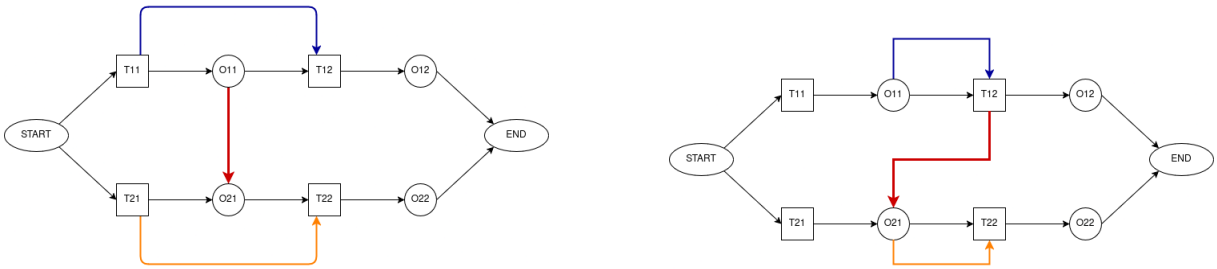
Figure 3.2: Spatial Graph Adapted from the Robot at Factory competition

To represent both production and transport processes, it is used a weighted directed

graph derived from a formulation used for the Flexible Job Shop Scheduling Problem with Transportation Resources described in Section 2.2.4 with additional consideration in order to extend this representation to the Hybrid Flow Shop Scheduling Problem with no buffers.

3.2.1 Including the lack of buffers in the graph

In producing the directed graph is where this representation differs from [15] to accommodate the no-buffer condition. In the Hybrid Flow Shop Scheduling Problem (HFSSP) graph (Figure 3.3a) the sequencing of production operations on the same machine is represented as a direct connection between operations O_{jn} and O_{kn} and the sequencing of transport operations on the same vehicle is likewise represented as a connection between T_{jn} and T_{kw}. This implies the existence of infinite input and output buffers in all machines as a delay in transport T_{jn+1} would not affect the start time of O_{kn} and a delay in the start time of operation O_{jn} would not affect the start time of transport T_{kw}. To solve this additional meaning is assigned to the T and O nodes, as now node T_{jn} indicates the *start* and node O_{jn} indicates the *end* of operation T_{jn}. The practical implication is that now a sequencing on the same machine between operations O_{jn} and O_{kn} is represented by a direct connection from T_{jn+1} to O_{kn} and a sequencing on the same vehicle between operations T_{jn} and T_{kw} is represented by a direct connection from O_{jn} to T_{kw}.



(a) Original solution for Figure 2.10

(b) No-buffer solution for Figure 2.10

Figure 3.3: Difference between graph representations for infinite and zero buffer scenarios

3.3 Scenarios

The scenarios progressively introduce additional layers of complexity—from pure scheduling under ideal conditions to fully integrated scheduling and routing under realistic constraints. By examining each scenario separately, it is possible to pinpoint the impact of various system modifications on the overall efficiency and reliability of the scheduling solution.

Each instance of each scenario is composed of a defined spatial environment (encompassing both machine position and production time), a defined number of vehicles and a defined 4-part setup (The initial color of each of the 4 parts in the Incoming Warehouse). The genetic algorithm optimization is applied for each of these instances, defining the final makespan as the longest path from node START to node END.

The problem space for each scenario is composed of six different environments, all of them spatially equal but with different production times for all machines (the production times are defined as a proportion of the average transport time T, such that the 6 possibilities are 0, $0.01T$, $0.1T$, $1T$, $10T$ and $100T$). For each production time one instance is created for each combination of parts. Since there are 4 parts in each round and each part can be in one of 3 stages, there is a total of $3^4 = 81$ starting combinations. Each of these possible instances is run for 1,2,3 and 4 vehicles.

3.3.1 Scenario 1: Pure Scheduling

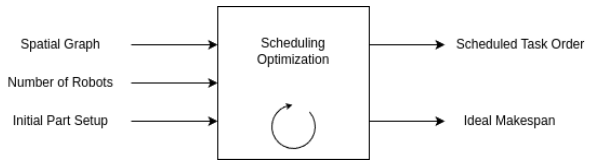


Figure 3.4: Diagram of scenario 1

In the first scenario, the focus is solely on task scheduling. Given the number of robots available and the initial configuration of parts the task order is optimized to obtain the lowest possible makespan.

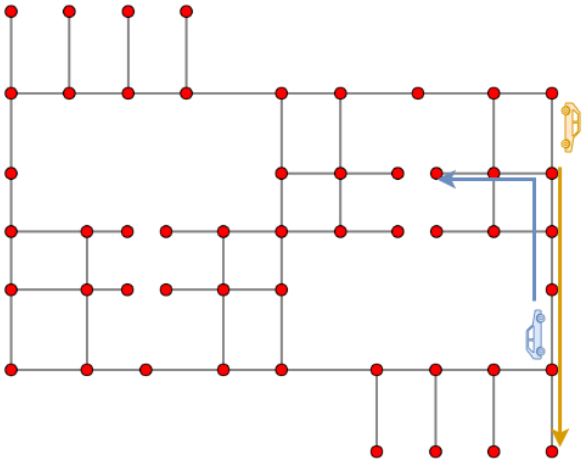


Figure 3.5: In Scenario 1 the routes are based only in the shortest path to the next destination, creating situations where collisions may occur

In this environment every transport operation follows the shortest path from start to destination. Here, collision considerations are completely ignored and vehicles are allowed to pass freely through one another, creating paths such as the one in Figure 3.5.

3.3.2 Scenario 2: Scheduling with Complete Space Separation

In this scenario, while still relying on the idealized routing (shortest paths with A*), spatial constraints are introduced to avoid collisions. Each vehicle is assigned an exclusive zone, ensuring that no two vehicles can occupy the same node simultaneously (in most cases). This enables an evaluation the impact of different number of vehicles in a heavily constrained scenario.

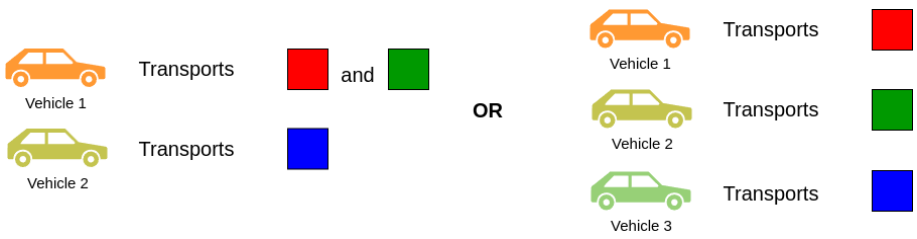
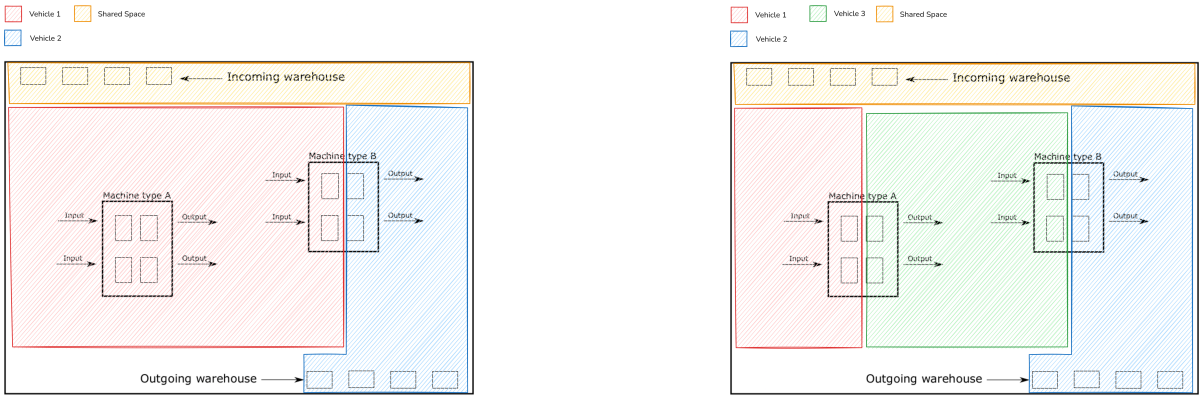


Figure 3.6: Representation of vehicle responsibility used in scenario 2

This complete separation is only possible in the first place because the RobotAtFactory

(RaF) environment implements separate physical location for pickup and delivery of parts. This way each vehicle can be constrained to be responsible for a single type of part (Figure 3.6), thus enabling the vehicles to act in separate spheres of influence.



(a) Example division for 2 vehicles

(b) Example division for 3 vehicles

Figure 3.7: Possible division of space for multiple vehicles

Since this approach depends on fixing vehicle allocations for each stage, this strategy only really makes sense for combinations of two and three vehicles. Furthermore while this approach greatly decreases the common area where vehicles interact it does not eliminate it, as all stages may appear in the incoming warehouse (as shown in the area separations represented in Figure 3.7).

3.3.3 Scenario 3: Routing with Time-Extended A*

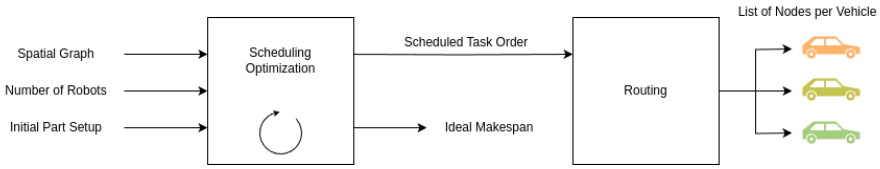


Figure 3.8: Diagram of Scenario 3

In this scenario, the aim is to generate conflict-free paths with the Time-Extended A* for the solutions found in scenario 1. Unlike in previous scenarios, the routing method takes into account the position of all vehicles when planning each path. This method allows the

system to adapt to potential conflicts by planning alternative routes that may be longer than the shortest path but are free from collisions. The focus here is to quantify how these adjusted routes compare to the ideal scheduling times.

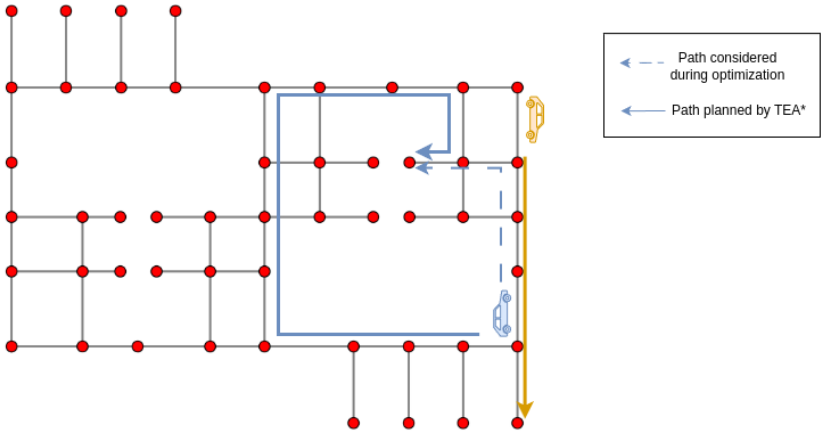


Figure 3.9: A situation where the task order was optimized with the assumption that the transport task would take the time to travel the shortest path, while in reality the routing algorithm planned a much longer path to avoid collisions

This approach may result in suboptimal scheduling, as the scheduling order may have been optimized for a different route than the one planned by the TEA* router. This may lead to situations such as the one in Figure 3.9, where the time taken to complete a task ends up being higher than expected.

3.3.4 Scenario 4: Joint Scheduling and Routing with TEA*

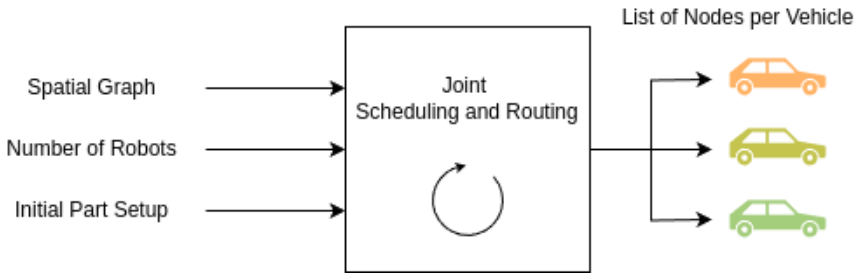


Figure 3.10: Diagram of scenario 4

Scenario 4 combines scheduling and routing into a unified optimization process specifically to avoid problems such as the one in Figure 3.9. In the integrated approach of Figure 3.10, the fitness evaluation for the GA is not based on a simple shortest-path approximation but is derived directly from the TEA* algorithm's conflict-free paths. This scenario examines whether the incorporation of realistic routing delays into the fitness function yields significant differences in optimization performance when compared to the more simple evaluation metrics used in previous scenarios.

