Modélisation et optimisation des Hoist Scheduling Problems

Jianguang Feng

université
PARIS-SACLAY

NNT : 2017SACLC043

CentraleSupélec

Thèse de Doctorat
de
Northwestern Polytechnical University
et de
l'Université Paris-Saclay
Préparée à CentraleSupélec

ÉCOLE DOCTORALE N°573
Interfaces : approches interdisciplinaires / fondements, applications et innovation

Spécialité de doctorat : Sciences et technologies industrielles

Par

Monsieur Jianguang FENG

Modeling and Optimization for Hoist Scheduling Problems

Thèse présentée et soutenue à Xi'an, le 24 août 2017

Composition du Jury :

M. Feng WU	Professeur, Xi'an Jiaotong University	Président du Jury
M. Antoine JOUGLET	Maître de conférences HDR, Université de Technologie de Compiègne	Rapporteur
M. Hua LI	Professeur, Xidian University	Rapporteur
M. Vincent MOUSSEAU	Professeur, CentraleSupeléc, Université Paris-Saclay	Examinateur
M. Chengbin CHU	Professeur, CentraleSupeléc, Université Paris-Saclay	Directeur de thèse
M. Ada CHE	Professeur, Northwestern Polytechnical University	Co-directeur de thèse

Acknowledgement

I am honored to have this opportunity to express my gratitude to those who helped, supported and accompanied me during my Ph.D. study.

This thesis was supported by a co-tutelage program between CentraleSupélec, Université Paris-Saclay (France) and Northwestern Polytechnical University (China). I would like to thank the Laboratoire Génie Industriel (LGI) of CentraleSupélec and the School of Management of Northwestern Polytechnical University for providing me this wonderful opportunity and good working conditions, and the China Scholarship Council for financially supporting my research in France.

I am grateful to the members of my thesis jury, to Professors Antoine JOUGLET (Université de Technologie de Compiègne) and Hua LI (Xidian University) for spending their valuable time reviewing this thesis and producing review reports, to Professors Vincent MOUSSEAU (CentraleSupélec, Université Paris-Saclay) and Feng WU (Xi'an Jiaotong University) for having accepted to take part in the jury to evaluate this work. Thank for their insightful comments, constructive suggestions and fruitful remarks they provided in their reports and during the defense.

Particularly, I would like to offer my sincerest gratitude to my two supervisors, Prof. Ada CHE and Prof. Chengbin CHU, for giving me the opportunity of Ph.D. study, for teaching me academic research skills, and for helping me finish this thesis. I have learned a lot from them, which I benefit not only for research works, but also for my career life.

Prof. CHU always took time out of his busy schedule to help me whenever I came up against difficulties. He can give me novel and insightful ideas each time we have a discussion. When I write this thesis, he read the manuscript carefully and corrected the typos word by word. His rigorous attitude to knowledge and minded guidance deeply impressed me. Without his help, you could not see this thesis as what it is right now. Many thanks to Prof. CHE for his continuous support throughout my four-year Ph.D. study, after having supervised my three-year master research. I benefited a lot from his profound scientific knowledge. He guided me into this research area and teach me how to do deal with a problem hand by hand. His rigorous attitude and insightful perspectives deeply influenced me. His positive attitude encouraged me to overcome the difficulties in both research and life.

I would also like to express my gratitude to my former and present colleagues in CentraleSupélec and Northwestern Polytechnical University. Special thanks go to Yu CAO,

Jie LIU, Shouyu MA, Jing PENG, Sheng YU and Lei ZHAO, who were always willing to give their warm helping hands when I lived in Paris.

Last but not least, I would like to show special thanks to my family. Thank my parents for their unselfish love and great confidence in me all the time. I am also very grateful to my sisters for taking care of the whole family when I was outside. They are always standing by me to overcome difficulties.

Jianguang FENG
24 August, 2017

Abstract

This thesis studies hoist scheduling problems (HSPs) arising in automated electroplating lines. In such lines, hoists are often used for material handing between tanks. These hoists play a crucial role in the performance of the lines and an optimal schedule of the hoist operations is a key factor in guaranteeing product quality and maximizing productivity. We focus on extended lines (i.e. with multi-function and/or multi-capacity tanks) with a single hoist. This research investigates three hoist scheduling problems: robust optimization for cyclic HSP, dynamic jobshop HSP in extended lines and cyclic jobshop HSP in extended lines.

We first study the robust optimization for a cyclic HSP. The robustness of a cyclic hoist schedule is defined in terms of the free slacks in hoist traveling times. A bi-objective mixed-integer linear programming (MILP) model is developed to optimize the cycle time and the robustness simultaneously. It is proved that the optimal cycle time strictly increases with the robustness, thus there is an infinite number of Pareto optimal solutions. We established lower and upper bounds of these two objectives. Computational results on several benchmark instances and randomly generated instances indicate that the proposed approach can effectively solve the problem.

We then examine a dynamic jobshop HSP with multi-function and multi-capacity tanks. We demonstrate that an existing model for a similar problem can lead to sub-optimality. To deal with this issue, a new MILP model is developed to generate an optimal reschedule. It can handle the case where a multi-function tank is also multi-capacity. Computational results on instances with and without multi-function tanks indicate that the proposed model always yields optimal solutions, and is more compact and effective than the existing one.

Finally, we investigate a cyclic jobshop HSP with multi-function and multi-capacity tanks. An MILP model is developed for the problem. The key issue is to formulate the time-window constraints and the tank capacity constraints. We adapt the formulation of time-window constraints for a simpler cyclic HSP to the jobshop case. The tank capacity constraints are handled by dealing with the relationships between hoist moves so that there is always an empty processing slot for new parts. Computational experiments on numerical examples and randomly generated instances indicate that the proposed model can effectively solve the problem.

Keywords: hoist scheduling, optimization, mixed-integer linear programming

Résumé

Dans cette thèse, nous étudions des Hoist Scheduling Problems (HSP) qui se posent fréquemment dans des lignes automatiques de traitement de surface. Dans ces lignes, des ponts roulants sont utilisés pour transporter les pièces entre les bains. Ainsi, les ponts roulants jouent un rôle essentiel dans la performance de ces lignes; et un ordonnancement optimal de leurs mouvements est un facteur déterminant pour garantir la qualité des produits et maximiser la productivité. Les lignes que nous étudions comportent un seul pont roulant mais peuvent être des lignes de base ou des lignes étendues (où des bains sont à fonctions et/ou capacités multiples). Nous examinons trois Hoist Scheduling Problems : l'optimisation robuste d'un HSP cyclique, l'ordonnancement dynamique d'une ligne étendue de type job shop et l'ordonnancement cyclique d'une telle ligne.

Pour l'optimisation robuste d'un HSP cyclique, nous définissons la robustesse comme la marge dans le temps de déplacement du pont roulant. Nous formulons le problème en programmation linéaire en nombres mixtes à deux objectifs pour optimiser simultanément le temps de cycle et la robustesse. Nous démontrons que le temps de cycle minimal augmente avec la robustesse, et que par conséquent la frontière Pareto est constituée d'une infinité de solutions. Les valeurs minimales et maximales des deux objectifs sont établies. Les résultats expérimentaux à partir de benchmarks et d'instances générées aléatoirement montrent l'efficacité de l'approche proposée.

Nous étudions ensuite un problème d'ordonnancement dynamique dans une ligne étendue de type job shop. Nous mettons en évidence une erreur de formulation dans une un modèle existant pour un problème similaire mais sans bains multi-fonctions. Cette erreur peut rendre l'ordonnancement obtenu sous-optimal voire irréalisable. Nous construisons un nouveau modèle qui corrige cette erreur. De plus il est plus compact et s'applique au cas avec des bains à la fois à capacités et à fonctions multiples. Les résultats expérimentaux menés sur des instances avec ou sans bains multi-fonctions montrent que le modèle proposé conduit toujours à une solution optimale et plus efficace que le modèle existant.

Nous nous focalisons enfin sur l'ordonnancement cyclique d'une ligne étendue de type job shop avec des bains à fonctions et capacités multiples. Nous construisons un modèle mathématique en formulant les contraintes de capacité du pont roulant, les intervalles des durées opératoires, et les contraintes de capacité des bains. Nous établissons également des contraintes valides. Les expériences réalisées sur des instances générées aléatoirement

montrent l'efficacité du modèle proposé.

Mots clés : hoist scheduling, optimisation, programmation linéaire mixte en nombres entiers

Contents

List of Figures

List of Tables

1 Introduction

Contents

1.1 Background and relevance

This thesis focuses on production scheduling where material handling devices play a key role in the manufacturing process. To be more specific, we are particularly interested in problems commonly called hoist scheduling problem or simply HSP in the literature. As we can see hereafter, effectively solving such problems can contribute very much to the performance of the considered production systems. Despite the fact that many efforts have been made during the last decades, much academic research is still needed to deal with some complex but relevant situations.

In such manufacturing systems, the transformation of raw materials into semi-finished and then finished products involves a sequence of processing stages which are executed at different workstations or processing units. After a processing stage of a part is completed, the part should be transported to the next workstations to pursue its subsequent processing stages. The transportation operations from one workstation to another are often performed by material

handling devices for technical, economic and security reasons. Such material handling devices may be Automatic Guided Vehicles (AGVs), traveling cranes/hoists, and mobile or rotational robots, according to operating environments or specific technical requirements. Figure 1.1 illustrates an automatic electroplating line with five processing tanks and a material handling hoist.

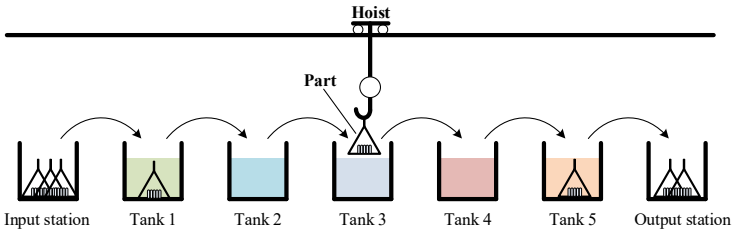

Figure 1.1: Illustration of a basic automatic electroplating line

In such an electroplating line, the raw parts waiting to be processed are stored in the input station, while finished parts are stored in the output station. Each tank contains a specific electroplating solution used for a particular processing stage. To complete the processing, each raw part is first unloaded from the input station, then successively treated in a sequence of tanks in a prescribed order and finally loaded on the output station. The hoist is in charge of the material handling operations between tanks and stations. To make sure that the finished products conform to the specifications, there may be various constraints on the processing time of each stage.

There are a variety of lines. In a *basic line*, each part visit exactly once each tank and each tank can process at most one part at a time. When a tank is visited more than once by a part, it is a multi-function or re-entrant tank. If a tank can process more than one part at a time, it is a multi-capacity tank. Furthermore, the parts can be identical or different. A production line including multi-function and/or multi-capacity tanks are called an *extended line*. The production can be organized in a *cyclic* mode or a *non-cyclic* one. In a cyclic production, the tanks and the hoist perform a same set of operations repeatedly. This production mode is particular relevant in mass production. The length of the interval between two successive repetitions is called *cycle time*. The shorter the cycle time is, the higher is the productivity.

Material handling devices are used for technical, economic as well as safety/security reasons. On the one side, using material handling devices can not only improve the throughput and reduce the production cost, but also guarantee the quality of parts. On the other side, safety and security in production have become very important concerns of enterprises. Using material handling devices in industrial manufacturing can limit harms to workers, especially in environments with hazardous materials, as in electroplating lines involving chemical processes, and boring and repetitive tasks.

From economic point of view, it has been reported that material handling operations have a significant influence in production, especially for high-value products. In this context, a well-planned schedule of material handling operations can significantly reduce manufacturing cost

and consequently improve profits. Tompkins, White, Bozer, and Tanchoco (2010) indicated that material handling can account for as high as 50% of the total operating cost within manufacturing. Kumar (1994) reported that as high as 20% reduction in mean part waiting time and 50% improvement in standard deviation of cycle time can be achieved by scheduling. For a semiconductor manufacturer, Dawande, Geismar, Pinedo, and Sriskandarajah (2010) showed that a 4.19% average increase in throughput and about $1.34 million in weekly revenue can be attained by scheduling a dual-gripper robot in a 15 - stage cell. Due to the potential economic gain, research on effective and efficient scheduling of material handling operations has attracted much attention from scholars and practitioners. The objective is mainly to maximize the productivity by effectively coordinating the material handling operations and processing operations.

Because of random events in production systems, the robustness of the schedules is also a major concern of managers. We also study robustness issues in this thesis, especially randomness related to the traveling times of the hoist.

1.2 Research Methodology

As we can see in the next chapter, there is a large variety of hoist scheduling problems, according to the production mode, the number of part-types, and characteristics of the tanks. But the majority of them have been proven to be NP-hard (Lei & Wang, 1989). In the literature, a large number of algorithms and approaches of different nature have been used to formulate and solve hoist scheduling problems, such as branch-and-bound algorithms, Mixed-Integer Linear Programming (MILP), heuristics, and meta-heuristics.

In this thesis, we mainly focus on mathematical formulation, in particular in forms of MILP models. The challenge in formulating these problems into MILP models is to appropriately define decision variables and formulate the constraints. It has been demonstrated that models with different ways of defining variables and expressing constraints can yield very different performances. On the other hand, there are many techniques to improve the performance of MILP models by adding cuts and reformulating constraints (Cornuéjols, 2008). Among these techniques, one can find valid inequalities that can tighten bounds. A valid inequality is added to the initial model by exploring properties of the problems to reduce the solution space or even obtain its convex hull. Furthermore, such inequalities can be integrated into an MILP model without influencing the remaining part of the model. This makes it possible to use sophisticated inequalities in similar MILP models.

To use MILP approaches for hoist scheduling problems, the model usually needs slight modification when the problem slightly varies. In general, to handle complex problems, the MILP models for simpler but similar problems can provide a good basis. For example, in the work of Phillips and Unger (1976), the authors extended their MILP model initially for a basic line to handle multi-function tanks by integrating the constraints that avoid collisions

related to multi-function tanks into the original model without modifying other constraints. Furthermore, MILP models can be solved by general mathematical programming solvers. Nowadays many general solvers, whether commercial or academic, are available and becoming more and more powerful, for Linear Programs (LP), Integer Programs (IP) and MILP models. State-of-the-art solvers supporting MILP models contain commercial software packages such as CPLEX (IBM ILOG, 2009), GUROBI (Gurobi Optimization, 2016), and open source software such as SCIP (Achterberg, 2009) and so on. On the one hand, these general solvers can be used as black boxes without considering the implementation details; on the other hand, many modern solvers also support interfaces to communicate with the user during solving processes by querying incumbent solutions and adding cuts, etc.

1.3 Contributions

Previous works mainly focus on HSPs in basic lines, or in the extended lines but with identical parts. This thesis studies three types of HSPs extended from basic lines and extended lines. They are respectively: robust cyclic hoist scheduling problem, dynamic hoist scheduling problem in extended lines, and cyclic robotic jobshop scheduling problem in extended lines. We first give an overview of the literature. Then the considered hoist scheduling problems are addressed in the following chapters, respectively. The following results are achieved:

(1). A method is proposed to measure the robustness of a cyclic hoist schedule subject to delays in hoist traveling times. A bi-objective MILP model is formulated to minimize the cycle time and maximize the robustness. It is proved that the cycle time strictly increases with the robustness, thus a Pareto optimal solution can be obtained by solving a single-objective HSP to minimize the cycle time for a given value of robustness or maximize the robustness for a specific cycle time.

(2). A dynamic hoist scheduling problem with multi-capacity and multi-function tanks is addressed. We point out a flaw in an article which may lead to sub-optimal solutions or fail to find a feasible solution. An MILP model is developed to improve and extend this work. This new model can furthermore handle the scenario where a multi-function tank is also multi-capacity.

(3). An MILP model is formulated for a cyclic robotic jobshop scheduling problem with multi-capacity and multi-function tanks. To handle the multi-capacity tanks used by multiple types of parts, the formulation for cyclic hoist scheduling with a single part-type is extended to the multiple part-types situation. For each loaded hoist move, the number of parts being processed in a tank at the start of a cycle, arriving at the tank before the move and leaving the tank before the move are calculated. Based on this, the capacity constraints for tanks are formulated.

1.4 Outline

The remainder of this thesis is organized as follows.

Chapter 2 provides a literature review of previous contributions related to hoist scheduling problems. The description is divided into several sections according to the features of the considered problem, including the type of production lines (basic lines, extended lines with multi-function tanks and/or multi-capacity tanks), the number of hoists (single-hoist and multi-hoist), the degrees of cyclic schedules, the number of part-types and the production mode (cyclic and non-cyclic). Existent works and results are analyzed, and the research gap is identified.

Chapter 3 deals with the robust optimization for cyclic hoist schedules considering delays in hoist traveling. The influence of the delays in loaded and unloaded hoist moves are analyzed. The measure of the robustness of a cyclic hoist schedule is defined as its ability to remain stable in the presence of such delays. With such a definition, we propose a method to measure the robustness of a cyclic hoist schedule. A bi-objective MILP model is developed, which aims to simultaneously optimize the cycle time and the robustness. We investigate the relationship between the optimal cycle time and the robustness. Furthermore, we derive the so-called ideal and nadir points that define the lower and upper bounds for the objective values of the Pareto front, respectively. It is proved that a Pareto optimal solution can be obtained by solving a single-objective HSP to minimize the cycle time for a given value of robustness or maximize the robustness for a specific cycle time. Computational experiments on several benchmark instances and randomly generated instances are conducted to evaluated the proposed approach.

