Tampere University

Jinhan Gao

DYNAMIC MULTIVARIABLE OPTIMIZATION FOR ROUTING IN HIGH-DENSITY MANUFACTURING TRANSPORTATION SYSTEMS

Master of Technology Thesis
Faculty of Information Technology and Communication Sciences
Examiner: Prof. Jose Luis Martinez Lastra
Luis Enrique Gonzalez Moctezuma
Prof. Azwirman Gusrialdi
Prof. Joni Kämäräinen
August 2023

ABSTRACT

Jinhan Gao: Dynamic multivariable optimization for routing in high-density manufacturing transportation systems
Master of Technology Thesis
Tampere University
August 2023

Intelligent manufacturing systems play a pivotal role within the framework of Industry 4.0. It encompass an array of diverse resources, including robots, conveyors, sensors, among others. The effectiveness of such systems hinges upon the adept scheduling of all available resources to successfully execute production tasks. Hence, scheduling systems assume particular significance. They facilitate a comprehensive and harmonized approach to planning and scheduling across all resources, ensuring the efficient utilization of resources while concurrently expediting the accomplishment of designated tasks.

The central aim of this thesis is to conceptualize and execute a scheduling system encompassing job shop scheduling and route planning components. This system is proficient in generating scheduling plans that concurrently fulfill multiple objectives, including minimizing work time, attaining balanced workloads, and enabling collision-free movement.

Through scenario modeling, this scheduling system is represented as a bilevel optimization problem. Each level constitutes a mixed-integer linear programming problem, that is able to be resolved autonomously through the prediction of interconnected parameters.

The system comprises three core components: parameter prediction, an manufacturing assignment scheduling problem solver, and a route planning problem solver. This configuration equips the system with the capability to effectively handle unforeseen circumstances, such as mechanical failures, while concurrently alleviating the complexity associated with resolving the primary problem.

The simulation outcomes affirm the system's capacity to generate solutions that are both effective and aligned with the stipulated objectives.

Keywords: Smart Manufacturing, Bilevel Optimization, JSSP, Route Planning

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

PREFACE

I would like to extend appreciation to Prof. Jose Martinez Lastra, Prof. Azwirman Gusrialdi, Wael Mohammed, and Luis Gonzalez Moctezuma for their assistance in both the project development and writing work. I also thank FastLab for providing me the chance to work on this project and my colleagues for their support.

I appreciate my parents, aunt, and uncle for their care and support during my studies in Finland. Lastly, I am thankful for my life partner, Aowen, whose support has been crucial to my success in the master's study.

Handan, 22nd August 2023

Jinhan Gao

CONTENTS

1. INTRODUCTION

This chapter serves as an introduction, delving into the backdrop of smart manufacturing, the research context, and challenges explored within this paper. It proceeds to elucidate the targeted objectives and provides an overview of the thesis structure.

1.1 Background

As information technology and AI techniques continue to advance rapidly, the integration of intelligence and information technology within the manufacturing sector has grown in significance. This evolution has coincided with the rise of concepts such as smart factories and unmanned factories, which has gained significant attention in recent times.

Amidst a myriad visions of the future of manufacturing, two overarching points of consensus have emerged:

1. The future factory will witness diminished human involvement, with a substantial portion of operations autonomously executed by machines.
2. Resources will be harnessed to their fullest extent and judiciously orchestrated in the planning process.

These aspects are inherently intertwined with the evolution and exploration of scheduling and planning systems. In alignment with the pursuit of building the future factory, this thesis is motivated to conceive and actualize a planning system.

1.2 Scenario statement and research questions

The scenario explored in this paper entails the requirement for a scheduling algorithm to ensure the coherent allocation of resources within an intelligent manufacturing system encompassing industrial robots and pallet transportation systems. The resultant outputs of this algorithm encompass the scheduling of production tasks and the planning of routes for pallet transportation. This scenario encompasses two distinct components: the production scheduling and the route planning. The research endeavors of this thesis are directed towards addressing the research questions:

1. What is the manufacturing assignment scheduling model of the entrusted testbed?

2. What is the route planning model of the entrusted testbed?
3. How to integrate the job shop scheduling problem and route planning problem?
4. How to find an optimal or close-to-optimal solution within an reasonable computational time?
5. What is the trade-off of the proposed solution?

1.3 Objectives

This thesis is aiming to formulate of a scheduling system that optimizes the speed of producing essential products. To construct this scheduling system, the given scenario is converted into a mathematical model, while the relevant solution methodologies are explored in the literature review segment. The objectives pursued through the implementation of this system encompass:

1. The ability to attain solutions within an acceptable timeframe. 2. Capability to effectively manage unforeseen contingencies, including system failures. 3. Achievement of equitable work distribution among robots. 4. Facilitation of pallet movement to designated locations in minimal time, without collisions.

1.4 Outline

The thesis follows the subsequent structure: Chapter 2 delves into the technical underpinnings of modeling and optimization algorithms. Chapter 3 expounds upon the modeling process and outlines the design of the optimizer. The implementation process is detailed in Chapter 4. Chapter 5 presents the simulation results and their subsequent analysis. Lastly, Chapter 6 concludes the research endeavors and deliberates upon potential avenues for future exploration.

2. LITERATURE REVIEW

The primary objective of this chapter is to critically assess the articles that related to research problems. The sections are organized thematically. Firstly, the section 2.1 presents the concept of smart manufacturing and the essentiality of intelligent schedule algorithm in this field. Secondly, the section 2.2 focus on the main problems, current gap, and typical solutions in the job shop scheduling area. Then, the section 2.3 focus on the literature about the methodology that is used in this paper. Finally, the section 2.4 summarizes past research work and present findings.

2.1 Smart Manufacturing

The ICT (Information and communication technology) has extremely improved competitiveness of the manufacturing industry. Due to the development of ICT, Smart Manufacturing is identified as the fourth revolution and a new paradigm [22]. It consists of various techniques including multi sensors system, cloud techniques, and smart devices, which Human can manage and communicate between themselves [36]. Furthermore, it becomes a new paradigm of manufacturing industry and effects global trends [22].

However, scheduling algorithms in an efficient and effective manner are necessary to achieve optimal output with all the recent developments in smart manufacturing systems [3]. The scheduling solutions provide the ability for smart manufacturing to adapt to dynamic and real-time environments [37]. Past scheduling algorithms have focused on the research of fixed production models, however, the study of scheduling algorithms that deal with more flexible and dynamic scenarios has been increasingly significant in recent years [3].

To meet the growing need of efficient production in manufacturing industries, the investigation of solving job shop schedule problem is therefore apparent.

2.2 Job Shop Scheduling Problem

The field of manufacturing assignments scheduling pertains to the investigation of the most efficient scheduling of numerous production machines. The utilization of this technology is prevalent in various industries and has resulted in a substantial volume of aca-

demic investigations pertaining to diverse implementation contexts [51]. The conventional formulation is depicted in the subsequent manner: given a collection of machines $M = M_1, M_2, M_3, ..., M_m$ in the work space, and a batch of jobs $J = J_1, J_2, J_3, ..., J_n$ is waiting for manufacturing in this work space, each job J_i comprises a series of operations $O = O_1, O_2, O_3, ..., O_n$, and normally, the dominate target is to minimize the makespan or other criterion. The figure 2.1 illustrates an example work space of the job shop problem.

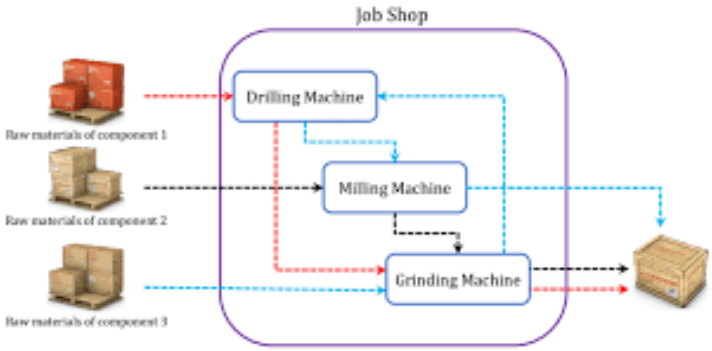


Figure 2.1. *JSSP [53]*

Based on previous studies, the job shop scheduling problem (JSSP) commonly exhibits the following constraints, which necessitate appropriate consideration [54]:

1. If a robot is restricted to performing only one operation at any given moment.
2. If there exists a predefined order of operations for executing a certain job.
3. If a hierarchy of significance exists for different jobs.
4. It is imperative to note that the machine is not susceptible to interruption during the execution of an ongoing operation.

In accordance with the specified constraints and production processes, JSSP can be classified by the following categories [58]:

1. Basic JSSP.
2. Flexible JSSP (FJSSP).
3. Multi-resources FJSSP (MrFJSSP).
4. Multi-plants-based MrFJSSP.

Since the research problem discussed in this thesis is a variant of FJSSP, a more detailed review of it is presented in Section 2.2.1.

Besides the makespan which has been mentioned before, there are other measure performance existing as objects in JSSP [51]. The common types are summarized by [35]: criteria related to processing time, criteria related to working jobs, criteria related to usage, and criteria related to the priority of jobs. Table 2.1 lists some elementary criteria of

these four categories. In JSSP, there can be more than one objective as target measure performance, which is called multi-objective optimization problem.

Table 2.1. *Classification of the JSSP Criterion*

Types	Criterion
Time-based	Makespan
	Cycle Time
	Tardiness
	Sum of completion time
Cost-based	Cost of raw materials per machine
	Total cost per job
	Average cost per job
Work-in-process	Number of jobs in process
	Average number of jobs
	Percentage of tardy jobs
Due-date related	Mean lateness
	Earliness

Over the past few decades, a substantial body of research is presented to the efficient resolution of Job Shop Scheduling Problem (JSSP) that is generally formulated by integer optimization, and various variants based on it are applied to different JSSP, which means that finding a solution to JSSP is actually finding an algorithm to solve its MILP model [32]. The common solutions to JSSP are classified into three types: Mathematical Programming, Heuristic Method, and Artificial Intelligence Method [54]. The table 2.2 presents some typical algorithms of each type.

Table 2.2. *Classification of the JSSP Solving Algorithms*

Types	Example
Exact Algorithms	Branch-and-bound
	Lagrangian Relaxation
Hearistic	Genetic Algorithm
	Hill Clibing Algorithm
	Beam search Algorithm
Data Driven Methods	Reinforcement Learning
	Neural Network
	Graph Learning

2.2.1 Job Shop Scheduling Problem Solving Algorithms

A basic definition of FJSSP has been given in the preceding section, and FJSSP has the subsequent specific assumptions and constraints [50]:

1. The start time of all operations are before zero.
2. Each operation in the context is limited to be processed in only one machine.
3. It is a limitation that each individual machine is capable of executing only a single operation at any given moment.
4. The absence of precedence constraints between operations of distinct jobs is due to their independent nature.
5. The predetermined sequence of operations is established for each occupation.

FJSSP can be further divided into two categories: Total FJSSP means that each machine can handle all operations, and Partial FJSSP means that not necessarily all machines can execute all operations, but at least one of them can [21].

Mathematical planning methods, or exact algorithms, aim to determine the optimal solution of the FJSSP. R.S.Hansmann [18] proposed an approach to solve FJSSP combining fast greedy algorithm and branch-and-bound method. Specifically, he developed a MILP model was developed for a railcar maintenance scenario in which a series of maintenance tasks on railcars contain a fixed order and busy machines block access to subsequent tasks. Zhou [61] performed Petri Net modeling of the flexible manufacturing system and presented an algorithm that incorporates priority setting in the branch-and-bound. AitZai [2] proposed an improved version branch-and-bound method aimed at resolving the manufacturing assignment scheduling while considering the blocking constraint. The primary distinction between the two approaches lies in the formation of the conjunctive graph, which incorporates disjunctive and no-wait arcs. The approach relies by exhaustively exploring potential solutions and incorporates several graph theory techniques, including the identification of strongly connected components, employment of reduced graphs.

Recent years, numerous studies have concentrated on the Surrogate Lagrangian Relaxation Method. The present study [60] introduces the surrogate subgradient approach, which enables the derivation of a suitable direction without the need to solve all subproblems optimally. The algorithm's convergence has been proven, and it is only necessary to optimise one subproblem approximately to obtain a suitable surrogate subgradient direction. This approach has the capability to yield accurate guidance with reduced exertion and presents a novel methodology that is particularly efficacious for issues of considerable magnitude. Bragin [5] devised a decomposition and coordination strategy based on pricing, which leverages the linear contraction of complication and geometric conver-

gence capability of Polyak's stepsizing formula to achieve nearly optimal solutions efficiently. The primary innovative approach for determining stepsizes is based solely on decision-making. Specifically, a new auxiliary constraint satisfaction problem is utilised to deduce suitable stepsizes.

The Surrogate Lagrangian Relaxation Method is largely applied in job shop scheduling problem. Sun [44] expands the Lagrangian relaxation technique to detect and mitigate solution oscillation, and introduces a normalised surrogate subgradient algorithm to expedite the convergence of Lagrangian multipliers. The experimental findings indicate that mitigating solution oscillation leads to the attainment of superior schedules, thereby substantially enhancing conventional approaches. Fang [16] expands upon the Lagrangian relaxation method by proposing a technique to detect and mitigate solution oscillation. The empirical findings indicate the mitigation of solution oscillation leads to the attainment of superior schedules, thereby resulting in a noteworthy enhancement over conventional approaches. Yan [52] devised a methodical formulation for enhancing the precision of a given system. This approach relied on the conversion of constraints and vertices, as well as the projection of vertices. The objective is to optimise a specific component's formulation as the limitations of machine capacity across the entire system are alleviated. The proposed method involves the relaxation of integrality constraints imposed on discrete decision variables. This relaxation leads to the generation of vertices based on the numerical values of the constraints in the resultant LP-relaxed problem. The formulation can be considered tight if all integer decision variables maintain their integer value at every vertex. If non-integer values are not feasible, they will be rounded up or down to the nearest integer.

Cui [11] investigates a steelmaking continuous casting (SCC) scene, which is a hybrid flowshop scheduling (HFS) problem.A MILP model containing a 0/1 variable is formulated for the SCC process and a constraint containing two decision variables is relaxed into the objective function by Lagrangian Relaxation Method. Because the obtained relaxed function is still difficult to solve, it is converted into a DC programming problem through Sum of Squares formula which is easy to solve by cplex. Deflected surrogate subgradient method is applied for updating the Lagrangian multipliers. The convergence of DC algorithms and deflected surrogate subgradient method have been proven.A constructive heuristic method is applied to fix this solution since the Lagrangin Method usually only returns a lower bound. Torabi [47] formulated a novel MINLP model to address a prevalent lot-scheduling problem involving multiple products in a common cycle. Additionally, an iterative enumeration technique was suggested to solve the problem.

Exact algorithms can ensure the identification of an optimal solution but at the cost of running speed, Heuristic methods are fast enough and get good enough solutions in large-scale optimization problems. Nature has created complex life phenomena and natural wonders, providing an inexhaustible source of wisdom for the intelligent process of

humans. inspired by this, computational intelligence algorithms are presented, such as genetic, simulated annealing algorithm, etc. [54]. [34] introduces an improved genetic algorithm (GA) for FJSSP. The presented GA enhances certain strategies that have been previously documented in biography and combines them to determine the optimal criteria at every stage of the algorithm. The process of selecting a criterion is undertaken from among three options. Various genetic operators are employed for the purposes of recombination and transformation, alongside the implementation of an intelligent mutation assignment operator. Wang [49] proposed the HGA-TS methodology which is a hybrid approach that utilizes a combination of the exploitation capabilities of Tabu Search (TS) and the exploration capabilities of Genetic Algorithm (GA) to address the flowshop manfacturing assignment that has a requirement of predefined times of the order. The approach employs a succinct encoding technique for denoting a pair of sub-problems, efficient genetic operators for executing selection, crossover, and mutation operations, and novel neighborhood structures and diversification functions to enhance its local search capability. Empirical investigations conducted on datasets derived from established benchmark instances indicate that HGA-TS is capable of achieving favorable outcomes within brief iterations. Ding [14] suggests a hybrid HLO-PSO algorithm that solves FJSP by combining different variations of the presented enhanced PSO and suggested scheduling schemes. In addition to being simple to implement and embed in other environment and scenarios, it supports individual learning capacity and enhances search capability. Most single-objective FJSSP scenarios can be solved successfully using it.