Chapter 4

Development

The development of this work has three foundational pillars that form its base: the genetic algorithm optimization, the graph representation and the simulation validation. Section 4.1 explains how scheduling is optimized through a genetic algorithm. Section 4.2 describes how the graph representation is used to decode the chromosome of each solution and calculate the time for the expected makespan. Section 4.3 describes the simulation implementation for validating the optimized solutions.

4.1 A Genetic Algorithm Approach to Scheduling

In a complex system like the RaF competition, the number of possible scheduling combinations is enormous due to the simultaneous assignment of production tasks and routing decisions for vehicles. Therefore, optimization is essential. By applying a Genetic Algorithm (GA) tailored for permutation problems, a large solution space is efficiently explored and converges on near-optimal scheduling plans that balance production times, routing constraints, and resource allocation. The implementation of the Genetic Algorithm was made in Python with the Pymoo [36] library version 0.6.1.3.

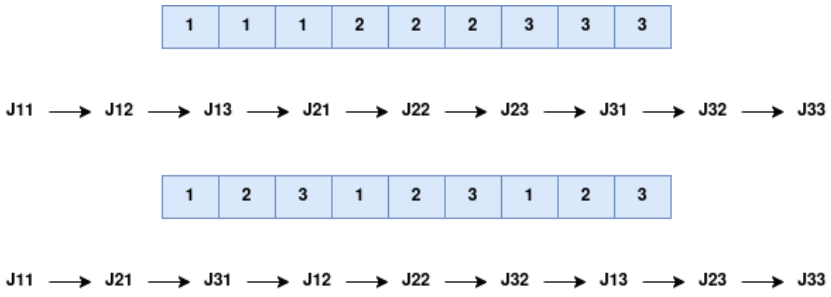
The algorithm execution was significantly sped up by the use of Pymoo's starmap parallelization, enabling the computation of multiple chromosomes in parallel inside each generation.

Chromosome representation

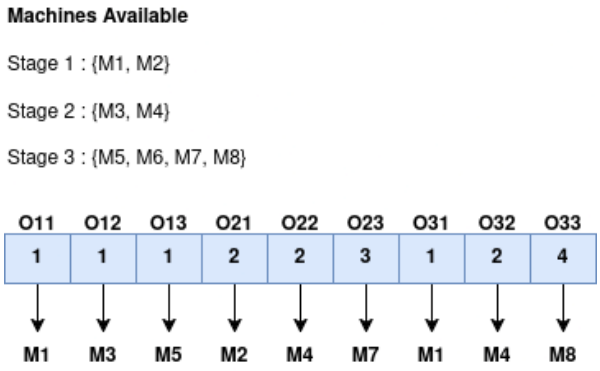
The right choice of encoding is essential for a good optimization and should accurately reflect the decision variables of this problem. The chosen encoding divides the chromosome in three parts of equal length (as shown in Figure 4.1), representing the execution order for the operations, the machine allocated for each operation and the vehicle allocated for the transportation of each operation.

Figure 4.1: Chromosome encoding

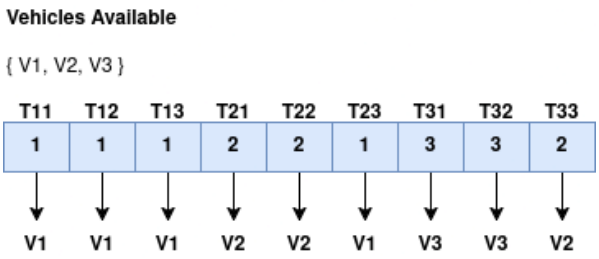
The first vector represents a global order of execution for all operations, to be used in cases where the model must make an ordering decision. This means that when two operations are assigned to the same resource (machine or vehicle), their order in this resource's queue is defined by this vector. It is represented using repetitive permutations, with each operation represented only by the number of their job and their index inside the job being defined by their position in relation to other numbers of equal value (see Figure 4.2a). This ensures that invalid states (such as operation O_{12} coming before operation O_{11}) are impossible to represent in the encoding.



(a) Job Ordering



(b) Machine Allocation



(c) Vehicle Allocation

Figure 4.2: Example of chromosome encoding: (a) presents two examples of possible job ordering. In the upper example all tasks of Job 1 are executed, then all tasks of Job 2 and then all tasks of Job 3. In the lower example the first task of all jobs is executed, then the second task of all jobs and then the third task of all jobs. (b) presents the machine allocation for each task, where the number inside each gene indicates the index of the chosen machine in the vector of machines available for that stage. The stage of an operation corresponds to its position inside its job (that is, operation O_{X1} is of stage one, operation O_{X2} is of stage 2, ...). (c) presents the allocation of vehicles for the transport operation of each task. As the vehicle availability does not differ for different stages, the number corresponds to vehicles in a single global vector.

The machine (Figure 4.2b) and robot (Figure 4.2c) allocation vectors establish which resource will be assigned to that operation from the possible pool of resources. The corresponding operation is based on the vector position, such that the first position correspond to the first operation of the first job, the second position to the second operation of the first job, until the last operation which will correspond to the last operation of the last job. Each position in the machine vector has a different range of values as each stage can have a different number of possible machines. The machine allocation on the last stage,

due to it being the end of the system, needs to have distinct allocations for every job (that is, the last operation in the machine vector for jobs 1,2,3 and 4 will always be a permutation of 1,2,3 and 4).

Fitness function

The quality of each individual solution is measured by the total makespan — that is, the difference between the start time of the first operation and the completion time of the last operation. To be able to estimate this value a conversion must be made from the chromosome encoding to a graph representation.

This conversion is a two-step process: First the three-vector representation is translated from the chromosome to a set of ordered operations for each resource in the system; Last the correct constraints must be added to the graph representation to ensure that the order of operations in each resource is respected.

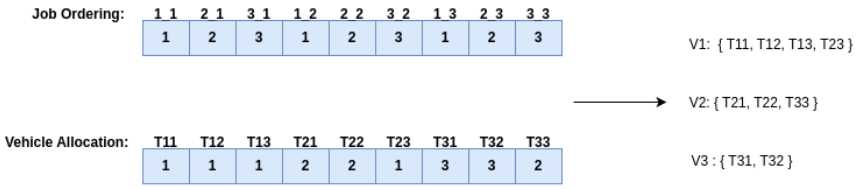


Figure 4.3: Conversion of the job ordering and vehicle allocation vectors into the ordered task list of each vehicle

To accomplish the first part a separate empty set is established for each distinct value in the vector. After, for each of these values, each operation marked with this value is added to its respective set in the order seen in the job ordering vector. The result of these steps can be seen in Figure 4.3, where the vehicle allocation is translated into a set of tasks for each vehicle to be executed in the order established by the job ordering vector.

Crossover operator

The chosen crossover operator is the Generalized Order Crossover. To understand its action consider a crossover operation between the two chromosomes from Figure 4.2a.

This operation starts by assigning one chromosome as the donor and the other as the receiver chromosome, and annotating each chromosome with the actual *job_operation* code for each position. Then, a section is selected from the donor chromosome to be transplanted into the receiver chromosome (Figure 4.4).

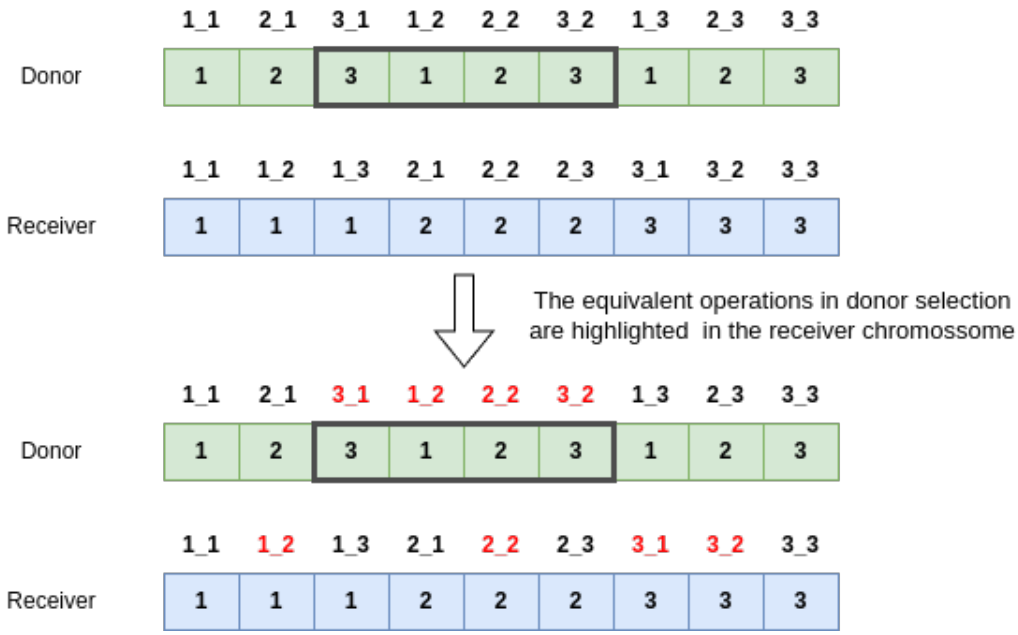


Figure 4.4: After determining one parent chromosome as the donor chromosome and the other as the receiver chromosome, a section in the donor chromosome is selected as the donated section. The repetitive permutation encoding is decoded for all genes, with the operations in the selected section of the donor chromosome marked in the receiver chromosome

The selected section is inserted into the receiver chromosome at the same location it resided in the donor chromosome, shifting the local genes to the right (Figure 4.5).

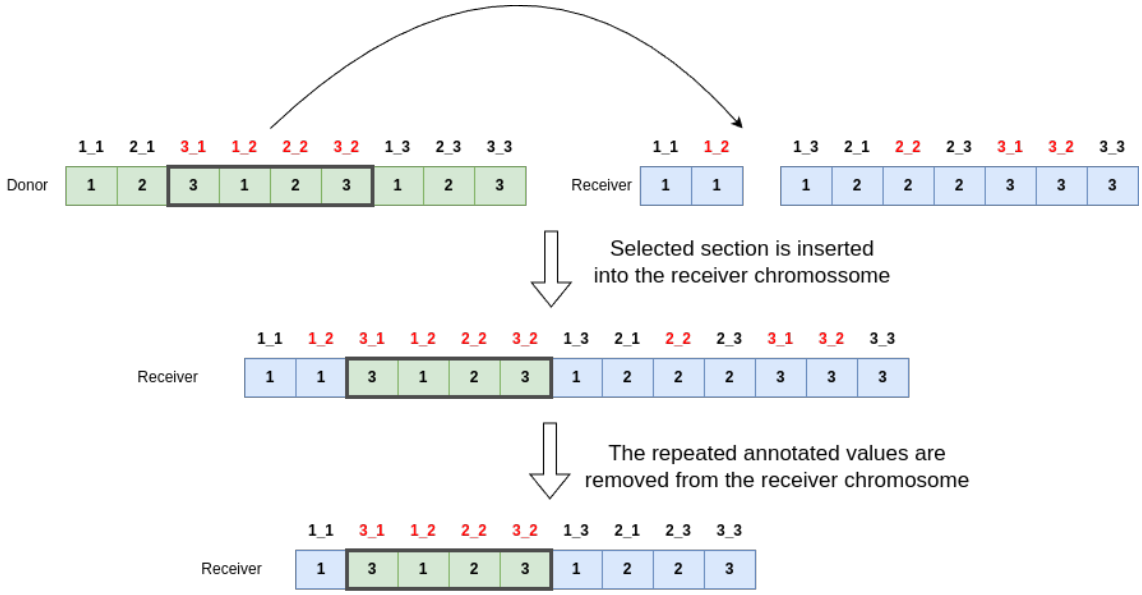


Figure 4.5: The selected section from the donor chromosome is inserted in its original position in the receiver chromosome, and the repeated operations from the original receiver chromosome are removed

Considering the *annotated* values, the elements in the receiver chromosomes which are also presented in the donated section are removed, as shown in the last step of Figure 4.5.