Chapter 4 deals with a dynamic hoist scheduling problem with multi-capacity and multi-function tanks, where a multi-function tank can be multi-capacity. Parts to be processed are of different types and dynamically arrive at the input station. Once a rescheduling is triggered according to the current state of the system, a new schedule should be generated to schedule the newly arriving parts and reschedule the parts in process so that the makespan is minimized. An MILP model is developed to generate such an optimal reschedule, which improves an existing model in the literature in several aspects. The proposed model is solved by optimization software package CPLEX. A large number of instances with and without multi-function tanks are used to evaluate the proposed model and compare it with an existent model.

Chapter 5 studies a cyclic robotic jobshop scheduling problem with multi-capacity tanks and multi-function tanks. We consider cyclic schedules with different types of parts. We extend the time-window formulation for cyclic hoist scheduling with identical parts to the multiple part-types situation. For each loaded hoist move, the number of parts being processed in a tank at the start of a cycle, those arriving at the tank and those leaving the tank before the move are calculated. To ensure the capacity of each tank is respected, at least one empty processing unit must be available at the moment a part arrives at the tank. An MILP model is developed for the problem. To evaluate the proposed model, a set of instances are randomly

generated and solved.

Finally, Chapter 6 concludes this work, discusses the limitations of the present research and suggests potentially promising directions for future research.

2

Problem Description and State of the Art

Contents

In this chapter, we describe in detail the hoist scheduling problems under various settings. It describes at the same time the notions and terms that will be used throughout the thesis. As we can see, there are a large variety of models possible. We then give a state-of-the-art overview of the numerous works that have been done in the last four decades. This overview shows that despite the large number of studies, much work is still needed to address complicated situations, especially in terms of mathematical formulation.

2.1 Hoist scheduling problem and diversity

In an automatic electroplating line, the processing from raw parts to semi-finished or final products involves a sequence of processing stages (Phillips & Unger, 1976; Levner, Kats, & Levit, 1997; Lee, Lei, & Pinedo, 1997; Levner, Kats, Alcaide López de Pablo, & Cheng, 2010; Manier & Lamrous, 2008; Che & Chu, 2007b; Yan, Che, Yang, & Chu, 2012). The line consists of a series of tanks, an input station, an output station and one or more hoists. Figure 1.1 in chapter 1, shown again and explained in detail hereafter for the sake of self-consistency, illustrates an automatic electroplating line with five processing tanks and a material handling hoist (see Figure 2.1).

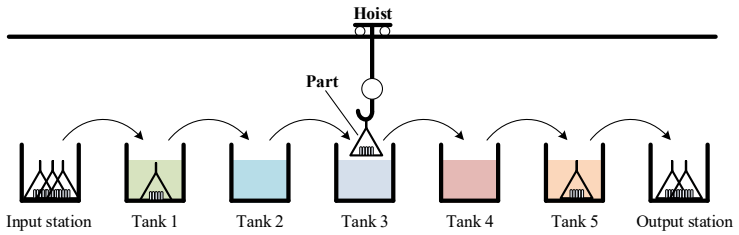

Figure 2.1: Illustration of a basic automatic electroplating line

The input station is used for holding the raw parts to be processed, while the output station for completed or finished parts. Each tank contains a specific electroplating solution used for a particular processing stage. To complete the processing, each raw part is firstly unloaded from the input station, then successively treated in a sequence of tanks in a prescribed order and finally loaded on the output station. The hoists are in charge of the material handling operations between tanks. To guarantee the quality of the parts, the processing time at each stage may be subject to restrictions. In general, the processing time at each stage of each part

is bounded with a predefined interval, called time-window. That is, the actual processing time must be no less than a minimum amount and no more than a maximum one. Otherwise, the part will become defective due to insufficient treatment or excessive exposure to chemical solution. To avoid excessive exposure to open air, there is no storage buffer between tanks. Thus, after the processing in a tank is completed, the part should be directly transferred to the next tank without any delay. To complete such a transfer operation, the hoist needs to unload the part from the origin tank, transports it to the destination tank, and then loads the part into it. The required transportation times are not negligible compared to the processing times in tanks. Once a part is unloaded from the input station, before it arrives at the output station as a finished part, it will be either being processed in a tank or being held by a hoist.

As all transportation tasks are carried out by hoists, the sequence of the hoist operations is a critical factor that affects the productivity of the system. A hoist scheduling problem aims to sequence the hoist transfer operations so that the throughput is maximized while all constraints are respected. The challenge is to effectively coordinate the hoist operations and processing stages so that the resources are fully utilized and related losses are cut down.

An intuitive hoist schedule is that the hoist waits at the tank after loading a part into it until the processing is completed, and then unloads the part and transfers it to the next tank. In such a schedule, at most one tank is used at any time and the others are idle, the productivity thus is low. In order to maximize the productivity, one would like all resources (tanks and hoists) to work as much as possible. In a sophisticated schedule, multiple parts are processed simultaneously in order to improve the use rate of tanks and hoists. For instance, Figure 2.1 shows a line where three parts are processed simultaneously in the production line. In this situation, after loading a part into a tank, the hoist, instead of waiting at the tank, may travel to another tank to execute the next transportation operation.

Many variants of hoist scheduling problems have been addressed in the literature. To distinguish hoist scheduling problems from one another and to classify them, Manier and Bloch (2003) developed a four-field notation system according to production modes, physical parameters, logical parameters and criteria that are associated with the problems. New paragraphs presents this diversity.

2.1.1 Basic vs. extended lines

In a *basic line* (Liu, Jiang, & Zhou, 2002), there is a one-to-one correspondence between the processing stages of a part and the tanks, and each tank can handle at most one part at a time. In other words, a tank is responsible for a specific processing stage, and each part will visit each tank once during its whole processing. Figure 2.1 illustrates a basic automatic electroplating line.

Many practical systems involve complex configurations such as re-entrant tanks and tanks with parallel processing slots. A re-entrant tank, also called a *multi-function* tank, is visited more than once by a part during its processing route; while a tank with parallel processing

slots, also called a *multi-capacity* tank, can treat multiple parts simultaneously and each processing slot can handle at most one part at a time. A production line with multi-function tanks and/or multi-capacity tanks is usually called *an extended line* in the literature (Che & Chu, 2005a; Che & Chu, 2007b; Liu et al., 2002). An extended line has advantages in several aspects. On the one hand, multi-function tanks can avoid deploying (or investing in) extra tanks and shorten the production line; on the other hand, multi-capacity tanks are used for stages requiring disproportionately long processing time compared with those of other stages. In a more general sense, a multi-function tank can also be multi-capacity, which is referred to as a multi-function multi-capacity tank. However, the presence of multi-function tanks and multi-capacity tanks makes the corresponding hoist scheduling problem more challenging to deal with.

2.1.2 Cyclic vs. non-cyclic production

Automatic electroplating lines can operate in cyclic mode (Phillips & Unger, 1976; Chen, Chu, & Proth, 1998; Liu et al., 2002; El Amraoui, Manier, El Moudni, & Benrejeb, 2013b; Lei & Wang, 1994; Manier & Lamrous, 2008) or in non-cyclic mode (Yih, 1994; Zhao, Fu, & Xu, 2013b; Sun, Lai, Lam, & So, 1994; Chauvet, Levner, Meyzin, & Proth, 2000; Lamothe, Correge, & Delmas, 1994; Yan, Che, Cai, & Tang, 2014). For non-cyclic scheduling, the parts to be processed are waiting at the input station beforehand or dynamically arrive at the input station. The objective is usually to sequence the parts and hoist operations so that the makespan (the time that the last part arrives at the output station) is minimized. This criterion is equivalent to maximizing the throughput. As for cyclic scheduling, the production system periodically repeats a fixed sequence of operations. Each repetition of the operations is called a *cycle*. At the end of a cycle, the system will reach the same state as at the start of the cycle. The duration of a complete cycle is defined as *cycle time*. The reverse of the cycle time represents the throughput rate or the productivity of the system. Thus, minimizing the cycle time is equivalent to maximizing the throughput. Due to its simplicity of implementation and ease of management, especially in mass production, which is often the case, cyclic scheduling has attracted extensive attention from researchers and practitioners.

One important character of cyclic robotic scheduling is that, the same number of parts enter and leave the system within each cycle. With this feature, the parts to be processed can be represented by a fixed ratio according to the numbers of parts of different types (Zhao, Fu, & Xu, 2013a; Lei, Che, & Chu, 2014). The parts are partitioned into multiple identical small sets, called Minimal Part Set (MPS), according to this ratio, where minimal means there is no common divisor among the elements of such a set. For example, suppose that the demands for parts of types A, B and C are 200, 200, and 400 respectively, they follow the ratio A : B : C = 1 : 1 : 2. Thus each MPS contains four parts: one part of type A, one part of type B and two parts of type C. The *degree* of a cyclic schedule is defined as the number of repetitions of the MPS in a cycle (Lei & Wang, 1994; Leung, Zhang, Yang, Mak, & Lam,

2004). It is worth noting that in each cycle, the newly entering parts and the completed parts are not necessarily the same. The cyclic schedule with multiple MPSs introduced in a cycle are commonly referred to as a multi-degree cyclic schedule (Che & Chu, 2009; Che, Zhou, Chu, & Chen, 2011b; Li & Fung, 2014; Li, Chan, & Chung, 2015; Lei & Wang, 1994; Leung et al., 2004) or a multi-cyclic schedule (Zhou, Che, & Yan, 2012).

An MPS may contains one part or multiple parts according to different configuration. On the one hand, when all parts to be processed are identical, i.e., a flowshop is considered, an MPS contains only one part and the degree of cyclic schedules is exactly equal to the number of parts introduced within a cycle. As the simplest case, a one - degree cyclic schedule is commonly referred to as a *simple cyclic schedule* (Shapiro & Nuttle, 1988). On the other hand, when different parts are processed, i.e., a jobshop is considered, an MPS is composed of several parts. In this case, one - degree cyclic schedules are usually considered, where an MPS is unloaded from the input station in a fixed sequence defined by the schedule, and a not-necessarily-the-same MPS is loaded onto the output station during each cycle. Therefore, such a cycle is periodically executed until the demands for parts are satisfied. Note that the entering sequence of the parts in MPS into the production line will affect the entire schedule of the production.

2.1.3 Diversity related to hoist traveling times

In practical systems, the hoist travel times can be influenced by the layout of the production line, such as the position of processing tanks and input/output stations. These times can be represented by an adjacent matrix. In the literature, three types of matrices are usually considered (Dawande, Geismar, Sethi, & Sriskandarajah, 2007): additive, constant, and Euclidean. For the additive situation, the times required for the hoist to travel between any two adjacent tanks are given, and the hoist travel time between two non-adjacent tanks can be calculated by adding up the travel times between the adjacent tanks located between them. For the constant matrices, the hoist travel time between any two tanks, whether adjacent or non-adjacent, are given. In the Euclidean situation, the hoist travel time between any two tanks can be different from one another, and they are assumed to satisfy triangle inequalities. Besides, other situations have also been considered by some works such as Kats and Levner (1998, 2009), Feng, Che, and Wang (2014), where no particular assumptions are made about these travel times.

2.1.4 Representation of hoist schedules

As the material handling operations are executed by the hoists, a schedule can be uniquely represented by a sequence of all related hoist transfer operations. To intuitively demonstrate the physical positions of the hoist and whether a tank is handling a part or not as time goes on, a hoist schedule can be represented by a time-way diagram (Liu et al., 2002), similar to a Gantt

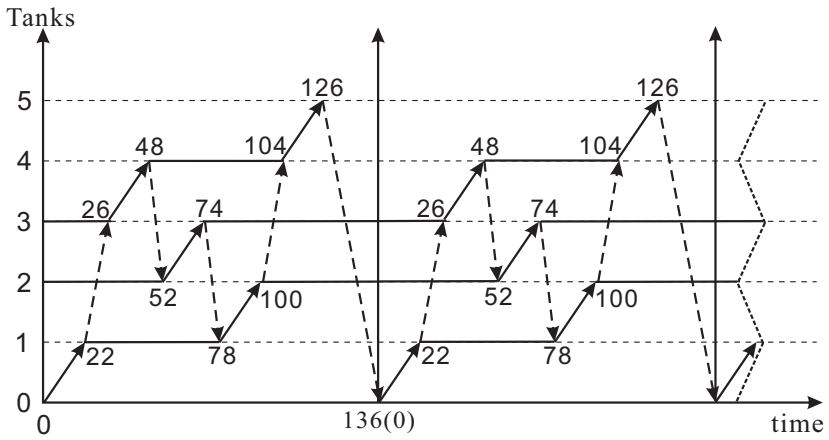

Figure 2.2: Illustration of the time-way diagram for a cyclic hoist schedule

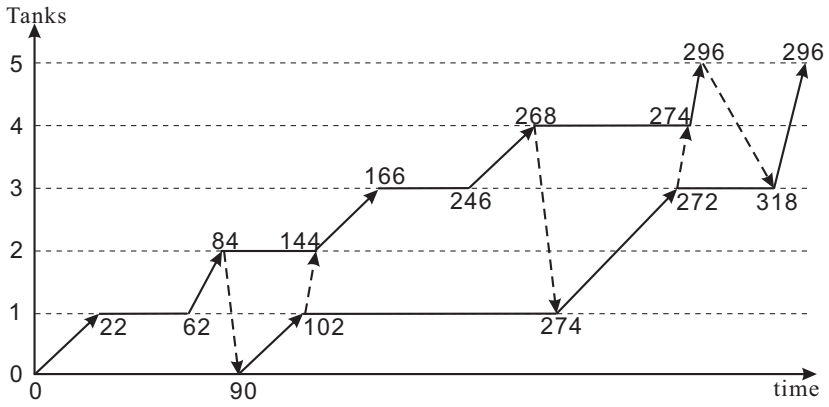

Figure 2.3: Illustration of the time-way diagram for a non-cyclic hoist schedule

chart in classical machine scheduling. Such a time-way diagram is depicted in a rectangular coordinate system. The horizontal axis is a time line while the vertical axis represents the tanks. A part's processing in a tank is denoted by a horizontal solid segment, and loaded and unloaded hoist traveling operations are represented by solid and dashed arrows respectively, where the arrows indicate the traveling direction of the hoist. Figure 2.2 and Figure 2.3 illustrate the time-way diagrams for a cyclic and a non-cyclic hoist schedules, respectively.

In Figure 2.2, a cycle starts by unloading a part from the input station. After loading the part into tank 1, the hoist travels to tank 2 to unloads a part, and then travels to tank 3 to loads the part into it. Since then, the hoist waits at tank 3 for the completion of the part's processing. Once the processing is completed, the hoist transfers the part to tank 4 for its next operation. The hoist travels back to tank 1 to unload the part that is loaded at the start of the cycle and transport it to tank 3. After this, the hoist returns to tank 4 and unloads a part from it. When all processing stages of a part has completed, the part is moved to the output station and the hoist returns to the input station to start a new cycle. Note that one part is in process in tank 2 at the start of a cycle and two parts are handled simultaneously during a cycle.

The non-cyclic hoist schedule in Figure 2.3 can be interpreted in a similar way except that no cycle can be defined.

2.1.5 Summary

In the past four decades, the hoist scheduling problem and many variants have been widely studied by researchers from academy and industry. Many models and algorithms of various nature, both exact and approximate, have been proposed for such problems. Due to the rich literature, a number of survey papers have reviewed the contributions on hoist scheduling and related problems from different perspectives. Lee et al. (1997) presented a brief review on cyclic hoist scheduling and related scheduling in robotic cells. Levner et al. (2010) reviewed the computational complexity results of several kinds of cyclic scheduling problems. Kats and Levner (2012) reviewed various methods based on prohibited intervals and proposed for cyclic hoist scheduling problems with zero-width processing time-windows, also called no-wait systems. Crama, Kats, van de Klundert, and Levner (2000) surveyed the modeling and complexity results of cyclic scheduling problems in robotic flowshops. More works on hoist scheduling and related robotic scheduling can be found in Brauner (2008), Dawande et al. (2007), Dawande et al. (2010), Lee (2008) and the references therein.

The following sections review the works on HSPs under different settings. The presentation is divided into four sections. Section 2.2 describes the works on cyclic hoist scheduling with time-window constraints. It covers the models and algorithms proposed for the cyclic hoist problems raising in both basic and extended lines. Section 2.3 reviews the research works related to non-cyclic hoist scheduling problems. It covers the models and algorithms proposed for dynamic and static hoist scheduling in both basic and extended lines. Section 2.4 reviews the literature related to hoist scheduling with zero-width time-windows. The research gap and the motivation of this research are discussed in Section 2.5.