Numerous studies have started to concentrate on using reinforcement learning and neural networks to solve MILPs as artificial intelligence develops, and these methods have caught the attention of both academia and industry due to their superior generalization. Baer [4] proposed a distributed method with multiple agents which controlling products. This approach offers greater flexibility and provides complete system information to each agent, resulting in a centralized agent architecture. Gaining a comprehensive understanding of the entire malleable processing system is essential to make informed decisions regarding resource allocation. This paper introduces an efficient scheduling agent that selects an appropriate service from a list of candidate services for each incoming task. The approach utilizes a deep reinforcement learning framework, specifically employing the deep Q-learning method, to address the Dynamic Scheduling and Service Matching (DSSM) problem at hand. Zhang [55] proposed automatically learning PDRs to incorporate the states encountered during the solving process. They take advantage of the distributed graph representation and proposed an improved method based on Graph Neural Networks. The resulting policy network is size-independent, which makes it possible to generalize on large-scale cases. The agent performs well when tested against the top PDRs currently in existence, according to experiments, and can train high-quality PDRs from scratch using simple raw features. On considerably larger situations not encoun-

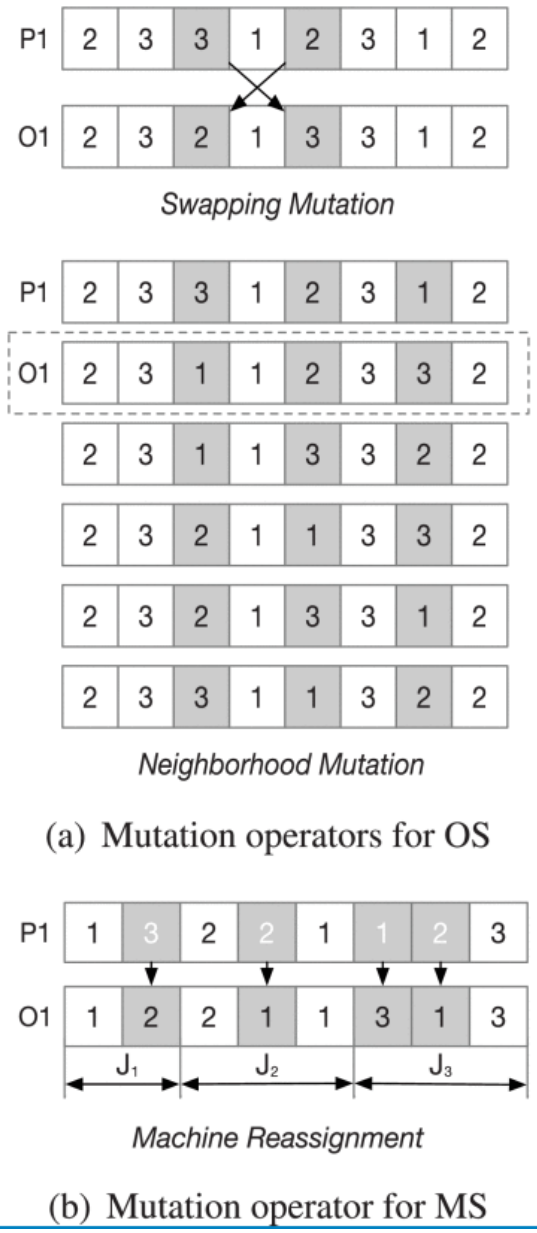


Figure 2.2. *Mutation Operators [49]*

tered during training, the learnt policies also work well. Park [33] proposed a framework for learning to make schedule for FJSSP using reinforcement learning (RL) and a graph neural network (GNN). To take into account the structure of FJSSP, it is conceptualized as a subsequent decision-making challenge, depicted using a graph-based representation of the state. A suggested framework uses a GNN to discover how node features are utilized to encapsulate and incorporate the dimensional architecture within the graph representation. of the FJSSP and to generate the best scheduling policy, which converts the ingrained node features into the most effective scheduling action. The modules are trained end-to-end using an RL technique based on Proximal Policy Optimization (PPO). Hu [19] proposed a DRL method. The proposed Automated Guided Vehicles true-time planning structure has two layers: the jobs assignment layer and the DQN-based agent

scheduling layer. In the first layer, true-time production system data is collected from Internet of Things (IoT) devices. This data is transmitted to the System State module, which extracts and sends relevant state information to the Q-network module for training and learning. The Q-network module processes the data and generates learned information. The learned information is then forwarded to the Combined Action module, which computes the appropriate dispatching rule and selects the optimal AGV for the task. This information is then communicated to the first layer as a command to show solution to the AGVs of their tasks. This approach enables efficient ture-time scheduling of AGVs in the dynamic production processing environment by utilizing the power of Deep Q-Network (DQN) based reinforcement learning techniques.

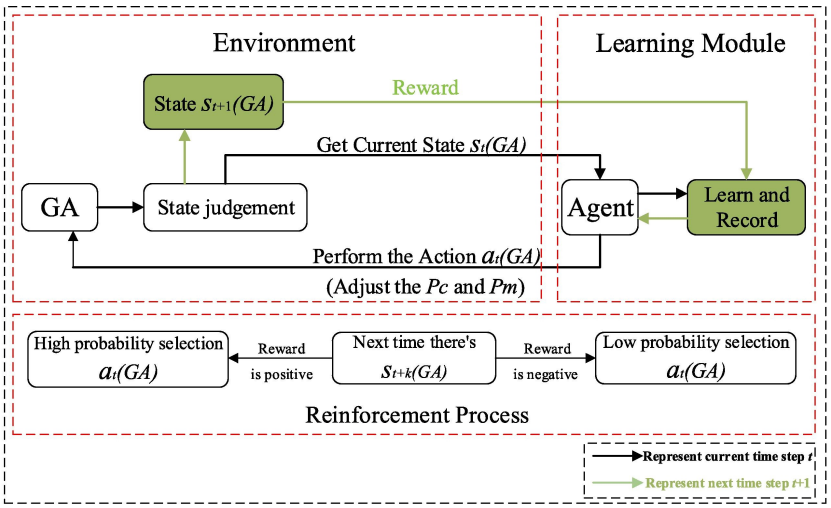


Figure 2.3. *Linked GA and RL structure [7]*

2.2.2 Multi-objective Optimization

The definition is given below [62]:

$$\min f(X) = \min[f_1(X), f_2(X), ..., f_m(X)]$$
$$X = (x_1, x_2, ..., x_n) \in R^n$$

The function $f_i(X)$ represents the i^{th} one of objective functions, where X represents a solution vector and R^n denotes a decision variable space. A solution X^* is considered Pareto optimal if there is not an existing any solution $X \in R^n$ that dominates X^*. There are several methods to find a Pareto optimal solution: Weighted global criterion method, Weighted sum method, Weighted min-max method, etc [28].

This section will focus on reviewing solutions of scheduling optimization problem with

multiple objectives. Moslehi [30] study introduces a novel approach that combines the particle swarm and local search algorithms to address the flexible manufacturing assignment scheduling problem with multi objective. Zhu [62]investigates the challenge of production assignment scheduling problem in a flexible environment, where job precedence constraints exit (FJSSP-JPC). The proposed approach considers ranked job priority constraints specified by Bills-of-Materials (BOMs) and presents an evolutionary multi-objective grey wolf optimizer that is effective in minimizing the a few significantly objectives in the context of the production processing concurrently. The approach incorporates the enhanced social ranking and a differing upper scheme to separately augment the convergence time and population diversification. Li [24] introduces a novel approach, namely the Elitist Nondominated Sorting Hybrid Algorithm (ENSHA), to address the production assignment scheduling problem in a flexible environment with multi objective in the context of sequence-dependent composition times/costs. The primary focal points of the manuscript include the operation-based sequence model, tasks allocation strategies that are dependent on the problem. The utilization of the arrangement model represents a novel approach in MOFJSSPs as it marks the initial instance wherein the TSC has been regarded as a distinct objective. Zhang [59] proposed a distributed ant colony system as a potential solution to address the MOFJSSP and investigate the Pareto front. The solution contains a PheromoneMap, which is shared among ants from different colonies, has been designed to influence their pathfinding behavior. Figure 2.4 presents the exploration directions of ant colonies.

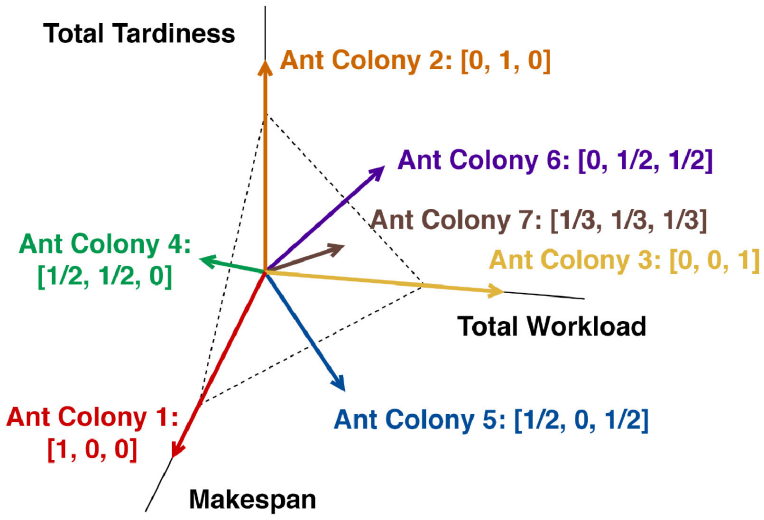


Figure 2.4. *Exploration directions of ant colonies [59]*

Chen [6] investigated the the subject of cost-effective decentralized no-idle recombination flow-shop scheduling and proposed a collaborative optimization algorithm (COA) consisting of two heuristics, multiple search operators, local intensification strategies, and speed adjusting strategies for minimizing makespan and the sum of energy usage. Wang [48] introduces a multi-objective whale swarm algorithm (MOWSA) as a potential solution

for the energy saving dynamic project scheduling problem (DPFSP). A novel problem-specific local search technique has been devised to enhance the exploitation potential of MOWSA. Furthermore, a mechanism for improving development is introduced to improve energy generating speed while maintaining productivity. The Figure 2.5 shows the PMX crossover operator in this solution.

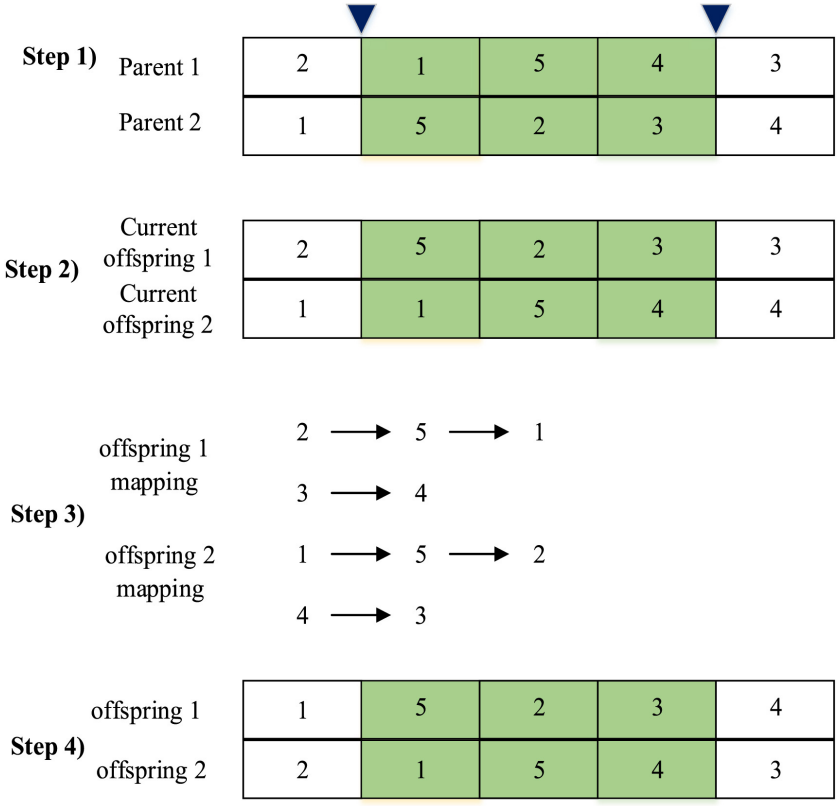


Figure 2.5. *PMX crossover operator [48]*

In order to tackle a multiple goals mixed manufacturing assignment scheduling problem, Sven [39] combines three strategies into one model. To identify a three-dimensional Pareto front for the following objectives — makespan, entirety of energy expenses, and peak usage — a novel continual local search technique has been created.

Heuristic methods are efficient and most research in MOFJSSP focus on it, however, exact algorithms can still find the optimal answer for multi-objective optimization. A new parallel branch and bound algorithm is introduced in this work [41]. Its upper bound is initialised using the NSGA-II algorithm. It uses a coexisting order list to save and distribute outstanding sub-problems and is built for shared memory architectures. Additionally, the solution domain is shown in a structured format.

2.2.3 Dynamic Optimization

In the context of actual production systems, the environment is characterised by a high degree of dynamism, as evidenced by unforeseeable occurrences such as machinery breakdowns and worker absenteeism. Consequently, there arises a pressing requirement for dynamic scheduling which is a prominent area of focus in the field of production scheduling research [29].

According to reference [45], the dynamic events have been categorised into four distinct classes:

1. Workpiece related events, which can be characterised by uncertain processing times, random arrival of workpieces, changing delivery dates, and dynamic shifts in priority and order.
2. The occurrence of machine-related events, which have significant impacts on production processes. For instance, machine damage can impose limitations on the availability of load, while conflicts may arise between productivity and the actual utilisation.
3. Instances about the procedure, which delay in the procedure, rejection of quality and instability in production.
4. Other potential events that may impact operations include the unavailability of personnel, delayed arrival of raw materials, and defects in raw materials.

The common solution to Dynamic JSSP (DJSSP) are Heuristic Method and Reinforement Learning. Zhang [56] outlines a newly developed framework, which utilises a two-stage GPHH approach including feature selection, which facilitate the automatic evolution of scheduling heuristics specifically for DFJSSP. The framework is devised to develop heuristics based solely on picked features.. Concurrently, proposed are particular variation approaches that leverage the situation of both the picked features and the considered instances. Zhang [57] introduces a new approach that utilises surrogate assistance in an evolutionary multitask algorithm, with the aim of enhancing both the efficiency and efficacy of training. Phenotypic characteriszation is employed to assess the scheduling rule behaviours and construct surrogates for individual tasks, thereby enhancing efficiency and knowledge transfer among a substantial pool of potential candidates. Figure 2.6 show the surrogate-assisted multitask respecting the technique for transferring surrogate and knowledge transfer.

Heuristic Method is more time consuming, and reinforcement learning methods can quickly select the best scheduling solution at each rescheduling point. The present study [26] has developed a deep Q-network (DQN) as a solution to tackle an aforementioned issue. The proposal suggests a few synthesized scheduling rules that determine a suitable operation and assign it to an accessible machine in instantaneously., upon the finish of the

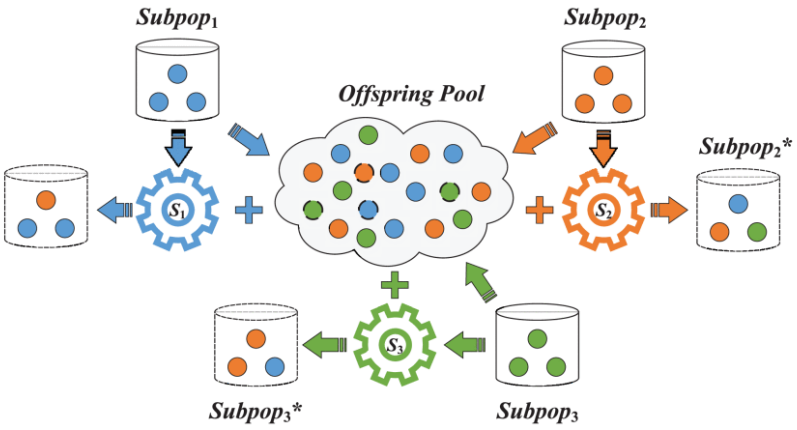


Figure 2.6. *Surrogate-assisted multitask [57]*

operation or arrival of a novel job. The production status when rescheduling occurs is represented by extracting seven generic state features. The DQN is subjected to training through the utilisation of deep Q-learning (DQL), which has been augmented based on enhancements, namely double DQN and soft target weight improvement. The utilisation of a "softmax" action selection policy serves to elevate rules by greater Q-values when simultaneously preserving the policy entropy. Chen [8] introduces an innovative framework based on deep reinforcement learning aimed at addressing the typical JSSP. The JSSP is modelled through the utilisation of the attention module and distributed graph embedding technique, while a transformer architecture that has been enhanced by a multi head attention procedure is employed. The system exhibits proficiency in acquiring knowledge of distant relationships and resolving extensive scheduling predicaments.

2.3 Route Planning

Within the realm of intelligent manufacturing, the transportation of products and raw materials is increasingly dependent on AGV or intelligent transportation systems. Intelligent transportation systems can significantly reduce labor and time costs, and can simultaneously assure the stability of transportation. Path planning is one of the software foundations of transportation systems, and efficient planning algorithms can greatly improve the efficiency of transportation systems.

Route planning problem can be defined below: there exits n transporters on the transport system, and they have their respective starting positions S_n, and a defined end positions G_n. In the basic scenario, the transporters need to reach their predefined goals within the shortest time. In some variants, the transporters need to reach the specified positions within a specified time period or need to consider carrying passengers.

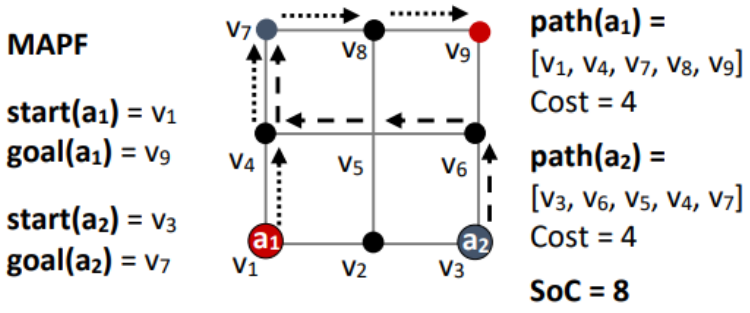


Figure 2.7. *A multi-agent path finding (MAPF) instance with two agents [46]*

However, all path planning problems are concerned with one important constraint: all transporters must not have collisions. There are four common types of collision, showing in the Figure 2.8.

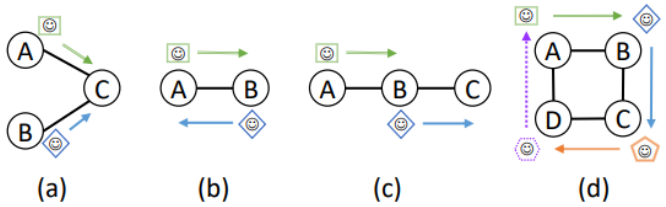


Figure 2.8. *4 possible collision or crash scenarios [42]: (a) Collision in the vertical direction, (b) Collision when switching positions, (c) a following struggle, and (d) Dead Loop.*

Therefore, this section review focuses on various algorithms that allow transporters to attain the desired locations with the fastest speed without facing conflicts. Path planning algorithms can be categorized into two main approaches: centralized methods and distributed methods, which depend on whether individual agents make independent decisions. A centralized approach means considering all transmitters simultaneously and planning routes for them as a whole. In the context of distribution, individual transporters are assigned their own optimisation subproblems, which they independently solve to plan their respective routes and prevent potential conflicts with other transporters.

The selection of solution technique and the efficiency of solution can be significantly influenced by the optimisation modelling. Hence, this section will provide an overview of various prevalent target formalisms prior to delving into particular algorithms.

i. Constraint Satisfaction Problem(CSP) CSP is comprised of three principal components: variables, domains, and constraints. In route planning problem, variables are the positions or states of transporters. Domains are available position set for transporters at each time step. Constraints represent what is allowed or prohibited.

ii. Boolean Satisfiability (SAT) Problem The utilisation of the SAT problem is a viable approach for modelling and resolving specific facets of route planning. The solutions are found by determining if assigned truth values can satisfy given Boolean formula (encoded original problem).

iii. Answer Set Programming (ASP) It is a programming paradigm that is declarative in nature and enables the representation and resolution of problems related to combinatorial optimisation. The application of this method is especially advantageous in addressing issues that entail discrete decision variables and intricate constraints. ASP programmes are formulated using rules and facts, and the resulting solutions, which are referred to as answer sets, are determined by the logical coherence of these rules and facts.