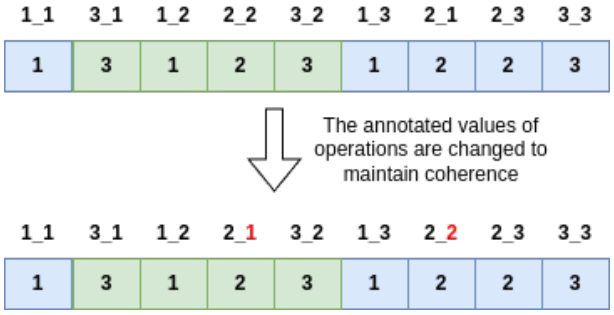


Figure 4.6: The child chromosome after the execution of the Generalized Order Crossover

After successfully applying the operator to the job ordering vector, the equivalent machine and vehicle assignments of the operations contained in the donated section from the donor chromosome are transferred to the receiver chromosome after coherence is established, as shown in Figure 4.6. If one of the donated genes references the machine allocation of the last stage, the operator ensures that each job at the last stage is assigned to a distinct machine.

Mutation Operator

The mutation operation is essential for ensuring that the GA is capable of reaching solutions that are not present in the initial population. With that consideration the Random Resetting mutation, an extension of the bit flip operator for integer values, is applied over the machine and robot allocation arrays. The operator works by assigning random values to randomly selected positions in each array, with each value being inside the accepted range for the operation referenced in the execution order array as can be seen on the first two mutations in Figure 4.7.

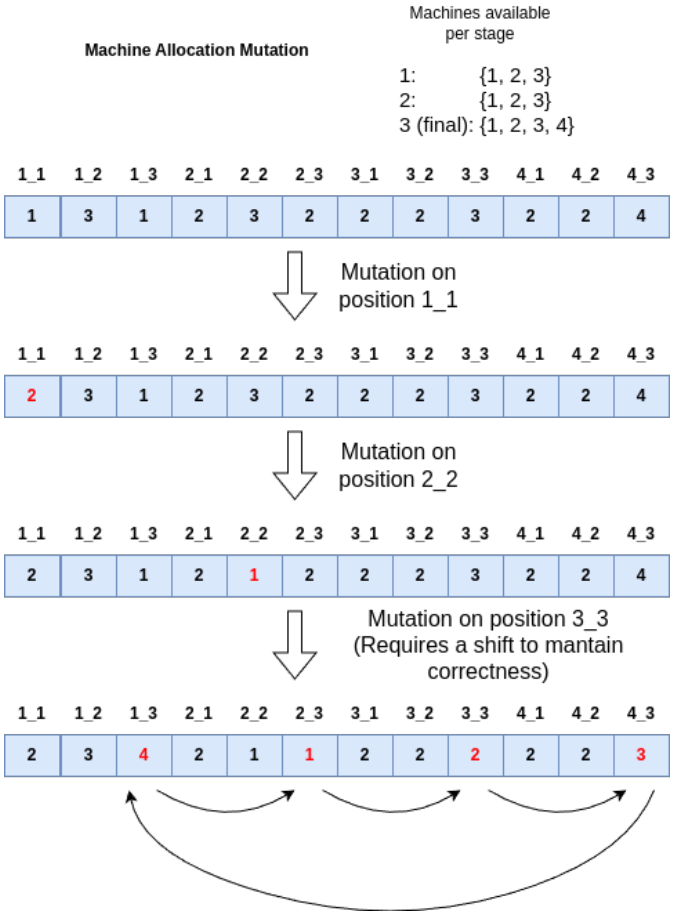


Figure 4.7: How Random Resetting mutation works on a machine allocation array: The chosen location in the first two mutations is randomly changed to a value bounded by the IDs of available machines based on the operation stage (number to the right above each gene). In the special case of stage 3 (where the parts must stay until the end of the competition), the position among all stage 3 genes is shifted every time one of them is mutated

As with the Crossover operator, this operation has a peculiar behavior when applied to stage 3, due to its special position as the end of the system. As shown in the last mutation of Figure 4.7, if the position affected assigns the machine to the last stage of operations (in this case stage 3), a right shift between all assignments of the last stage substitutes the random value assignment.

4.2 Graph Implementation

The implementation for the graph representation of the Hybrid Flow Shop Scheduling problem was made in Python with the aid of the NetworkX [37] library version 3.4.2.
The graph is represented as a MultiDiGraph object from NetworkX, meaning that it is a directed graph which allows multiple edges between each pair of nodes. The graph is built incrementally in 3 stages:

1. Base environment graph: This graph contains all the necessary nodes, but only contains environment-derived edges (such as those due to proper stage ordering)

2. Complete graph: This graph adds the edges derived from the decision variables, such as ordering and assignment of resources (vehicles and machines)

3. Complete Weighted Graph: This graph add the correct weight to each edge according to the selected path planning algorithm

Since the first stage will be equal for every instance in a single run of the optimizer, stage 1 is controlled by a static method and configured in a class-wide level, while stages 2 and 3 are configured in the instance level through the class constructor.

4.2.1 Building a Base Graph from the Environment

The base graph is built from the hybrid flow shop problem characteristics: The total number of stages, the total number of jobs and the number of skipped stages for each job. Figure 4.8 shows how these variables impact the size of the graph.

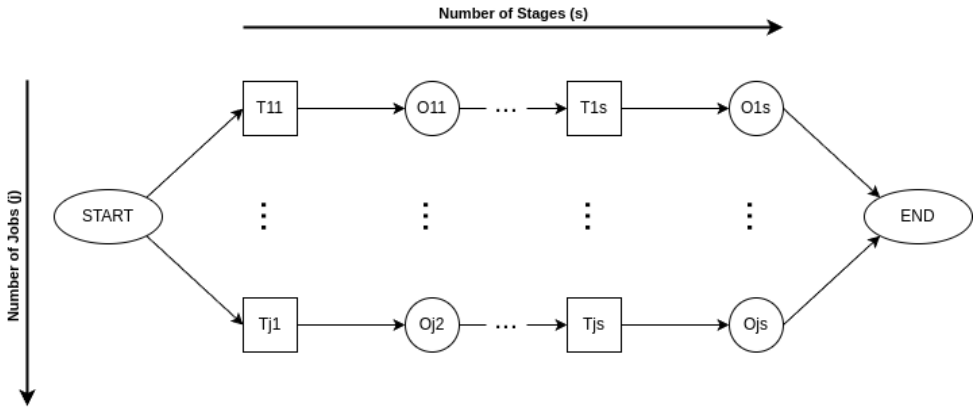


Figure 4.8: Variation of the base graph for the environment variables for a number j of jobs, each job with at most s stages. T represents a transport operation, and O represents a processing operation performed on a machine

Since these variables depend only on the problem environment and do not vary across individual solutions, a single graph is created as a static variable for the class during the GA initialization. This enables the algorithm to copy this graph when building the detailed version within each instance of the optimization.

4.2.2 Integrating the Decision Variables

With the base graph established, the next step is to determine which edges must be added to satisfy the constraints imposed by the solution of each chromosome encoding.

The constraints are derived from two independent situations: The sharing of a single machine by two or more processing tasks and the sharing of a single vehicle by two or more transportation tasks. Each of these is calculated by combining the job ordering section of the chromosome with, respectively, the machine allocation section and the vehicle allocation section.

For the machine constraints, the combination of job ordering and machine allocation is used to build an ordered list for each machine of its assigned operations.

$$machineList = [..., O_{xy}, O_{kw}, ...]$$

Each pair of operations O_{xy} and O_{kw} in the list corresponds to one edge going from operation T_{xy+1} to operation O_{kw}, indicating that the operation O_{kw} can only start after operation O_{xy} has been completed *and* removed from the respective machine.

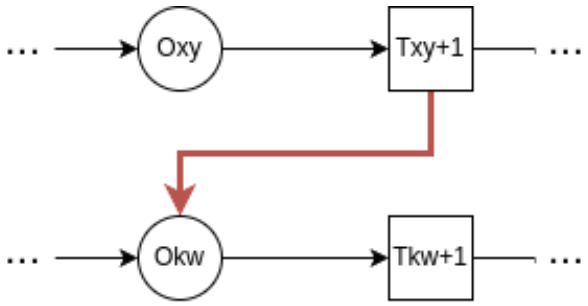


Figure 4.9: The edge enforcing the precedence constraint between O_{xy} and O_{kw} when they are in the same machine

For the vehicle constraints one extra step is needed. The vehicle lists are built in the same manner of the machine lists, combining the job ordering and vehicle allocation to build one ordered list for each vehicle.

$$VehicleList = [T_{ij}, ..., T_{xy}, T_{kw}, ...]$$

Before evaluating the pairs, however, a special constraint must be included from the $START$ node to the first element of each list, representing the time spent by the vehicle getting from its starting position in the shop to the location of its first transportation task. In this case this would represent an edge going from node $START$ to node T_{ij}.

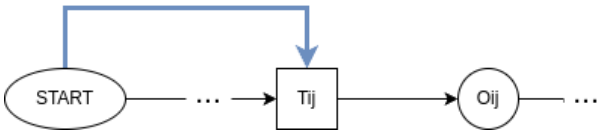


Figure 4.10: The edge enforcing the constraint imposed by the travel time from the initial position to the first task of a vehicle

After the start node is correctly linked to the first element of each list it is possible to evaluate the pairs of transport operations, with each pair of operations T_{xy} and T_{kw}

corresponding to an edge going from operation O_{xy} to operation T_{kw}, indicating that the vehicle can only start operation T_{kw} after it has successfully completed operation T_{xy}.

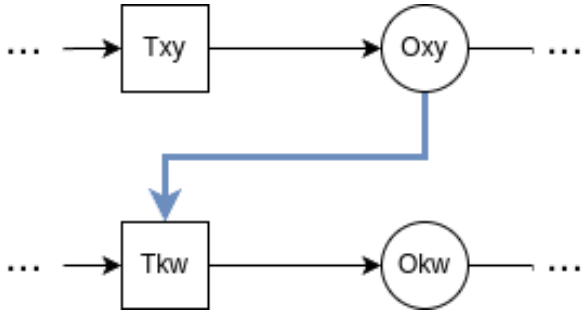


Figure 4.11: The edge enforcing the precedence constraint between T_{xy} and T_{kw} when they are in the same vehicle

4.2.3 Graph Weight Calculation

The weight w of each arc is dependent on the types of operations involved in each side and are described with the auxiliary operators $a(O_{jn})$ for returning the machine m allocated to operation O_{jn}; $p(m)$ for returning the processing time of a specific machine; $i(m)$ for returning the position of the input location of a specific machine; $o(m)$ for returning the output location of a specific machine; and $d(p_1, p_2)$ for returning the time spent moving from position p_1 to position p_2. To ease visualization, the shorthand $in_{jn} := i(a(O_{jn}))$ and $out_{jn} := o(a(O_{jn}))$ describe the position of the input and output of a machine assigned to operation O_{jn}.

With this established notation the 6 possible types of edges are defined:

- $\boxed{START} \rightarrow \boxed{T_{jn}} : d(initial\_position, out_{jn})$

  Marks the beginning of the transport operation for the first operation in a job, establishing the starting condition of the scheduling sequence. The weight represents the time taken by a vehicle from its position at time 0 to one of the boxes containing the part that will be carried in operation T_{jn}. Due to the established convention, the positions at the incoming warehouse are treated as incomplete machines, as they only possess their output component.

- $\boxed{T_{jn}} \rightarrow \boxed{O_{jn}} : d(out_{jn-1}, in_{jn})$

  Represents the transportation time between operations. This edge's weight is calculated by the distance between the location where the operation O_{jn-1} was collected and the location where operation O_{jn} must be delivered. It ensures that the time taken for a vehicle to move between machines is accounted for.

- $\boxed{O_{jn}} \rightarrow \boxed{T_{jn+1}} : p(a(O_{jn}))$

  Indicates the processing time required on a machine before the subsequent transport operation can begin. Its weight is equal to the processing time of the machine executing the operation O_{jn}.