2.2 Cyclic hoist scheduling problems

In mass production environments, a large number of products of each type must be produced. In this case, the production can be implemented in a cyclic way. Cyclic hoist scheduling problems can thus be widely encountered in practice and have attracted extensive attention from researchers. In this section, we review the works related to cyclic hoist scheduling. We first introduce the study on basic lines, including simple cyclic schedules, multi-degree cyclic schedules with identical parts and cyclic schedules with multiple types of parts. Then studies on extended lines with multi-function and/or multi-capacity tanks are addressed. Finally, the works on cyclic multi-hoist scheduling are surveyed.

2.2.1 Basic scheduling problems

In hoist scheduling, we have to determine an optimal sequence of hoist operations. But because of the upper bounds of the processing times, not all sequences of hoist operations are feasible. That is why a lot of works have been done to develop methods to check the

feasibility and to calculate the starting time of each move so that the cycle time is minimized for a given sequence of hoist operations. Such a problem is called *basic scheduling problem*. Lei (1993) introduced a search procedure to determine the optimal integer starting times of the hoist moves for a basic scheduling problem, which can also be used to check the feasibility of the sequences of hoist moves for a given cycle time. Levner and Kats (1998) formulated the basic scheduling problem as a parametric critical path problem and solved it by a modified Bellman-Ford algorithm. Chen et al. (1998) considered in parallel the same problem and transformed it into cycle time evaluation problems in bi-valued graphs and proposed a polynomial algorithm of time complexity $O(n^4m^2)$, where n and m are the number of vertices and the number of arcs of the graph. Even though it has a higher worse-case complexity than that of Levner and Kats (1998), but is computationally more effective. Kats, Lei, and Levner (2008) showed that the multi-degree basic scheduling problem with setup times is equivalent to a parameter critical path problem, and proposed a strongly polynomial algorithm. Kats and Levner (2011b) studied a 2 - degree scheduling problem and proposed a polynomial algorithm of complexity $O(m^8 \log m)$, where m is the number of tanks. The authors further developed an improved algorithm of reduced complexity $O(m^8)$ for the same problem (Kats & Levner, 2011a). Levner, Kats, and De Pablo (2007) suggested a parametric critical path algorithm to optimize the multi-hoist scheduling with multiple part-types. The complexity of the proposed algorithm is $O(m^4)$, where m is the number of tanks.

2.2.2 Cyclic hoist scheduling problems in basic lines

2.2.2.1 With identical parts and simple cyclic schedules

Phillips and Unger (1976) were the first to use mathematical programming for solving hoist scheduling problems. They proposed the first MILP model for a basic line with a single hoist to obtain an optimal simple cyclic schedule. The model is tested on an instance derived from a real industrial line. The instance has become a benchmark widely used in subsequent works. Later, researchers have proposed various branch-and-bound algorithms for the problem (Shapiro & Nuttle, 1988; Lei & Wang, 1994; Armstrong, Lei, & Gu, 1994; Chen et al., 1998; Yan, Chu, Yang, & Che, 2010) and heuristics or meta-heuristics (Baptiste, Legeard, & Varnier, 1992; Baptiste, Legeard, Manier, & Varnier, 1993; Spacek, Manier, & Moudni, 1999; Lim, 1997; Yan et al., 2012).

Among the diverse branch-and-bound algorithms, the main differences lie in branching rules and bounding schemes. Shapiro and Nuttle (1988) showed that a schedule can be obtained as soon as the number of parts simultaneously processed in a cycle and the precedence between the processing stages are known. Then, the objective became searching the minimum offset between two successive parts. To this end, a branch-and-bound algorithm is proposed based on enumerating the number of parts and their processing sequence at each stage. Furthermore, the algorithm is extended to handle multi-capacity tanks. Lei and Wang (1994) proposed a

time-window algorithm, which is based on a branch-and-bound procedure. The branching is carried out by enumerating the sequences of hoist moves in a cycle. In order to effectively check the feasibility of candidate schedules, the earliest and latest start times of hoist moves are calculated to rule out infeasible candidates as soon as they are identified. Armstrong et al. (1994) introduced a sequence-dependent parameter to estimate the minimal time interval between two hoist moves. They showed that this parameter can provide a good lower bound of the cycle time, and integrated it into a branch-and-bound search procedure. Chen et al. (1998) developed a branch-and-bound algorithm consisting of two nested branching procedures. The upper-level branching enumerates the part distribution at the start of a cycle, which indicates whether a part is in process or not in a tank. Once such a part distribution is obtained, the lower-level branching enumerates possible sequences of hoist moves. The maximum possible number of parts simultaneously processed in a cycle and a lower bound of the cycle time based on partial sequences of hoist moves are used to enhance the algorithm. The relaxed problems at each node of the search tree were transformed into a difference system and solved with the longest path algorithm in bi-valued graphs which are equivalent to parameterized graphs defined by Kats and Levner (2011a). Yan et al. (2010) applied the method of prohibited intervals which is usually used for the situations with zero-width time-windows (Levner et al., 1997). The objective is to enumerate the non-prohibited intervals for the cycle time. A dynamic branch-and-bound algorithm is proposed to enumerate all possible intervals for cycle time.

Both MILP method and branch-and-bound algorithms aim to obtain an optimal solution. However, due to the NP-hardness of the problem, the amount of computation time for large-sized instances may become unacceptable. To cope with this, approximation algorithms have also been investigated. Different from exact algorithms, the purpose of approximation algorithms are to obtain satisfactory or near-optimal solutions in a reasonable amount of time. Baptiste et al. (1992) and Baptiste et al. (1993) introduced a Constraint Logic Programming (CLP) approach. Spacek et al. (1999) suggested a max- and min-algebra model for the problem and introduced heuristics to handle the conflicts in the use of the tanks and the hoist. The model was also extended to multi-hoist and multi-degree situations. Lim (1997) designed a genetic algorithm (GA) to solve the problem by representing the sequences of hoist moves as chromosomes. Yan et al. (2012) developed a tabu search procedure to divide the solution space into subspaces according to the number of work-in-process parts. Furthermore, a repairing procedure is proposed to try to obtain feasible solutions out of infeasible ones.

2.2.2.2 Problems with different parts and/or multi-degree schedules

All the above works deal with simple cyclic schedules, i.e., 1-degree cyclic scheduling. It has been reported that multi-degree or multi-cyclic schedules can yield strictly higher productivity than their simple-cyclic counterparts in many situations (Lei & Wang, 1994; Kats, Levner, & Meyzin, 1999; Zhou et al., 2012). However, the related problems become more complicated,

and the modeling and optimization are much more challenging as well.

In fact, Lei and Wang (1994) remarked that their branch-and-bound algorithm can also deal with 2-degree cyclic scheduling, and an optimal 2-degree cyclic schedule for the benchmark problem proposed by Phillips and Unger (1976) was reported. For general multi-degree cases, Che et al. (2011b) proposed a branch-and-bound algorithm. The algorithm implicitly enumerates, with two search trees, the part distributions at the start of a cycle and the sequences of hoist moves, respectively. For the same problem, an MILP model was also developed and solved by Zhou et al. (2012).

In the above works, all the parts are identical, thus the problem related to the entering sequence of parts vanishes. That is, the precedence among the parts does not matter. However, this is not the case for systems with different types of parts. It is worth noting that for two-part cyclic scheduling which is a special case of general ones, whether the two parts are identical or not, their entering sequence can be neglected because of the cyclic nature (Zhou et al., 2012). Lei and Liu (2001) proposed a branch-and-bound algorithm for such a problem with two different parts in an MPS to minimize the cycle time. The processing routes of the two different parts are identical but their processing time-windows in each tank may be different. The authors also remarked that the proposed algorithm can be extended to system with multiple part-types. A different branch-and-bound approach for the problem has also been studied by Mateo and Companys (2006). El Amraoui, Manier, El Moudni, and Benrejeb (2008) developed an MILP model for the problem and extended it to the case where the hoist can wait during loaded moves which is also considered by Ng (1996), Liu et al. (2002) and Che and Chu (2007b). This work was further extended by El Amraoui, Manier, El Moudni, and Benrejeb (2012) to handle configurations such as load-unloaded buffers which is characterized by the time-window at input station, multi-function tanks and multi-capacity tanks.

As mentioned before, it is necessary to determine the part entering sequence if there are more than two part-types. For such a problem, Varnier and Jeunehomme (2000) suggested an approach to construct a feasible hoist schedule by combining the hoist schedules with identical parts in a specific way. El Amraoui, Manier, El Moudni, and Benrejeb (2013a) studied the cyclic scheduling where each part has its specific processing time-windows in the tanks. An MILP model and a GA are developed to minimize the cycle time. El Amraoui et al. (2013b) developed an MILP model similar to that proposed by Zhou et al. (2012). A heuristic based on Earliest Start Time (EST) was suggested by El Amraoui, Manier, El Moudni, and Benrejeb (2011) to handle multiple part-types. The algorithm aims to sequence the hoist moves so that a cyclic schedule is achieved. Note that the obtained schedule may be multi-degree. Lei et al. (2014) proposed a branch-and-bound algorithm composed of three nested branching procedures. The first two procedures are mainly designed for enumerating all possible entering sequences of the parts, while the third one for enumerating the sequences of hoist moves.

The above studies on cyclic scheduling in basic lines are summarized in Table 2.1. In the table, *#D* represents the degree of the cyclic schedule; *#PT* and *#H* represent the number of

part-types and the number of hoists, respectively.

Table 2.1: Studies on the cyclic HSPs in basic lines

#D	*#PT*	*#H*	*Method*	*Features*	*Reference*
1	1	1	MILP		Phillips and Unger (1976)
1	1	1	Branch-and-bound		Shapiro and Nuttle (1988)
1/2	1	1	Branch-and-bound		Lei and Wang (1994)
1	1	1	Branch-and-bound		Armstrong et al. (1996)
1	1	1	Branch-and-bound		Chen et al. (1998)
1	1	1	Branch-and-bound		Yan et al. (2010)
1	1	1	CLP		Baptiste et al. (1992)
					Baptiste et al. (1993)
$1/N$	1	$1/N$	Heuristics		Spacek et al. (1999)
1	1	1	Genetic algorithm		Lim (1997)
1	1	1	Tabu search		Yan et al. (2012)
N	1	1	Branch-and-bound		Che et al. (2011b)
N	1	1	MILP		Zhou et al. (2012)
$2/N$	1	1	Branch-and-bound		Lei and Liu (2001)
2	2	1	Branch-and-bound		Mateo and Companys (2006)
2	2	1	MILP		El Amraoui et al. (2008)
2	2	1	MILP	Load-unload buffer	El Amraoui et al. (2012)
N	N	1	Heuristics		Varnier and Jeunehomme (2000)
N	N	1	MILP and GA	Identical processing routes	El Amraoui et al. (2013a)
N	N	1	MILP	Identical processing routes	El Amraoui et al. (2013b)
N	N	1	Heuristics	Identical processing routes	El Amraoui et al. (2011)
N	N	1	Branch-and-bound		Lei et al. (2014)

2.2.3 Cyclic hoist scheduling problems in extended lines

2.2.3.1 Simple cyclic schedules with identical parts

In addition to basic lines, extended lines which consist of multi-function tanks and/or multi-capacity tanks have also been studied in the literature. In fact, Phillips and Unger (1976) also discussed multi-function tanks in their work, and suggested how to extend their MILP model to handle multi-function tanks. Shapiro and Nuttle (1988) and Lei and Wang (1994) also showed that their branch-and-bound algorithms for basic lines can be extended to deal with multi-capacity tanks. Ng (1995) pointed out the fact that forcing use of all parallel processing slots of a multi-capacity tank may worsen the performance of the system. Thus in their study, the number of actually used parallel slots of a multi-capacity tank is treated as decision variables instead of constants. The algorithm proposed by Shapiro and Nuttle (1988) was adapted by enumerating all possible numbers of parallel processing slots to use. For the problem, Che and Chu (2007b) proposed a branch-and-bound algorithm, where the branching is based on the number of actually used parallel processing slots and the sequences

of hoist moves. Lower and upper bounds of the processing slots and the maximum number of work-in-process parts are investigated and applied to enhance the algorithm. Zhou and Li (2003) dealt with a similar problem but the required times for loaded hoist moves are constant and all parallel slots of a multi-capacity tank are forced to be used. An MILP model is developed and solved by a general optimization solver. Furthermore, the relationship between the number of parallel processing slots, the time-window constraints associated with bottleneck stages and the cycle time is analyzed and used to improve the computational performance.

Unlike previous research, Ng (1996) treated the hoist traveling times as decision variables that are no less than a given lower bound and developed a branch-and-bound algorithm. This configuration provides the flexibility of hoist loaded moves and may achieve a better solution than the settings with constant traveling times. Liu et al. (2002) published a representative work, which extends Phillips and Unger's model to handle both multi-function tanks and multi-capacity tanks. Both the number of actually used parallel processing slots and the loaded hoist move times are treated as decision variables as suggested by Ng (1995, 1996). Furthermore, they pointed out the fact that forcing the use of all parallel processing slots of a multi-capacity tank can lead to infeasibility.

Due to the challenge on both mathematical formulation and problem-solving, some researchers resorted to hybrid approaches. Riera and Yorke-Smith (2002) developed a CP - MILP hybrid model for a problem with multi-capacity tanks, which combines Constraint Programming (CP) and MILP methods. Rodosek, Wallace, and Hajian (1999) also developed a hybrid approach by integrating CLP techniques and MILP methods. Compared with MILP, CP and CLP have some advantages in handling logic constraints.

2.2.3.2 Multi-degree schedules or problems with different parts

Multi-degree cyclic hoist scheduling in extended lines has also been investigated by researchers. Li and Fung (2014) studied such a problem with multi-function tanks. They first proposed an MILP model for the problem without re-entrance, which is similar to the one proposed by Zhou et al. (2012). The difference is that the hoist moves were defined according to the executing sequence at each stage rather than for each part. Thus the related decision is to determine which pair of loaded and unloaded hoist moves operate the same part. The proposed model is then extended to handle multi-function tanks by enumerating all possible conflicts in the tanks. Recently, Li et al. (2015) investigated the multi-degree cyclic scheduling with multi-capacity tanks, but the parts are identical. In such a case, the entering sequence does not have any effect. Given the part entering sequence on input station, the processing sequence on any other tank is determined at the same time and remains the same during the whole production; that is, the earlier a stage starts, the earlier it ends.

As for the cyclic scheduling with multiple part-types, the entering sequence of parts really matters and is part of the decision. In addition, for multi-capacity tanks, it also involves the

Table 2.2: Studies on the cyclic HSPs in extended lines

Line	*#D*	*#PT*	*#H*	*Method*	*Reference*
Multi-function	1	1	1	MILP	Phillips and Unger (1976)
Multi-capacity	1	1	1	Branch-and-bound	Shapiro and Nuttle (1988)
Multi-capacity	1	1	1	Branch-and-bound	Ng (1995)
Multi-capacity	1	1	1	Branch-and-bound	Ng (1996)
Multi-capacity, Multi-function	1	1	1	MILP	Liu et al. (2002)
Multi-capacity, Multi-function	1	1	1	Branch-and-bound	Che and Chu (2007b)
Multi-capacity, Multi-function	1	1	1	MILP	Zhou and Li (2003)
Multi-capacity	1	1	1	CP and MIP	Riera and Yorke-Smith (2002)
Multi-capacity	1	1	1	CLP and MIP	Rodosek et al. (1999)
Multi-function	*N*	1	1	MILP	Li and Fung (2014)
Multi-capacity	*N*	1	1	MILP	Li, Chan, and Chung (2015)
Multi-capacity	*N*	*N*	1	MILP	Zhao, Fu, and Xu (2013a)
Multi-capacity	*N*	*N*	1	MILP	Fu, Zhao, Xu, and Ho (2013)

share of tanks among different parts. Zhao et al. (2013a) studied the cyclic hoist scheduling with multiple part-types and multi-capacity tanks but without multi-function tanks. They introduced an MILP model for the problem. However, their formulation may identify feasible solutions as infeasible and also implies that each part uses all parallel processing slots of a multi-capacity tank. Fu, Zhao, Xu, and Ho (2013) investigated a similar problem, for which instead of a fixed number of parallel processing slots, a decision should be made on how to assign a given number of additional processing slots to the tanks so that the throughput is maximized. It is thus a bi-objective problem. An MILP model is formulated and an iterative framework is designed to obtained a series of Pareto solutions that balance the cost for configuring additional processing slots and the increased throughput. However, the tank capacity constraints and the time-window constraints are formulated in a similar way to that in Zhao et al. (2013a). As a consequence, it may also lead to sub-optimal solutions or even infeasibility.

The above studies on cyclic hoist scheduling in extended lines are summarized in Table 2.2. In the table, *Line* denotes the main features of the considered line, including multi-function tanks, multi-capacity tanks. *#D*, *#PT* and *#H* have the same meaning with those in Table 2.1.

2.2.4 Cyclic multi-hoist scheduling problems

The motivation to employ multiple hoists is that the material handling operations may become the bottleneck of the production system when involving a large number of material handling tasks. Multiple hoists can simultaneously perform several transportation tasks, which can relieve the bottleneck at some level. However, it also introduces additional issues such as the assignment of transportation tasks to multiple hoists and the avoidance of potential collisions between the hoists. Besides, whether the processing routes of parts are unidirectional or

bidirectional is another critical factor affecting problem-solving. In general, bidirectional processing routes involve more potential collisions than unidirectional ones. In order to balance the increased productivity and the additional complexity, researchers have also investigated many special cases of the general multi-hoist scheduling, including two-hoist scheduling, multi-hoist scheduling with non-overlapping zones on a single track, or multi-hoist scheduling with parallel tracks, which simplify or eliminate potential collisions among hoists.