2.3.1 Centralized Approach

The centralised approach involves a central entity, such as a computer, that possesses comprehensive information and devises planning for all transporters or agents. The central entity assumes the duty of communicating with all agents, gathering data and disseminating directives.

The path planning model is an integer or mixed integer model, which allows for centralised resolution through employment of exact solution methods. Divya [1] suggests a lagrangian relaxation-based optimisation algorithm as a solution to the route planning problem with time windows. The issue at hand involves the incorporation of optimal Lagrangian multipliers, with the utilisation of the subgradient method to determine the suitable multipliers. The algorithm has undergone implementation and tested on some problems, and the outcome shows the proposed approach is not only competitive but also superior to the optimal solution.The path planning model utilised is an integer model, also known as a mixed integer model, which allows for centralised resolution through the use of exact solution methods. Van [12] introduces a proficient algorithm designed for the purpose of path planning for multiple robots. The approach involves breaking down the problem into a few small sub-problems that is possible to be solved in a sequential manner, with a focus on minimising the dimension of the subproblem with the highest dimensionality. The optimal sequence is derived by the algorithm through planning solely within configuration spaces that possess a dimension that is either less than or equal to the aforementioned minimum. The algorithm is comprehensive and universally applicable to robots of varying types and degrees of freedom.

In addition to exact algorithms, heuristic algorithms are also largely applied to solve of path planning. Faiza [17] presents the Aquila Optimisation Algorithm, which has been recently developed and tailored to perform Multi-Robot space exploration, which is a novel amalgamation. Jose [20] devises heuristic techniques for the allocation of tasks and the planning of collision-free paths for a trio of robots operating within a shared workspace.

The task allocation was performed using a genetic algorithm (GA), while path planning was accomplished through the utilisation of an A* algorithm. The present study's findings were superior to the preceding research conducted by Liu and Kroll (2012). The reason for this outcome can be attributed to the utilisation of Genetic Algorithm (GA) for task scheduling and allocation of an optimal number of tasks to each robot. This approach encoded a greater number of environmental conditions in the GA-string.

Chen advocates [9] for a comprehensive strategy that merges the precomputation of optimal potential routes with online routes retrieval and dynamic improvement, in order to develop a dynamic navigation strategy within a centralised structure. A heuristic approach is employed to increment link weights and construct a partially disjoint set of candidate paths prior to the trip. The algorithm under consideration exhibits several noteworthy features. Firstly, it demonstrates an action time for navigation requests that is almost linear with the size of the network and is not significantly influenced by system load. Secondly, it enhances the accuracy of pretrip route planning by incorporating travel time reliability. Lastly, the algorithm allows for system optimisation while still accommodating driver preferences.

Li [23] introduces a motion planning approach for multi-AGV systems that is centralised in nature. This method is characterised by high computational requirements, but exhibits a heightened responsiveness to the quality of solutions obtained. The scenario is cast as an optimal control problem, wherein the dynamics of the AGV, mechanical limitations, and external constraints are described using differential algebraic equations. The study involved conducting comprehensive simulations on tasks related to the reconfiguration of 10-AGV formations. The results of the simulations indicate that the centralised planner introduced in the study has the potential to be validated, unified, and implemented in real-time.

2.3.2 Distributed Approach

Compared to the centralized algorithm, each transporter makes an independent path planning in the distributed solution. The computational cost of the centralized algorithm is large, while the communication cost between the central entity and transporters increases as the number of transporters increases. A typical strategy for addressing a multi-agent route finding problem in a decoupled manner involves three distinct stages: (i) determining the plans for each individual agent, (ii) establishing the sequence or priority of agents whose plans will be restructured to achieve a conflict-free solution, and (iii) restructuring the particular plans in accordance with an established sequence from the previous stage [10].

Certain exact algorithms that facilitate the decomposition of the optimization problem exhibit high proficiency in assigning the primary problem to individual transporters. The

present study puts forth a technique for planning routes in a distributed manner for multiple automated guided vehicles (AGVs). The approach presented by [31] exhibits a distinctive attribute whereby Automated Guided Vehicle (AGV) autonomously generates a routeing plan that is in close proximity to the optimal solution. This is achieved through iterative communication between the distributed agents and local computation for each individual agent. The proposed method's solution optimality is assessed through a comparison with an optimal solution.

Another common approach in distributed algorithms is reinforcement learning, which allows training strategies directly on a individual transporter. Sartoretti [38] proposes a framework named PRIMAL which incorporates reinforcement learning and imitation learning techniques to facilitate the acquisition of fully decentralised policies. It presents an exposition of proficient MAPF planning techniques, including the utilisation of expert demonstrations during the training process, reward shaping, and environment sampling. After acquiring knowledge of the policy, it can be replicated across multiple agents and adapted to varying team sizes and world dimensions.

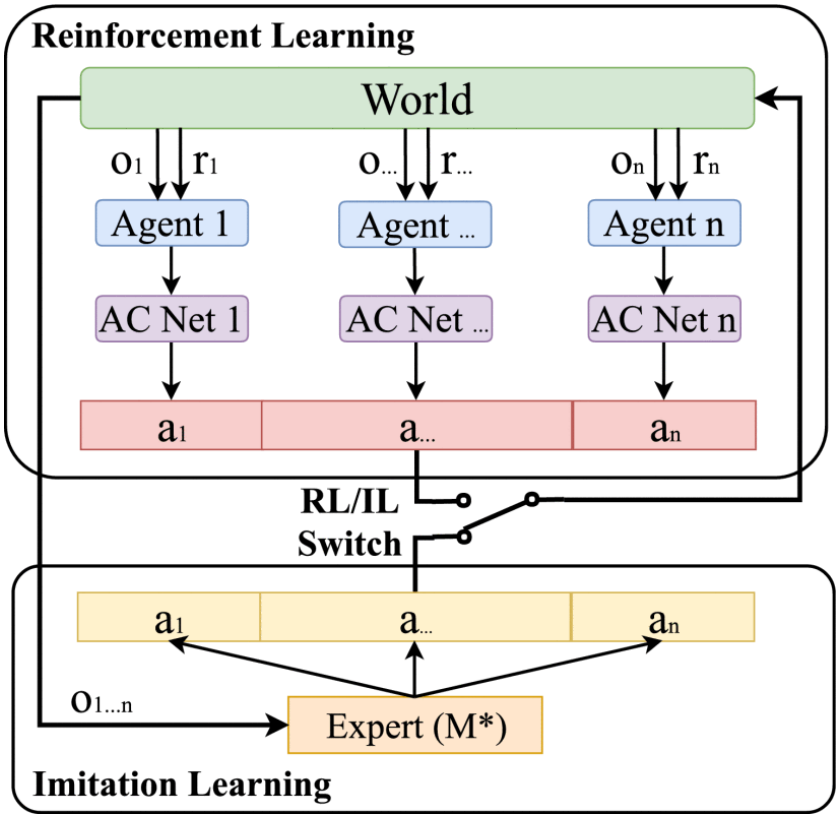


Figure 2.9. *PRIMAL framework [38]*

Zolfpour-Arokhlo [63] introduces a novel approach for devising a multi-agents path finding problem that relies on multi-agent reinforcement learning techniques. They presented a model that employs QVDP and Boltzmann distribution techniques to generate a prioritised route plan for vehicles, taking into account various factors such as climate, traffic statistics,

security, and power capacity. Luviano [27] introduces a novel approach for planning multi-agent systems in continuous time, which involves the utilisation of a fuzzy Q-iteration algorithm and an improved Wolf-PH algorithm. The algorithm has been implemented in the context of a collaborative mission involving a pair of mobile Khepera robots, and the empirical findings indicate its efficacy.

In the context of distributed algorithms, the occurrence of deadlock behaviour can be attributed to the conflicting objectives pursued by individual transmitters, each of which operates based on its own planning and optimisation strategies. Solving the deadlock problem is one of the problems that must be solved in practical applications of distributed approaches. A methodology is introduced by [13] for constructing a grid-oriented Multi-Agent Path Finding (MAPF) scenario, which is commonly demanded by contemporary MAPF solving algorithms.

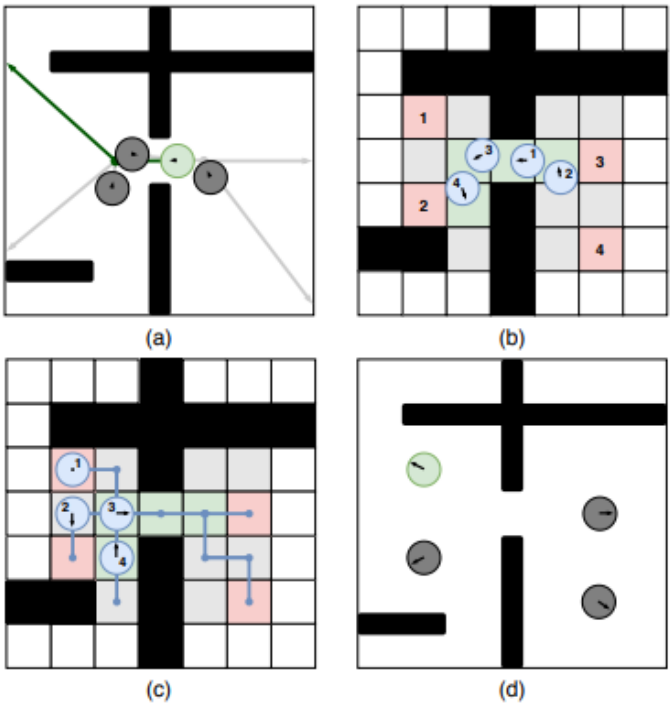


Figure 2.10. *An instance of solving the deadblock. [13]*

2.4 Bilevel Optimization

In the previous sections, studies on task assignment and path planning problems in smart manufacturing were reviewed. In practical applications, they are often linked together to form a bilevel problem. This is because in an intelligent manufacturing system, task assignment requires consideration of transportation time, which in turn depends on task assignment. Hence, this section strives to offer a all-inclusive overview of the current body of literature pertaining to bilevel optimization.

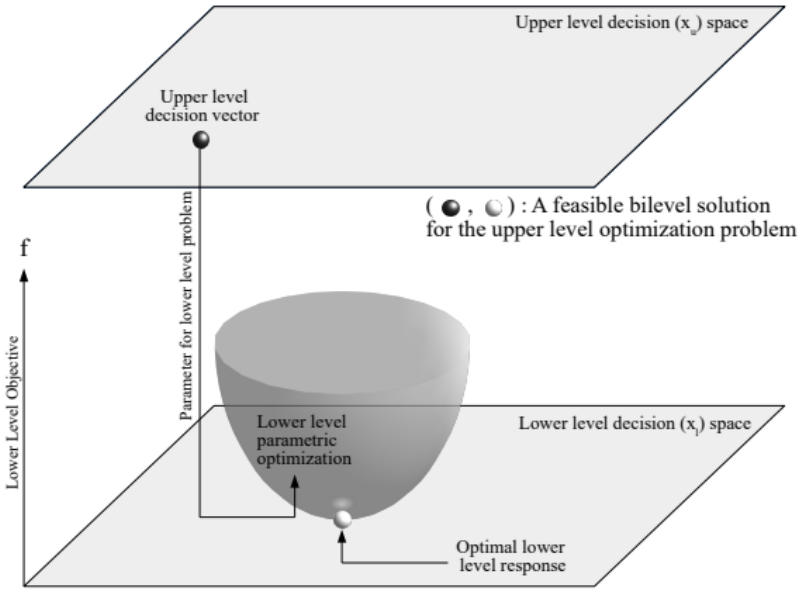


Figure 2.11. *An illustration of a typical bilevel optimization problem. [40]*

A typical bilevel optimisation problem comprises a leader level optimisation problem and a follower level optimisation problem that seeks to be linked [40]. If the bilevel optimization problem demonstrates convexity, it is apparent that there are a few techniques which can be used to integrate two optimization problems to one problem. The primary objective is to identify the circumstances in which the problem at the follower level achieves optimality and to utilize this as a constraint on the problem at the leader level.

In integer programming or mixed integer programming problems, the lower level problems exhibit discontinuity, thereby rendering the KKT conditions inapplicable. When nested approach is used, the problem size and solution difficulty become significant. Liu [25] introduces a novel algorithmic framework, aimed at mitigating the challenges posed by the non-convex follower structure in Bi-Level Optimisation (BLO). The proposed method incorporates an auxiliary initialisation strategy to facilitate the optimisation dynamics and devises a conservative trajectory truncation mechanism to construct a dependable approximation of the original BLO when the LLC hypothesis is absent.

Stouraitis [43] suggests a proficient bilevel paradigm that merges network discovery techniques with motion optimisation. This approach empowers intelligent individuals to adjust their strategies in real-time, in response to modifications in the dyadic task. This system represents a novel approach to enhancing agent capabilities by enabling them to engage in online planning within hybrid spaces. Specifically, the system optimises multi parameters, thereby providing agents with a comprehensive suite of tools to enhance their performance. In the context of co-manipulation of large objects, wherein there is a need for frequent changes in grasp-holds and plan adaptation, this aspect assumes significant importance.

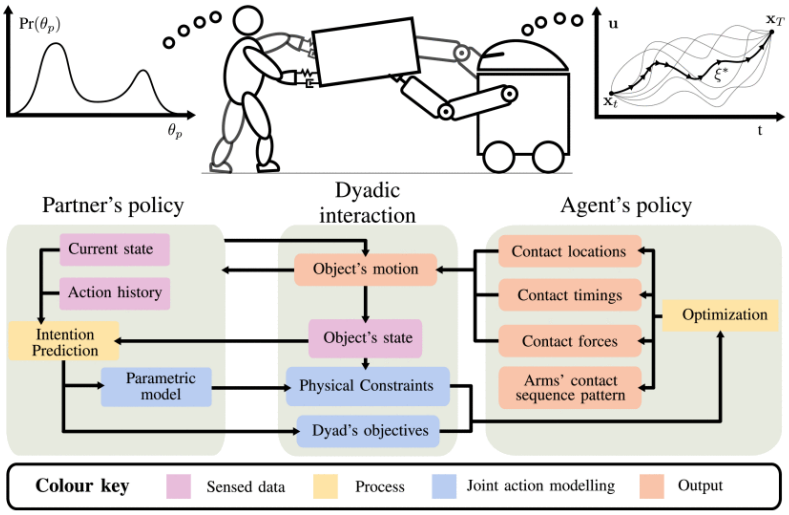


Figure 2.12. *An illustration of a human-machine cooperative transport system. [43]*

2.5 Summary

This section provides a comprehensive review of a substantial body of literature pertaining to the Job Shop Scheduling Problem and Route Planning Problem. The examination of solutions to these optimization problems reveals that the identification of an optimal approach is challenging. Each approach exhibits distinct merits and demerits, necessitating the consideration of trade-offs depending on particular scenarios.

3. DESIGN OF THE ALGORITHM TO SOLVE THE BILEVEL OPTIMIZATION PROBLEM

3.1 A Scenario Statement for Smart Manufacturing Systems

The FAST-Lab at the Tampere University has a complete experimental environment for simulating engineering production, including multiple industrial robots and pallet transfer systems. Presently, the FAST-Lab is in the process of enhancing its pallet transfer system by integrating a novel version, thus establishing an intelligent manufacturing configuration in conjunction with the existing industrial robots.

The novel pallet system consists of 120 diminutive blocks that leverage magnetic levitation, thereby enabling the pallet's movement in a state devoid of physical contact. The industrial robots undertake the execution of manufacturing tasks, while the conveyance of products amid these robotic entities is effectively facilitated by the integrated pallet mechanism.

In a conventional manufacturing context, a prevalent scenario encompasses the fabrication of a set of cellphones, comprising three discrete constituents: namely, the screen, the frame, and the keypad. The manufacturing system has to assemble the three parts, each of which has its own color. The extent of operations achievable by an industrial robot is contingent upon its specifications and model attributes. The pallet, in this context, operates as a conduit for conveying unfinished components across multiple robotic stations, culminating in the transportation of the finalized product to its predetermined point of reception.

There are two actions for a certain robot: 1. Draw product. 2. Change pen. The time required for a pen change is around 40 seconds. However, the time it takes to draw a product varies and is dependent on the specific product being drawn. This drawing time can range from approximately 30 to 60 seconds, depending on the particular product being processed.

Each pallet can perform nine operations on the blocks: move horizontally to the right, move horizontally to the left, move vertically up, move vertically down, move up along the diagonal left, move up along the diagonal right, move down along the diagonal left, move

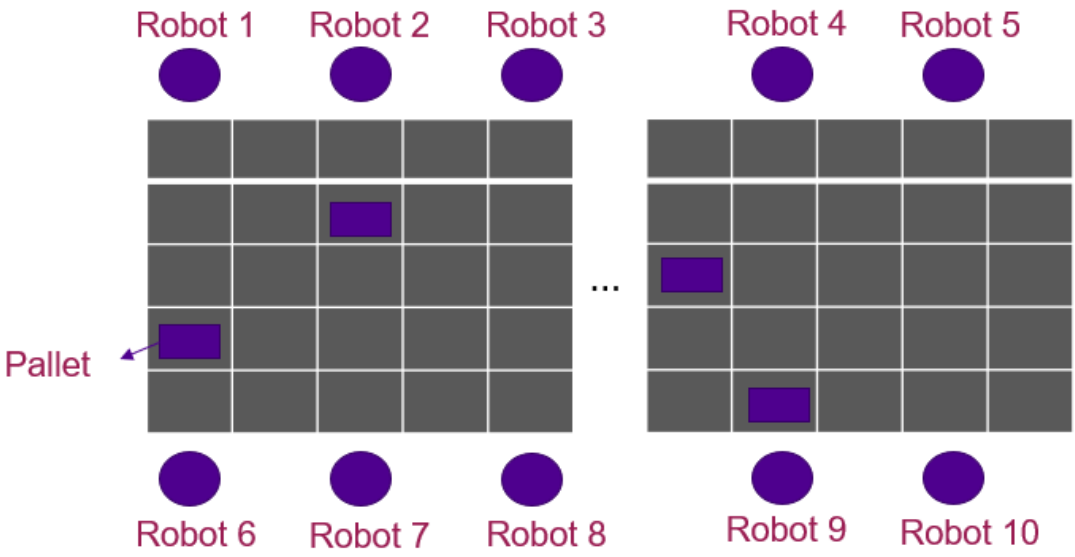


Figure 3.1. *The illustration of the pallet system.*

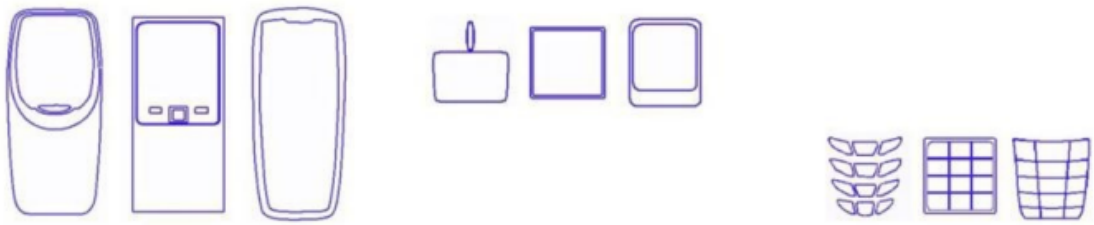


Figure 3.2. *The cell phone screens, frames, and keypads. [15]*

down along the diagonal right, and stay.