- $\boxed{O_{jn}} \rightarrow \boxed{T_{kw}} : d(in_{jn}, out_{kw-1})$

  Captures the sequencing of two transport operations assigned to the same vehicle, as the node O_{jn} represents both the start of its production operation and the end of transport operation T_{jn}. Its value is the time spent moving from the output location of operation O_{jn} to the input location of operation O_{kw-1}.

- $\boxed{T_{jn+1}} \rightarrow \boxed{O_{kn}} : 0$

  Captures the sequencing of two production operations in stage n assigned to the same machine, as the operation O_{kn} can only start after operation O_{jn} has been removed from the machine (thus the connection from T_{jn+1}). The actual value of this edge is 0 as there is no actual relation between the two connected operations, with the edge acting only to propagate any delays that may have affected the start of transport operation T_{jn+1}.

- $\boxed{O_{jn}} \rightarrow \boxed{END} : 0$

Connects the final operation of a job to the dummy END node, ensuring that the makespan calculation covers the complete schedule from the initial START through to the final delivery. As it represents the final delivery, its weight is 0.

To validate this solution as a possible scheduling for the problem, the final graph must be a Directed Acyclic Graph (DAG), as any cycle in this graph highlights the presence of circular dependencies in the solution which will lead to a logical deadlock.

After a solution has been deemed valid there is still the concern of order of evaluation. While it is possible to use functions that evaluate to the same result independently of the order in which they are executed, in others it is desirable to only evaluate an arc after the value for every one of its dependencies have already been determined. An example of this behavior on this research is the difference between evaluating transport times with the A* versus the TEA* algorithm: while the A* is used to calculate the shortest path between two points and only depends on the initial and final point, the TEA* algorithm is concerned with the specific time window in which the vehicle is located at every node along the way in order to avoid collisions. This extra necessity can be satisfied by providing the start time of the operation which, if all the dependencies values are known, is only a matter of determining the length of the longest path from the node START to the current node.

Conveniently, every directed graph that is also a DAG has an order of evaluation in which every dependency is evaluated with respect to precedence, and its called a topological array and is shown in Figure 4.12.

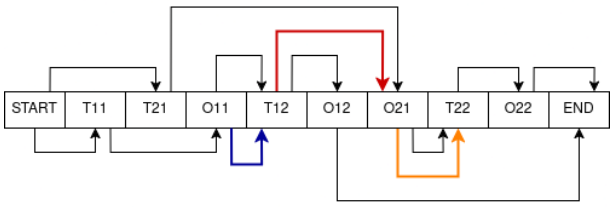


Figure 4.12: Topological array of Figure 3.3b

Finally, with this weighted conjunctive graph representing the solution, the expected

makespan is obtained by evaluating the length of the longest path from node START to node END.

4.3 Simulation Implementation

To validate and visualize the optimization results, a simulation environment was developed for the RaF competition within the GAMA simulator. This environment executes the optimizer proposed routes and records their outcomes step-by-step. Integration between the optimization algorithm and the simulation is achieved via a SQLite database (Figure 4.13), where scheduling and routing outputs are written by the optimizer and then retrieved by the simulator to reconstruct that specific shop execution.

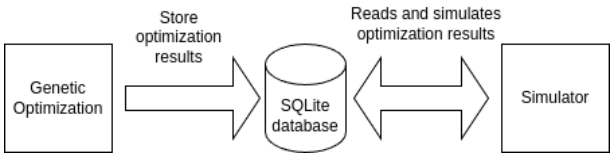


Figure 4.13: Relation between optimization algorithm and simulation

The database schema shown in Figure 4.14 was developed with the aim of storing one experiment record for every distinct situation (combination of `graph_id` and `starting_setup`) evaluated by the optimizer. These experiments can then each be simulated an arbitrary number of times, with each simulation storing a different record in the simulation table.

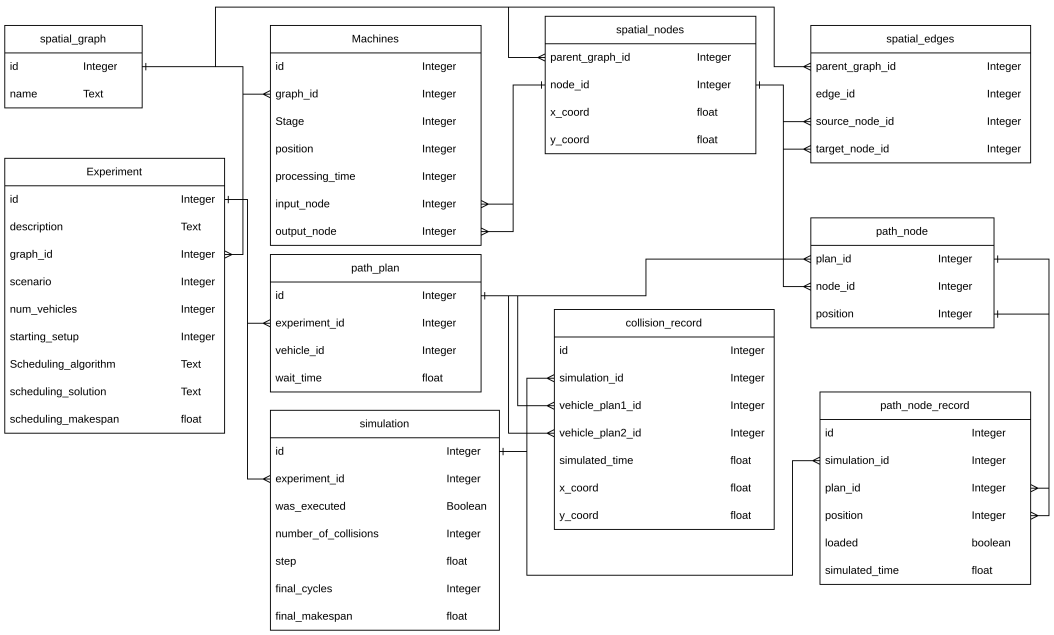


Figure 4.14: Schema of the database used to store optimization and simulation results

The simulation implementation in this section is divided between two models: `Shop` and `ExperimentValidator`. The `Shop` model is the more complex one by a significant margin, encapsulating all the logic for one execution of the RaF environment. The `ExperimentValidator` model is a simple implementation leveraging the comodel feature of GAMA to loop over all untested experiments and generate simulations for them, allowing the automation of this work without the need for external scripts.

4.3.1 The `Shop` Model

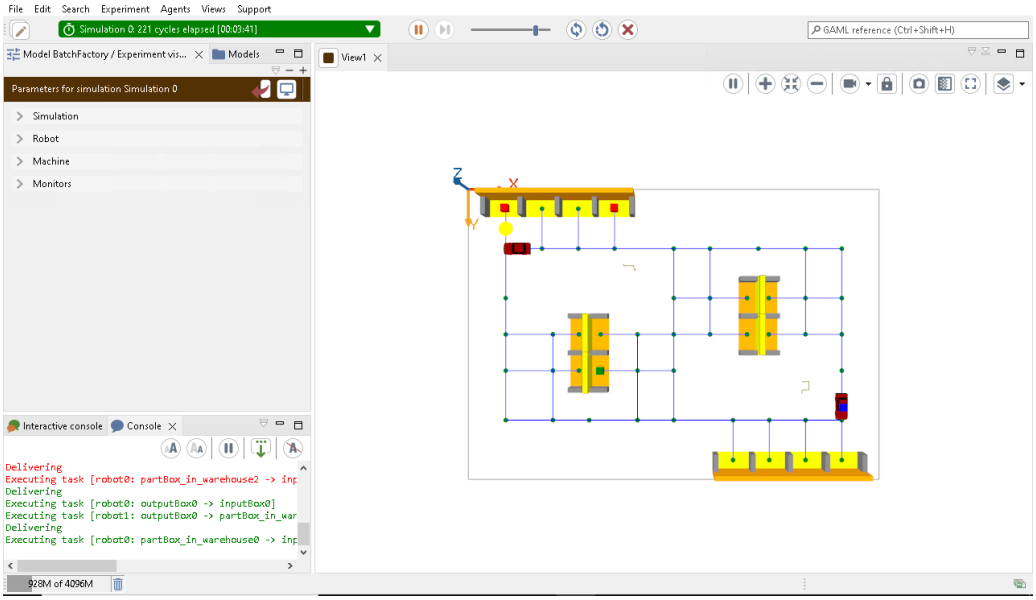


Figure 4.15: Screenshot of the `Shop` simulation running

The `Shop` model reproduces the functioning of a single execution of the RobotAtFactory circuit for a given set of parts while recording the results in the database (A screenshot of this model mid-execution is present in Figure 4.15). To be executed, the only information needed by the shop is the ID of the experiment that must be simulated. With this ID the shop creates an empty simulation record and loads the following associated information from the database:

- The spatial graph data: X and Y coordinates of each node and a pair list of all undirected edges.
- Machine characteristics: details such as the processing time, optional input and output nodes and machine name.
- The Starting Setup: The color of the 4 parts present in the incoming warehouse.
- Routing Information: The specific routes (the sequence of nodes) calculated for each vehicle. These routes are derived from either the TEA* algorithm or A* algorithm, depending on the scenario.

- Waiting Time: The delay that must be observed when a node repeats itself in the sequence. This represents a wait in place action and is crucial for enabling collision avoidance for the TEA* algorithm.

The spatial graph data is loaded in two phases: First all nodes are created in their respective coordinates and then all edges are built from the list of ID pairs. After the graph is properly created then the machine information is loaded. Each machine is associated with one node for its input location and one node for its output location. The incoming warehouse is represented as machines with only the output location defined and the outgoing warehouse is represented as machines with only the input location defined.

The starting setup is stored in the database as a single base-3 encoded integer value s, where the tuple of 4 values $(p1, p2, p3, p4)$ is decoded from Equation 4.1.

$$s = p_1 \cdot 3^0 + p_2 \cdot 3^1 + p_3 \cdot 3^2 + p_4 \cdot 3^3 \tag{4.1}$$

where every part $p_n \in 0, 1, 2$ with each value corresponding to a possible part color:

$$red := 0, \quad green := 1, \quad blue := 2$$

Based on the number of routes linked to the experiment ID an equivalent number of vehicles is created. The routes for each vehicle are then loaded as an ordered set of IDs, such as:

$$54 \rightarrow 53 \rightarrow 1 \rightarrow 2 \rightarrow ...$$

where each ID is related to a node in the database, indicating the sequence of steps to be taken. After loading each route in its respective vehicle they strictly follow this route, taking different actions only when they arrive at a machine node. When arriving at a machine node the vehicle will await indefinitely until it can either receive or deliver a part, depending on the type of node (input or output). Since this is a no-buffer environment, the machines only accept a new part when it does not have another part in the output node.