2.2.4.1 Problems with parallel tracks

Another method to handle multiple hoists is the utilization of multiple parallel tracks. In such a situation, each hoist has its own track, thus no collision between the hoists will occur. The remained problem is to assign all material handling tasks to the hoists so that the throughput is maximized. Manier and Lamrous (2008) developed an evolution algorithm for simultaneously optimizing the number of hoists and the throughput. The encoding of the solution is based on the sequence of unloaded hoist moves instead of loaded hoist moves.

2.2.4.2 With non-overlapping zones on a single track

When multiple hoists share a single track, the collision-avoidance is often achieved by partitioning the track into non-overlapping segments or zones. Such a method is called zone-partition. It should be noted that this partition may be given in advance or part of the decision. Between any two adjacent zones, there is a boundary tank that can be seen as the output buffer of one zone and the input buffer of the other. After the partition, each zone can be regarded as a single hoist scheduling problem. Such a method has been investigated in many works in the literature.

Lei and Wang (1991) studied a two-hoist cyclic scheduling problem and proposed a heuristic algorithm. To avoid the collisions between the hoists, the shared track is partitioned into two non-overlapping zones, and each hoist is restricted to one of them. The risk of collisions between the hoists can only happen at the boundary tank. For each zone, a sub-problem involving a single hoist is solved. The remaining problem is to combine the two single-hoist cyclic schedules by calculating a common cycle time. Zhou and Li (2009) used a similar zone-partition method for the problem. Instead of heuristics, an MILP model was developed to obtain a final schedule for a given hoist assignment. Yang, Ju, Zheng, and Lam (2001) also considered a multi-hoist scheduling problem with zone-partition method. Nevertheless, the partition of the track is part of the decision instead of being given. A simulated annealing algorithm is proposed to minimize the cycle time.

Manier, Varnier, and Baptiste (2000) proposed a different zone-partition method, for which the production line is partitioned according to material handling operations instead of the physical positions of tanks. Each hoist is assigned to handle a subset of operations. By enumerating and analyzing all possible hoist collisions, a series of rules are defined to avoid these collisions. Zhou and Liu (2008) considered two-hoist scheduling with three

non-overlapping zones. Each hoist is responsible for one of the end zones but they share the middle zone. Several heuristic rules are used to generate the sequences of hoist moves and assign each move to one of the hoists. For the middle zone, twenty possible collision situations are identified and analyzed to formulate collision-avoidance constraints. After a feasible hoist assignment is obtained, a linear program is solved to obtain an optimal schedule. Chtourou, Manier, and Loukil (2013) proposed an iterative heuristic to deal with a two-hoist scheduling problem. The heuristic suggested by Zhou and Liu (2008) was adapted to generate the sequences of hoist moves. For each generated sequence, the moves are assigned to the hoists according to their relative positions. An MILP model is developed to obtain a candidate schedule whose feasibility is checked by a collision-test procedure. Li and Fung (2013) extended the work to multi-degree cyclic scheduling, where the line is manually divided into non-overlapping zones in advance. An MILP model is developed to obtain an optimal schedule for the given zone-partition. Li et al. (2015) developed an MILP model for the same problem but with multi-capacity tanks. The problem becomes more complicated as the assignment of multiple parts to parallel processing slots should be handled as well.

2.2.4.3 Direct collision avoidance

Besides zone-partition and parallel tracks, researchers have also directly studied problems with potential hoist collisions on a single track. Leung et al. (2004) developed an MILP for such a multi-hoist scheduling problem. The direction of loaded hoist moves is consistent with the parts' processing route; that is, all parts are transported toward to the same direction. Che and Chu (2004) proposed a branch-and-bound algorithm for the same problem. Jiang and Liu (2014) studied a similar problem and formulated an MILP model by enumerating all possible collisions between hoists. A branch-and-bound algorithm is designed to solve the problem. Recently, Che, Lei, Feng, and Chu (2014) pointed out that previous works usually assume the loaded hoist moves cannot across cycles, i.e., a loaded hoist move should start and end within the same cycle. They demonstrated that allowing loaded hoist moves to cross adjacent cycles can lead to better solutions. To this end, an improved MILP model was developed by reformulating the time-window constraints in Leung et al. (2004).

There are also works considering bidirectional processing routes of parts. Varnier, Bachelu, and Baptiste (1997) adapted an approach similar to the zone-partition method, and proposed a heuristic algorithm for a multi-hoist cyclic scheduling problem with bidirectional part processing routes by using CLP. Leung and Zhang (2003) considered a problem similar to Leung et al. (2004) but the processing routes of parts are bidirectional. In this case, two loaded hoists traveling in opposite directions can meet each other halfway. Thus collisions between hoists are more likely than with unidirectional processing routes. An MILP model was formulated by considering all collision situations. Riera and Yorke-Smith (2002) developed a hybrid model for generic hoist scheduling problems, which integrates CLP and MILP techniques and addresses both multiple hoists and bidirectional processing routes of parts.

Table 2.3: Studies on the cyclic multi-hoist scheduling

Features	*#D*	*#PT*	*#H*	Method	Reference
Unidirectional; Zone-partition	1	1	2	Heuristics	Lei et al. (1991)
Unidirectional; Zone-partition	1	1	2	MILP	Zhou et al. (2009)
Unidirectional; Zone-partition	1	1	*N*	Simulated annealing	Yang et al. (2001)
Unidirectional; Zone-partition	1	1	*N*	CLP	Manier et al. (2000)
Unidirectional; Zone-partition	1	1	2	Heuristic and MILP	Zhou et al. (2008)
Unidirectional; Zone-partition	1	1	2	Heuristic and MILP	Chtourou et al. (2013)
Unidirectional; Zone-partition	*N*	1	*N*	MILP	Li et al. (2013)
Multi-capacity; Unidirectional; Zone-partition	*N*	1	*N*	MILP	Li et al. (2015)
Unidirectional; Parallel track	1	1	*N*	Evolution algorithm	Manier et al. (2008)
Unidirectional	1	1	*N*	MILP	Leung et al. (2004)
Unidirectional	1	1	*N*	Branch-and-bound	Che et al. (2004)
Unidirectional	1	1	*N*	Branch-and-bound	Jiang et al. (2014)
Unidirectional, Moves span cycles	1	1	*N*	Branch-and-bound	Che et al. (2014)
Bidirectional; Zone-partition	1	1	*N*	CLP	Varnier et al. (1997)
Bidirectional	1	1	*N*	MILP	Leung et al. (2003)
Bidirectional	1	1	*N*	CLP and MIP	Riera et al. (2002)
Zone-partition; Given cycle time	1	1	*N*	Heuristics	Lei et al. (1993)
Zone-partition; Given cycle time	1	1	*N*	Greedy algorithm	Armstrong et al. (1996)

Due to the complication of multi-hoist scheduling, researchers also studied sub-problems such as multi-hoist scheduling with given cycle times (Lei, Armstrong, & Gu, 1993; Armstrong, Gu, & Lei, 1996). Lei et al. (1993) developed a heuristic to obtain a cyclic schedule for a given cycle time such that the number of hoists that share a single track is minimized. The algorithm partitions the line into non-overlapping zones each of which is handled by a single hoist. Then the problem is equivalent to minimizing the number of non-overlapping zones. They also proved that the obtained solution is optimal if and only if two hoists are needed. Armstrong et al. (1996) proposed a greedy algorithm for the problem. By partitioning all hoist moves into groups and then assigning each group to a single hoist, minimizing the number of hoists is equivalent to maximizing the number of hoist moves in each group, which can be solved by dealing with a shortest path problem.

The above studies on cyclic multi-hoist scheduling is summarized in Table 2.3.

2.3 Non-cyclic hoist scheduling problems

In cyclic hoist scheduling, the hoist(s) repeatedly execute(s) a sequence of operations. However, it may be not applicable for multi-product and small-batch production. In such situations, coupling all types of parts in one cycle is usually impossible, and splitting each type of parts into a cycle can involve frequent reset and reconfiguration of the production for changing from

one type of parts to another. Another situation where cyclic hoist scheduling is not preferred is when the types and the numbers of parts are not known in advance. To deal with such problems, non-cyclic hoist scheduling have also been investigated by researchers.

Non-cyclic hoist scheduling problems are divided into three classes by Manier and Bloch (2003): predictive hoist scheduling problems (PHSP), dynamic hoist scheduling problem (DHSP), and reactive hoist scheduling problem (RHSP). In particular, PHSP denotes the static hoist scheduling problems except CHSP, which usually considers the following two kinds of problems: the calculation of the minimum duration for transiting from one cyclic production to another; the scheduling of a set of parts' processing such that the makespan is minimized. DHSP aims to generate a new hoist schedule when new parts arrive so that the productivity is maximized. RHSP is to dynamically assign hoist operations so that particular criteria are achieved.

In the literature, few works have been done to deal with PHSP. One important reason is that scheduling a large number of parts is usually computationally difficult or even impossible as it involves sequencing numerous operations. From this view, DHSP and RHSP can be seen as two different compromise approaches to deal with this problem. DHSP assumes that parts dynamically arrive at the system, thus each time a new predictive schedule involving one or several new parts (short-term predictive schedule) are generated. In this context, a series of short-term predictive schedules are dynamically generated one by one until all parts are scheduled. For RHSP, no predictive schedule is generated and the hoist(s) are dynamically dispatched to handle transportation operations with rules either predefined or dynamically defined according to system states.

2.3.1 Predictive hoist scheduling problems

Song, Zabinsky, and Storch (1993) developed a heuristic algorithm for a single-hoist scheduling problem. The algorithm is based on the EST heuristic. Fleury, Gourgand, and Lacomme (2001) proposed stochastic meta-heuristics to determine a hoist schedule for which the consequences of the variations in hoist transportation times on the makespan and the violation of the time-windows are minimized. It is also assumed that only one new part is introduced into the system at a time. Zhang, Manier, and Manier (2012) proposed a model for jobshop scheduling with and without storage buffers between stages. A genetic algorithm with local tabu search is proposed to solve it. Zhang, Manier, and Manier (2014) developed a modified shifting bottleneck heuristic and disjunctive graph modeling. A modified disjunctive graph including processing nodes, transportation nodes and storage nodes is used to represent the problem. It is shown that a feasible solution is associated with a path that contains no positive cycle. Bloch, Varnier, and Baptiste (1996) compare the efficiency of several meta-heuristics (stochastic descent, tabu search, kangaroo algorithm, etc.).

2.3.2 Reactive hoist scheduling problems

As mentioned before, no complete schedule for RHSPs is generated, and each hoist move is scheduled according to the real-time state of the system. In other words, the executing sequence of hoist operations are not known in advance (or off-line), and are reactively determined on-line. During the production, the state of the system is first identified, the next hoist move to be executed is then determined based on the state. Thesen and Lei (1986, 1990) introduced an expert system to improve the productivity and reduce defective parts, which consists of a data base for collecting system states and a rule base for selecting the best rule for specific system states. During scheduling, a part is loaded only if all parts in the tanks will not become defective. Sun et al. (1994) investigated the three dispatching rules proposed by Thesen and Lei (1986, 1990) for a multi-hoist scheduling problem. The difference is that a part is loaded if the assignment of the first future hoist move is unaffected. Mak, Gupta, and Lam (2002) investigated a multi-hoist scheduling problem with multi-function tanks and multi-capacity tanks. A knowledge-based simulation (KBS) system was proposed, including a part loading heuristic and seven hoist dispatching heuristics. The part loading heuristic was adopted from the one suggested by Sun et al. (1994), while the hoist dispatching heuristics include the four rules proposed by Thesen and Lei (1986, 1990) and three new heuristic rules. A heuristic is also proposed to choose appropriate dispatching rules for new parts, instead of defining all dispatching heuristics and selection rules suggested by Thesen and Lei (1990).

Jegou, Kim, Baptiste, and Lee (2006) proposed an intelligent agent system composed of two distinct multi-agent systems. One is used to determine the entering time of the new parts while the other is designed to assign transportation tasks to hoists. Inside the systems, the task dispatching is based on an auction and bidding process. However, it can produce defective parts as the bid calculation is based on the flexibility in the current schedule. Furthermore, neither multi-capacity tanks nor multiple part-types are supported. Goujon, Lacomme, Norre, and Traoré (1996) developed a heuristic which uses dispatching rules that designate a higher priority to the most urgent part. Though the dispatching rules are simple to implement, defective parts cannot be avoided due to the nature of dispatching rules.

2.3.3 Dynamic hoist scheduling problems

With dynamic scheduling, researchers investigated three situations concerning moves already scheduled but not yet started: (i) the starting times and hence the precedence relations among them are not allowed to be modified in the rescheduling; (ii) the starting times are modifiable but not the precedence relations; (iii) the precedence relations and therefore the starting times are modifiable.

2.3.3.1 Without changing starting times of already scheduled moves

With this assumption, the transportation and processing operations for the new part are inserted into the idle time of the hoist and tanks in the initial schedule, respectively. Yih and Yin (1992) studied a dynamic single-hoist scheduling problem under this assumption and with only one new part considered each time. A two-phase heuristic is developed to explore the flexibility of processing time-windows of the new part. In its first phase, the entering time of the new part is updated so that no conflict in the use of tanks occurs. Its second phase deals with possible hoist conflicts by updating the part's entering time and exploring the flexibility of processing time-windows. Not allowing the already scheduled operations to change may reduce the complexity of the problem, but it prevents the potentials that improve the productivity by rescheduling all operations.

2.3.3.2 Without changing precedence relations among already scheduled moves

The above approach (Yih & Yin, 1992) was further refined by Yih (1994) by allowing previously scheduled operations to be rescheduled. However, the sequence of the already scheduled hoist moves are not allowed to changed. In other words, the already scheduled hoist moves can be shifted forward or backward. Thus it can explore the flexibility of the processing time-windows for both the already scheduled parts and the new one. The two heuristics (Yih & Yin, 1992; Yih, 1994) were reported to be superior to their respective basic versions where the required processing times are fixed to their minimums. Ge and Yih (1995) presented a branch-and-bound-based heuristic for the problem. A necessary and sufficient condition for feasible system states is proposed and transformed into the feasibility checking of a linear program. The heuristic is based on enumerating possible sequences of hoist moves and is applicable for multi-hoist situations. Cheng and Smith (1997) proposed a precedence constraint posting heuristic to schedule hoist moves, which is motivated by the heuristic and the multi-lift algorithm suggested by Smith and Cheng (1993) and E. G., Garey, and Johnson (1978), respectively.

Hindi and Fleszar (2004) treated the problem as a constraint satisfaction problem and proposed a heuristic. Two checking procedures, a forward checking and a backtracking, were developed to evaluate the feasibility of candidate schedules. Zhou and Li (2002) proposed a heuristic procedure to obtain feasible sequences of hoist moves. An optimal schedule is then calculated by solving linear programs for such sequences. Paul, Bierwirth, and Kopfer (2007) developed an adaptive time-window heuristic for a similar problem and compared it with previously published methods such as the EST heuristic (Song et al., 1993) and the permutation-based algorithm (Yih & Yin, 1992; Yih, 1994).

Chauvet et al. (2000) proposed a polynomial algorithm for a multi-hoist scheduling problem with multi-capacity tanks. Nevertheless, the possible collisions between hoists are not taken into consideration. Yan et al. (2014) proposed a two-phase branch-and-bound algorithm for a dynamic hoist scheduling problem considering disturbances defined as the deviation of

parts' completion times between the initial and new schedules. The objective is to minimize the makespan of the new schedule under a limited disturbance.

2.3.3.3 Allowing rescheduling of already scheduled moves

In the above works, the processing sequence of the already scheduled parts is not changed in the rescheduling. In some restrictive cases, even the start and end times of hoist moves are not allowed to change. Furthermore, it is assumed that the rescheduling contains only one new part or multiple new parts with a given entering sequence. All these may reduce the complexity, but the flexibility of the scheduled parts vanishes at the same time. To overcome this, researchers have also investigated the schedules where all hoist moves scheduled in the initial schedule but not executed yet can be rescheduled.

Lamothe et al. (1994) demonstrated that allowing the previously scheduled but not yet started hoist moves to be rescheduled may yield higher throughput but at the cost of extra computational efforts. They introduced a branch-and-bound algorithm and showed that it can deal with multiple new parts. To further improve the performance, Lamothe, Correge, and Delmas (1995) proposed an improved branch-and-bound algorithm, which integrates a solution quality checking procedure and a dynamic backtracking strategy (Ginsberg, 1993). Furthermore, the information of initial schedules is used in the rescheduling procedure whenever possible. A simulation on the benchmark instance given by Phillips and Unger (1976) showed a better computational performance.