The intelligent manufacturing system is required to manufacture products in accordance with the user's specifications and has the capacity to process approximately 100 units on a daily basis. The processing time of a system is significantly affected by various scheduling algorithms due to the vast array of products involved. The objective of this portion is to illustrate the structure and functionality of an effective scheduling system.

3.2 Problem Formulation

This section outlines the process of translating a given scenario into a mathematical model. The initial phase of integrating optimization methodologies encompasses the creation of a mathematical framework that accurately portrays the provided scenario. The act of formulating this model holds substantial significance, given that an apt approach to modeling can augment the algorithm's efficacy, whereas an inadequately constructed model can significantly amplify the complexity of deriving a solution. This section will

elaborate on the strategies employed for both the process of model formulation and the subsequent optimization thereof.

This indeed constitutes a bilevel optimization quandary comprising a pair of distinct optimization problems:

1. Upper Problem: Job Shop Scheduling Problem: This optimization problem investigates how to assign the tasks to be produced to the industrial robots in the system to achieve goals while satisfying the constraints. When attempting to get the optimal schedule, it is essential to consider the duration of the transporting process for the pallet transporting item. The duration of transporting is derived from the route planning problem, thus the solution to it serves as a constraint on the variables.
2. Lower Problem: Route Planning Problem: The lower problem aims to plan the transportation routes for each pallet. The input of the lower problem is the schedule obtained from the upper problem, and the output is the shortest path planning without collision. The lower problem will generate a plan by considering the initial and final locations of individual pallets at various time steps.

The primary goal of an intelligent manufacturing system resides in attaining the utmost efficiency in production for a specified array of tasks. Correspondingly, the aim of the presented bilevel optimization quandary is to minimize the makespan. An integral constituent of the system's makespan is attributed to the pallet transportation time, which is ascertained via computations from the lower-tier problem. Stated differently, the acquisition of the task assignment schedule necessitates precise knowledge of the transportation time, while conversely, the precise transportation time hinges on awareness of the schedule. The subsequent segment will expound upon the approach employed for both the formulation and resolution of the aforementioned optimization predicament.

Regarding the previously described scenario, there are specific strategies that can contribute to achieving a resolution, including the establishment of an extensive set of predefined rules. Nevertheless, attaining an optimal or nearly optimal solution within this complex context presents a challenge. The utilization of optimization methods emerges as the most suitable approach to attain a scheduling solution that is suitably efficient.

3.3 Solution Design

A trade-off emerges between expeditiously solving a bilevel optimization predicament and attaining an optimal solution. Achieving the optimal solution frequently demands a notable computational expenditure. For instance, employing precise algorithms such as the Lagrangian Relaxation method mandates iterative resolution of the subproblem multiple times to ascertain the lower bound. Conversely, techniques proficient in efficiently tackling optimization quandaries do not assure the production of optimal solutions. The solution

posited by this thesis endeavors to secure an optimal or closely optimal resolution within a reasonable temporal investment.

3.3.1 Overview

The bilevel optimisation problem comprises an upper and a lower problem, both of which can be represented as mixed integer programming models. As indicated by the preceding review of pertinent literature, the resolution of a bilevel optimization predicament conventionally requires adopting a nested strategy that transforms the lower-tier problem into an upper-tier problem constraint. This adaptation engenders a notable proliferation of the exploratory domain. The envisaged methodology is grounded in the divide-and-conquer principle, wherein the two optimization problems will be independently formulated and resolved. Ultimately, an intrinsic interrelation between the two optimization problems becomes evident. The variables indispensable for the upper-tier problem are prognosticated by leveraging the optimal conditions derived from the lower-tier problem.

If the two optimisation problems are solved independently, the ensuing considerations will appear:

1. The value assigned to the transport time variable within the upper-tier problem can come into discord with the corresponding value in the lower-tier problem, leading to instances where the anticipated collision-free transportation duration between pallets and their intended destination may not always be achievable.
2. If a substantial margin is allotted to the optimal transport time, the upper and lower problems will not exhibit conflicting outcomes. However, this will result in a significant increase in the makespan.

To achieve a satisfactory solution within a reasonable time, it is imperative to ensure that the estimated transport time closely approximates the actual solution.

The subsequent sections are organized as: 1. Present the model and solution of the upper problem (Job Shop Scheduling Problem) and lower problem (Path Finding Problem). 2. Present an approach to integrate two problems.

3.3.2 Processing Task Allocation

Model Job Shop Scheduling Problem

There are N industrial robots in use, ($0 \leq N \leq 10$), and there are M products to be processed, each product has T tasks to be completed.

Initially, it is important to establish the objective function for the processing task allocation scenario. The first aim is to minimize the makespan, where the scheduling scheme of

production is expected to complete the production of all products in the shortest possible time. Assuming that the time for robot k to finish all the tasks assigned to it is C_k, then makespan can be expressed as the maximum value in C_k.

$$Makespan = Max\left\{C_k\right\}, \forall k = 1, 2, ..., N$$

In addition to minimizing makespan, achieving a balanced workload distribution among the robots is also significant. This is due to the fact that if one robot is overloaded for an extended length of time, it will undergo maintenance earlier than the other robots and will be more likely to sustain damage. The second objective can be then formulated as the following objective function.

$$WB = \frac{1}{M}\sum_{k=1}^{M}\left|W_k - \bar{W}\right|, \forall k = 1, 2, ..., N$$

W_k stands for the workload of robot k in the formula, and $\bar{W}$ is the average of the total sum of all robot workloads. Minimizing the workload variance (WB) serves to equalize the distribution of tasks across all robots.

After presenting objective functions to be optimized, three decision variables are introduced: x_{ij}^k, y_{ijgh}^k, S_{ij}, $\forall k = 1, 2, ..., M$, $\forall i, g = 1, 2, ..., P$, $\forall j, h = 1, 2, ..., T$. In the optimisation problem, the decision variables define the search space, and in the scenario discussed in this work, the decision variables are required to be able represent a schedule.

x_{ij}^k and y_{ijgh}^k are integer variables and S_{ij} is a continuous variable. x_{ij}^k is used to indicate whether task j of product i is scheduled to robot k. y_{ijgh}^k is employed to denote the sequential arrangement of tasks. If the task j of product i and the task g of product h both perform on the robot k and the task j of product i performs before the task g of product h, y_{ijgh}^k is equal to 1. S_{ij} indicates the start time of the task j of product i.

Following the deliberation on the objective function and the determination of decision variables, the ensuing exposition will elucidate the constraints inherent to this Job Shop Scheduling Problem. These constraints delineate and confine the scope of exploration in accordance with the attributes of the intelligent manufacturing system.

There are some general considerations of JSSP.

1. Each task (task i of product j) can be assigned to only one robot.

$$\sum_{i=1}^{N} x_{ij}^k = 1, \qquad \forall k = 1, 2, ..., N \quad j = 1, 2, ..., T \tag{1}$$

2. The start time of each task (task i of product j) must be greater than or equal to 0.

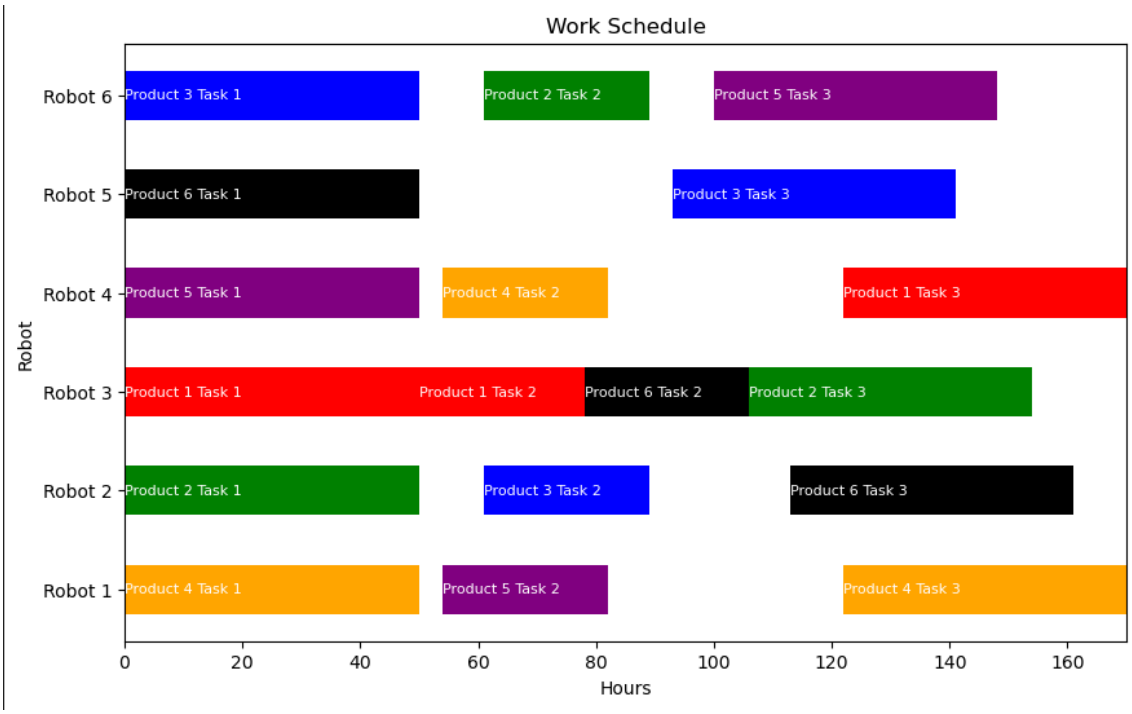


Figure 3.3. *An example solution of the JSSP.*

$$S_{ij} \geq 0, \qquad \forall i = 1, 2, ..., N, \quad j = 1, 2, ..., T \quad k = 1, 2, ..., M \tag{2}$$

3. It is necessary to complete the entire task for each product. T is the number of tasks per product, and in this scenario, T is generally equal to 3.

$$\sum_{k}^{R} \sum_{j}^{T} x_{ij}^{k} = T, \qquad \forall i = 1, 2, ..., M \tag{3}$$

In addition to these common constraints of JSSP, constraints specific to this system must be considered.

1. A fixed sequence governs the arrangement of diverse tasks within each product. As depicted in Figure 3.3, a temporal gap exists between the initiation of task 1 for product i and the commencement of task 2 for the same product. This interval necessitates a minimum duration exceeding the execution time of task 1. Moreover, the prospective transportation time warrants consideration. If sequential tasks of the identical product are designated to a common robot, the transportation time is rendered as 0. Conversely, when successive tasks for the same product are assigned to distinct robots, the temporal span between the initiation times of the two tasks is either equivalent to or greater than the combined duration of the processing

time and the transportation time..

$$S_{i(j+1)} - (S_{ij} + E_{ij} + R_{ij}) \geq 0, \qquad \forall i = 1, 2, ..., N \quad j = 1, 2, ..., T-1 \quad (4)$$

In the above formula, E_{ij} represents the estimated processing time of the task j of the product i and it is a predefined constant variable, R_{ij} represents the transport time of the task j of the product i. R_{ij} equals 0 when i is equal to j.

2. There exits an order between tasks that are assigned to the same robot. The variable y^k_{ijgh} is initially defined to restrict the start time duration of two consecutive tasks on the same robot.

$$y^k_{ijgh} + y^k_{ghij} = x^k_{ij} x^k_{gh}, \qquad \forall i, g = 1, 2, ..., N \quad j, h = 1, 2, ..., T \quad k = 1, 2, ...M \quad (5)$$

In the given expression, if task j of product i and task h of product g are not executed on the same robot, the product of x^k_{ij} and x^k_{gh} will be zero for all k (representing all robots). Consequently, both y^k_{ijgh} and y^k_{ghij} will be equal to zero. This setup serves the purpose of excluding discussions about tasks that are not performed on the same robot.

However, when both task j of product i and task h of product g are executed by the same robot, the combined values of y^k_{ijgh} and y^k_{ghij} equate to 1, illustrating that one of these variables assumes a value of 0 while the other adopts a value of 1. The determination of which variable corresponds to 0 and which corresponds to 1 hinges on the selection that optimally minimizes the objective function. This constraint ensures the establishment of a distinct order for task execution on the shared robot, a sequence that can be succinctly conveyed through the utilization of the variables y^k_{ijgh}.

Upon attaining the definition of the binary variable y, the formulation of the temporal constraint governing the commencement times of tasks executed by the same robot can be succinctly articulated as follows:

$$\begin{aligned} & y^k_{ijgh}(s_{gh} - (s_{ij} + E_{ij} + W_{ij})) \geq 0 \\ & , \forall i, g = 1, 2, ..., N \quad j, h = 1, 2, ..., T \quad k = 1, 2, ...M \end{aligned} \quad (6)$$

W_{ij} represents the potential pen changing time, the formula for W_{ij} will be given later. When the colors required for adjacent tasks are the same, the value of W_{ij} is set to 0. However, if the required colors differ before and after, W_{ij} is assigned the time needed to replace the corresponding pen.

Constraint (5) and Constraint (6) establish the start time duration between individ-

Figure 3.4. *Adjacent Tasks.*

ual tasks within the proposed framework. Constraint (5) accounts for the potential transport time between adjacent tasks of the same product, while Constraint (6) addresses the possible replacement time required for the pen between adjacent tasks on the same robot. Together, these constraints encompass all scenarios pertaining to task order and determine the appropriate start time duration.

3. To compute the pen replacement time, as specified in Constraint (6), it is necessary to introduce the new variable. This variable is introduced to address the requirement of a binary variable that indicates whether two tasks are assigned to the same robot and are adjacent to each other. The introduction of the new variable is essential because the existing variable y^k_{ijgh} only indicates the order of task execution without explicitly identifying adjacency between tasks.

   The newly introduced variable, z^k_{ghij}, is a five-dimensional variable similar to y^k_{ghij}. When task j of product i and task h of product g are executed on the same robot, and task h of product g is the subsequent task following task j of product i, z^k_{ghij} is set to 1. The following constraint is the definition of z^k_{ghij}.

$$z^k_{ijgh} = min\left\{\sum_a^P\sum_b^T z^k_{abij} - \sum_a^P\sum_b^T z^k_{abgh} - 1, 1\right\} + 1,$$

$$\forall a, i, g = 1, 2, ..., P \quad b, j, h = 1, 2, ..., T \quad k = 1, 2, ...M \tag{7}$$

$$z^k_{ijgh} \leq y^k_{ijgh} \qquad \forall a, i, g = 1, 2, ..., P, \quad b, j, h = 1, 2, ..., T \quad k = 1, 2, ...M \tag{8}$$

   Within Constraint (7), the initial term in the minimum operation expression signifies the disparity in the count of tasks that precede each specific task. As delineated in the accompanying figure 3.4, when two tasks are contiguous, this disparity equates to 1. In all other instances, the disparity surpasses 1. Consequently, the variable z^k_{ijgh} is assigned a value of 0 when the disparity is 1, and 1 when the disparity exceeds 1. However, in actuality, z^k_{ijgh} should indeed be 1 when tasks are adjacent, and 0 otherwise. Hence, the inclusion of one in the formula effectively introduces a negation (NOT) operation, whereby z^k_{ijgh} becomes 1 when tasks are adjacent and 0 otherwise. There are two possible scenarios for a pair of adjacent tasks: either task j of product i precedes task h of product g, or task h of product g precedes task j of product i. To calculate the potential pen changing time accurately, the order of the adjacent tasks must be specified. Consequently, Constraint (8) mandates that

z_{ijgh}^{k} is equal to 1 when task j of product i is the last task of task h of product g.