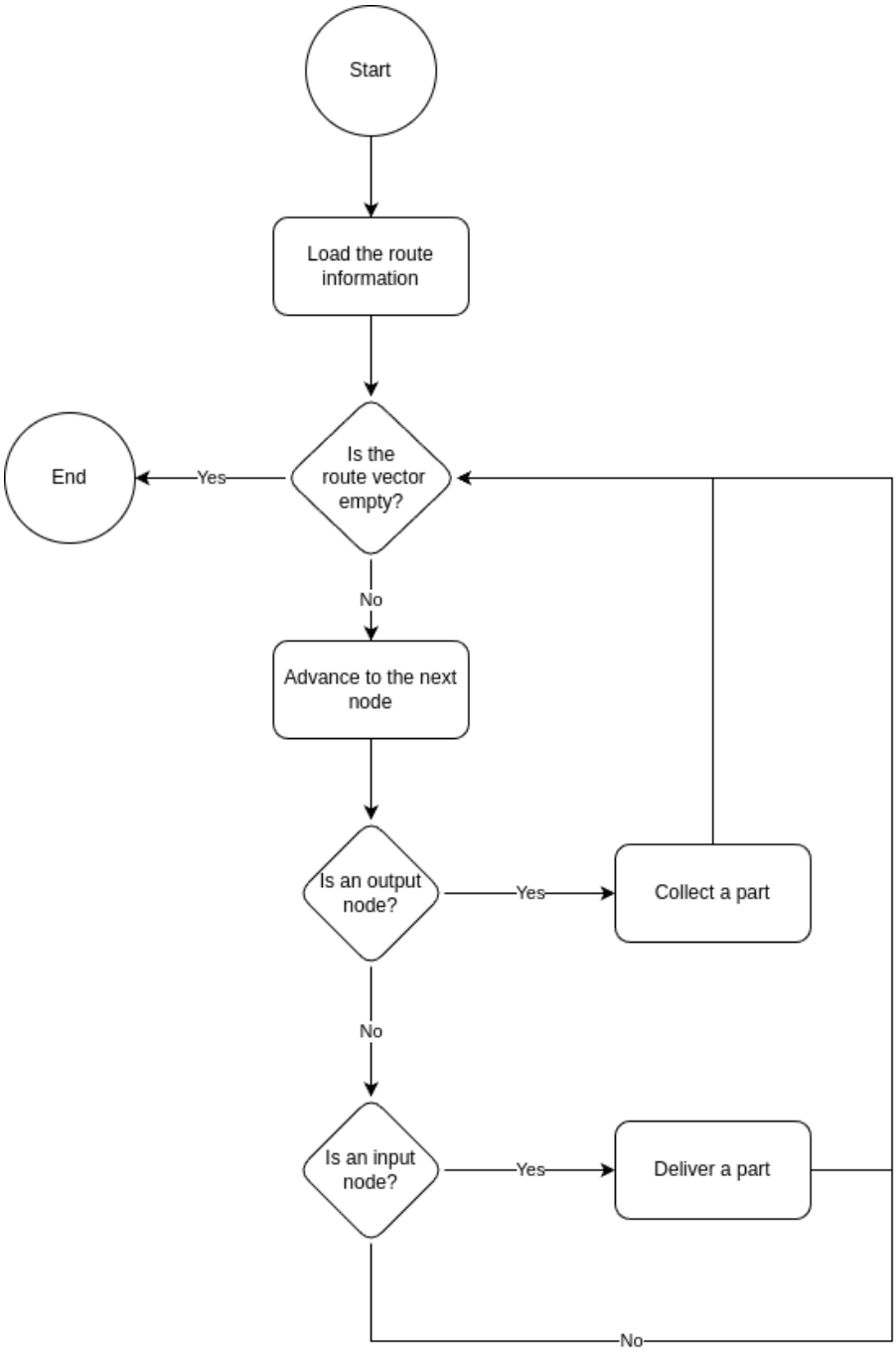


Figure 4.16: Vehicle action flowchart

As can be seen in Figure 4.16, the vehicles do not have any way to verify/check if they actually have a part when delivering or if they are empty when collecting. This is by design, as the simulation should not be relied upon to assist in solving the problem.

An incorrect route will simply result in an infinite simulation, which will be forcefully terminated after extending the maximum time. Since the simulation will never reach the final step of updating its record in the database, a search for every simulation with the boolean flag `was_executed` set to false can discover if the algorithm is outputting incorrect solutions.

Collision between vehicles do not affect the behavior of the agents in any way, but they generate a collision record that is stored in the database with time and location of the collision.

4.3.2 The `ExperimentValidator` Model

Through the comodel abstraction GAMA allows the modeler to instantiate one simulation inside another. This has been leveraged to create a simple loop to instantiate Shop models for each experiment that does not have a matching complete simulation.

Running the simulation over all routes produced guarantees two things: That all algorithms produce valid routes and that the routes produced by two different algorithms can be accurately compared.

Chapter 5

Results and Discussions

This chapter is divided in 4 sections, each one aiming to answer one of the proposed research question from Section 3.1. Section 5.1 will show the impact of the number of vehicles on the total completion time. Section 5.2 will examine if this impact is maintained when the movement of the vehicles is heavily constrained. Section 5.3 will compare scenarios 1 and 3 to determine how much time is lost when avoiding collisions with the TEA* algorithm. Section 5.4 will evaluate if there are differences when the graph representation uses TEA*-derived paths instead of the shortest paths inside the GA's fitness evaluation.

5.1 Results for Research Question 1

In Scenario 1, the aim is to explore the impact of the number of vehicles on the total completion time for different combinations of parts and different machine processing times. As this idealized environment assumes that each vehicle follows the shortest possible path without interference from other vehicles, the results found on this section establish a best-case scenario for similar starting conditions.

The central element to this problem is the joint impact that both the transport time and the production time have on scheduling, separating it from a pure path-planning or flow shop scheduling problem. For that purpose it is analyzed how the increase in the

number of vehicles affects situations with 6 production times (0, 10, 100, 1000, 10000, 100000). Considering that the average time of a transport operation in this scenario is of 900 units, these production times correspond to an approximated ratio between the production time and the average transport time: 0, $0.01T$, $0.1T$, $1T$, $10T$, $100T$. How these ratios of production and transport time affect the problem also can't be dissociated from the starting combination of the 4 parts present in the system, as they direct what operations are necessary to complete the problem. These 4 parts produce the 81 possible starting combinations of red, green and blue parts. At last, it is evaluated how these systems behave for 1, 2, 3 and 4 vehicles, using the values for 1 vehicle as the benchmark for worst possible case. This amounts to a total of $6 \cdot 81 \cdot 4 = 1944$ unique situations to be optimized and simulated.

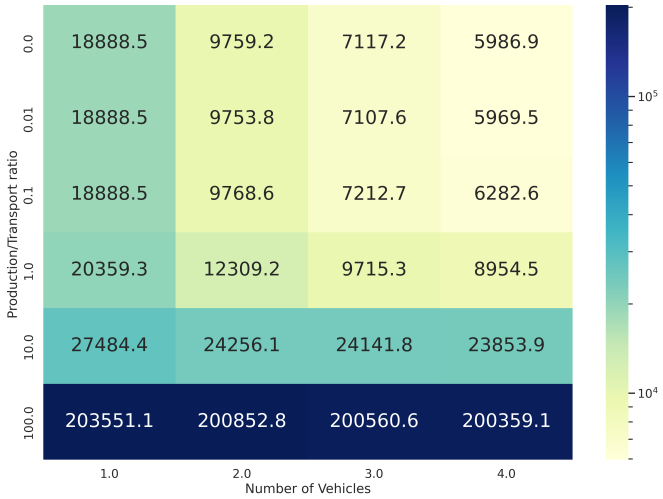


Figure 5.1: Average Makespan by Number of Vehicles and Production/Transport time ratio for Scenario 1

The results on Figure 5.1 display an interesting behavior: The production time has no impact on the makespan for one vehicle when production time is smaller than the average transport time. When visualizing the simulations, it's possible to observe that this happens because the travel time for the vehicle to get from the input node of one machine to its output node is higher than the processing time, making it seem like an effective instant operation from the point of view of the vehicle. On the opposite end

of the spectrum, while the number of vehicles has a small but noticeable impact when production takes 10 times as long as the average transport, the benefits vanish completely beyond that.

To get a more precise view of how much the number of vehicles impacts the total makespan of the system, the efficiency gain of a number V of vehicles when compared to its equivalent with one vehicle is calculated with Equation 5.1.

$$efficiency\_gain(V) = \frac{makespan(1) - makespan(V)}{makespan(1)} \tag{5.1}$$

As there are 486 different instances for each number of vehicles, the efficiency gain of Equation 5.1 is calculated for the instances with 2,3 and 4 vehicles for a total of 1458 values.

The mean and standard deviation of these values are calculated for each pair of P/T ratio and number of vehicles, and the results are presented in Table 5.1. The results show a high gain when the P/T ratio is low, meaning that the gain is higher when the transport operations are the dominant variable in the makespan. A lower gain is observed when production is more dominant, with little deviation from the mean when the relationship between these 2 variables is clearly skewed towards one side.

Table 5.1: Average efficiency gain in makespan when compared to the same scenario with one vehicle. In this table a higher value is better, as it indicates that the overall makespan is lower.

Production\Transport Ratio	Number of Vehicles					
	2		3		4	
	Mean(%)	σ	Mean(%)	σ	Mean(%)	σ
0	48.19	1.02	62.22	1.23	67.85	0.73
0.01T	48.37	0.96	62.37	0.85	68.03	0.53
0.1T	48.93	0.59	61.98	1.20	66.81	0.64
1.0T	38.64	5.27	51.87	3.22	55.43	6.37
10T	11.17	3.11	11.53	2.96	12.61	5.00
100T	1.05	0.95	1.16	1.08	1.25	1.21

There is one clear outlier in Table 5.1, and that is the case where the transport and production times are roughly equal. As can be seen more clearly when plotted in Figure

5.2, the case where $P = T$ has a much bigger deviation from the mean than the other cases and the cause is not immediately clear. This is possibly related to different starting setups being more or less susceptible to the number of vehicles when the production and transport times are equal.

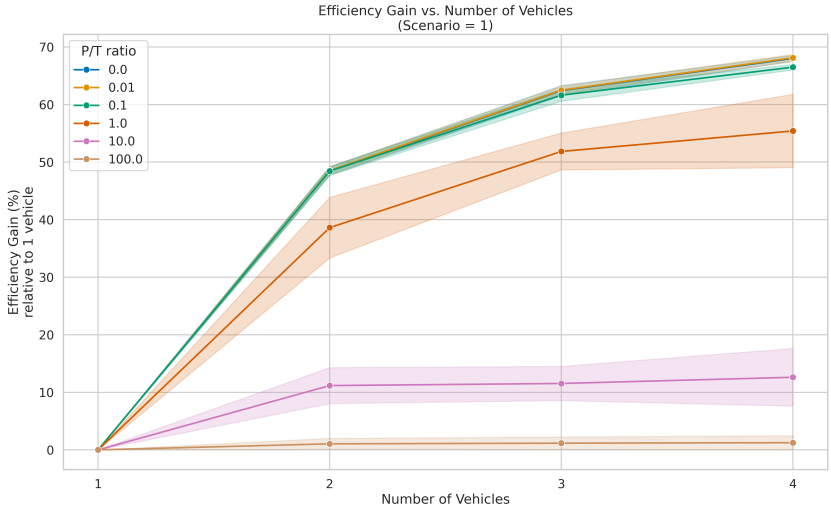


Figure 5.2: Increase in Efficiency when increasing the number of vehicles, for different processing times

Since it is unreasonable to plot all 81 possible starting setups in a single graph and still obtain a good visualization, three special cases have been singled out: The case where all initial parts are red, the case where all initial parts are green and the case where all initial parts are blue. By plotting the efficiency of multiple vehicles for these 3 cases the results in Figure 5.3 are reached.

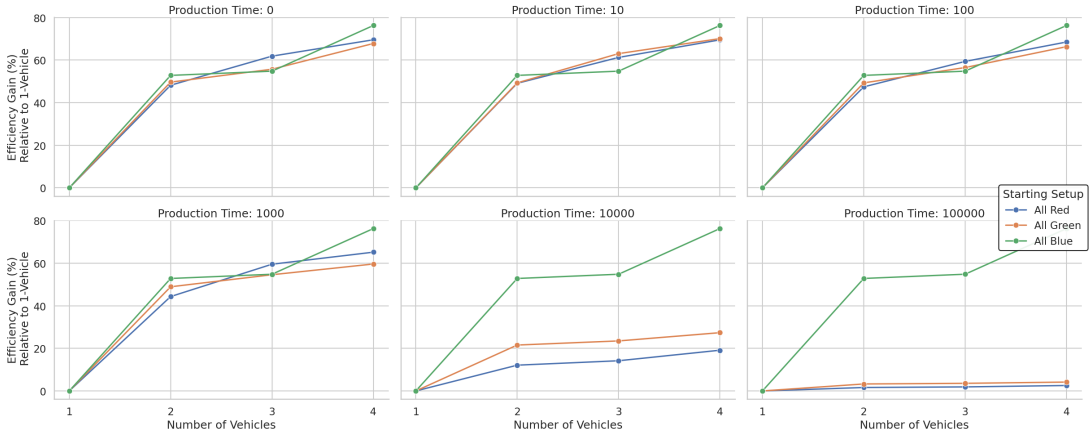


Figure 5.3: Efficiency gain for each homogeneous part combination separated for each production time

On most scenarios in Figure 5.3 the lines for red and green parts usually stay close to each other, indicating that the efficiency is much more related to how much time the part must stay in a given machine than to how many times it must be processed. The case where this does not hold as much is exactly the one of interest to us, where $P = T$. This difference in efficiency implies that the change in the number of operations (as red parts go through 2 operations while green parts go through only 1) brings a change to the *balance* between production and transport operation by slightly changing the proportions of their effects on the total makespan. The all-blue setup is a special case in this environment, because its special rule (that it does not have a production time) means that it produces the same result in all scenarios for all production times. Since it is a problem with only transport operations it is also expected that it will have the highest gains from additional vehicles.