Lamothe, Thierry, and Delmas (1996) extended the above work to a multi-hoist scheduling problem by taking into consideration the possible hoist collisions. Fargier and Lamothe (2001) studied a similar problem, but the required processing times are treated as soft constraints which can be violated. The quality of parts, which is measured by their actual processing times, is represented by fuzzy numbers. A bi-criteria problem that optimizes both part quality and makespan is solved by dealing with a series of classic hoist scheduling problems.

Kujawski and Swiatek (2011) proposed a decomposition algorithm called cyclogram unfold method. The parts are first divided into groups according to their types. Then, the original problem is decomposed into several independent problems, and each of them handles a single group of parts and is solved by adapting existing cyclic schedules. Finally, the obtained schedules are combined together to create a final schedule. On the one hand, the method is fast as it relies on the cyclic schedules calculated in advance. On the other hand, the performance of the method closely relies upon the quality of these cyclic schedules.

2.3.3.4 Extended lines and multiple new parts

The above works mainly considered basic lines without multi-function tanks or multi-capacity tanks. Furthermore, most of them consider only one new part in the rescheduling procedure. That is to say, it is assumed that only one new part is involved in the rescheduling at a time, either in the case a new part arrives at the input station only after another has left, or multiple

parts simultaneously arrive at the input station but their entering sequence is known in advance. Zhao et al. (2013b) developed an MILP model to deal with a general problem with multiple new parts and multi-capacity tanks. This problem is more complicated because it has to simultaneously determine the entering sequence of the parts as well as the start times of hoist moves. However, there are several flaws in this model. Firstly, the model involves a large number of variables and constraints even for small-sized instances. Secondly, it may identify feasible schedule as infeasible or lead to suboptimal solutions or infeasibility due to its flawed formulation of some constraints. Tian, Che, and Feng (2013) reformulated some constraints of the model, which significantly reduces the number of constraints and variables and consequently improves the computational performance. However, the flaws that may lead to suboptimal solutions or infeasibility were not addressed.

The above studies on non-cyclic hoist scheduling are summarized in Table 2.4.

Table 2.4: Studies on non-cyclic HSPs

Main features	*#H*	Method	Reference
PHSP	1	Heuristic	Song et al. (1993)
PHSP, Variable traveling times	1	Meta-heuristic	Fleury et al. (2001)
PHSP	1	Genetic algorithm	Zhang et al. (2012)
PHSP	1	Heuristics	Zhang et al. (2014)
PHSP	1	Meta-heuristics	Bloch et al. (1996)
RHSP	1	Heuristics	Thesen and Lei (1986, 1990)
RHSP	1	Heuristics	Sun et al. (1994)
RHSP, Re-entrant, Multi-capacity	N	Knowledge system	Mak et al. (2002)
RHSP	N	Multi-agent system	Jegou et al. (2006)
RHSP	N	Heuristics	Goujon et al. (1996)
DHSP, Fixed start times of moves	1	Heuristics	Yih and Yin (1992)
DHSP, Fixed sequence of moves	1	Heuristics	Yih (1994)
DHSP, Fixed sequence of moves	$1/N$	Heuristics	Ge and Yih (1995)
DHSP, Fixed sequence of moves	1	Heuristics	Cheng and Smith (1997)
DHSP, Fixed sequence of moves	1	Heuristics	Hindi and Fleszar (2004)
DHSP, Fixed sequence of moves	1	Heuristics, MILP	Zhou and Li (2002)
DHSP, Fixed sequence of moves	1	Heuristics	Chauvet et al. (2000)
DHSP, Fixed sequence of moves	1	Heuristics	Paul et al. (2007)
DHSP, Fixed sequence of moves	1	Branch-and-bound	Yan et al. (2014)
DHSP	1	Branch-and-bound	Lamothe et al. (1994)
DHSP	1	Branch-and-bound	Lamothe et al. (1995)
DHSP	N	Branch-and-bound	Lamothe et al. (1996)
DHSP	N	Heuristics	Kujawski and Swiatek (2011)
DHSP, Multi-capacity	1	MILP	Zhao et al. (2013b)
DHSP, Multi-capacity	1	MILP	Tian et al. (2013)

2.4 Hoist scheduling with zero-width time-windows

Due to the NP-hardness introduced by time-window constraints, the hoist scheduling with fixed processing times has also been studied in the literature. It can be seen as a special case of a time-window for which the lower bound is exactly equal to the upper bound, i.e., a zero-width time-window. Such problems are also called no-wait scheduling problems in the literature. The effort made for these problems mainly focuses on complexity issues.

2.4.1 Simple cyclic schedules

For simple cyclic schedules, Kats and Mikhailetskii (1980) studied the problem with a single hoist to minimize the cycle time, and suggested an algorithm of time complexity $O(m^6)$, where m stands for the number of tanks in the line. Kats and Levner (1997a) considered the same problem and proposed an algorithm of time complexity $O(m^4)$. They also demonstrated that the time complexity for its re-entrant version is $O(m^5)$. An improved algorithm of time complexity $O(m^3 \log m)$ was proposed by Levner et al. (1997).

2.4.2 Multi-degree schedules

As for multi-degree schedules, researchers have also developed various algorithms. Levner, Kats, and Sriskandarajah (1996) proposed a geometric algorithm for solving a 2-degree cyclic scheduling problem. The authors did not assess the computational complexity, but conjectured that the algorithm is polynomial. The geometrical scheme was further enhanced by Kats and Levner (2006), and an improved algorithm of complexity $O(m^5 \log m)$ was developed. Che, Chu, and Levner (2003) proposed a polynomial algorithm of complexity $O(m^8 \log m)$ for the problem, and demonstrated that the algorithm can be directly used for the situation where the two parts introduced during a cycle are of different types. Chu (2006) improved the above work and presented a faster algorithm of time complexity $O(m^5 \log m)$. Kats and Levner (2009) showed that the complexity for the general non-Euclidean case remained open, and presented an algorithm of time complexity $O(m^5 \log m)$ by adapting the geometrical scheme suggested by Levner et al. (1996).

The above works all considered 2 - degree cyclic scheduling. A general multi-degree cyclic schedule can involve more than two parts in a cycle. For such a problem, Song et al. (1993) developed an MILP model for maximizing the production rate of a given number of parts. They suggested a heuristic procedure based on the EST heuristic. The obtained schedule may be a simple cyclic one or a multi-degree one depending on specific instances. They also proposed two modifications to improve the obtained schedules by considering hoist traveling times when determining the earliest start times and performing a local search. Kats et al. (1999) studied the multi-degree cyclic scheduling with integer input data. The problem is transformed into a linear program based on the prohibited intervals method. To obtain an

optimal schedule, a sieve method is proposed to search over all possible integer values of its cycle time. The complexity of the algorithm is $O(rm^3u^r)$, where r is the degree of cyclic schedules, u is the number of integer values tested in the solution procedure. For the general multi-degree cyclic scheduling, Che, Chu, and Chu (2002) proposed different algorithms and showed that the complexity is $O\left(r^4(m^3/r)^{2+r(r-1)/2}\right)$ if $m > r$ and $O\left(r^4m^{r(r-1)+4}\right)$ if $m \leq r$, where m is the number of tanks and r is the degree of the considered schedule.

2.4.3 Jobshop scheduling

For the above works, all parts to be processed are identical. If multiple part-types are involved, the problem become more complicated due to similar reasons for problems with general processing time-windows. Thus only a few works have addressed such problems. Agnetis (2000) investigated a two- or three-tank system with different part-types and proposed an algorithm of complexity $O(m \log m)$, where m is the number of parts. Che, Yan, Yang, and Chu (2010) studied a more general problem with more tanks. A branch-and-bound algorithm is proposed with a complexity of $O\left(r^{10}(m^3/r)^{2+r(r-1)/2}\right)$ if $r \leq m$ and $O\left(r^{10}m^{4+r(r-1)}\right)$ if $r > m$. That is, the algorithm is polynomial in the number of tanks for a given value of r, but exponential if it is arbitrary.

2.4.4 Extended lines

There are also works on cyclic hoist scheduling in extended lines. Che and Chu (2005a) studied such a problem with both multi-capacity tanks and multi-function tanks, and proposed an algorithm of time complexity $O\left(n^6m\right)$, where n and m are the number of stages and the number of tanks, respectively. Che and Chu (2007a) proposed a faster algorithm for a special case of the problem where a multi-function tank cannot be multi-capacity. The time complexity of the algorithm is reduced to $O\left(n^3m \log n\right)$ due to this restriction.

2.4.5 Multi-hoist schedules

As for cyclic multi-hoist scheduling, Kats and Levner (1997b) proposed an algorithm of time complexity $O\left(m^5\right)$ for obtaining the minimum number of hoists required for all possible cycle lengths. However, the hoists were assumed to be running on parallel tracks and no collision-avoidance constraint was involved. Later, Kats and Levner (2002) extended their previous work for basic line (Levner et al., 1997) to the multi-hoist case with parallel tracks, where the tanks served by each hoist are known in advance. They proposed an algorithm which aims to minimize the cycle time in $O\left(m^3 \log m\right)$ time, where m is the number of tanks as before.

Leung and Levner (2006) and Che and Chu (2008) proposed effective polynomial algorithms with time complexity $O\left(m^5\right)$ and $O\left(m^6k\right)$, respectively, for the problem with a single track, where k is the number of hoists. In this case, the collisions between hoists should

be considered, which make it much more complicated than with parallel tracks. Jiang and Liu (2007) considered a more general problem where the tanks are not arranged in the same order with the part processing sequence. A polynomial algorithm of time complexity $O(m^6k)$ is developed, where m and k are the numbers of tanks and hoists, respectively. Che, Chabrol, Gourgand, and Wang (2012) considered a similar problem with multi-function tanks. The authors developed a polynomial algorithm for obtaining the minimum number of hoists for all feasible values of cycle time in $O(m^5)$ time. Consequently, the optimal cycle time for a given number of hoists can be obtained with the same complexity.

2.4.6 Multi-degree multi-hoist schedules

Researchers have also investigated multi-degree multi-hoist cyclic scheduling. Che and Chu (2005b) addressed a multi-degree cyclic scheduling problem with two hoists running on parallel tracks. They proposed an algorithm to minimize the cycle time. It was proved the complexity of the algorithm is $O\left(m^3/r)^{r(r-1)/2}(r^5+m)m^6r\right)$ if $r \leq m$ and $O\left(m^{r(r-1)+4}r^8\right)$ if $r > m$. The work was generalized to the scenario with multiple hoists by Che and Chu (2009). They showed that for a given degree r, the problem can be solved in $O\left((m^{6+3r(r-1)/2}(m^2+r^5)r^3\right)$ time. As mentioned before, the assumption that multiple hoists are running on separated parallel tracks simplifies the problem to some extent. Some pioneer works have attempted to relax this assumption. Che, Hu, Chabrol, and Gourgand (2011a) studied 2-degree multi-hoist cyclic scheduling where the hoists are running on a single track. They proved that the problem is solvable in $O(m^7)$ time.

The above works on hoist scheduling with zero-width time-windows are summarized in Table 2.5.

2.5 Summary

In the previous sections, we first introduced hoist scheduling problems under various settings and then reviewed the studies on cyclic and non-cyclic hoist scheduling problems. As a special case of time-window constraints, the studies on hoist scheduling problems with fixed processing times were also surveyed. It can be seen that hoist scheduling problems has been widely studied in the literature. However, there are still a lot of works to do in this field, especially in the formulation and optimization of HSPs with complex configurations. Based on this overview, we make the following observations:

(1). The cyclic single-hoist scheduling problems have received extensive attention from researchers. Diverse formulations and algorithms have been developed for these problems. Most of the studies deal with basic lines or extended lines with one more feature such as multi-degree schedules, multiple part-types, multi-function tanks and multi-capacity tanks. In particular, multi-degree schedules or multiple part-types are usually considered

Table 2.5: Studies on the HSPs with fixed processing times

Main Features	#D	#PT	#H	Complexity	Reference
		1	1	$O(m^6)$	Kats and Mikhailetskii (1980)
		1	1	$O(m^4)$	Kats and Levner (1997a)
Multi-function	1	1	1	$O(m^5)$	Kats and Levner (1997a)
		1	1	$O(m^3 \log m)$	Levner et al. (1997)
Non-Euclidean	2	1	1	Geometric algorithm	Levner et al. (1996)
		2	1	$O(m^5 \log m)$	Kats and Levner (2006)
		2	1, 2	$O(m^8 \log m)$	Che et al. (2003)
		2	1	$O(m^5 \log m)$	Chu (2006)
Non-Euclidean	2	1	1	$O(m^4 \log m)$	Kats and Levner (2009)
		1	N	Heuristic	Song et al. (1993)
Integer Solution	N	1	1	$O(rm^3u^r)$	Kats et al. (1999)
$m > r$	N	1	1	$O(r^4(m^3/r)^{2+r(r-1)/2})$	Che et al. (2002)
$m \leq r$	N	1	1	$O(r^4m^{4+r(r-1)})$	Che et al. (2002)
2- and 3 tanks	N	N	1	$O(m \log m)$	Agnetis (2000)
$r \leq m$	N	N	1	$O(r^{10}(m^3/r)^{2+r(r-1)/2)})$	Che et al. (2010)
$r > m$	N	N	1	$O(r^{10}m^{r(r-1)+4})$	Che et al. (2010)
Multi-function	1	1	1	$O(m^6n)$	Che and Chu (2005a)
Multi-capacity					
Multi-function	1	1	1	$O(n^3m \log n)$	Che and Chu (2007a)
Multi-capacity					
Parallel tracks	1	1	N	$O(m^5)$	Kats and Levner (1997b)
Parallel tracks	1	1	N	$O(m^3 \log m)$	Kats and Levner (2002)
Single-track	1	1	N	$O(m^5)$	Leung and Levner (2006)
Single-track	1	1	N	$O(m^6k)$	Che and Chu (2008)
Bidirectional	1	1	N	$O(m^6k)$	Jiang and Liu (2007)
Multi-function	1	1	N	$O(m^5)$	Che et al. (2012)
Parallel track, $r \leq n$	N	1	2	$O((m^3/r)^{r(r-1)/2}(r^5 + m)m^6r)$	Che and Chu (2005b)
Parallel track, $r > n$	N	1	2	$O(m^{r(r-1)+4}r^8)$	Che and Chu (2005b)
Parallel track	N	1	N	$O((m^{6+3r(r-1)/2}(m^2 + r^5)r^3)$	Che and Chu (2009)
Single track	2	1	N	$O(m^7)$	Che et al. (2011a)

in basic lines. The research on cyclic scheduling that combine several features needs more investigation.

(2). The work on multi-hoist scheduling, especially in lines with bidirectional part processing routes has not received much attention. Few works have been done on multi-hoist scheduling in extended lines with multi-function tanks and multi-capacity tanks. It is known that the problems become more challenging when multi-degree schedules and multiple part-types are simultaneously considered. It usually needs more attention on the collision-avoidance constraints for hoists running on a single track.

(3). Most of the works study problems with fixed parameters without considering random events that can alter these parameters. The schedules obtained without considering the

variations or disturbances caused by random events may not effectively be implemented in practice. Defective parts and deadlocks may occur due to random events that can disrupt the schedule.

(4). Hoist scheduling with multiple part-types in extended lines with multi-function tanks and multi-capacity tanks has not received much attention under either cyclic or non-cyclic settings. The only existing model involves some flaws which will be analyzed and reformulated in this study.

This thesis mainly addresses the last two points. It aims to develop mathematical models for several hoist scheduling problems so that the throughput is maximized while all constraints are respected. In particular, we will address the following problems: formulation and optimization for robust cyclic hoist scheduling that considers delays in hoist traveling operations, formulation and optimization of hoist scheduling in extended lines with multi-function multi-capacity tanks, whether in cyclic or non-cyclic mode.

3

Robust Optimization for Cyclic Hoist Scheduling Problem

Contents

3.1 Introduction

As mentioned above, cyclic hoist scheduling is usually used in mass production due to its ease of management and implementation. In this situation, an optimal cyclic hoist schedule is computed in advance according to the deterministic parameters such as the required processing time-windows at all stages and the hoist transportation times. Then, the hoist is

programmed to perform the obtained cyclic schedule repeatedly without any disruption until the demand is satisfied. However, in real industrial environments, the hoist transportation times between tanks and stations are subject to variations due to diverse random events, such as variations in part weights, failures of unloading, material-handling engine problems and communication-delay between control system and hoists (Fleury et al., 2001). In a cyclic operating mode, the schedule is usually more compact than that in a non-cyclic mode (Brucker, Burke, & Groenemeyer, 2012). Thus under a cyclic operating mode, any such variations can cause the hoist to be behind schedule. The consequences may be serious because it influences the processing of parts not only in the current cycle but also in the subsequent ones. In this case, some final parts may be defective due to violations of processing time-windows, or more seriously, deadlocks may occur.

To the best of our knowledge, Fleury et al. (2001) is the first study to investigate variations of hoist traveling times and proposed a stochastic meta-heuristic framework to determine a non-cyclic hoist schedule, for which the consequences of the variations on the makespan and the violation of the time-windows are minimized. However, defective parts may be produced because of violations of processing time requirements. Different from their work, we study a hoist scheduling problem subject to variations in the hoist transportation times from a robust perspective. That is, we try to generate a cyclic hoist schedule that is "robust" within certain extent of variations in the hoist transportation times. In other words, as long as the hoist transportation times vary within some range due to random events, the generated robust cyclic hoist schedule can still be implemented without altering its corresponding cycle time and without violating the processing time-windows. Thus no part will be defective.