In summary, the mathematical model and notations are shown below:

Objective Functions (MultiObjectives) :

$$Minimize \quad Makespan = Max\left\{C_k\right\}, \forall k = 1, 2, ..., M$$

$$Minimize \quad WB = \frac{1}{M}\sum_{k=1}^{M}\left|W_k - \bar{W}\right|, \forall k = 1, 2, ..., M$$

Subject to (constraints) :

$$\sum_{i=1}^{N} x_{ij}^{k} = 1, \qquad \forall i = 1, 2, ..., N \quad j = 1, 2, ..., T \tag{1}$$

$$S_{ij} \geq 0, \qquad \forall i = 1, 2, ..., N \quad j = 1, 2, ..., T \quad k = 1, 2, ..., M \tag{2}$$

$$\sum_{k}^{R}\sum_{j}^{T} x_{ij}^{k} = T, \qquad \forall i = 1, 2, ..., M \tag{3}$$

$$S_{i(j+1)} - (S_{ij} + E_{ij} + R_{ij}) \geq 0, \qquad \forall i = 1, 2, ..., N \quad j = 1, 2, ..., T-1 \tag{4}$$

$$y_{ijgh}^{k} + y_{ghij}^{k} = x_{ij}^{k} x_{gh}^{k}, \qquad \forall i, g = 1, 2, ..., N \quad j, h = 1, 2, ..., T \quad k = 1, 2, ...M \tag{5}$$

$$y_{ijgh}^{k}(s_{gh} - (s_{ij} + E_{ij} + W_{ij})) \geq 0,$$
$$\forall i, g = 1, 2, ..., N, \quad j, h = 1, 2, ..., T \quad k = 1, 2, ...M \tag{6}$$

$$z_{ijgh}^{k} = min\left\{\sum_{a}^{P}\sum_{b}^{T} z_{abij}^{k} - \sum_{a}^{P}\sum_{b}^{T} z_{abgh}^{k} - 1, 1\right\} + 1,$$
$$\forall a, i, g = 1, 2, ..., P \quad b, j, h = 1, 2, ..., T \quad k = 1, 2, ...M \tag{7}$$

$$z_{ijgh}^{k} \leq y_{ijgh}^{k} \qquad \forall i, g = 1, 2, ..., P \quad j, h = 1, 2, ..., T \quad k = 1, 2, ...M \tag{8}$$

$$x_{ij}^{k}, y_{ijgh}^{k}, z_{ijgh}^{k} \in \left\{0, 1\right\}, \qquad i = 1, 2, ..., N \quad g = 1, 2, ..., N \quad i \neq g, \quad h, j = 1, 2, ..., T \tag{9}$$

Notation of objective functions :

$Makespan$: $System\ makespan$

C_k : $Completion\ time\ of\ robot\ k$

WB : $Robots\ workload\ balance$

W_k : $Working\ time\ of\ robot\ k$

$\bar{W}$: $Average\ Working\ time\ of\ robot\ k$

N : $The\ total\ number\ of\ products$

T : $The\ total\ number\ of\ tasks\ of\ one\ product$

M : $The\ total\ number\ of\ manufacturing\ robots$

Notation of constraints :

t_{ij} : $task\ j\ of\ product\ i$

S_{ij} : $Start\ time\ of\ t_{ij}\ processed\ on\ robot\ k$

x_{ij}^k : $Equals\ 1\ if\ task\ T_{ij}\ is\ processed\ by\ robot\ k,\ 0\ otherwise$

y_{ijgh}^k : $Equals\ 1\ if\ t_{ij}\ is\ performed\ before\ t_{gh}\ on\ robot\ k,\ 0\ otherwise$

z_{ijgh}^k : $Equals\ 1\ if\ t_{ij}\ is\ performed\ before\ t_{gh}\ and\ t_{ij}\ is\ adjacent\ to\ t_{gh}\ on\ robot\ k$, $0\ otherwise$

E_{ij} : $Estimated\ processing\ time\ of\ task\ T_{ij}$

Notation of decision variables :

s_{ij} : $Strat\ time\ of\ task\ ij$

x_{ij}^k : $Equals\ 1\ if\ t_{ij}\ is\ performed\ on\ robot\ k,\ 0\ otherwise$

y_{ijgh}^k : $Equals\ 1\ if\ t_{ij}\ is\ performed\ before\ t_{gh}\ on\ robot\ k,\ 0\ otherwise$

Job Shop Scheduling Problem Solver

The model designed for the given situation results in a challenging Job Shop Scheduling Problem. This issue involves complex connections between constraints (multiple decision variables within a single constraint) and a significant quantity of variables. The ability to efficiently identify the optimization within a certain time period is of utmost importance when employing an algorithm.

To tackle this challenge, a solver has been created using the ortool library. This solver, known as CP-SAT, employs methods like linear relaxation and Large Neighborhood Search (LNS). To speed up the search, initial solutions are used as a starting point for further exploration.

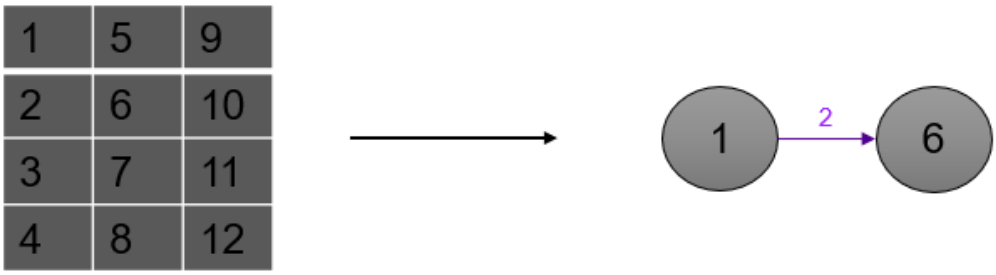


Figure 3.5. *An example movement from one node to another. The numbers indicated on the arrows represent the cost associated with moving from node 1 to node 6.*

3.3.3 Route Planning With Time Window

The primary aim of the route planning module is to facilitate the streamlined movement of individual pallets, each containing a distinct product, to their assigned destinations—these destinations align with the robots as indicated by the schedule generated from the job shop scheduling solver. This collaborative methodology fosters a unified and optimized operational flow, fostering efficient synchronization between task scheduling and pallet transport within the system.

Model Route Planning Problem

In the context of this optimization problem related to path-finding, the scenario involves K pallets and T time steps, with the availability of M magnetic levitation blocks to facilitate pallet movement.

To establish a mathematical model for path-finding, the initial step is to define the decision variables. Illustrated in the figure 3.5, each individual block step can be treated as a node, while the transporter navigates between these nodes.

The decision variables are defined as $x_{ijt}^k, i, j = 1, ..., M, t = 1, .., T, k = 1, ..., P$. The i and j encompass all nodes within the system, while t represents the time step and k denotes the specific pallet. When x_{ijt}^k is equal to 1, it signifies that pallet k is carried from node i to node j within the time interval t. When x_{ijt}^k is equal to 1, and i is equal to j, it signifies that pallet k's action at time t is to come to a halt at a particular node.

In this route planning problem, the primary objective is to formulate an optimal transportation strategy that ensures the prompt conveyance of each pallet to its assigned destination, while minimizing the total time taken for transportation.

The definition of variable y is:

$$\sum_{i \in N_j} x^k_{i,j,t} = 1 - y_{k,t} \quad j \in G_k, t = 1, .., T, k = 1, ..., P$$
$$- y_{k,t} + y_{k,t+1} \leq 0 \quad t \leq W_{kq} - 1, k = 1, ..., P, q = 1, ..., Q$$

The objective function is:

$$Minimize \quad \sum_k \sum_t y_{k,t} \quad i, j = 1, ..., M, t = 1, .., T, k = 1, ..., P$$

The first two equations impose constraints on the variable y. Specifically, when pallet k doesn't arrive at its destination at time step t, y_k, t equals 0, and it equals 1 when pallet k reaches its destination at time step t. The summation $\sum_k \sum_t y_{k,t}$ calculates the total cost associated with transporting pallets to their destinations. The cost function is defined as the sum of the y variables, and a smaller value indicates that less time is required to confirm that a pallet has been successfully transported to its destination.

The constraints are more important in this scenario, they are presented from four aspects:

- Definition of decision variable x.
- Constraints about collision avoidance.
- Constraints about movement in different directions.
- Completion conditions.

1. Definition of the decision variable

$$\sum_{j \notin N_i} x^k_{i,j,t} = 0 \quad i, j = 1, ..., M, t = 1, ..., T, k = 1, ..., P \tag{1}$$
$$\sum_i \sum_j x^k_{i,j,t} = 1 \quad i, j = 1, ..., M, t = 1, ..., T, k = 1, ..., P \tag{2}$$
$$\sum_{j \in N_i} x^k_{j,i,t} = \sum_{n \in N_i} x^k_{i,n,t+1} \quad i, j = 1, ..., M, t = 1, ..., T, k = 1, ..., P \tag{3}$$

N_i indicates that how many notes are connected to the node i. The constraint 1 presents that the pallet can not move to the node i from nodes that are not connected to the node i. The constraint (2) indicates that the pallet must and only has one action at each time step: move to an adjacent node or stay at the same node. The constraint (3) indicates that actions of each pallet is continuous at each time step. In summary, the three constraints defined above restrict the basic definition

of the variable x.

2. Collision avoidance

$$\sum_{k}\sum_{i\in N_j} x^k_{i,j,t} \leq 1 \quad j = 1, ..., M, t = 1, ..., T \tag{4}$$

$$\sum_{k}(x^k_{i,j,t} + x^k_{j,i,t}) \leq 1 \quad i, j = 1, ..., M, t = 1, ..., T \tag{5}$$

The two constraints above restrict two pallets from entering the same node, and they cannot swap positions.

3. Movement Constraints

$$F^k_{it}(\sum_{j\in D_i}^{M} x^k_{j,j,t+1} + \sum_{j\in D_i}^{M} x^k_{j,j,t+2}) = 1 \quad i = 1, ..., M, t = 1, ..., T, k = 1, ..., P \tag{6}$$

This pallet system grants the pallet the capability to traverse horizontally, vertically, and diagonally. Specifically, the pallet requires only one time unit for horizontal or vertical movement, while diagonal movement necessitates two time units.

4. Start Positions

$$x^k_{R_k,R_k,0} = 1 \quad k = 1, ..., P \tag{7}$$

R defines the initial position of all pallets. At the moment 0, all pallets should be in their initial position.

5. Completion Conditions

$$\sum_{i\in N_j} x^k_{i,j,t} = 1 \quad j \in G_{kq}, t \in W_k, k = 1, ..., P, q = 1, ..., Q \tag{8}$$

G represents a set of the goal positions of each task of each pallet.The number of tasks of each pallet Q depends on the number of products.This constraint enforces that the pallets must arrive at their assigned target locations at the predetermined time. The timing is determined by the planning scheme derived from the upper-level problem.

In summary, the mathematical model and notations are shown below:

Notation of objective functions :

$$Minimize \quad \sum_{k}\sum_{t} y_{k,t} \quad i,j=1,...,M, t=1,..,T, k=1,...,P$$

Subject to (**constraints**) :

$$\sum_{j\notin N_i} x^k_{i,j,t} = 0 \quad i,j=1,...,M, t=1,...,T, k=1,...,P \tag{1}$$

$$\sum_{i}\sum_{j} x^k_{i,j,t} = 1 \quad i,j=1,...,M, t=1,...,T, k=1,...,P \tag{2}$$

$$\sum_{j\in N_i} x^k_{j,i,t} = \sum_{n\in N_i} x^k_{i,n,t+1} \quad i,j=1,...,M, t=1,...,T, k=1,...,P \tag{3}$$

$$\sum_{k}\sum_{i\in N_j} x^k_{i,j,t} \leq 1 \quad j=1,...,M, t=1,...,T \tag{4}$$

$$\sum_{k}(x^k_{i,j,t} + x^k_{j,i,t}) \leq 1 \quad i,j=1,...,M, t=1,...,T \tag{5}$$

$$F^k_{it}(\sum_{j\in D_i}^{M} x^k_{j,j,t+1} + \sum_{j\in D_i}^{M} x^k_{j,j,t+2}) = 1 \quad i=1,...,M, t=1,...,T, k=1,...,P \tag{6}$$

$$x^k_{R_k,R_k,0} = 1 \quad k=1,...,P \tag{7}$$

$$\sum_{i\in N_j} x^k_{i,j,t} = 1 \quad j\in G_{kq}, t\in W_k, k=1,...,P, q=1,...,Q \tag{8}$$

$$\sum_{i\in N_j} x^k_{i,j,t} = 1 - y_{k,t} \quad j\in G_k, t=1,..,T, k=1,...,P \tag{9}$$

$$-y_{k,t} + y_{k,t+1} \leq 0 \quad t\leq W_{kq}-1, k=1,...,P, q=1,...,Q \tag{10}$$

Notation of objective functions :

$y_{k,t}$: $Equals\ 0\ if\ the\ pallet\ k\ arrives\ its\ target\ positions\ on\ the\ time\ step\ t$, $1\ otherwise$

P : $The\ number\ of\ pallets$

T : $The\ number\ of\ time\ steps$

M : $The\ number\ of\ nodes$

Notation of constraints :

$x^k_{i,j,t}$: $Equals\ 1\ if\ the\ pallet\ k\ move\ from\ node\ i\ to\ node\ j\ on\ the\ time\ step\ t$, $0\ otherwise$

F^k_{ij} : $Equals\ 1\ if\ the\ pallet\ k\ move\ from\ node\ i\ to\ node\ j\ diagonally\ on\ the\ time\ step\ t$, $0\ otherwise$

R_k : $Start\ positions\ of\ each\ pallet$

D_i : $The\ set\ of\ nodes\ that\ connets\ to\ node\ i$

G : $Goal\ positions\ of\ each\ task\ of\ each\ pallet$

W : $Target\ time\ of\ \ reaching\ the\ goal\ position$

3.3.4 Integration Of Task Allocation And Path Finding

After formulating the upper and lower problems, it becomes evident that they both belong to the realm of mixed-integer linear programming problems. Moreover, considering the scale of parameters within this system, traditional solving methods would prove highly challenging, and the considerable latency could render the system inadequately responsive to sudden failures. In this context, confirming the optimality of a solution becomes extraordinarily NP-hard.

To be specific, the task of verifying a solution's optimality is NP-hard in this context. In conventional approaches, acquiring and employing High-Point Problem Point (HPP) becomes necessary within the solution algorithm. The intricacy of the constraints in both the leader and follower optimization problems is such that constructing a High-Point Relaxation (HPR) can be a formidable task.

This framework for resolving two-tier optimization problems ensures that the two optimization problems are tackled independently. The crux of merging these two problems hinges on accurately predicting the parameters essential for the first problem. If it becomes feasible to predict the transmission time for each pallet required in the first problem with precision or within a reasonable deviation range, then the bilevel problem can be addressed separately.

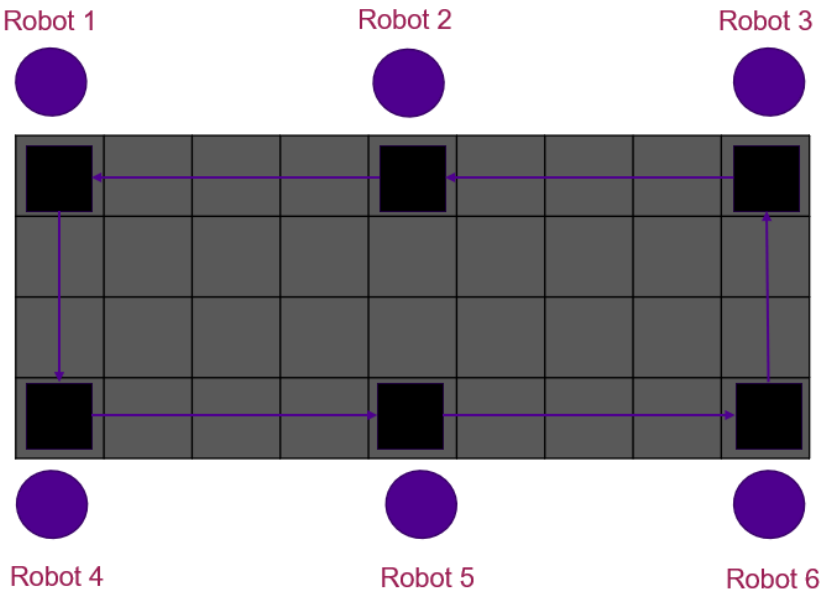


Figure 3.6. *Example Route Plan 1*

The Prediction of Transport Time

Initially, an assumption is made that all pallets can be transported within an ideal timeframe, representing the transportation duration without factoring in collisions. In reality, the transportation duration for all pallets should equal or surpass the ideal time. This differentiation categorizes pallet transport times into "ideal transport time" and "non-ideal transport time."

Before delving further into these two classifications, an essential observation needs highlighting: the concurrent existence of pallets with transport tasks should not surpass the count of operational robots. Each product is positioned on an individual pallet, and when managing a substantial product inventory, there may be twenty to thirty pallets present within the system. These pallets can be segregated into two groups: those with transport tasks and those without. Having a transport task signifies that, as per the planning scheme, a robot awaits the pallet's transportation to its designated location. Hypothetically, if all ten robots are engaged in work, a maximum of ten robots could await the transportation of the pallets and their products, leading to a maximum of ten pallets with transport tasks. These tasks pertain to the transfer of pallets from one robot's position to another robot's location.

Based on this conclusion, the following transportation scenarios are discussed:

1. Pallet's ideal routes would not have overlapped.

   In Figures 3.6 and 3.7, the black square symbolizes the pallet, while the presence of purple color arrows signifies the intended movement direction of the pallet. As illustrated in the figures, when the optimal pathways of the pallets remain

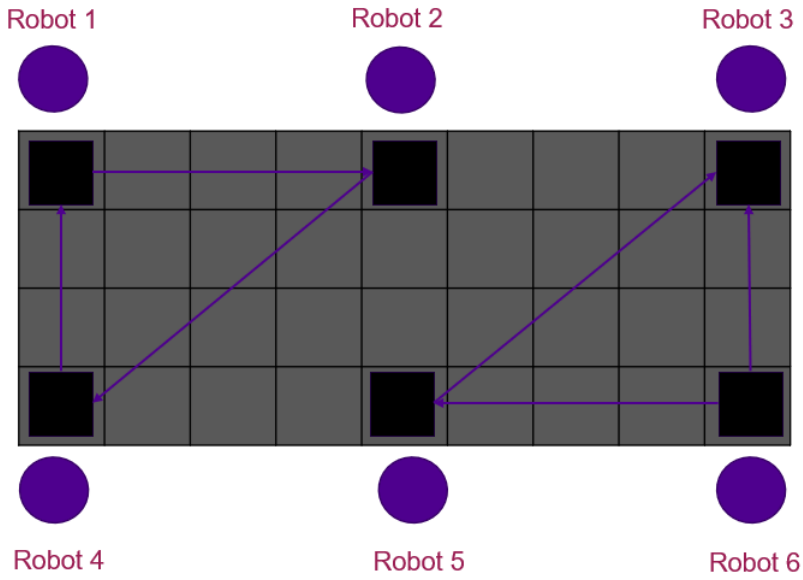


Figure 3.7. *Example Route Plan 2*

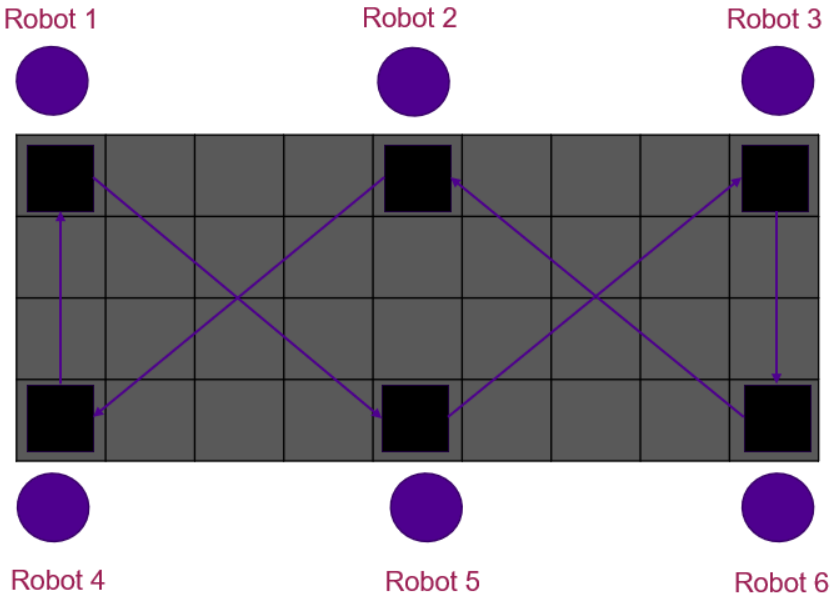


Figure 3.8. *Example Route Plan 3*

non-intersecting, all the pallets can successfully reach their designated destinations within the expected timeframe.