From these results it is possible to conclude that the added complexity of multiple vehicles only justifies itself in situations where the average production time is equal or less than the average transport time, and even then this benefit only extends to at most 3 vehicles.

5.2 Results for Research Question 2

As described in Section 3.3, scenario 2 proposes to examine how a simple heuristic such as restricting the vehicles to operate in different areas with little overlap can reduce the collision rate while still avoiding the added complexity of multi-vehicle path-planning.

Before analyzing the results of scenario 2, consider how prevalent collisions are in the routes of scenario 1. Table 5.2 show a high rate of collisions for every instance with two or more vehicles, with increasing odds of collision with higher number of vehicles.

Table 5.2: Scenario 1 Collision Data

Vehicles	Simulations	Simulations with collision	% of simulations with collisions
2	486	342	70.37%
3	486	403	82.92%
4	486	419	86.21%

While this result is not unexpected, given that each vehicle's path was planned independently of the others, it introduces uncertainty regarding whether the observed potential gains from using multiple vehicles would be achievable in real-world scenarios. Scenario 2 is designed to provide a partial answer to this question without increasing the complexity of the problem.

After running the simulations for 2 and 3 vehicles and collecting the collision data in Table 5.3, it is already possible to see much more promising results, with a very low collision rate in both situations when compared with Table 5.2.

Table 5.3: Collision data for scenario 2

Vehicles	Simulations	Simulations with collisions	% of simulations with collisions
2	486	20	4.12%
3	486	33	6.79%

This reduction in collisions comes at a very heavy cost, as the data in Figure 5.4 shows. When plotting the efficiency of these results in relation to the equivalent situations with a

single vehicle in Figure 5.5, a weaker gain can be observed for 2 vehicles when compared to Figure 5.2 and a complete stagnation for 3 vehicles, with effectively no advantage when compared to the scenario with 2 vehicles.

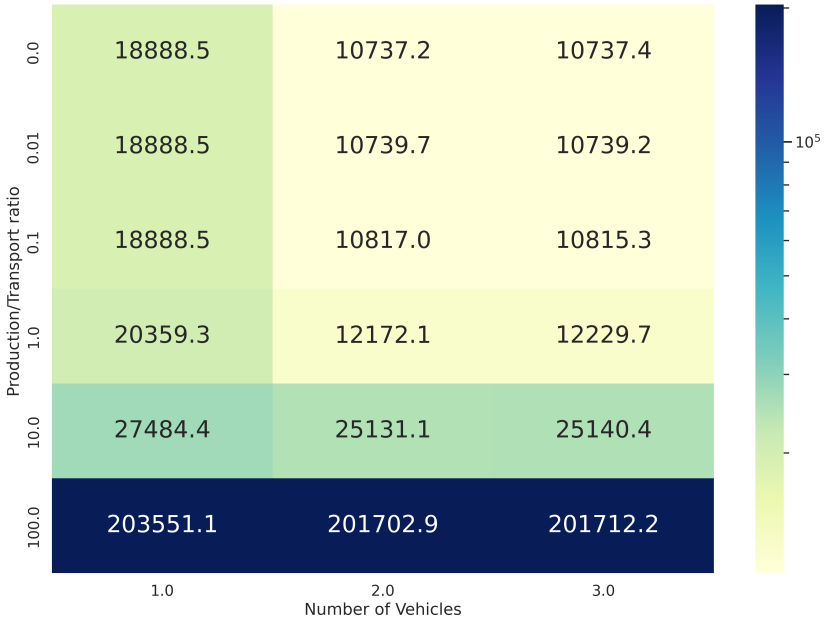


Figure 5.4: Average Makespan by Number of Vehicles and Production Time for Scenario 2

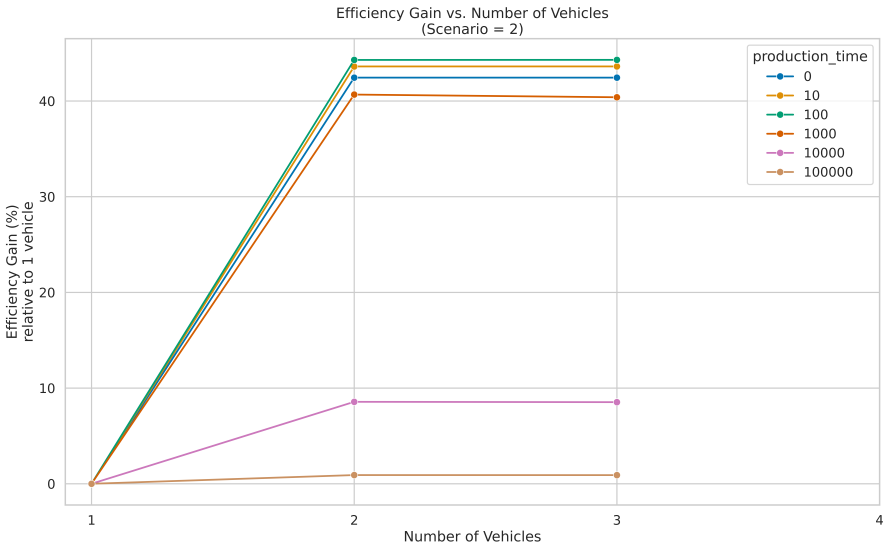


Figure 5.5: Efficiency gain for multiple vehicles when compared with the same scenario for a single vehicle

The results in Figure 5.5 indicate that the efficiency advantage maintains itself for two vehicles when adding spatial restrictions, but the advantage is not maintained for three vehicles. This shows that when considering very restricted scenarios there is no justification to use more than two vehicles, and even then there is considerable room for improvement with more complex approaches.

5.3 Results for Research Question 3

As the results in Section 5.2 show, limiting the scenario is a viable option to diminish the number of collision but it heavily penalizes the total makespan of the system. In search of results with a better performance the use of multi-vehicle path-planning in the form of the TEA* algorithm is considered, using it to build collision-free routes to the scheduling results of scenario 1. To see the impact from these new routes (that is, the time lost to perform additional maneuvers to avoid collisions) the percentage difference between the makespan in scenario 3 and the makespan in scenario 1 is calculated for each pair of number of vehicles V and starting setup s as written in Equation 5.2.

$$difference(V, s) = \frac{makespan_{scn3}(V, s) - makespan_{scn1}(V, s)}{makespan_{scn1}(V, s)} \tag{5.2}$$

Equation 5.2 is applied for all instances of scenario 1 that have more than one vehicle and that *had some form of collision* as shown in Table 5.2, meaning that this difference is collected for a total of 1164 simulations (342 for 2 vehicles, 403 for 3 vehicles and 419 for 4 vehicles). This is done because it does not make sense to apply the TEA* algorithm to a collection of paths known to have no collision, as the algorithm will return the exact same path. This could skew some results towards 0, as there is no difference between scenarios 1 and 3 in these cases.

The average increase in makespan between scenarios 1 and 3 for all combination of parts is shown in table 5.4 as the percentage mean and standard deviation for each number of vehicles and production times. A first glance at the results shows that, as expected

for any change that only affects the transport time, the application of TEA* has no measurable effect on those instances where the makespan is dominated by the production time($10T$ and $100T$).

Moving to the instances where there is an observable difference, the coefficient of variation of these averages (that is, if each standard deviation is divided by its mean) for almost all of them (with the exception of production time $1T$) have values higher than 1, indicating that there is high variability in these results and that the mean values may be misleading. It is possible that the time lost maneuvering to avoid collision may be correlated to the initial setup of parts in each instance, leading to this high variability when all the different combinations are aggregated.

Table 5.4: Average relative difference between the total makespan of scenario 3 and the total makespan of scenario 1 for the same combination of parts. In this table a lower value is better, as it shows that there is less time lost to avoid collisions

Production\ Transport Ratio	Number of Vehicles					
	2		3		4	
	Mean(%)	σ	Mean(%)	σ	Mean(%)	σ
0	1.26	1.53	4.03	4.22	6.02	4.79
0.01T	1.28	1.58	3.2	3.88	5.04	4.63
0.1T	1.32	1.50	3.08	3.80	4.18	4.38
1.0T	2.58	1.09	3.92	2.23	4.36	2.46
10T	0.03	0.15	0.03	0.04	0.04	0.05
100T	0.03	0.16	0.03	0.04	0.04	0.05

To observe how the difference in makespan is distributed across the different configurations of parts, four different groupings were established:

$$\text{Part Combination} = \begin{cases} \text{Mostly Red} & \text{if at least 3 parts are red} \\ \text{Mostly Green} & \text{if at least 3 parts are green} \\ \text{Mostly Blue} & \text{if at least 3 parts are blue} \\ \text{Mixed} & \text{otherwise} \end{cases}$$

The behavior for each of these groups was recorded for the 4 initial ratios, as the

values for the last 2 do not present enough variability for a meaningful analysis. Table 5.5 holds these results, where it can be observed that the variability of the results, although still high, has decrease when compared to the results in Table 5.4. This strengthens the hypothesis that part of the variation can be due to the part combination, as certain configurations may be prone to accumulating vehicles in certain areas, incentivizing collisions.

Table 5.5: Average relative difference between makespan of scenarios 1 and 3 grouped by characteristics of the setup. As with Table 5.4, a lower mean is better as it indicates that less time is lost to avoid collisions

Production\Transport Ratio	Part Combination	Number of Vehicles					
		2		3		4	
		Mean(%)	σ	Mean(%)	σ	Mean(%)	σ
0	Mostly Red	0.69	0.26	2.00	1.29	3.33	2.10
	Mostly Green	1.20	0.51	1.84	0.84	3.91	3.02
	Mostly Blue	2.04	1.03	3.40	2.45	4.33	2.74
	Mixed	1.18	0.62	2.78	2.19	3.79	1.89
0.01T	Mostly Red	0.38	0.41	2.25	0.74	2.34	1.90
	Mostly Green	1.20	0.61	2.47	1.62	2.90	2.10
	Mostly Blue	1.18	0.68	2.71	2.33	3.12	2.80
	Mixed	1.17	0.69	2.30	2.15	3.35	2.23
0.1T	Mostly Red	1.50	0.81	1.60	1.04	2.21	0.90
	Mostly Green	1.25	0.69	3.70	3.46	5.46	2.85
	Mostly Blue	1.59	1.24	2.67	2.37	3.56	2.92
	Mixed	1.09	0.49	2.48	1.88	2.70	1.90
1.0T	Mostly Red	3.17	0.70	5.01	1.69	6.76	2.47
	Mostly Green	2.61	0.88	3.83	1.87	5.45	2.47
	Mostly Blue	2.15	1.34	4.07	3.27	3.82	4.6
	Mixed	2.62	0.97	3.91	1.98	4.07	1.76

Considering the results in Figure 5.6, it can be observed more clearly how there is a consistent increase in makespan when the number of vehicles increases, possibly indicating that the higher density of vehicles needs bigger compromises from the routing system on the length of the routes, which may be one of the causes for higher delays.