In the literature, the robustness of a schedule is usually defined as its ability to perform well under dynamic and uncertain operational environments (Dooley & Mahmoodi, 1992; Billaut, Moukrim, & Sanlaville, 2010). Two types of robustness are usually considered in the literature: solution robustness and quality robustness (Herroelen & Leus, 2004; Sevaux & Sörensen, 2004; Van de Vonder, Demeulemeester, Herroelen, & Leus, 2005; Briskorn, Leung, & Pinedo, 2011). Solution robustness is measured in terms of the deviation between the planned and actual start times of operations, while quality robustness is measured in terms of the deviation of the criteria (e.g. makespan, flow time, and customer service level) between the planned and realized schedules (Van de Vonder et al., 2005). Most studies in the literature considered the quality robustness, such as the deviation of makespan between the planned schedules and the realized ones (Storer, Wu, & Vaccari, 1992; Rahmani & Heydari, 2013; Xiong, Xing, & Chen, 2013), the deviation of flow time (Lu, Lin, & Ying, 2012) or customer service level (Ranjbar, Davari, & Leus, 2012). Al-Hinai and ElMekkawy (2011) considered three measures of solution robustness: (1) the average deviation between the completion times of predicted and realized operations, (2) the total deviation between the completion times of predicted and realized operations, and (3) the average deviation between the completion times of affected predicted operations and the affected realized ones. In other studies, both the

solution robustness and quality robustness are addressed (Wu, Storer, & Chang, 1993; LEON, WU, & STORER, 1994; Rangsaritratsamee, Ferrell Jr, & Kurz, 2004).

The above works related to robust scheduling consider the robustness in production systems without material handling devices. To the best of our knowledge, no work in the literature has addressed robust optimization for cyclic hoist scheduling problems. In this chapter, we deal with the robust optimization for a cyclic hoist scheduling problem. The work presented in this Chapter contributes to the area in the following aspects. Firstly, we propose a method to measure the robustness of a cyclic hoist schedule in terms of free slacks in hoist transportation. Secondly, a bi-objective MILP model is proposed for the problem, which aims to optimize the cycle time and the robustness. Thirdly, it is proved that the optimal cycle time strictly increases with the robustness, thus the problem has an infinite number of Pareto optimal solutions. The so-called ideal and nadir points are derived to describe the lower and upper bounds for the objective values of the Pareto front. A Pareto optimal solution can be obtained by solving a single-objective problem to minimize the cycle time for a given value of robustness or to maximize the robustness for a specific upper bound of the cycle time.

The remainder of this chapter is organized as follows. A formal problem description and the notation are given in Section 3.2. In Section 3.3, the measurement of the robustness for a given cyclic hoist schedule is defined. Section 3.4 is devoted to the formulation of the bi-objective MILP model. In Section 3.5, the proposed model is analyzed and discussed. In Section 3.6, computational experiments on benchmark instances and randomly generated instances are executed to evaluate the proposed approach. Finally, the work is concluded in Section 3.7.

3.2 Problem description and notation

The production line studied is physically similar to those widely studied in the literature (Che & Chu, 2007b; Chen et al., 1998; Leung et al., 2004; Liu et al., 2002; Phillips & Unger, 1976; Shapiro & Nuttle, 1988) and called a basic line in the literature and the previous chapters. Similar problem description can be found in these works. For the sake of self-consistency, we recall this description and the notation used throughout the chapter, which are similar to those in the literature. This production line is a basic line composed of a single computer-controlled hoist and $W+2$ tanks. The tanks are arranged in a line according to the processing routine, and are denoted by $M_0, M_1, \ldots, M_{W+1}$ from left to right, where the input station and the output station are respectively indexed as M_0 and M_{W+1}. In some cases, the input station and output station may be the same physical tank. All the parts to be processed are identical and waiting at M_0. The processing route of a part can be described as follows. The part is first moved away by the hoist from M_0, then is processed in $M_1, M_2, \ldots, M_W$ successively and finally loaded onto M_{W+1}. The material handling operations between the tanks are carried out by the hoist. M_0 and M_{W+1} are assumed to be of unlimited capacity, while each tank can process at

most one part at a time. There is no intermediate storage buffer between the tanks. After the processing in a tank is completed, the part must be moved away and directly transported to the next tank by the hoist. The hoist is not allowed to wait during the transportation. The actual processing time of each part in tank M_i must be within its required time-window, which is defined as a closed interval $[L_i, U_i]$, $1 \leq i \leq W$. The hoist operation of transporting a part from tank M_i to tank M_{i+1} is called *loaded move* i, and denoted by O_i, $0 \leq i \leq W$. The time required to execute move O_i is D_i, $0 \leq i \leq W$. In contrast to a loaded move, the hoist operation that travels from a tank to another without holding a part is called an *empty move*. The time required for the empty move from tank M_i to tank M_j is $E_{i,j}$, $0 \leq i, j \leq W + 1$. Furthermore, it is assumed that the hoist traveling times satisfy the following triangular inequalities, as it is generally the case in practice.

$$E_{i,k} + E_{k,j} \geq E_{i,j}, \text{ for all } 0 \leq i, j, k \leq W + 1 \tag{3.1}$$

$$D_i \geq E_{i,i+1}, \text{ for all } 0 \leq i \leq W \tag{3.2}$$

In a cyclic schedule, the hoist repeatedly performs a fixed sequence of operations. Each repetition of such a sequence of operations is called a cycle, and its duration is called cycle time and denoted as T. A cyclic hoist schedule can be defined by a sequence of loaded moves and their associated start times in a cycle. Hence, a cyclic hoist schedule can be represented by $(T, s_{[0]}, s_{[1]}, \ldots, s_{[W]})$, where $[i]$ is the index of the i^{th} loaded move that the hoist performs during a cycle and $s_{[i]}$ is its start time, $0 \leq i \leq W$. For instance, $[2] = 3$ means that the second loaded move executed by the hoist within a cycle is move O_3, i.e., the move that transports a part from tank M_3 to tank M_4. Note that each loaded move is followed by an empty move in a cyclic hoist schedule. After loading a part into some tank, the hoist will travel to the tank at which the next loaded move will start. If the hoist just waits until the completion of the processing after loading a part into a tank, the associated empty move can be treated as a dummy move. To simplify the expression, the empty move immediately follows loaded move O_i is called *empty move* i, and denoted by G_i hereafter. Thus, for schedule $(T, s_{[0]}, s_{[1]}, \ldots, s_{[W]})$, the hoist first performs loaded move $O_{[0]}$, then empty move $G_{[0]}$, loaded move $O_{[1]}$, ..., loaded move $O_{[W]}$ and empty move $G_{[W]}$ within a cycle.

In general, a cyclic schedule is obtained off-line with given deterministic data and then implemented. During the execution of the schedule, inevitable random events may lead to delays in the hoist transportation times, which may further cause a deadlock or defective parts due to violations of processing time-windows. It can seriously degrade the performance of the production line. To deal with this problem, it is desirable to obtain a robust schedule that is able to remain stable in the presence of variations in the hoist transportation times.

With the problem described above, it is preferable to obtain a cyclic hoist schedule that optimizes both the cycle time and the robustness with respect to all related constraints. To this end, a bi-objective MILP model is developed to optimize the cycle time and the robustness simultaneously. The developed bi-objective model will serve as a framework to minimize the cycle time for a given value of robustness, as done in this chapter, or maximize the robustness

for a specific cycle time according to the preference of a decision-maker. We also show that the obtained solutions are Pareto optimal in both cases. Before proceeding, the measurement of the robustness for a given cyclic hoist schedule is defined and analyzed.

3.3 Measuring the robustness for a cyclic hoist schedule

In this study, the robustness of a given cyclic hoist schedule is defined as the schedule's ability to remain stable in the presence of a certain degree of delays in the hoist transportation times. With this definition, the robustness of a given planned cyclic hoist schedule (referred to as the initial schedule) can be measured as the free slacks in the transportation times of loaded and empty moves within which the initial hoist schedule can still be implemented without altering its corresponding cycle time and without violating the processing time-windows. As the delays occurred during loaded moves and that occurred during empty moves influence the hoist schedule in different ways, we will separately consider their influences on a given cyclic hoist schedule $(T, s_{[0]}, s_{[1]}, \ldots, s_{[W]})$ in the following analysis.

We first consider the delay occurred during an empty move and analyze its influence on the cyclic hoist schedule. Note that tanks $M_{[i]}$ and $M_{[i]+1}$ are the departure tank and the destination tank of move $O_{[i]}$, respectively. After executing loaded move $O_{[i]}$, the hoist will perform empty move $G_{[i]}$ by traveling from tank $M_{[i]+1}$ to tank $M_{[i+1]}$. If empty move $G_{[i]}$ is delayed, it may affect the start time of its subsequent move, i.e., loaded move $O_{[i+1]}$. We define the free slack $\bar{q}_{[i]}$ as the amount of time that empty move $G_{[i]}$ can delay without affecting the start of move $O_{[i+1]}$. That is to say, as long as the time required for executing empty move $G_{[i]}$ is not more than $E_{[i]+1,[i+1]} + \bar{q}_{[i]}$ (i.e. with a delay not more than $\bar{q}_{[i]}$), loaded move $O_{[i+1]}$ can still start at its expected time $s_{[i+1]}$ and consequently, the initial hoist schedule can still be continued without any change. In what follows, we derive the free slack $\bar{q}_{[i]}$ for each empty move $G_{[i]}$ of a given cyclic hoist schedule $(T, s_{[0]}, s_{[1]}, \ldots, s_{[W]})$.

Note that if no random event occurs, the hoist will arrive at tank $M_{[i+1]}$ at time $s_{[i]} + D_{[i]} + E_{[i]+1,[i+1]}$ upon the completion of move $O_{[i]}$. To guarantee that loaded move $O_{[i+1]}$ can be executed at its expected time, i.e., $s_{[i+1]}$, the free slack for empty move $G_{[i]}$ is $s_{[i+1]} - (s_{[i]} + D_{[i]} + E_{[i]+1,[i+1]})$. Hence, we have:

$$\bar{q}_{[i]} = s_{[i+1]} - s_{[i]} - D_{[i]} - E_{[i]+1,[i+1]}, \text{ for all } 0 \leq i \leq W-1 \tag{3.3}$$

It is worth noting that if $s_{[i+1]} = s_{[i]} + D_{[i]} + E_{[i]+1,[i+1]}$ for some empty move $G_{[i]}$, the associated $\bar{q}_{[i]}$ would be zero. That is, no delay is allowed for empty move $G_{[i]}$. In particular, after loaded move $O_{[W]}$, the last in a cycle, is completed, the hoist must have sufficient time to travel back to $M_{[0]}$ to execute the first loaded move of the next cycle. Hence, we have:

$$\bar{q}_{[W]} = T - \left(s_{[W]} + D_{[W]} + E_{[W]+1,[0]}\right) \tag{3.4}$$

Now, we consider the delay during a loaded move and analyze its influence on the cyclic hoist schedule. If the hoist spends more time than scheduled to perform loaded move $O_{[i]}$ due

to random events, the hoist would arrive at tank $M_{[i]+1}$ later than expected. On the one hand, the delay will affect the processing time of the part in tank $M_{[i]+1}$ by delaying the start time of its processing. On the other hand, it can also affect the start time of loaded move $O_{[i+1]}$ by delaying the start time of empty move $G_{[i]}$, which can further affect the actual processing time in tank $M_{[i+1]}$. To handle this case, we define free slack $\bar{p}_{[i]}$ as the amount of time that loaded move $O_{[i]}$ can vary without violating the processing time-window in tank $M_{[i]+1}$ and without delaying the start of move $O_{[i+1]}$.

For a given hoist schedule $(T, s_{[0]}, s_{[1]}, \ldots, s_{[W]})$, we first derive the free slack for loaded move $O_{[i]}$ without violating the processing time-window in tank $M_{[i]+1}$. If no random event occurs, a part will arrive at tank $M_{[i]+1}$ at time $s_{[i]} + D_{[i]}$ and departs from the tank at time $s_{[i]+1}$. Hence, the actual processing time of the part in tank $M_{[i]+1}$ is $s_{[i]+1} - s_{[i]} - D_{[i]}$ if move $O_{[i]+1}$ is executed after move $O_{[i]}$ within a cycle; and the actual processing time is $T + s_{[i]+1} - s_{[i]} - D_{[i]}$ otherwise. To guarantee the actual processing time in tank $M_{[i]+1}$ is within its required time-window $[L_{[i]+1}, U_{[i]+1}]$, the free slack for loaded move $O_{[i]}$ is $s_{[i]+1} - s_{[i]} - D_{[i]} - L_{[i]+1}$ in the first case and it is $T + s_{[i]+1} - s_{[i]} - D_{[i]} - L_{[i]+1}$ in the second.

For a given hoist schedule $(T, s_{[0]}, s_{[1]}, \ldots, s_{[W]})$, we now derive the free slack for loaded move $O_{[i]}$ without delaying the start of loaded move $O_{[i+1]}$. Note that in case of no random event, upon the completion of move $O_{[i]}$, the hoist will arrive at tank $M_{[i+1]}$ at time instant $s_{[i]} + D_{[i]} + E_{[i]+1,[i+1]}$. To guarantee move $O_{[i+1]}$ can be executed at its expected time $s_{[i+1]}$, the free slack for loaded move $O_{[i]}$ is $s_{[i+1]} - s_{[i]} - D_{[i]} - E_{[i]+1,[i+1]}$. From the above analysis, we have:

$$\bar{p}_{[i]} = \min\left(s_{[i]+1} - s_{[i]} - D_{[i]} - L_{[i]+1}, \bar{q}_{[i]}\right), \ \forall 0 \le [i] \le W-1, s_{[i]+1} > s_{[i]}. \tag{3.5}$$

$$\bar{p}_{[i]} = \min\left(T + s_{[i]+1} - s_{[i]} - D_{[i]} - L_{[i]+1}, \bar{q}_{[i]}\right), \ \forall 0 \le [i] \le W-1, s_{[i]+1} < s_{[i]}. \tag{3.6}$$

$$\bar{p}_W = \bar{q}_W. \tag{3.7}$$

According to the definition, the robustness of a given cyclic hoist schedule (the initial schedule) must be less than or equal to the minimum free slack time among all loaded and empty moves. If the delay does not exceed the robustness, the initial hoist schedule can still be executed without altering its associated cycle time and without violating the processing time-window in any tank. Note that $\bar{p}_{[i]} \le \bar{q}_{[i]}$ always holds according to equalities (3.5)–(3.7). Thus the robustness of a given hoist schedule, which is denoted by R, can be formulated as:

$$R = \min_{1 \le i \le W}\left\{\bar{p}_{[i]}, \bar{q}_{[i]}\right\} = \min_{1 \le i \le W}\left\{\bar{p}_{[i]}\right\} \tag{3.8}$$

Note that the robustness considered in this study can be seen as both solution robustness and quality robustness. That is, both the start times for all loaded and empty moves and the cycle time of the schedule will remain unchanged as long as the delay does not exceed the robustness. With the above definition, every loaded move $O_{[i]}$, (resp. every empty move $G_{[i]}$) can be delayed with up to $\bar{p}_{[i]}$ (resp. $\bar{q}_{[i]}$) time units while the initial hoist schedule can still be executed without altering its associated cycle time and without violating the processing

time-windows. However, it should be noted that the delay in a loaded hoist move and that in its subsequent empty hoist move are not independent. From the above analysis, if both loaded move $O_{[i]}$ and empty move $G_{[i]}$ are delayed, their delays cannot exceed $\bar{p}_{[i]}$ and $\bar{q}_{[i]}$, respectively. Furthermore, the sum of their delays should not exceed the value of $\bar{q}_{[i]}$ to guarantee the initial hoist schedule can be implemented as planned.

Table 3.1: Time-windows and move times for the example

Tank	M_0	M_1	M_2	M_3	M_4
L_i (s)	–	60	30	20	30
U_i (s)	–	90	60	40	49
D_i (s)	12	15	17	10	13

We now use an example to illustrate the concept of robustness. Figure 3.1 shows a cyclic hoist schedule for an electroplating line with four processing tanks. In the figure, M_0 and M_5 are the input and output stations, respectively. The empty move times are estimated as $E_{i,j} = E_{j,i} = |i - j|/0.4$ s, for all $0 \le i, j \le 5$. The other parameters for the example are given in Table 3.1.