2. Ideal routes for pallet have intersections.

   Even in scenarios where the pallet's optimal path intersects with other paths, the pallet can still reach its designated destination within the expected timeframe.

   In Figure 3.9, the black numbers correspond to the time step at which pallet 1 reaches the present node (with time step commencing at 0), while the white numbers denote the time step at which pallet 2 attains the same node. Figure 3.8

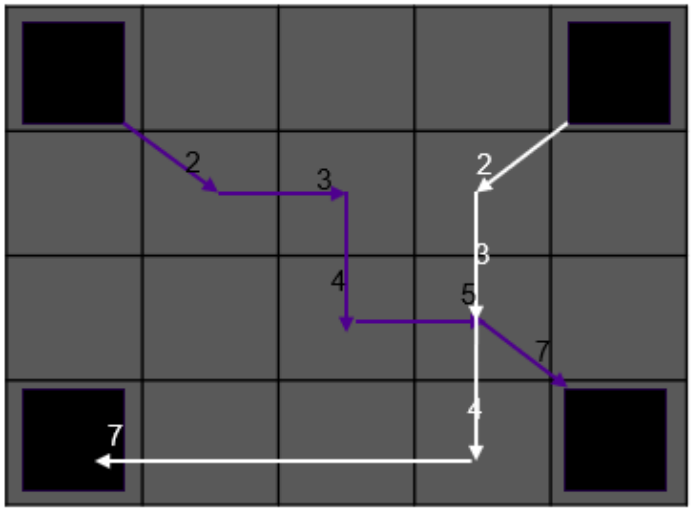


Figure 3.9. *Conflict avoidance solutions.*

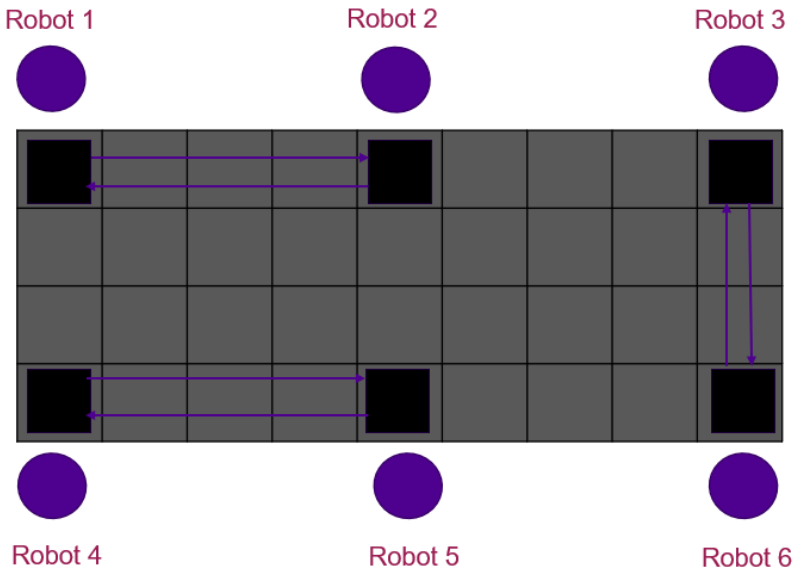


Figure 3.10. *Example Route Plan 3*

exemplifies a collision avoidance path scheme, yet numerous alternative schemes exist. As depicted, within the collision avoidance context, both pallets have adhered to the anticipated time frame to arrive at their designated destinations. As stated earlier, the number of pallets with simultaneous transportation tasks does not exceed the number of operating robots, so there is enough space on the system to support the desired route for collision avoidance.

3. Ideal routes overlap.

   As presented in Figure 3.10, when the routes of two pallets overlap, then at least

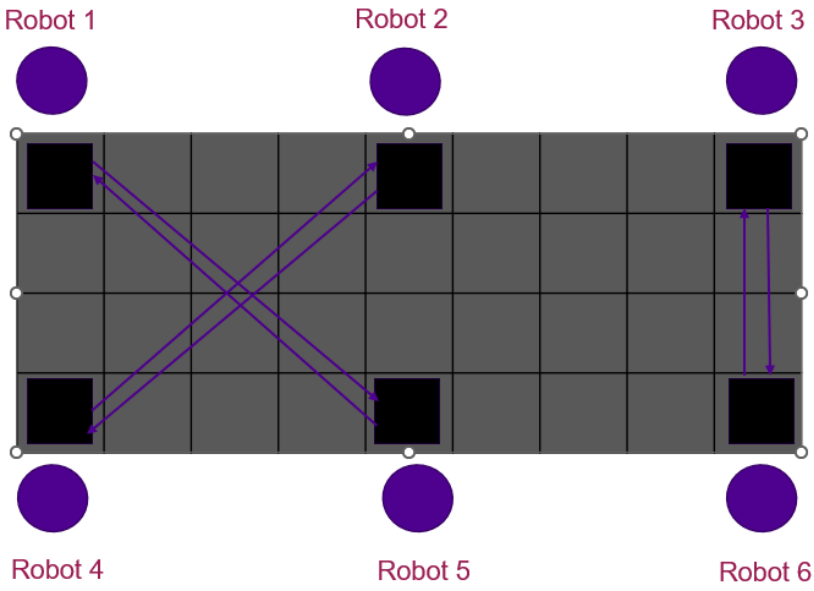


Figure 3.11. *Example Route Plan 4*

one of the pallets will not be able to reach the location at the ideal transportation time.

4. Ideal routes overlap and have intersections.

   In the Figure 3.11, the pallet not only fails to reach its intended destination within the desired timeframe but also necessitates the longest transit duration among all the scenarios.

To summarize, a rule is established to predict the transportation time: when the routes of pallets overlap, more than 2 time steps are preset as margin for these pallets.

In the following pseudocode, Algorithm 1 illustrates the prediction of transportation time based on the departure and destination nodes. This prediction algorithm is incorporated into the upper-level problem solver.

Algorithm 1 A function for transport time prediction.

```
function PredictTransportTime(StartNodes,GoalNodes)
    PredictTransportTime ← IdealTransportTime(StartNodes, GoalNodes)
    if RoutesOverlap(StartNodes, GoalNodes) then
        PredictTransportTime = PredictTransportTime + 1
        if RoutesIntersect(StartNodes, GoalNodes) then
            PredictTransportTime = PredictTransportTime + 1
        end if
    end if
    return PredictTransportTime
end function
```

Algorithm 2 outlines the framework for solving a bilevel Mixed-Integer Linear Programming

(MILP) problem, where the lower-level problem focuses on path finding.

Algorithm 2 A framework for solving the path finding based bilevel MILP problem.

Require: $NumOfProducts, NumOfRobots, ColorSetting \leftarrow Initialise()$

$Initialization \leftarrow NumOfProducts, NumOfRobots, ColorSetting$

$GoalPositions, TargetTime \leftarrow UPSolver(Initialization, PredictTransportTime())$

$Route \leftarrow LPSolver(GoalPositions, TargetTime)$

while $IsSolutionFeasible(Route)$ **do**

    $TimeMargin \leftarrow UpdatePredictTransportTime$

    $TargetTime \leftarrow TargetTime + TimeMargin$

    $Route \leftarrow LPSolver(GoalPositions, TargetTime)$

end while

return $Route$

4. IMPLEMENTATION OF THE ALGORITHM

The system design presented in this paper will be implemented in FastLab and this chapter presents the software implementation of that design. The system architecture is presented in Sec. 4.1. The modeling implementation including linearization is presented in Sec. 4.2. The implementation of solvers is presented in Sec. 4.3. The implementation of the integration is presented in Sec. 4.4.

4.1 Architecture

The three components of the system have been stated in the chapter 3 on algorithm design and the Figure 4.1 shows how the three components have been organized in the complete workflow, each column represents a parallel computation, assuming that all products are divided into B batches for execution.

There are three reasons for this design:

- To decrease the dimensions of the initial problem. During the experimental phase, the system could potentially handle a substantial influx of orders. Given the intricate nature of the model, as the problem's scale increases, the time required to attain an optimal solution grows considerably. In light of this, adopting batch processing emerges as a viable approach to mitigate the problem's magnitude and complexity. Nevertheless, this strategy comes at the expense of achieving sub-optimal solutions as opposed to optimal ones. Therefore, the segmentation of batches must be conducted judiciously. When the number of products to be allocated is relatively

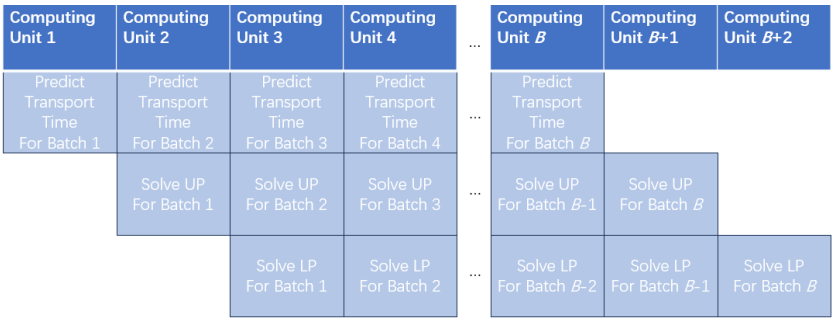


Figure 4.1. *Workflow*

modest, a small number of batches can be chosen (where a batch number of 1 resembles uniform planning for all products). On the contrary, when the number of products to be allocated is substantial, it becomes advantageous to divide the products into multiple batches for more manageable handling.

- Immediate response to emergencies In the event of hardware malfunction (e.g., pallet system breakdown, robot failure), the system must swiftly generate alternative strategies. Upon hardware failure, the initial planning scheme is discarded, and all operations come to a halt. Subsequently, the system will devise alternatives by leveraging the presently accessible resources. These alternatives will be designed in smaller batches. Concurrently, while the alternative solution is active, the system will proceed to generate larger batches following the optimal solution. The alternative strategy will remain operational until the superior solution is successfully implemented.
- Continuous data collection to improve prediction accuracy Information from each batch is collected and employed to forecast parameters. The current data prediction techniques are tailored to the volume of available data. Once a certain volume of data has been amassed, alternate methods such as neural networks can be applied for prediction purposes.

4.2 Modeling

The implementation employs the optimization library Or-tools, which is an open-source project developed by Google. Or-tools serves as a versatile tool for Linear Optimization, Integer Optimization, and Constraint Optimization. It proves especially apt for various practical scenarios, such as Assignment, Packing, Scheduling, Routing, and Network Flows, offering a range of solution strategies.

The model's formulation is achieved through the functions provided by Or-tools. The fundamental principles underlying model definition are as follows: 1. Strive to minimize the count of variables to streamline the model. A higher number of variables translates to greater computational resource demands. 2. Avert scenarios where multiple variables become interdependent or coupled.

4.2.1 Linearization

Given that both problems are characterized as linear mixed-integer programming issues, it's important to note that Or-tools exclusively accommodates linear expressions for the integer solver. Consequently, any nonlinear constraints embedded within the model necessitate linearization prior to being formulated as Or-tools expressions.

1. Product of two binary variables. Assume that there are two binary variables: x_1 and

x_2, obviously, Ax_1 and Bx_2 are linear expression(A,B is a constant number), $Ax_1 + Bx_2$ is also a linear expression. However, the product of x_1 and x_2 is an nonlinear expression.

Assume y represents the multiplication value of x_1 and x_2, analyzing the output of the calculations for diverse values of $y = x1 * x2$, the following table is available:

This table shows that the upper and lower boundaries of y are restricted by the variable x.

$$\begin{aligned} y &\leq x_1 \\ y &\leq x_2 \\ y &\geq x_1 + x_2 - 1 \end{aligned}$$

The first two equations restrict the upper bound on y to the smallest value of $x1$ and $x2$, and the third equation restricts the lower bound on y.

2. Product of one binary variable and one continuous variable. Assume that C represents a continuous variable and x represent an binary variable, and variable z represents the product of C and x. The upper bound of C is a big number M and the lower bound of C is zero. The linearization shows below:

$$\begin{aligned} z &\leq M \times x \\ z &\leq C \\ z &\geq C - (1 - x)M \\ z &\geq 0 \end{aligned}$$

It's crucial to recognize that when the variable x assumes a value of zero, the initial inequality ensures that z also becomes zero. Additionally, it's noteworthy that the third inequality simply establishes that z needs to surpass a negative threshold. Conversely, in situations where x equals one, the first inequality guarantees that z remains below the predefined "Big M" threshold, and this constraint is further reinforced by the second inequality. Ultimately, the last inequality dictates that z must be equal to or exceed the value indicated by C.

3. Maximum value of two variables. Assume that there exits a constraint that needs $\max\{x_1, x_2\}$, a new binary decision variable y that will be equal 1 if $x_1 \geq x_2$, otherwise

0. The following inequality constraints enforce that $X = \max\{x_1, x_2\}$:

$$\begin{aligned} X &\geq x_1 \\ X &\geq x_2 \\ X &\leq x_1 + M(1-y) \\ X &\leq x_2 + My \end{aligned}$$

The first two constraints restrict that the lower bound of X is $\max\{x1, x2\}$. The last two constraints indicate that X is equal to x_1 when x_1 is greater than x_2, X is equal to x_2 if x_2 is greater than x_1.

4.3 Solver

Within the realm of OR-Tools, there are two primary instruments available for addressing integer programming quandaries: the MPSolver and the CP-SAT solver. For the implementation of this system, the CP-SAT solver is selected, primarily due to its superior performance in tackling mixed-integer linear problems.

OR-Tools provide a python module named CpModel that contains all interface needed for using CP-SAT solver. What follows is a diagram that describe how to solve the MILP problem by CpModel.

4.3.1 Task Scheduling Solver

As depicted in the figure, resolving the planning problem necessitates the preliminary definition of variables, the types of which are detailed in the accompanying table:

Constant variables are established based on the prevailing environment. Decision variables wield influence over the ultimate planning arrangement. Intermediate variables arise from the linearization procedure, and their quantity greatly surpasses that of other variables, constituting a principal factor impacting algorithmic speed.

Within OR-Tools, constraints are categorized into two distinct classes: hard constraints and soft constraints. Hard constraints necessitate tight adherence to predetermined limits, while soft constraints suggest that optimal compliance with these limits is desired. Notably, for this scheduling dilemma, all constraints fall under the hard category.

The CP-SAT solver leverages a lazy clause generation mechanism built atop an SAT solver. In the course of implementation, the primary drivers influencing the CP-SAT

solver's computational velocity are the meticulous formulation of constraints and variables. The solver's intrinsic parameters have largely been configured to optimal default settings. The cardinal parameter available for tuning pertains to the count of search workers. CP-SAT encompasses rudimentary parallelism features; the initial thread defaults to executing fundamental strategies, the second thread adheres to user-specified strategies (if provided), threads three through six adopt variations of the fundamental approach, and all remaining threads conduct extensive domain explorations. In this specific system, the thread count is set to sixteen.. Concurrent linear searches for solutions are executed in parallel, and limited data exchanges occur between threads until an optimal solution is ascertained.

4.3.2 Path Finding Solver

The formulation of variables and constraints for the path planning problem within this system is comparably less intricate than that of the scheduling problem. Notably, to address the distinctive behaviors of the transporter when navigating diagonally versus moving linearly, the CpModel solely employs the "OnlyEnforceIf" function. This function operates via a binary variable that dictates the activation or deactivation of specific constraints.

5. DISCUSSION AND RESULTS

5.1 Simulation

This chapter is concerned with the software simulation of the implemented system. First the software environment will be presented, then the simulation results for JSSP, and then after that the simulation results for path planning will be shown.

Programming Language: Python 3.10.0 Modeling Tool: OR-Tools v9.7 Solver: OR-Tools v9.7, Gurobi Optimizer

5.1.1 Job Shop Scheduling Problem Simulation

In the simulation experiments, six robots and six pallets were invoked with a total of six products waiting to be produced. Each product entails three tasks. The processing duration for these tasks are specified as 50 seconds, 28 seconds, and 48 seconds, while an additional 40 seconds are required for robot tool changeovers. It is posited that a total of three colors – red, yellow, and blue – are at the disposal for utilization. The color requisites for each task and product are illustrated in the table 5.1. The pallets are running in a 4×14 grid.