Observing the behavior for each of the selected groupings, the group of mostly blue parts is consistently the one with the higher standard deviation, while the other 3 present more varied behavior in their standard deviations. The fact that the difference is considerably higher for setups with multiple stages (especially mostly red parts) when the

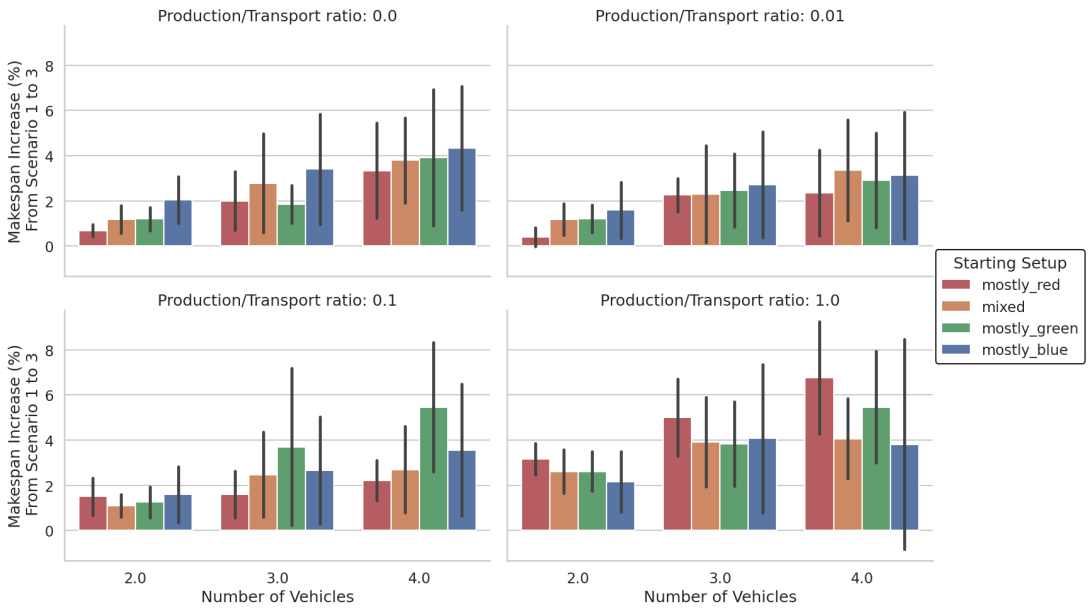


Figure 5.6: Categorized Difference in Makespan Between Scenarios 1 and 3 as shown in Table 5.5

Production/Transport ratio is at $1T$ may indicate that the delays caused by the TEA* algorithm are disrupting the some fine-tuned sequencing of operations by the optimization algorithm as parts do not arrive at their destined locations at the expected time. The group of mostly blue parts is especially interesting as its standard deviation suggests possible negative values. As this is not possible (since the A* in Scenario 1 is guaranteed to return the optimal paths), this indicates that there is a cluster of low values with few high outliers.

All these factors lead to the conclusion that collision avoidance does consistently lead to some delays, although their effect are not so significant. Most of these delays cause an impact of between 2% and 6% on the overall makespan, with the severity of them increasing as the production and transport times approach an equilibrium. The delay is consistently higher with a high number of vehicles, and a more balanced P/T ratio favors more delays for setups with red parts.

5.4 Results for Research Question 4

Considering the increase in makespan found in Section 5.3 due to the TEA*, a solution is proposed that might diminish it: The integration of the TEA* algorithm in the fitness function of the Genetic Algorithm. Since the original GA in scenario 1 optimizes its scheduling sequence with the expectation that the vehicles will follow the routes planned by the A* algorithm, it was hypothesized that it may be producing schedules that are unfavorable to the TEA*, such as placing the vehicles to constantly navigate crowded environments or expecting perfect timing in pickup and deliveries.

Table 5.6: Average difference between the total makespan of scenario 4 and the total makespan of scenario 3 for the same combination of parts

Production\Transport Ratio	Number of Vehicles					
	2		3		4	
	Mean(%)	σ	Mean(%)	σ	Mean(%)	σ
0	-0.14	1.48	-1.00	2.22	-1.46	2.66
0.01T	-0.14	1.70	-0.75	2.04	-0.76	3.15
0.1T	-0.22	1.39	-0.60	2.01	-0.90	2.46
1.0T	0.0	0.19	-0.20	0.64	-0.26	0.74
10T	0.0	0.04	-0.02	0.04	-0.03	0.05
100T	0.0	0.03	-0.02	0.04	-0.03	0.05

Considering the result in Table 5.6, in the best case the final makespan diminishes by 1.4%, and this result has enough variance to make it inconclusive if there is any benefit at all to this strategy.

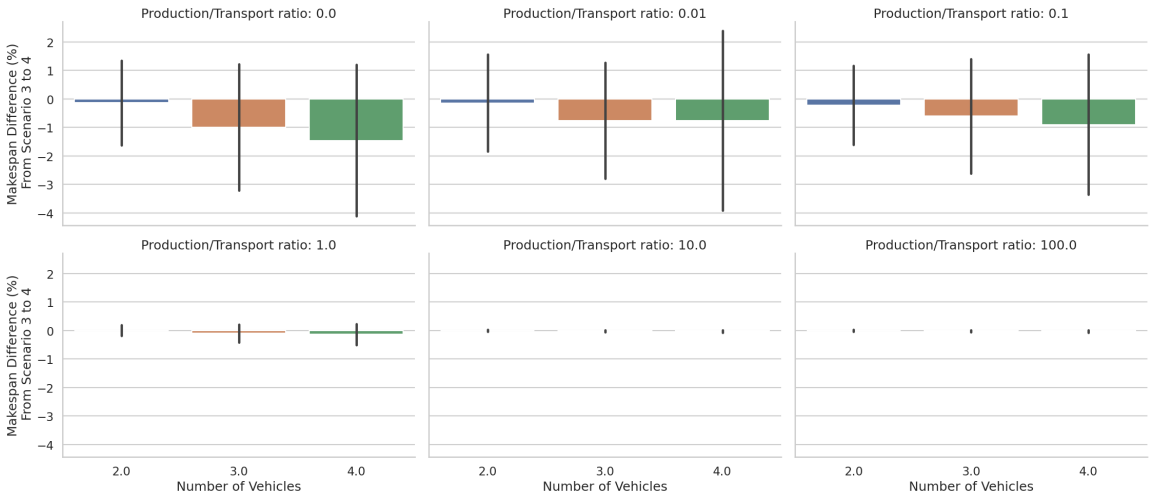


Figure 5.7: Average difference between the total makespan of scenarios 3 and 4

The data in Figure 5.7 shows more clearly how this approach does not achieve significant difference in any situation and does not justify the added complexity and computational cost needed for it.

5.5 Summary of the Results

To summarize what was found:

For the first research question: "*when applying a hybrid flow shop scheduling model to the RobotAtFactory environment - ignoring collision constraints — how significant is the impact of the number of vehicles on the total completion time for different combinations of parts and different machine processing times?*". Based on the results found in Section 5.1, it can be concluded that the number of vehicles is extremely significant when the average production time is less or equal than the average transport time, but the benefits drastically decrease for every vehicle that is added. It can also be concluded that the initial combination of parts does not have a significant effect on the impact of multiple vehicles.

The second research question "*given that the analysis conducted in the previous question indicates a potential reduction in completion time with increased vehicle numbers,*

does enforcing fixed spatial constraints (to prevent collisions, making each robot move on mutually exclusive sectors of the environment) maintain this efficiency advantage?" can be answered with a partial yes. As found in Section 5.2, the advantage is maintained for 2 vehicles although with approximately half the gain observed in Section 5.1 and is completely lost (that is, the advantage remains stagnant) when moving to the case with 3 vehicles.

For the third research question: "*considering that the ideal scheduling scenario (from Question 1) does not account for collision avoidance, the TEA* algorithm is used to plan conflict-free paths for all vehicles. How close do these TEA*-derived paths come to achieving the ideal completion time?*", it was found that the TEA*-derived paths do indeed impose some delays on the total makespan, especially when the production time and transport time are approximately equal. This is an interesting result as it also shows that instances populated primarily by red parts had higher delays when the production and travel times where equal, indicating a possible range of values where the amount of operations realized on the system is relevant to the importance of transportation times in the overall makespan.

For the fourth research question "*the GA's fitness value is computed using an approximated time (based on shortest paths) but may differ from the true execution time when more realistic routing (using TEA*) is applied. As the evaluation becomes increasingly realistic, is there a significant difference in optimization efficiency?*", it was found that there is no advantage to the use of joint routing and scheduling in comparison with the sequential approach used in the third research question. This indicates that the approximated times used during the fitness evaluation of Scenario 1 are close enough to the real paths to not negatively impact the optimization.

Chapter 6

Conclusions

This work aimed to study the challenges for warehouse scheduling in the context of the RobotAtFactory competition, exploring if better task schedules could be produced when optimized together with the associated multi-robot path-planning. In order to achieve that it was first necessary to understand how the competition structure fit inside of the theoretical framework of the many optimal job scheduling problems, how a solution to this problem could be represented as a graph structure and how this solution behaved when multi-vehicle scenarios where considered.

In accordance with the first objective in this sequence the competition was categorized as a Hybrid Flow Shop Scheduling Problem with Transportation Resources, as the competition features all the main characteristics of this type of problem such as a specific order of stages that is always followed, multiple machines to be chosen per stage and transportation between machines being done by a limited number of vehicles. As a theoretical representation was established, it was possible to adapt the graph representation for the Flexible Job Shop Scheduling Problem with Transportation Resources to fit the RobotAtFactory competition. The graph was changed on how it represented consecutive tasks on shared resources to account for the lack of buffers and the detail in the arguments for edge weight calculation was increased to account for different pickup and delivery locations for each machine. This achieves the goal specified in Objective 1 of developing an optimal job scheduling representation for the RobotAtFactory competition.

The first three research questions encompass what was needed to accomplish the goal of Objective 2, as they show how the environment benefits from different number of vehicles organized under more efficient (as in scenario 1) and more inefficient (as in scenario 2) scheduling schemes. The results from question 1 show how there is a massive gain in time saved when 2 vehicles are used for all cases where the transport time is at least mildly relevant to the final makespan, and how this advantage still exists in the severely restricted environment proposed in question 2. The results for question 3 showed a decrease in makespan that is still much smaller than the gains in the results of scenario 1, further confirming that these gains can be achieved on real situations. The diminishing returns are clear when adding more vehicles as they give smaller gains in scenario 1 and no gains at all in scenario 2, showing that the use of 3 or more vehicles can only be taken advantage of with the aid of more complex strategy for multi-robot path planning such as the one from scenario 3. With these results in mind, it was concluded that the ideal number of robots to solve the RobotAtFactory competition is 2, as they offer enough benefits even when used with very simple strategies, offering the best tradeoffs between makespan and problem complexity.

The fourth research question is directly related to Objective 3 of this work, that is, the development of a joint scheduling and routing algorithm. It was found that the joint approach brings no benefits on average, and its variation is well within what is expected from a heuristic algorithm such as GA. This leads us to the conclusion that, for this environment, there is no advantage in using a joint approach for scheduling and routing over executing them in a sequential manner. It is believed that this happens due to a lack of complexity in the environment, both in size and arrangement. A lack of complexity in size because there are not enough bottlenecks and the overall shape of the shop floor is extremely symmetrical, which results in multiple routes with approximately the same travel time, thus defeating the purpose of searching for alternative routes. The lack of complexity in arrangement refers to the impact of the machine assignments, as they lack diversity of location. Just as many similar routes defeat the purpose of looking for alternatives, when all machines available for a specific stage are located in very close

proximity an alternative choice of assignment does not impact the overall route, making the choice of machine irrelevant from the point of view of transport time.

In conclusion, the 3 proposed objectives at the beginning of this work were achieved. The proposed RaF representation for Objective 1 takes into account characteristics not seen in related works (as seen in Table 2.1), such as zero buffers and separate pickup and delivery locations. The analysis for Objective 2 shows how 2 vehicles always produces advantages when transport times are relevant, but larger numbers of vehicles are dependant on the scheduling strategy. The algorithm for joint optimization for Objective 3 points to a promising direction for research, even if it does not produce advantages in this environment.

Future Works

There are many avenues to be explored in future works. Works interested in expanding the RobotAtFactory model to dynamic problems could analyze how the model behaves when exposed to parts arriving at the incoming warehouse at random intervals. Works focused on the joint scheduling and routing problem could explore if changes in the the general infrastructure (quantity and location of machines, changes in the spatial graph) do cause the joint scheduling and routing algorithm to perform better.

Another possible avenue would be to explore the effects of vehicle diversity, as this work assumes that all vehicles are identical. Works could explore how the environment behaves for vehicles with different speeds or different drive types (differential wheeled versus omnidirectional robots, for example). As this would require a more complex movement model inside the simulation, it could also explore if the results found in this research hold for more realistic simulations.

Bibliography

[1] M. Carter, C. C. Price, and G. Rabadi, *Operations Research: A Practical Introduction*, 2nd ed. New York: Chapman and Hall/CRC, Aug. 6, 2018, 470 pp., ISBN: 978-1-315-15322-3. DOI: 10.1201/9781315153223.

[2] M. R. Garey, D. S. Johnson, and R. Sethi, "The Complexity of Flowshop and Jobshop Scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117–129, May 1976, ISSN: 0364-765X. DOI: 10.1287/moor.1.2.117. [Online]. Available: https://pubsonline.informs.org/doi/10.1287/moor.1.2.117 (visited on 01/21/2025).