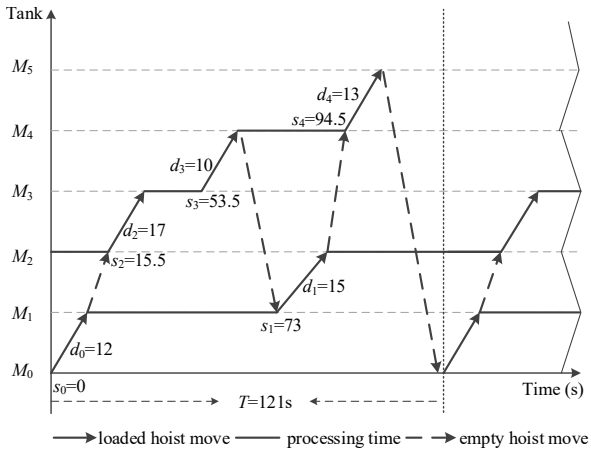

Figure 3.1: A robust cyclic hoist schedule for the example

The cyclic hoist schedule depicted in Figure 3.1 can be represented by ($T = 121$, $s_0 = 0$, $s_2 = 15.5$, $s_3 = 53.5$, $s_1 = 73$, $s_4 = 94.5$). Thus we can have: $[0] = 0$, $[1] = 2$, $[2] = 3$, $[3] = 1$ and $[4] = 4$. That is, the hoist successively performs loaded move O_0, empty move G_0, loaded move O_2, empty move G_2, loaded move O_3, empty move G_3, loaded move O_1, empty move G_1, loaded move O_4, and empty move G_4. Note that empty move G_2 is a dummy move as the hoist just waits at the tank for the completion of the processing after releasing a part into tank M_3.

With equations (3.3) and (3.4), we can obtain the free slack for each empty move as follows: $\bar{q}_0 = 1$s, $\bar{q}_1 = 1.5$s, $\bar{q}_2 = 21$s, $\bar{q}_3 = 2$ s, $\bar{q}_4 = 1$s. With equations (3.5)–(3.7), we can also obtain the free slack for each loaded move as follows: $\bar{p}_0 = 1$s, $\bar{p}_1 = 1.5$s, $\bar{p}_2 = 1$s, $\bar{p}_3 = 1$s, $\bar{p}_4 = 1$s. It can be seen from formula (3.8) that the robustness of the schedule is $R = 1$s. This

means that as long as the delay in any loaded move or empty move does not exceed 1s, the initial hoist schedule can still be executed without any modification. To illustrate this point, we take loaded move O_0 as an example. If O_0 is executed as scheduled, it will complete at time instant 12s, as illustrated in Figure 3.1. In this case, the actual processing time of the part in tank M_1 is $(73 - 12) = 61$s. Upon completion of O_0, the hoist travels to tank M_2 and arrives at the tank at time 14.5s in order to execute move O_3 at time instant 15.5s. Suppose that O_0 is delayed by 1s. In this case, the hoist will complete loaded move O_0 at time instant 13s. Thus the actual processing time of the part in tank M_1 becomes $(73 - 13) = 60$s, which is still within its required time-window $[60, 90]$. Furthermore, due to this delay, the hoist will depart from tank M_1 at time instant 13s, instead of 12s as scheduled. As a result, it will arrive at tank M_2 at time 15.5s. As move O_2 starts at time 15.5s, the initial schedule can still be executed without any modification.

3.4 Problem formulation

Let p_i and q_i be actual delays in the execution of loaded move O_i and empty move G_i, respectively. To formulate the model, we define the following binary variables.

$y_{i,j}$: binary variables. $y_{i,j} = 1$ if move O_j is executed after move O_i within a cycle; otherwise, $y_{i,j} = 0$, for all $1 \leq i \leq j \leq W$.

The considered cyclic hoist scheduling problem can be formulated as the following MILP model whose constraints will be explained in detail later.

P:

$$\text{Minimize} \quad T \tag{3.9}$$

$$\text{Maximize} \quad R \tag{3.10}$$

subject to:

Processing time-window constraints considering robustness:

$$s_1 - D_0 \geq L_1 + p_0. \tag{3.11}$$

$$s_1 - D_0 \leq U_1. \tag{3.12}$$

$$s_i - s_{i-1} - D_{i-1} \geq L_i + p_{i-1} - M\left(1 - y_{i-1,i}\right),\ \forall 2 \leq i \leq W. \tag{3.13}$$

$$s_i - s_{i-1} - D_{i-1} \leq U_i + M\left(1 - y_{i-1,i}\right),\ \forall 2 \leq i \leq W. \tag{3.14}$$

$$T + s_i - s_{i-1} - D_{i-1} \geq L_i + p_{i-1} - My_{i-1,i},\ \forall 2 \leq i \leq W. \tag{3.15}$$

$$T + s_i - s_{i-1} - D_{i-1} \leq U_i + My_{i-1,i},\ \forall 2 \leq i \leq W. \tag{3.16}$$

Hoist capacity constraints considering robustness:

$$s_0 = 0. \tag{3.17}$$

$$s_i \geq D_0 + E_{1,i} + q_0,\ \forall 1 \leq i \leq W. \tag{3.18}$$

$$T \geq s_i + D_i + E_{i+1,0} + q_i, \ \forall 1 \leq i \leq W. \tag{3.19}$$

$$s_i + D_i + E_{i+1,j} + q_i \leq s_j + M\left(1 - y_{i,j}\right), \ \forall 1 \leq i \leq j \leq W. \tag{3.20}$$

$$s_j + D_j + E_{j+1,i} + q_j \leq s_i + M y_{i,j}, \ \forall 1 \leq i \leq j \leq W. \tag{3.21}$$

Other constraints:

$$p_i \leq q_i, \ \forall 0 \leq i \leq W. \tag{3.22}$$

$$R \leq p_i, \ \forall 0 \leq i \leq W. \tag{3.23}$$

$$y_{i,j} \in \{0, 1\}, \ \forall 1 \leq i \leq j \leq W. \tag{3.24}$$

The processing time-window constraints are associated with each processing tank and ensure the actual processing time of each part in each tank must fall into its prescribed time-window. Violation of time-window constraints can cause defective parts. The processing time-window constraints without considering robustness can be formulated as (3.25)–(3.30) (Chen et al., 1998; Liu et al., 2002; Phillips & Unger, 1976).

$$s_1 - D_0 \geq L_1 \tag{3.25}$$

$$s_1 - D_0 \leq U_1 \tag{3.26}$$

$$s_i - s_{i-1} - D_{i-1} \geq L_i - M\left(1 - y_{i-1,i}\right), \ \forall 2 \leq i \leq W. \tag{3.27}$$

$$s_i - s_{i-1} - D_{i-1} \leq U_i + M\left(1 - y_{i-1,i}\right), \ \forall 2 \leq i \leq W. \tag{3.28}$$

$$T + s_i - s_{i-1} - D_{i-1} \geq L_i - M y_{i-1,i}, \ \forall \leq i \leq W. \tag{3.29}$$

$$T + s_i - s_{i-1} - D_{i-1} \leq U_i + M y_{i-1,i}, \ \forall \leq i \leq W. \tag{3.30}$$

If the execution of loaded move O_{i-1} is delayed, it can affect the start time of the processing in tank M_i, and consequently the actual processing time may violate the time-window constraints, as explained in Section 3.3. To ensure the time-window for tank M_i be respected, the delay in the execution of move O_{i-1} must satisfy (3.31)–(3.33).

$$p_0 \leq s_1 - D_0 - L_1. \tag{3.31}$$

$$p_{i-1} \leq s_i - s_{i-1} - D_{i-1} - L_i, \ \forall 2 \leq i \leq W, \text{ if } y_{i-1,i} = 1. \tag{3.32}$$

$$p_{i-1} \leq T + s_i - s_{i-1} - D_{i-1} - L_i, \ \forall 2 \leq i \leq W, \text{ if } y_{i-1,i} = 0. \tag{3.33}$$

Constraints (3.31)–(3.33) can be derived from (3.5)–(3.7) and the fact that $p_i \leq \bar{p}_i$ for all $0 \leq i \leq W$. From (3.25)–(3.33), we can obtain the time-window constraints (3.11)–(3.16) with robustness considered.

Hoist capacity constraints ensure that the hoist not be required to execute more than one move at a time. Without loss of generality, loaded move O_0 is assumed to be executed at the start of a cycle. Since all the moves are executed by the hoist, hoist capacity constraints ensure no conflict occurs in the use of the hoist between any pair of moves at any time. In particular, any loaded move O_i, $1 \leq i \leq W$, cannot start until loaded move O_0 is completed. After the last loaded move in a cycle is completed, the hoist must have sufficient time to travel back

to M_0 where it will execute the first loaded move of the subsequent cycle. The basic hoist capacity constraints without considering robustness can be formulated as (3.34)–(3.38) (Chen et al., 1998; Liu et al., 2002; Phillips & Unger, 1976).

$$s_0 = 0. \tag{3.34}$$

$$s_i + D_i + E_{i+1,j} \leq s_j + M\left(1 - y_{i,j}\right),\ \forall 1 \leq i \leq j \leq W. \tag{3.35}$$

$$s_j + D_j + E_{j+1,i} \leq s_i + M y_{i,j},\ \forall 1 \leq i \leq j \leq W. \tag{3.36}$$

$$s_i \geq D_0 + E_{1,i},\ \forall 1 \leq i \leq W. \tag{3.37}$$

$$T \geq s_i + D_i + E_{i+1,0},\ \forall 1 \leq i \leq W. \tag{3.38}$$

If the execution of an empty move is delayed, it may affect its subsequent loaded move. To guarantee that the execution of the loaded move starts at expected time, the delay during the empty move must satisfy:

$$q_0 \leq s_i - D_0 - E_{1,i},\ \forall 1 \leq i \leq W. \tag{3.39}$$

$$q_i \leq s_j - s_i - D_i - E_{i+1,j},\ \forall 1 \leq i \leq j \leq W,\ \text{if } y_{i,j} = 1. \tag{3.40}$$

$$q_i \leq s_i - s_j - D_j - E_{j+1,i},\ \forall 1 \leq i \leq j \leq W,\ \text{if } y_{i,j} = 0. \tag{3.41}$$

$$q_i \leq T - \left(s_i + D_i + E_{i+1,0}\right),\ \forall 1 \leq i \leq W. \tag{3.42}$$

Constraints (3.39)–(3.41) can be derived from constraints (3.1)–(3.3) and the fact that $q_{[i]} \leq \bar{q}_{[i]}$, $0 \leq i \leq W-1$, while constraint (3.42) can be derived from (3.1)–(3.4) and the fact that $q_{[W]} \leq \bar{q}_{[W]}$. With constraints (3.34)–(3.42), we can obtain the hoist capacity constraints (3.17)–(3.21) with robustness considered.

As explained above, both the delay for a loaded move and that for an empty move can affect the hoist schedule, and these two types of delays are not independent. Thus we have inequality (3.22) by considering constraints (3.5)–(3.7), while constraint (3.23) can be derived from (3.8) and (3.23). Constraint (3.23), together with inequality (3.22), means that the robustness of a cyclic hoist schedule is the minimum among all free slacks for loaded and empty moves. In addition, formula (3.24) indicates the values of binary variables.

Note that the robustness of a cyclic hoist schedule is measured as the minimum value among all free slacks for loaded and empty moves. We may define other measures of the robustness. For example, the robustness can be defined as the total sum of all the free slacks. In that case, it is sufficient to replace equation (3.23) with $R = \sum_{i=0}^{W}(p_i + q_i)$ while the other part of the model remains the same.

3.5 Analysis of the MILP model

As the schedule with a larger robustness may lead to a longer cycle time, the two objectives of the proposed MILP model (i.e. cycle time and robustness) may be conflicting. To derive the relation between them, we first define a new problem as follows.

$P(R_0)$:

$$\text{Minimize} \quad T$$

subject to:

$$R = R_0 \tag{3.43}$$

and constraints (3.11)–(3.24). Problem $P(R_0)$ aims to minimize the cycle time for a given robustness R_0. Let $T^*(R_0)$ be the optimal cycle time of $P(R_0)$. With the new defined problem, we can have the following Proposition.

Proposition 3.1. *The optimal cycle time of $P(R_0)$ strictly increases with R_0, i.e. for any two values R_1 and R_2, if $R_2 > R_1$, then $T^*(R_2) > T^*(R_1)$ always holds.*

Proof. We first show that if $R_2 > R_1$, then $T^*(R_2) \geq T^*(R_1)$ always holds. For any two robustness values R_1 and R_2, if $R_2 > R_1$, then constraint (3.23) of problem $P(R_2)$ is tighter than that of $P(R_1)$ while the other constraints of the two problems remain the same. Thus, the optimal cycle time of $P(R_2)$ is greater than or equal to that of $P(R_1)$, i.e., $T^*(R_2) \geq T^*(R_1)$. We now show that in this case, $T^*(R_2) = T^*(R_1)$ is impossible by contradiction. Assume that $T^*(R_2) = T^*(R_1)$. Let $\delta = R_2 - R_1$. Suppose that $(T^*(R_2), s_{[0]}, s_{[1]}, \ldots, s_{[W]})$ is an optimal cyclic hoist schedule for $P(R_2)$ with robustness R_2. From this schedule, we can always construct a new cyclic hoist schedule $(T^*(R_2) - \delta, s_{[0]} - \delta, s_{[1]} - \delta, \ldots, s_{[W]} - \delta)$ with robustness $R_2 - \delta$. According to the constraints of $P(R_0)$ and the definition of robustness, we can verify that the new hoist schedule $(T^*(R_2) - \delta, s_{[0]} - \delta, s_{[1]} - \delta, \ldots, s_{[W]} - \delta)$ satisfy all constraints of $P(R_1)$, and it is a feasible schedule with robustness $R_2 - \delta$ (i.e., R_1). By assumption $T^*(R_2) = T^*(R_1)$, it means we would obtain a feasible cyclic hoist schedule whose cycle time and robustness are $T^*(R_1) - \delta$ and R_1 respectively. This is in contradiction with the fact that $T^*(R_1)$ is the optimal cycle time for all cyclic hoist schedules with robustness R_1. Hence, it is impossible to have $T^*(R_2) = T^*(R_1)$ in this case. Consequently, if $R_2 > R_1$, then $T^*(R_2) > T^*(R_1)$ always holds. □

As R is a continuous (real) variable, we can have the following corollary.

Corollary 3.2. *Problem P has infinite number of Pareto optimal solutions.*

With Proposition 3.1 and Corollary 3.2, a Pareto optimal solution can be obtained by solving a single-objective optimization problem to minimize the cycle time for a given value of robustness or maximize the robustness for a specific cycle time. To derive the Pareto front, it is important for a decision-maker to know the lower and upper bounds of the robustness and the cycle time for a given instance. In what follows, we will deal with the problem of how to obtain lower and upper bounds of the robustness for a given instance.

Let R^L and R^U be a lower bound and an upper bound of R, respectively. We give in Proposition 3.3 an upper bound of R.

Proposition 3.3. $\min_{1\le i\le W}(U_i - L_i)$ *is a valid upper bound of* R *for any feasible cyclic hoist schedule.*

Proof. Given a feasible cyclic hoist schedule $(T, s_{[0]}, s_{[1]}, \ldots, s_{[W]})$, we first consider the case that $s_{[i]} < s_{[i]+1}$, $1 \le [i] \le W$. That is, move $O_{[i]+1}$ is executed after move $O_{[i]}$ within a cycle, $0 \le [i] \le W-1$. By considering (3.5), we have the following equation:

$$p_{[i]} \le s_{[i]+1} - s_{[i]} - D_{[i]} - L_{[i]+1},\ \forall 0 \le [i] \le W-1. \tag{3.44}$$

Furthermore, the processing time in tank $M_{[i]+1}$ must be less than or equal to its upper bound $U_{[i]+1}$ for a feasible schedule. Thus, we have:

$$s_{[i]+1} - s_{[i]} - D_{[i]} \le U_{[i]+1},\ \forall 0 \le [i] \le W-1. \tag{3.45}$$

With inequalities (3.44) and (3.45), we must have:

$$p_{[i]} \le U_{[i]+1} - L_{[i]+1},\ \forall 0 \le [i] \le W-1. \tag{3.46}$$

In a similar way, we can show that (3.46) also holds for the case that $s_{[i]} > s_{[i]+1}$. It follows from (3.23) and (3.46) that $R \le \min_{1\le i\le W}(U_i - L_i)$. Thus, the upper bound of R can be set as $R^U = \min_{1\le i\le W}(U_i - L_i)$. □

Corollary 3.4. $\left(T^*(R^U), R^U\right)$ *is a Pareto optimal point in the objective space for problem* P.

With Proposition 3.3 and Corollary 3.4, we can also know that R^U is a tight upper bound for R. We now derive a lower bound of R. First, let us consider the following standard non-robust cyclic hoist scheduling problem:
P^*:

$$\text{Minimize} \quad T$$

subject to constraints (3.24)–(3.30) and (3.34)–(3.38).

Let $\left(T^*, s_{[0]^*}, s_{[1]^*}, \ldots, s_{[W]^*}\right)$ be an optimal cyclic schedule for P^* and R^* the robustness of the schedule. We can have the following Proposition 3.5.

Proposition 3.5. $R^* = 0$ *holds for schedule* $\left(T^*, s^*_{[0]}, s^*_{[1]}, \ldots, s^*_{[W]}\right)$ *of* P^*.