The figure 5.1 and 5.2 show that the text and Gantt chart planning of the scenario in the simulation. The figure 5.3 shows that the impact of concurrent workers on the solver's solution discovery time. As elucidated in Section 4.3.1, the initial three threads undertake the foundational approach and its derivations, which demonstrate comparatively less

Table 5.1. *Color Requirements Of Products*

Product Id	Required Color for Task 1	Required Color for Task 2	Required Color for Task 3
1	Red	Red	Yellow
2	Yellow	Blue	Red
3	Blue	Yellow	Yellow
4	Red	Blue	Yellow
5	Blue	Red	Blue
6	Blue	Red	Yellow

	Task	Robot	Product	Start	Finish	task_duration
0	Product 1 Task 1	6	Product 1	0	50	50
1	Product 1 Task 2	2	Product 1	90	118	28
2	Product 1 Task 3	5	Product 1	122	170	48
3	Product 2 Task 1	5	Product 2	0	50	50
4	Product 2 Task 2	3	Product 2	61	89	28
5	Product 2 Task 3	2	Product 2	118	166	48
6	Product 3 Task 1	2	Product 3	0	50	50
7	Product 3 Task 2	5	Product 3	54	82	28
8	Product 3 Task 3	1	Product 3	122	170	48
9	Product 4 Task 1	1	Product 4	0	50	50
10	Product 4 Task 2	4	Product 4	54	82	28
11	Product 4 Task 3	6	Product 4	122	170	48
12	Product 5 Task 1	4	Product 5	0	50	50
13	Product 5 Task 2	1	Product 5	54	82	28
14	Product 5 Task 3	3	Product 5	96	144	48
15	Product 6 Task 1	3	Product 6	0	50	50
16	Product 6 Task 2	6	Product 6	54	82	28
17	Product 6 Task 3	4	Product 6	122	170	48

Figure 5.1. *Task Allocation*

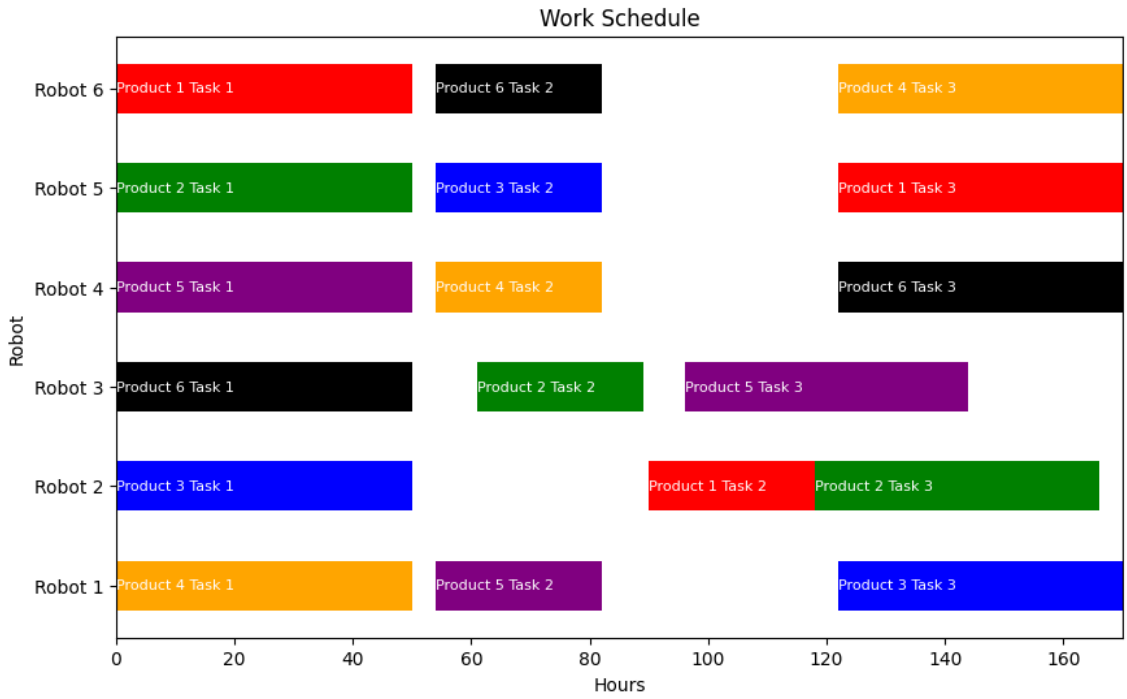


Figure 5.2. *Gantt Chart For Scheduling*

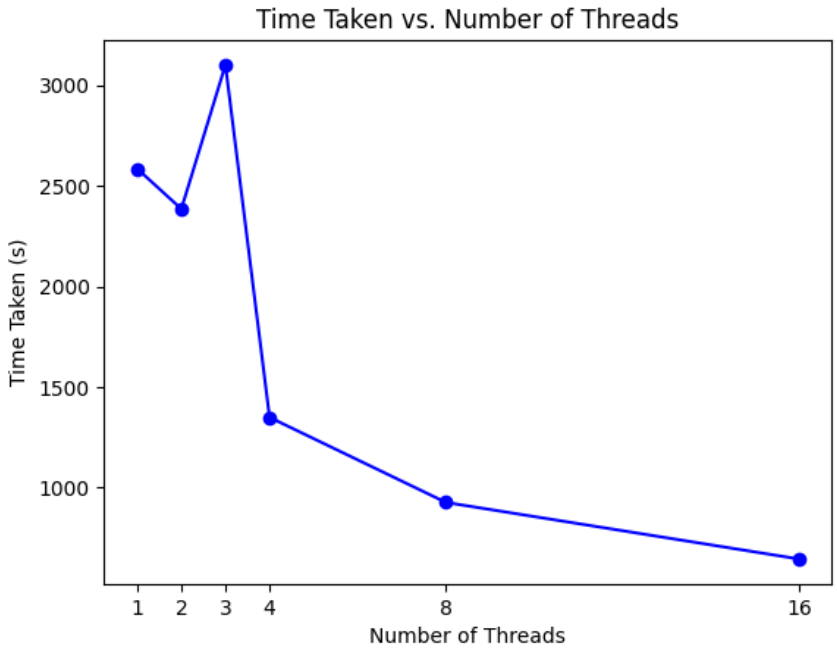


Figure 5.3. *Relationship between the number of workers and the running time of the solver.*

efficiency with their usage. Upon introducing the fourth thread, a noticeable reduction in runtime transpires. Subsequent increments in thread count further diminish execution time, albeit with diminishing returns as the thread count grows.

5.1.2 Route Planning Simulation

Derived from the simulation outcomes of the JSSP, the pallets movement solution is formulated by solving the route planning problem. The solution is shown in Figure 5.4, where each quaternion represents (node before move, node after move, time step, pallet id).

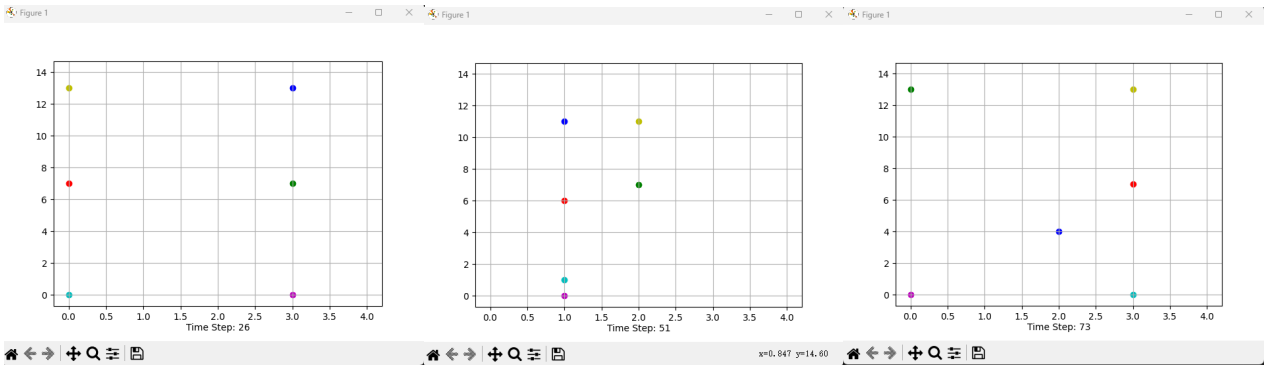
The figure 5.5 is a path visualization made using the matplotlib library. The table 5.2 show the target positions and time of each pallet. The initial assumption is that the starting position for each pallet is the location designated for performing its first task. Pallet 1 is represented by the dark green dot, Pallet 2 by the blue-green dot, Pallet 3 by the purple dot, Pallet 4 by the blue dot, Pallet 5 by the yellow dot, and Pallet 6 by the red dot. The figure 5.5 a shows the pallets stay in the target positions of the first task and waiting for robots processing manufacturing jobs. The figure 5.5 b shows that pallets are moving to positions for processing the second task. The figure 5.5 c shows that five pallets processing the second task and one is moving.

id	transporter_1	transporter_2	transporter_3	transporter_4	transporter_5	transporter_6
1	(55, 55, 0, 0)	(31, 31, 0, 1)	(28, 28, 0, 2)	(0, 0, 0, 3)	(3, 3, 0, 4)	(52, 52, 0, 5)
1	(55, 55, 1, 0)	(31, 31, 1, 1)	(28, 28, 1, 2)	(0, 0, 1, 3)	(3, 3, 1, 4)	(52, 52, 1, 5)
1	(55, 55, 2, 0)	(31, 31, 2, 1)	(28, 28, 2, 2)	(0, 0, 2, 3)	(3, 3, 2, 4)	(52, 52, 2, 5)
1	(55, 55, 3, 0)	(31, 31, 3, 1)	(28, 28, 3, 2)	(0, 0, 3, 3)	(3, 3, 3, 4)	(52, 52, 3, 5)
1	(55, 55, 4, 0)	(31, 31, 4, 1)	(28, 28, 4, 2)	(0, 0, 4, 3)	(3, 3, 4, 4)	(52, 52, 4, 5)
1	(55, 55, 5, 0)	(31, 31, 5, 1)	(28, 28, 5, 2)	(0, 0, 5, 3)	(3, 3, 5, 4)	(52, 52, 5, 5)
1	(55, 55, 6, 0)	(31, 31, 6, 1)	(28, 28, 6, 2)	(0, 0, 6, 3)	(3, 3, 6, 4)	(52, 52, 6, 5)
1	(55, 55, 7, 0)	(31, 31, 7, 1)	(28, 28, 7, 2)	(0, 0, 7, 3)	(3, 3, 7, 4)	(52, 52, 7, 5)
1	(55, 55, 8, 0)	(31, 31, 8, 1)	(28, 28, 8, 2)	(0, 0, 8, 3)	(3, 3, 8, 4)	(52, 52, 8, 5)
1	(55, 55, 9, 0)	(31, 31, 9, 1)	(28, 28, 9, 2)	(0, 0, 9, 3)	(3, 3, 9, 4)	(52, 52, 9, 5)
1	(55, 55, 10, 0)	(31, 31, 10, 1)	(28, 28, 10, 2)	(0, 0, 10, 3)	(3, 3, 10, 4)	(52, 52, 10, 5)
1	(55, 55, 11, 0)	(31, 31, 11, 1)	(28, 28, 11, 2)	(0, 0, 11, 3)	(3, 3, 11, 4)	(52, 52, 11, 5)
1	(55, 55, 12, 0)	(31, 31, 12, 1)	(28, 28, 12, 2)	(0, 0, 12, 3)	(3, 3, 12, 4)	(52, 52, 12, 5)
1	(55, 55, 13, 0)	(31, 31, 13, 1)	(28, 28, 13, 2)	(0, 0, 13, 3)	(3, 3, 13, 4)	(52, 52, 13, 5)
1	(55, 55, 14, 0)	(31, 31, 14, 1)	(28, 28, 14, 2)	(0, 0, 14, 3)	(3, 3, 14, 4)	(52, 52, 14, 5)
1	(55, 55, 15, 0)	(31, 31, 15, 1)	(28, 28, 15, 2)	(0, 0, 15, 3)	(3, 3, 15, 4)	(52, 52, 15, 5)
1	(55, 55, 16, 0)	(31, 31, 16, 1)	(28, 28, 16, 2)	(0, 0, 16, 3)	(3, 3, 16, 4)	(52, 52, 16, 5)
1	(55, 55, 17, 0)	(31, 31, 17, 1)	(28, 28, 17, 2)	(0, 0, 17, 3)	(3, 3, 17, 4)	(52, 52, 17, 5)
1	(55, 55, 18, 0)	(31, 31, 18, 1)	(28, 28, 18, 2)	(0, 0, 18, 3)	(3, 3, 18, 4)	(52, 52, 18, 5)
1	(55, 55, 19, 0)	(31, 31, 19, 1)	(28, 28, 19, 2)	(0, 0, 19, 3)	(3, 3, 19, 4)	(52, 52, 19, 5)
1	(55, 55, 20, 0)	(31, 31, 20, 1)	(28, 28, 20, 2)	(0, 0, 20, 3)	(3, 3, 20, 4)	(52, 52, 20, 5)
1	(55, 55, 21, 0)	(31, 31, 21, 1)	(28, 28, 21, 2)	(0, 0, 21, 3)	(3, 3, 21, 4)	(52, 52, 21, 5)
1	(55, 55, 22, 0)	(31, 31, 22, 1)	(28, 28, 22, 2)	(0, 0, 22, 3)	(3, 3, 22, 4)	(52, 52, 22, 5)
1	(55, 55, 23, 0)	(31, 31, 23, 1)	(28, 28, 23, 2)	(0, 0, 23, 3)	(3, 3, 23, 4)	(52, 52, 23, 5)
1	(55, 55, 24, 0)	(31, 31, 24, 1)	(28, 28, 24, 2)	(0, 0, 24, 3)	(3, 3, 24, 4)	(52, 52, 24, 5)
1	(55, 55, 25, 0)	(31, 31, 25, 1)	(28, 28, 25, 2)	(0, 0, 25, 3)	(3, 3, 25, 4)	(52, 52, 25, 5)
1	(55, 55, 26, 0)	(31, 31, 26, 1)	(28, 28, 26, 2)	(0, 0, 26, 3)	(3, 3, 26, 4)	(52, 52, 26, 5)
1	(55, 55, 27, 0)	(31, 31, 27, 1)	(28, 28, 27, 2)	(0, 0, 27, 3)	(3, 3, 27, 4)	(52, 52, 27, 5)
1	(55, 55, 28, 0)	(31, 31, 28, 1)	(28, 28, 28, 2)	(0, 0, 28, 3)	(3, 3, 28, 4)	(52, 52, 28, 5)
1	(55, 55, 29, 0)	(31, 31, 29, 1)	(28, 28, 29, 2)	(0, 0, 29, 3)	(3, 3, 29, 4)	(52, 52, 29, 5)
1	(55, 55, 30, 0)	(31, 31, 30, 1)	(28, 28, 30, 2)	(0, 0, 30, 3)	(3, 3, 30, 4)	(52, 52, 30, 5)
1	(55, 55, 31, 0)	(31, 31, 31, 1)	(28, 28, 31, 2)	(0, 0, 31, 3)	(3, 3, 31, 4)	(52, 52, 31, 5)
1	(55, 55, 32, 0)	(31, 31, 32, 1)	(28, 28, 32, 2)	(0, 0, 32, 3)	(3, 3, 32, 4)	(52, 52, 32, 5)
1	(55, 55, 33, 0)	(31, 31, 33, 1)	(28, 28, 33, 2)	(0, 0, 33, 3)	(3, 3, 33, 4)	(52, 52, 33, 5)
1	(55, 55, 34, 0)	(31, 31, 34, 1)	(28, 28, 34, 2)	(0, 0, 34, 3)	(3, 3, 34, 4)	(52, 52, 34, 5)
1	(55, 55, 35, 0)	(31, 31, 35, 1)	(28, 28, 35, 2)	(0, 0, 35, 3)	(3, 3, 35, 4)	(52, 52, 35, 5)
1	(55, 55, 36, 0)	(31, 31, 36, 1)	(28, 28, 36, 2)	(0, 0, 36, 3)	(3, 3, 36, 4)	(52, 52, 36, 5)
1	(55, 55, 37, 0)	(31, 31, 37, 1)	(28, 28, 37, 2)	(0, 0, 37, 3)	(3, 3, 37, 4)	(52, 52, 37, 5)

Figure 5.4. *Route Plan*

Table 5.2. *Target positions and time of pallets.*

Pallet Id	Goal Position 1	Time(s)	Goal Position 2	Time(s)	Goal Position 3	Time(s)
1	(3,7)	0	(3,7)	54	(3,0)	122
2	(0,0)	0	(3,0)	54	(0,7)	122
3	(3,0)	0	(0,0)	54	(3,13)	122
4	(3,13)	0	(0,13)	54	(0,0)	122
5	(0,13)	0	(3,13)	54	(0,13)	122
6	(0,7)	0	(0,7)	54	(3,7)	122



(a) ***(b)*** ***(c)***

Figure 5.5. *Route Visualization.*

5.2 Optimality of The Solution

This section examines the optimality of the solution while considering trade-offs. The solution is considered optimal when all products are produced in a single batch and when transportation times can be accurately predicted.

As discussed in Chapter 3, when ideal routes overlap or intersect, the transit times may not be ideal. In such cases, the predicted transit time includes both the ideal transit time and a time margin. It's important to note that the predicted transit time might be larger than the actual transit time in such situations. The solution ceases to be optimal when the predicted transportation time deviates from the actual time.

When the number of batches for processing products exceeds 1, the solution obtained is not optimal, regardless of whether the transportation time can be accurately predicted or not.

When multiple batches of products are processed, the solution for the pathfinding problem is much faster than the production allocation problem. This allows for multiple attempts to adjust the time margin value. Consequently, even if the predicted transportation time doesn't match the actual time, the resulting suboptimal solution is very close to the optimal solution. Sub-batch processing significantly reduces computation time but may not achieve an optimal solution, and the number of batches is determined based on the number of products and the desired computation time.

5.3 Observation and Analysis of Results

Certainly, in such bilevel optimization scenarios, the approach to problem resolution hinges on whether the lower-level problem exhibits convexity. When dealing with bilevel optimization problems featuring a convex lower-level problem, it is common practice to employ KKT conditional transformations. These transformations serve to reconfigure the lower-level problem into a constraint applicable to the upper-level problem.

On the other hand, when the lower-level problem takes the form of an integer programming problem, the academic realm has introduced various solution methods grounded in high-point relaxation (HPR). It is worth noting that, as of now, open-source libraries dedicated to tackling bilevel integer optimization problems in languages such as Julia or Python are not readily available. Consequently, developing tailored algorithms for addressing the intricacies of smart manufacturing systems, encompassing both production and transportation, is a necessity.

The complexity of the production scheduling problem is attributed to its numerous constraints and the intricate interplay among variables. This complexity renders HPR-based algorithms time-intensive, and they are typically applied to problems of smaller scale. In

the context of this thesis, we delve into the conditions dictating the attainment of optimality in the path planning problem. Our approach adopts a sequential methodology, dissecting the bilevel problem into separate components. Furthermore, we introduce a batch processing mode designed to approximate the time required to solve the two-layer problem to the time needed for addressing the upper-level problem exclusively. This strategic implementation minimizes the overall solving time while enhancing the system's adaptability to dynamic changes.

Within the framework of the manufacturing tasks scheduling problem, the most favorable outcome obtained through the utilization of two solvers, namely CP-SAT and Gurobi, amounted to a computation time of 3 minutes. Notably, by refining the modeling approach for JSSP, a notable reduction in solution time was achieved. However, there remains untapped potential for further enhancement in efficiency. The resolution of the JSSP issue is notably impacted by the underlying hardware, specifically the central processing unit (CPU). An adept CPU can yield a remarkable acceleration in solution speed. In a Route Planning problem, the quantity of variables expands exponentially in proportion to the number of time steps. Concerning linearization, the API furnished by OR-Tools for accommodating non-linear modeling is comparatively less potent than the manual application of linearization techniques.

6. CONCLUSION

This thesis is dedicated to the conception and realization of a planning system designed to generate schedules for production tasks and path planning for the pallet system. Comprising three integral components, this system is formulated through scenario analysis for parameter prediction and the subdivision of a complex two-tier optimization problem.