[3] R. de Koster, T. Le-Duc, and K. J. Roodbergen, "Design and control of warehouse order picking: A literature review," *European Journal of Operational Research*, vol. 182, no. 2, pp. 481–501, Oct. 16, 2007, ISSN: 0377-2217. DOI: 10.1016/j.ejor.2006.07.009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221706006473 (visited on 08/31/2024).

[4] V. P. Paulo Costa José Lima, *Rules for the robot at factory 4.0 competition*, https://github.com/P33a/RobotAtFactory/blob/04043003893b697020ca49383da484484b1ac2b8/RobotAtFactory_4_0_2024_Rules.pdf, Accessed: 2025-02-04, 2024.

[5] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. Springer, Feb. 10, 2016, 674 pp., ISBN: 978-3-319-26580-3.

[6] H. Xiong, S. Shi, D. Ren, and J. Hu, "A survey of job shop scheduling problem: The types and models," *Computers & Operations Research*, vol. 142, p. 105 731, Jun. 1, 2022, ISSN: 0305-0548. DOI: 10.1016/j.cor.2022.105731. [Online]. Available:

https://www.sciencedirect.com/science/article/pii/S0305054822000338 (visited on 08/28/2024).

[7] J. N. D. Gupta and E. F. Stafford, "Flowshop scheduling research after five decades," *European Journal of Operational Research*, vol. 169, no. 3, pp. 699–711, Mar. 16, 2006, ISSN: 0377-2217. DOI: 10.1016/j.ejor.2005.02.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221705001372 (visited on 04/10/2025).

[8] P. Brucker and R. Schlie, "Job-shop scheduling with multi-purpose machines," *Computing*, vol. 45, no. 4, pp. 369–375, 1990. DOI: 10.1007/BF02238804.

[9] R. Ruiz and J. A. Vázquez-Rodríguez, "The hybrid flow shop scheduling problem," *European Journal of Operational Research*, vol. 205, no. 1, pp. 1–18, Aug. 16, 2010, ISSN: 0377-2217. DOI: 10.1016/j.ejor.2009.09.024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221709006390 (visited on 01/08/2025).

[10] A. Hosseini, A. Otto, and E. Pesch, "Scheduling in manufacturing with transportation: Classification and solution techniques," *European Journal of Operational Research*, vol. 315, no. 3, pp. 821–843, Jun. 16, 2024, ISSN: 0377-2217. DOI: 10.1016/j.ejor.2023.10.013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221723007701 (visited on 12/04/2024).

[11] J. Błażewicz, W. Domschke, and E. Pesch, "The job shop scheduling problem: Conventional and new solution techniques," *European Journal of Operational Research*, vol. 93, no. 1, pp. 1–33, Aug. 23, 1996, ISSN: 0377-2217. DOI: 10.1016/0377-2217(95)00362-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0377221795003622 (visited on 04/10/2025).

[12] S. Dauzère-Pérès and J. Paulli, "An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search," *Annals of Operations Research*, vol. 70, no. 0, pp. 281–306, Apr. 1, 1997, ISSN: 1572-9338.

DOI: 10.1023/A:1018930406487. [Online]. Available: https://doi.org/10.1023/A:1018930406487 (visited on 04/10/2025).

[13] J. Hurink and S. Knust, "Tabu search algorithms for job-shop problems with a single transport robot," *European Journal of Operational Research*, Logistics: From Theory to Application, vol. 162, no. 1, pp. 99–111, Apr. 1, 2005, ISSN: 0377-2217. DOI: 10.1016/j.ejor.2003.10.034. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221703008221 (visited on 04/10/2025).

[14] P. Lacomme, M. Larabi, and N. Tchernev, "Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles," *International Journal of Production Economics*, vol. 143, no. 1, pp. 24–34, May 1, 2013, ISSN: 0925-5273. DOI: 10.1016/j.ijpe.2010.07.012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925527310002525 (visited on 04/10/2025).

[15] L. Berterottière, S. Dauzère-Pérès, and C. Yugma, "Flexible job-shop scheduling with transportation resources," *European Journal of Operational Research*, vol. 312, no. 3, pp. 890–909, Feb. 1, 2024, ISSN: 0377-2217. DOI: 10.1016/j.ejor.2023.07.036. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221723005969 (visited on 11/18/2024).

[16] A. Koubaa, H. Bennaceur, I. Chaari, *et al.*, "Introduction to Mobile Robot Path Planning," in *Robot Path Planning and Cooperation: Foundations, Algorithms and Experimentations*, A. Koubaa, H. Bennaceur, I. Chaari, *et al.*, Eds., Cham: Springer International Publishing, 2018, pp. 3–12, ISBN: 978-3-319-77042-0. DOI: 10.1007/978-3-319-77042-0_1. [Online]. Available: https://doi.org/10.1007/978-3-319-77042-0_1 (visited on 08/25/2024).

[17] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, "Path planning techniques for mobile robots: Review and prospect," *Expert Systems with Applications*, vol. 227, p. 120 254, Oct. 1, 2023, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2023.120254.

[Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741742300756X (visited on 08/20/2024).

[18] A. Loganathan and N. S. Ahmad, "A systematic review on recent advances in autonomous mobile robot navigation," *Engineering Science and Technology, an International Journal*, vol. 40, p. 101 343, Apr. 1, 2023, ISSN: 2215-0986. DOI: 10.1016/j.jestch.2023.101343. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2215098623000204 (visited on 08/20/2024).

[19] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. DOI: 10.1109/TSSC.1968.300136.

[20] J. Santos, P. Costa, L. F. Rocha, A. P. Moreira, and G. Veiga, "Time enhanced A*: Towards the development of a new approach for Multi-Robot Coordination," in *2015 IEEE International Conference on Industrial Technology (ICIT)*, Mar. 2015, pp. 3314–3319. DOI: 10.1109/ICIT.2015.7125589. [Online]. Available: https://ieeexplore.ieee.org/document/7125589 (visited on 11/12/2024).

[21] J. H. Holland, "Outline for a Logical Theory of Adaptive Systems," *Journal of the ACM*, vol. 9, no. 3, pp. 297–314, 3 Jul. 1962, ISSN: 1557-735X. DOI: 10.1145/321127.321128. [Online]. Available: https://articles.researchsolutions.com/10.1145/321127.321128 (visited on 09/15/2024).

[22] A. E. Ezugwu, A. K. Shukla, R. Nath, *et al.*, "Metaheuristics: A comprehensive overview and classification along with bibliometric analysis," *Artificial Intelligence Review*, vol. 54, no. 6, pp. 4237–4316, Aug. 1, 2021, ISSN: 1573-7462. DOI: 10.1007/s10462-020-09952-0. [Online]. Available: https://doi.org/10.1007/s10462-020-09952-0 (visited on 08/31/2024).

[23] M. Çolak and G. A. Keskin, "An extensive and systematic literature review for hybrid flowshop scheduling problems," *International Journal of Industrial Engineering Computations*, vol. 13, no. 2, pp. 185–222, 2022, ISSN: 19232926, 19232934. DOI: 10.

5267/j.ijiec.2021.12.001. [Online]. Available: http://www.growingscience.com/ijiec/Vol13/IJIEC_2021_31.pdf (visited on 01/02/2025).

[24] C. Bierwirth, "A generalized permutation approach to job shop scheduling with genetic algorithms," *Operations-Research-Spektrum*, vol. 17, no. 2, pp. 87–92, Jun. 1, 1995, ISSN: 1436-6304. DOI: 10.1007/BF01719250. [Online]. Available: https://doi.org/10.1007/BF01719250 (visited on 11/18/2024).

[25] W. de Paula Ferreira, F. Armellini, and L. A. De Santa-Eulalia, "Simulation in industry 4.0: A state-of-the-art review," *Computers & Industrial Engineering*, vol. 149, p. 106 868, Nov. 1, 2020, ISSN: 0360-8352. DOI: 10.1016/j.cie.2020.106868. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835220305635 (visited on 09/05/2024).

[26] A. P. Galvão Scheidegger, T. Fernandes Pereira, M. L. Moura de Oliveira, A. Banerjee, and J. A. Barra Montevechi, "An introductory guide for hybrid simulation modelers on the primary simulation methods in industrial engineering identified through a systematic review of the literature," *Computers & Industrial Engineering*, vol. 124, pp. 474–492, Oct. 1, 2018, ISSN: 0360-8352. DOI: 10.1016/j.cie.2018.07.046. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835218303693 (visited on 09/05/2024).

[27] S. Abar, G. K. Theodoropoulos, P. Lemarinier, and G. M. P. O'Hare, "Agent Based Modelling and Simulation tools: A review of the state-of-art software," *Computer Science Review*, vol. 24, pp. 13–33, May 1, 2017, ISSN: 1574-0137. DOI: 10.1016/j.cosrev.2017.03.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013716301198 (visited on 05/16/2025).

[28] P. Taillandier, B. Gaudou, A. Grignard, *et al.*, "Building, composing and experimenting complex spatial models with the GAMA platform," *GeoInformatica*, vol. 23, no. 2, pp. 299–322, Apr. 1, 2019, ISSN: 1573-7624. DOI: 10.1007/s10707-018-00339-6. [Online]. Available: https://doi.org/10.1007/s10707-018-00339-6 (visited on 12/02/2024).

[29] GAMA. "Using comodel." (2022), [Online]. Available: https://gama-platform.org/wiki/Comodel.

[30] R. Bürgy and H. Gröflin, "The blocking job shop with rail-bound transportation," *Journal of Combinatorial Optimization*, vol. 31, no. 1, pp. 152–181, Jan. 1, 2016, ISSN: 1573-2886. DOI: 10.1007/s10878-014-9723-3. [Online]. Available: https://doi.org/10.1007/s10878-014-9723-3 (visited on 05/18/2025).

[31] B. Zhou and Y. Lei, "Bi-objective grey wolf optimization algorithm combined Levy flight mechanism for the FMC green scheduling problem," *Applied Soft Computing*, vol. 111, p. 107 717, Nov. 1, 2021, ISSN: 1568-4946. DOI: 10.1016/j.asoc.2021.107717. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494621006384 (visited on 05/18/2025).

[32] X. Lyu, Y. Song, C. He, Q. Lei, and W. Guo, "Approach to Integrated Scheduling Problems Considering Optimal Number of Automated Guided Vehicles and Conflict-Free Routing in Flexible Manufacturing Systems," *IEEE Access*, vol. 7, pp. 74 909–74 924, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2919109. [Online]. Available: https://ieeexplore.ieee.org/document/8723142 (visited on 05/18/2025).

[33] T. Nishi, Y. Hiranaka, and I. E. Grossmann, "A bilevel decomposition algorithm for simultaneous production scheduling and conflict-free routing for automated guided vehicles," *Computers & Operations Research*, vol. 38, no. 5, pp. 876–888, May 1, 2011, ISSN: 0305-0548. DOI: 10.1016/j.cor.2010.08.012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054810001760 (visited on 05/18/2025).

[34] M. Saidi-Mehrabad, S. Dehnavi-Arani, F. Evazabadian, and V. Mahmoodian, "An Ant Colony Algorithm (ACA) for solving the new integrated model of job shop scheduling and conflict-free routing of AGVs," *Computers & Industrial Engineering*, Applications of Computational Intelligence and Fuzzy Logic to Manufacturing and Service Systems, vol. 86, pp. 2–13, Aug. 1, 2015, ISSN: 0360-8352. DOI: 10.1016/

j.cie.2015.01.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835215000121 (visited on 05/18/2025).

[35] J. Liu, B. Sun, G. Li, and Y. Chen, "An integrated scheduling approach considering dispatching strategy and conflict-free route of AMRs in flexible job shop," *The International Journal of Advanced Manufacturing Technology*, vol. 127, no. 3, pp. 1979–2002, Jul. 1, 2023, ISSN: 1433-3015. DOI: 10.1007/s00170-022-10619-z. [Online]. Available: https://doi.org/10.1007/s00170-022-10619-z (visited on 05/18/2025).

[36] J. Blank and K. Deb, "Pymoo: Multi-Objective Optimization in Python," *IEEE Access*, vol. 8, pp. 89 497–89 509, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2990567. [Online]. Available: https://ieeexplore.ieee.org/document/9078759 (visited on 05/16/2025).

[37] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11–15.