Commencing with Chapter 1, the background, significance, desired objectives, and key research inquiries underpinning the development of the planning system are expounded. In Chapter 2, a comprehensive technical backdrop is outlined, encompassing discussions and evaluations of diverse techniques and algorithms pertinent to solving Job Shop Scheduling Problems (JSSP), route planning, and bi-level optimization dilemmas. This chapter further categorizes and contrasts the merits and demerits associated with distinct solution methodologies.

Chapter 3 unveils the modeling process and solver design employed to tackle the optimization problem, delineating the journey from real-world scenarios to formal mathematical models. Chapter 4 elaborates upon the architectural framework of the system, elucidating the efficacy of this design in decomposing the complex problem and responding to unforeseen challenges. Furthermore, the application of nonlinear transformation techniques in the modeling phase and the iterative refinement of the solver during the implementation phase are duly presented.

The solution proposed in this theis, addressing the research problem presented in Chapter 1, unfolds as follows: The mathematical model encompassing the scheduling and transportation of production tasks within the system is cast as a two-layer integer optimization problem. To ensure that this problem can be solved within a reasonable timeframe, a POLICY-based integration approach is introduced. By scrutinizing the conditions governing optimality at the lower level, this method streamlines the problem-solving process.

Furthermore, to enhance solution efficiency and equip the system with the ability to navigate unforeseen circumstances, the focus shifts from the pursuit of the optimal solution to identifying a solution that effectively addresses the optimality requirements. This paradigm shift enables the system to adapt and respond more swiftly to dynamic challenges.

6.1 Future work

The planning system devised within this paper can be perceived as an underpinning infrastructure for forthcoming endeavors. Building upon the foundation laid in this study, the following avenues for future exploration are envisioned, aiming to further enhance solution speed:

1. Optimization of the JSSP model stands as a paramount endeavor. Given the intricacies inherent in real-world scenarios, the JSSP model also exhibits a certain level of complexity, characterized by an abundance of intermediate variables (essential for linear transformations) and intricate constraints. By delving into further refinement of the JSSP model, there lies the potential for achieving a substantial acceleration in solution speed.
2. The pursuit of formulating dedicated algorithms for JSSP resolution emerges as a promising avenue. While the conventional Mixed-Integer Linear Programming (MILP) solver generally yields commendable performance in tackling JSSP within this system, the creation of specialized algorithms tailored to this particular problem can contribute to heightened efficiency in the solution process.
3. The exploration of parallel solving algorithms compatible with Graphics Processing Units (GPUs) constitutes another potential research direction. Given the presence of multi-core GPUs within FastLab's experimental infrastructure, there exists the opportunity to devise parallel solving algorithms that harness the full potential of the available hardware resources.
4. This paper adopts a centralized algorithm for resolving the Route Planning problem, but it's also feasible to explore the development of distributed algorithms, such as the Lagrangian relaxation method, to address the same problem.

REFERENCES

[1] Divya Aggarwal, Vijay Kumar, and Ashish Girdhar. “Lagrangian relaxation for the vehicle routing problem with time windows”. In: *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*. 2017, pp. 1601–1606. DOI: 10.1109/ICICICT1.2017.8342810.

[2] Abdelhakim AitZai, Brahim Benmedjdoub, and Mourad Boudhar. “Branch-and-bound and PSO algorithms for no-wait job shop scheduling”. In: *Journal of Intelligent Manufacturing* 27 (2016), pp. 679–688.

[3] Duarte Alemão, André Dionisio Rocha, and José Barata. “Smart manufacturing scheduling approaches—Systematic review and future directions”. In: *Applied Sciences* 11.5 (2021), p. 2186.

[4] Schirin Baer et al. “Multi-Agent Reinforcement Learning for Job Shop Scheduling in Flexible Manufacturing Systems”. In: *2019 Second International Conference on Artificial Intelligence for Industries (AI4I)*. 2019, pp. 22–25. DOI: 10.1109/AI4I46381.2019.00014.

[5] Mikhail A Bragin and Emily L Tucker. “Surrogate “Level-Based” Lagrangian Relaxation for mixed-integer linear programming”. In: *Scientific Reports* 12.1 (2022), p. 22417.

[6] Jing-fang Chen, Ling Wang, and Zhi-ping Peng. “A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling”. In: *Swarm and Evolutionary Computation* 50 (2019), p. 100557. ISSN: 2210-6502. DOI: https://doi.org/10.1016/j.swevo.2019.100557. URL: https://www.sciencedirect.com/science/article/pii/S2210650219302652.

[7] Ronghua Chen et al. “A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem”. In: *Computers Industrial Engineering* 149 (2020), p. 106778. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2020.106778. URL: https://www.sciencedirect.com/science/article/pii/S0360835220304885.

[8] Ruiqi Chen, Wenxin Li, and Hongbing Yang. “A Deep Reinforcement Learning Framework Based on an Attention Mechanism and Disjunctive Graph Embedding for the Job-Shop Scheduling Problem”. In: *IEEE Transactions on Industrial Informatics* 19.2 (2022), pp. 1322–1331.

[9] Yanyan Chen, Michael G. H. Bell, and Klaus Bogenberger. “Reliable Pretrip Multipath Planning and Dynamic Adaptation for a Centralized Road Navigation System”.

In: *IEEE Transactions on Intelligent Transportation Systems* 8.1 (2007), pp. 14–20. DOI: 10.1109/TITS.2006.889437.

[10] Satyendra Singh Chouhan and Rajdeep Niyogi. “Dmapp: A distributed multi-agent path planning algorithm”. In: *AI 2015: Advances in Artificial Intelligence: 28th Australasian Joint Conference, Canberra, ACT, Australia, November 30–December 4, 2015, Proceedings 28*. Springer. 2015, pp. 123–135.

[11] Haijuan Cui, Xiaochuan Luo, and Yuan Wang. “Scheduling of steelmaking-continuous casting process using deflected surrogate Lagrangian relaxation approach and DC algorithm”. In: *Computers & Industrial Engineering* 140 (2020), p. 106271.

[12] Jur van Den Berg et al. “Centralized path planning for multiple robots: Optimal decoupling into sequential plans.” In: *Robotics: Science and systems*. Vol. 2. 2.5. 2009, pp. 2–3.

[13] Stepan Dergachev and Konstantin Yakovlev. “Distributed Multi-Agent Navigation Based on Reciprocal Collision Avoidance and Locally Confined Multi-Agent Path Finding”. In: *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. 2021, pp. 1489–1494. DOI: 10.1109/CASE49439.2021.9551564.

[14] Haojie Ding and Xingsheng Gu. “Hybrid of human learning optimization algorithm and particle swarm optimization algorithm with scheduling strategies for the flexible job-shop scheduling problem”. In: *Neurocomputing* 414 (2020), pp. 313–332. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.07.004. URL: https://www.sciencedirect.com/science/article/pii/S0925231220311085.

[15] Mahboob Elahi. “Design and Implementation of a policy-driven orchestrator for production planning”. MA thesis. 2020.

[16] Lei Fang, Peter B Luh, and H Chan. “Improving the Lagrangian relaxation approach for large job-shop scheduling”. In: *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No. 00CH37187)*. Vol. 1. IEEE. 2000, pp. 552–557.

[17] Faiza Gul et al. “A Centralized Strategy for Multi-Agent Exploration”. In: *IEEE Access* 10 (2022), pp. 126871–126884. DOI: 10.1109/ACCESS.2022.3218653.

[18] Ronny S Hansmann, Thomas Rieger, and Uwe T Zimmermann. “Flexible job shop scheduling with blockages”. In: *Mathematical Methods of Operations Research* 79 (2014), pp. 135–161.

[19] Hao Hu et al. “Deep reinforcement learning based AGVs real-time scheduling with mixed rule for flexible shop floor in industry 4.0”. In: *Computers Industrial Engineering* 149 (2020), p. 106749. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2020.106749. URL: https://www.sciencedirect.com/science/article/pii/S036083522030468X.

[20] Kelin Jose and Dilip Kumar Pratihar. “Task allocation and collision-free path planning of centralized multi-robots system for industrial plant inspection using heuristic methods”. In: *Robotics and Autonomous Systems* 80 (2016), pp. 34–42. ISSN:

0921-8890. DOI: https://doi.org/10.1016/j.robot.2016.02.003. URL: https://www.sciencedirect.com/science/article/pii/S0921889016000282.

[21] Imed Kacem, Slim Hammadi, and Pierre Borne. “Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 32.1 (2002), pp. 1–13.

[22] Hyoung Seok Kang et al. “Smart manufacturing: Past research, present findings, and future directions”. In: *International journal of precision engineering and manufacturing-green technology* 3 (2016), pp. 111–128.

[23] Bai Li et al. “Centralized and optimal motion planning for large-scale AGV systems: A generic approach”. In: *Advances in Engineering Software* 106 (2017), pp. 33–46.

[24] Z.C. Li et al. “An elitist nondominated sorting hybrid algorithm for multi-objective flexible job-shop scheduling problem with sequence-dependent setups”. In: *Knowledge-Based Systems* 173 (2019), pp. 83–112. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2019.02.027. URL: https://www.sciencedirect.com/science/article/pii/S0950705119300887.

[25] Risheng Liu et al. “Towards gradient-based bilevel optimization with non-convex followers and beyond”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 8662–8675.

[26] Shu Luo. “Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning”. In: *Applied Soft Computing* 91 (2020), p. 106208.

[27] David Luviano and Wen Yu. “Continuous-time path planning for multi-agents with fuzzy reinforcement learning”. In: *Journal of Intelligent & Fuzzy Systems* 33.1 (2017), pp. 491–501.

[28] R Timothy Marler and Jasbir S Arora. “Survey of multi-objective optimization methods for engineering”. In: *Structural and multidisciplinary optimization* 26 (2004), pp. 369–395.

[29] Jatoth Mohan, Krishnanand Lanka, and A Neelakanteswara Rao. “A review of dynamic job shop scheduling techniques”. In: *Procedia Manufacturing* 30 (2019), pp. 34–39.

[30] Ghasem Moslehi and Mehdi Mahnam. “A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search”. In: *International Journal of Production Economics* 129.1 (2011), pp. 14–22. ISSN: 0925-5273. DOI: https://doi.org/10.1016/j.ijpe.2010.08.004. URL: https://www.sciencedirect.com/science/article/pii/S0925527310002938.

[31] T. Nishi, M. Ando, and M. Konishi. “Distributed route planning for multiple mobile robots using an augmented Lagrangian decomposition and coordination technique”. In: *IEEE Transactions on Robotics* 21.6 (2005), pp. 1191–1200. DOI: 10.1109/TRO.2005.853489.

[32] Cemal Özgüven, Lale Özbakır, and Yasemin Yavuz. “Mathematical models for job-shop scheduling problems with routing and process plan flexibility”. In: *Applied Mathematical Modelling* 34.6 (2010), pp. 1539–1548. ISSN: 0307-904X. DOI: https://doi.org/10.1016/j.apm.2009.09.002. URL: https://www.sciencedirect.com/science/article/pii/S0307904X09002819.

[33] Junyoung Park et al. “Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning”. In: *International Journal of Production Research* 59.11 (2021), pp. 3360–3377.

[34] F. Pezzella, G. Morganti, and G. Ciaschetti. “A genetic algorithm for the Flexible Job-shop Scheduling Problem”. In: *Computers Operations Research* 35.10 (2008). Part Special Issue: Search-based Software Engineering, pp. 3202–3212. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2007.02.014. URL: https://www.sciencedirect.com/science/article/pii/S0305054807000524.

[35] R Ramasesh. “Dynamic job shop scheduling: A survey of simulation research”. In: *Omega* 18.1 (1990), pp. 43–57. ISSN: 0305-0483. DOI: https://doi.org/10.1016/0305-0483(90)90017-4. URL: https://www.sciencedirect.com/science/article/pii/0305048390900174.

[36] Daniel Rossit and Fernando Tohmé. “Scheduling research contributions to Smart manufacturing”. In: *Manufacturing Letters* 15 (2018). Industry 4.0 and Smart Manufacturing, pp. 111–114. ISSN: 2213-8463. DOI: https://doi.org/10.1016/j.mfglet.2017.12.005. URL: https://www.sciencedirect.com/science/article/pii/S2213846317300871.

[37] Daniel Rossit and Fernando Tohmé. “Scheduling research contributions to Smart manufacturing”. In: *Manufacturing Letters* 15 (2018), pp. 111–114.

[38] Guillaume Sartoretti et al. “PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning”. In: *IEEE Robotics and Automation Letters* 4.3 (2019), pp. 2378–2385. DOI: 10.1109/LRA.2019.2903261.

[39] Sven Schulz, Janis S. Neufeld, and Udo Buscher. “A multi-objective iterated local search algorithm for comprehensive energy-aware hybrid flow shop scheduling”. In: *Journal of Cleaner Production* 224 (2019), pp. 421–434. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2019.03.155. URL: https://www.sciencedirect.com/science/article/pii/S0959652619308467.

[40] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. “A review on bilevel optimization: From classical to evolutionary approaches and applications”. In: *IEEE Transactions on Evolutionary Computation* 22.2 (2017), pp. 276–295.

[41] Carlos Soto et al. “Solving the multi-objective flexible job shop scheduling problem with a novel parallel branch and bound algorithm”. In: *Swarm and evolutionary computation* 53 (2020), p. 100632.

[42] Roni Stern et al. “Multi-agent pathfinding: Definitions, variants, and benchmarks”. In: *Proceedings of the International Symposium on Combinatorial Search*. Vol. 10. 1. 2019, pp. 151–158.

[43] Theodoros Stouraitis et al. “Online hybrid motion planning for dyadic collaborative manipulation via bilevel optimization”. In: *IEEE Transactions on Robotics* 36.5 (2020), pp. 1452–1471.

[44] Tao Sun, Peter B Luh, and Min Liu. “Lagrangian relaxation for complex job shop scheduling”. In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE. 2006, pp. 1432–1437.

[45] V Suresh and Dipak Chaudhuri. “Dynamic scheduling—a survey of research”. In: *International journal of production economics* 32.1 (1993), pp. 53–63.

[46] Pavel Surynek. “Problem Compilation for Multi-Agent Path Finding: a Survey”. In: ().

[47] S.A. Torabi, B. Karimi, and S.M.T. Fatemi Ghomi. “The common cycle economic lot scheduling in flexible job shops: The finite horizon case”. In: *International Journal of Production Economics* 97.1 (2005), pp. 52–65. ISSN: 0925-5273. DOI: https://doi.org/10.1016/j.ijpe.2004.05.005. URL: https://www.sciencedirect.com/science/article/pii/S0925527304002208.

[48] Guangchen Wang et al. “Energy-efficient distributed permutation flow shop scheduling problem using a multi-objective whale swarm algorithm”. In: *Swarm and Evolutionary Computation* 57 (2020), p. 100716. ISSN: 2210-6502. DOI: https://doi.org/10.1016/j.swevo.2020.100716. URL: https://www.sciencedirect.com/science/article/pii/S2210650220303692.

[49] Yilun Wang and Qianwen Zhu. “A Hybrid Genetic Algorithm for Flexible Job Shop Scheduling Problem With Sequence-Dependent Setup Times and Job Lag Times”. In: *IEEE Access* 9 (2021), pp. 104864–104873. DOI: 10.1109/ACCESS.2021.3096007.

[50] Jin Xie et al. “Review on flexible job shop scheduling”. In: *IET Collaborative Intelligent Manufacturing* 1.3 (2019), pp. 67–77.

[51] Hegen Xiong et al. “A survey of job shop scheduling problem: The types and models”. In: *Computers Operations Research* 142 (2022), p. 105731. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2022.105731. URL: https://www.sciencedirect.com/science/article/pii/S0305054822000338.

[52] Bing Yan, Mikhail A Bragin, and Peter B Luh. “Novel formulation and resolution of job-shop scheduling problems”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3387–3393.

[53] Maziar Yazdani et al. “Optimizing the sum of maximum earliness and tardiness of the job shop scheduling problem”. In: *Computers Industrial Engineering* 107 (Mar. 2017). DOI: 10.1016/j.cie.2017.02.019.

[54] Yingchen Yu. “A research review on job shop scheduling problem”. In: *E3S Web of Conferences*. Vol. 253. EDP Sciences. 2021, p. 02024.

[55] Cong Zhang et al. “Learning to Dispatch for Job Shop Scheduling via Deep Reinforcement Learning”. In: *Advances in Neural Information Processing Systems*. Ed.

by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1621–1632. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/11958dfee29b6709f48a9ba0387a24 Paper.pdf.

[56] Fangfang Zhang et al. “Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling”. In: *ieee transactions on cybernetics* 51.4 (2020), pp. 1797–1811.

[57] Fangfang Zhang et al. “Surrogate-assisted evolutionary multitask genetic programming for dynamic flexible job shop scheduling”. In: *IEEE Transactions on Evolutionary Computation* 25.4 (2021), pp. 651–665.

[58] Jian Zhang et al. “Review of job shop scheduling research and its new perspectives under Industry 4.0”. In: *Journal of Intelligent Manufacturing* 30 (2019), pp. 1809–1830.

[59] Sicheng Zhang et al. “Multi-objective optimisation in flexible assembly job shop scheduling using a distributed ant colony system”. In: *European Journal of Operational Research* 283.2 (2020), pp. 441–460. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2019.11.016. URL: https://www.sciencedirect.com/science/article/pii/S0377221719309270.

[60] Xing Zhao, Peter B Luh, and Jihua Wang. “Surrogate gradient algorithm for Lagrangian relaxation”. In: *Journal of optimization Theory and Applications* 100 (1999), pp. 699–712.

[61] MengChu Zhou, Hua-Sheng Chiu, and H Henry Xiong. “Petri net scheduling of FMS using branch and bound method”. In: *Proceedings of IECON’95-21st Annual Conference on IEEE Industrial Electronics*. Vol. 1. IEEE. 1995, pp. 211–216.

[62] Zhenwei Zhu and Xionghui Zhou. “An efficient evolutionary grey wolf optimizer for multi-objective flexible job shop scheduling problem with hierarchical job precedence constraints”. In: *Computers Industrial Engineering* 140 (2020), p. 106280. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2020.106280. URL: https://www.sciencedirect.com/science/article/pii/S0360835220300140.

[63] Mortaza Zolfpour-Arokhlo et al. “Modeling of route planning system based on Q value-based dynamic programming with multi-agent reinforcement learning algorithms”. In: *Engineering Applications of Artificial Intelligence* 29 (2014), pp. 163–177.