Proof. First, it is quite clear that $R^* \ge 0$. In the following, we demonstrate by contradiction that $R^* > 0$ is impossible. To this end, we assume that $R^* > 0$. With a similar way described in the proof of Proposition 3.1, we can obtain a new schedule $\left(T^* - R^*, s^*_{[0]} - R^*, s^*_{[1]} - R^*, \ldots, s^*_{[W]} - R^*\right)$. According to the constraints of P^* and the definition of robustness, the obtained new schedule satisfies all constraints of P^*. This means that there would exist a feasible schedule for problem P^* with cycle time $T^* - R^* < T^*$, which is in contradiction with the fact that T^* is the optimal cycle time of P^*. Hence, the robustness of the optimal schedules for P^* must be zero, which also implies that zero is a tight lower bound of R. □

Corollary 3.6. $(T^*, 0)$ *is a Pareto optimal point in the objective space for problem P.*

As mentioned above, a decision-maker may also want to maximize the robustness for a specific cycle time. In this case, it is desirable to know the lower and upper bounds of the cycle time for a given instance, which can be determined by the following proposition.

Proposition 3.7. *It is sufficient to restrict the cycle time within the closed interval* $\left[T^*, T^*(R^U)\right]$ *in order to obtain the Pareto front of problem P.*

Proof. First, it is obvious that the lower bound of the cycle time is T^* since it is the minimum cycle time obtained by solving the single-objective optimization problem P^*. Note that $T^*\left(R^U\right)$ is the minimum cycle time corresponding to robustness R^U. As R^U is an upper bound of the robustness, any schedule whose cycle time is greater than $T^*\left(R^U\right)$ must be dominated by any schedule corresponding to $\left(T^*\left(R^U\right), R^U\right)$. Thus, to obtain the Pareto front, it is unnecessary to consider the values of the cycle time greater than $T^*\left(R^U\right)$. □

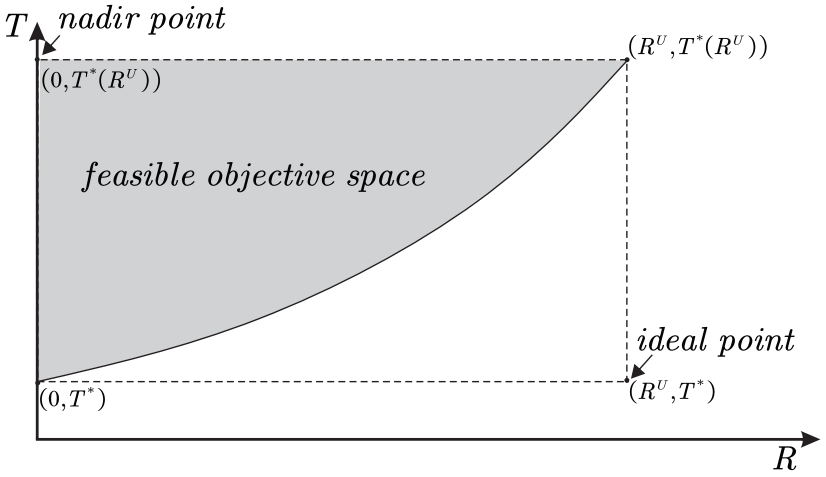

Figure 3.2: Illustration of the ideal and nadir points

Now we can define the so-called *ideal point* and *nadir point* in the objective space, which define the lower and upper bounds for the objective function values of Pareto front. The ideal point (objective vector) is composed of the optimal values for all objective functions when a single objective is optimized. The nadir point (objective vector) represents the upper bound of each objective function to be minimized or the lower bound of each objective to be maximized in the Pareto front (not in the entire solution space). In other words, a nadir point and an ideal point together define upper and lower bounds on the objective function values of Pareto front (Branke, Deb, Miettinen, & Slowinski, 2008).

According to the above analysis, the ideal and nadir points of P are $\left(T^*, R^U\right)$ and $\left(T^*\left(R^U\right), 0\right)$, respectively, as shown in Figure 3.2. In the figure, the region above the solid curve is the feasible objective space. With Proposition 3.7, the Pareto front is located in the shadow region of Figure 3.2.

Furthermore, we can obtain Proposition 3.8.

Proposition 3.8. *Problem $P(R_0)$ can be transformed into a standard non-robust cyclic hoist scheduling problem (i.e., problem P^*) with a new loaded move times $D'_i = D_i + R_0$ for each move O_i, $0 \le i \le W$, and a new upper bound $U'_i = U_i - R_0$ for the processing time in each tank M_i, $1 \le i \le W$.*

Proof. With constraints (3.22) and (3.23), we can substitute all p_i's and q_i's with R_0 in the $P(R_0)$ without changing the feasible solution space. After this substitution, we can replace all $D_i + R_0$ in constraints (3.11), (3.13), (3.15), and (3.18)–(3.21) with D'_i. Note that the triangular inequalities still hold and these constraints with D'_i are equivalent to constraints (3.25), (3.27), (3.29), and (3.35)–(3.38) of P^* with new loaded move times D'_i.

We now consider constraints (3.12), (3.14) and (3.16). Take constraint (3.14) for example, it is equivalent to the following inequality:

$$s_i - s_{i-1} - (D_{i-1} + R_0) \le U_i - R_0 + M\left(1 - y_{i-1,i}\right), \ \forall 2 \le i \le W. \tag{3.47}$$

By substituting $D_{i-1} + R_0$ and $U_i - R_0$ in (3.47) with D'_{i-1} and U'_i, respectively, (3.14) is equivalent to (3.28) with new loaded move times D'_i and new upper bounds of processing times U'_i. Similarly, we can show that (3.12) and (3.16) are equivalent to (3.26) and (3.30) respectively with D'_i and U'_i. After the above transformation, problem $P(R_0)$ is equivalent to a standard non-robust cyclic hoist scheduling problem that minimizes the cycle time T subjecting to (3.25)–(3.30) and (3.34)–(3.38) with new loaded move times D'_i and new upper bounds of processing times U'_i. □

With Proposition 3.8, problem $P(R_0)$, which aims to minimize the cycle time for a given robustness R_0, can be solved with any of existing various approaches for classical non-robust cyclic hoist scheduling problems in the literature, such as Chen et al. (1998), Lim (1997), and Yan et al. (2012). However, the proposed bi-objective MILP model is still valuable for the following reasons. Firstly, in some circumstances, a decision-maker may want to maximize the robustness for a specific cycle time. In that case, the proposed bi-objective MILP model can be reduced to a single-objective MILP model that maximizes the robustness for a given cycle time. Secondly, as mentioned above, the robustness may be defined in an alternative way such as the sum of all the free slacks $\bar{p}_{[i]}$ and $\bar{q}_{[i]}$, $0 \le i \le W$, in some situations. In these cases, the corresponding problem can no longer be transformed into a standard non-robust cyclic hoist scheduling problem in the same way. However, with the proposed bi-objective MILP model, we can obtain the Pareto front of the problem with multi-objective optimization approaches such as ε–constraint method (Kirlik & Sayın, 2014; Zhang & Reimann, 2014).

3.6 Computational results

In this section, we use six well-known benchmark instances as well as a set of randomly generated instances to evaluate the proposed approach. The instances are solved by CPLEX

(Version 12.5) via its C++ API. The default parameters setting of CPLEX is used for all instances. The computation is done on a personal computer with a 3.1 GHZ CPU and 4.0 GB RAM. Each instance is solved to guaranteed optimality.

3.6.1 Benchmark instances

The six benchmark instances are called *Phillips*, *Ligne*1, *Ligne*2, *Bo*1, *Bo*2 and *Mini*, respectively. Their data can be obtained from Phillips and Unger (1976), Manier and Lamrous (2008) and Leung et al. (2004). The computational results are given in Table 3.2. From the table, we see that the theoretical upper bound of R ranges from 10 to 60 for these instances. Note that this is a theoretical upper bound of R. In industrial applications, the data about the variations in the hoist travel times can be obtained from historical data. The actual upper bound of R is usually determined based on both its theoretical upper bound and instance data. In our experiments, the actual upper bound of R is set as 10. We continually solved problem $P(R)$ for $R = 0, 1, \ldots, 10$ and obtained 11 Pareto optimal solutions for each instance. With Proposition 3.8, problem $P(R)$ is solved by dealing with its corresponding standard non-robust cyclic hoist scheduling problem (i.e., problem P^*) with $D'_i = D_i + R$ and $U'_i = U_i - R$.

Table 3.2: Cycle times for the benchmark instances

Instance	*Phillips*	*Ligne*1	*Ligne*2	*Bo*1	*Bo*2	*Min*
$R = 0$	521.00	392.00	712.00	281.90	279.30	287.00
$R = 1$	566.00	426.00	714.00	305.40	281.30	295.00
$R = 2$	576.00	436.00	716.00	335.00	325.00	303.00
$R = 3$	679.00	446.00	718.00	341.10	329.00	402.00
$R = 4$	690.00	456.00	720.00	356.90	337.00	407.00
$R = 5$	701.00	468.00	722.00	378.90	342.40	412.00
$R = 6$	712.00	494.00	724.00	384.90	364.40	417.00
$R = 7$	723.00	507.00	726.00	390.90	375.40	422.00
$R = 8$	734.00	523.00	749.00	406.70	386.40	427.00
$R = 9$	807.00	534.00	761.00	413.60	397.40	432.00
$R = 10$	816.00	545.00	773.00	423.00	408.40	437.00

Table 3.3: Computation times for the benchmark instances (in CPU seconds)

Instance	*Phillips*	*Ligne*1	*Ligne*2	*Bo*1	*Bo*2	*Min*
$R = 0$	0.289	0.321	0.227	0.240	0.192	0.149
$R = 1$	0.277	0.350	0.250	0.203	0.209	0.193
$R = 2$	0.232	0.273	0.191	0.259	0.266	0.135
$R = 3$	0.251	0.327	0.191	0.220	0.270	0.150
$R = 4$	0.250	0.386	0.200	0.219	0.280	0.147
$R = 5$	0.254	0.336	0.224	0.192	0.229	0.128
$R = 6$	0.250	0.279	0.281	0.217	0.267	0.137
$R = 7$	0.254	0.266	0.275	0.189	0.234	0.129
$R = 8$	0.188	0.263	0.261	0.204	0.233	0.167
$R = 9$	0.226	0.255	0.254	0.228	0.220	0.166
$R = 10$	0.220	0.201	0.264	0.171	0.222	0.145

From Table 3.2, we can see the cycle time T increases with the robustness R which verifies Proposition 3.1. Table 3.3 shows the computation times for the instances, and indicates that these benchmark instances can be solved in very short time.

We now examine in detail how the cycle time varies when the robustness changes. For this purpose, we uniformly divide the robustness interval $[0, 10]$ with 1000 points; that is, there is a step of size 0.01 between two adjacent points. Then, we solve a set of problems $P(R)$ for $R = 0, 0.01, 0.02, \ldots, 10$ and obtain 1001 Pareto optimal solutions for each instance. The objective values for the obtained Pareto optimal solutions are plotted in Figure 3.3, which illustrates how the optimal cycle time changes as the value of R increases.

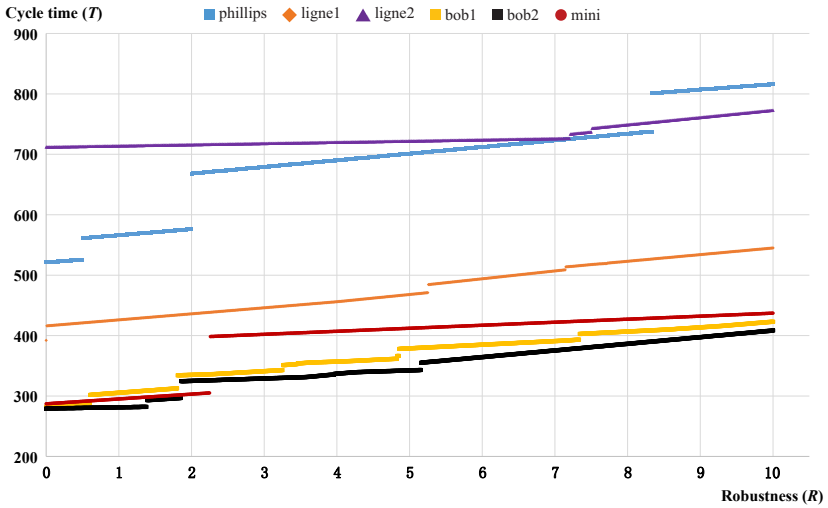

Figure 3.3: Partial Pareto fronts for the benchmark instances

From Figure 3.3, it is interesting to see that the optimal cycle time T is a piecewise linear increasing function of the robustness R. That is, the optimal cycle time increases linearly with the robustness in some interval, and it increases discontinuously at some special points. This phenomenon can be explained as follows.

To simplify the discussion, we analyze how the optimal cycle time for P^* varies when the robustness increases from $R = 0$ to $R = \delta$, where δ is an arbitrarily small positive value. As mentioned above, the cyclic hoist schedule $\left(T^*, s^*_{[0]}, s^*_{[1]}, \ldots, s^*_{[W]}\right)$, whose robustness is zero, is an optimal schedule of problem P^*. Let $t^*_{[1]}, t^*_{[2]}, \ldots, t^*_{[W]}$ be its corresponding actual processing times in tanks $M_{[1]}, M_{[2]}, \ldots, M_{[W]}$, respectively.

We now consider the cyclic hoist scheduling problem with fixed processing times $t^*_{[1]}, t^*_{[2]}, \ldots, t^*_{[W]}$. According to Levner et al. (1997), the optimal cycle time for the problem can be formulated as a set of prohibited intervals of the cycle time as follows.

$$T \notin \mathbf{Q} = \left\{ (-\infty, \beta) \cup \bigcup_{j=0}^{W-1} \bigcup_{i=j+1}^{W} \bigcup_{k=1}^{W} \left(\frac{Z_i - Z_j - D_j - E_{j+1,i}}{k}, \frac{Z_i - Z_j + D_i + E_{i+1,j}}{k} \right) \right\}$$

where $\beta = \max_{1 \le i \le W} \left(t^*_i + D_i + E_{i+1,i-1} + d_{i-1} \right)$, $Z_i = \sum_{j=1}^{i} \left(d_{j-1} + t^*_j \right)$, $1 \le i \le W$.

As demonstrated by Levner et al. (1997), **Q** is a union of open prohibited intervals which are not necessarily disjoint. However, **Q** can be considered as a union of disjoint prohibited intervals after merging the intersecting ones. With the above formulation, it can be proved that the optimal cycle time T^* is necessarily the upper bound of the first open disjoint prohibited interval (Levner et al., 1997). Note also that the upper bound of any disjoint prohibited interval, after merging, is necessarily an upper bound of one of the prohibited intervals before merging. This means that the optimal cycle time satisfies:

$$T^* \in \{\beta\} \cup \left\{ \frac{Z_i - Z_j + D_i + E_{i+1,j}}{k}, 0 \le j \le W-1, j+1 \le i \le W, 1 \le k \le W \right\}$$

Furthermore, we can also have:

$$\beta = \max_{1 \le i \le W} \left(t_i^* + D_i + E_{i+1,i-1} + D_{i-1} \right) = \max_{1 \le i \le W} \left(Z_i - Z_{i-1} + D_i + E_{i+1,i-1} \right)$$

By combining the above sets, we have:

$$T^* \in \left\{ \frac{Z_i - Z_j + D_i + E_{i+1,j}}{k}, 0 \le j \le W-1, j+1 \le i \le W, 1 \le k \le W \right\}$$

Without loss of generality, we assume that $T^* = \left(Z_{i^*} - Z_{j^*} + D_{i^*} + E_{i^*+1,j^*} \right) / k^*$ for some i^*, j^* and k^*, which is called the optimal upper bound hereafter. Note that the robustness of schedule $\left(T^*, s^*_{[0]}, s^*_{[1]}, \ldots, s^*_{[W]} \right)$ is zero. There are two possible cases concerning the optimal cycle time as the robustness increases from $R = 0$ to $R = \delta$:

1. The optimal upper bound for $R = 0$ is still the optimal upper bound for $R = \delta$. This implies that $t^*_{[1]}$, $t^*_{[1]}$, …, $t^*_{[W]}$ are still the optimal processing times in tanks $M_{[1]}$, $M_{[2]}$, …, $M_{[W]}$ for the problem with $R = \delta$, respectively. Consequently, the new optimal cycle time can be represented by $T^\delta = \left(Z_{i^*} - Z_{j^*} + D_{i^*} + E_{i^*+1,j^*} \right) / k^* + (i^* - j^* + 1)\,\delta / k^* = T^* + (i^* - j^* + 1)\,\delta / k^*$. As a result, the increase of the optimal cycle time, i.e. $(i^* - j^* + 1)\,\delta / k^*$, is a linear function of δ.

2. The optimal upper bound for $R = 0$ is no longer the optimal upper bound for $R = \delta$. In this case, the initial optimal cycle time falls in some prohibited intervals and becomes infeasible when the robustness increases from $R = 0$ to $R = \delta$. As a result, $t^*_{[1]}$, $t^*_{[2]}$, …, $t^*_{[W]}$ are not necessarily the actual processing times in tanks $M_{[1]}$, $M_{[2]}$, …, $M_{[W]}$ for $R = \delta$, respectively. In this case, the optimal cycle time may suffer from a discontinuous increasing when the robustness increases from $R = 0$ to $R = \delta$.

3.6.2 Randomly generated instances

In addition, three sets of randomly generated instances are used to further test the proposed approach. The instances are generated using integer uniform distributions. Let $\mathbf{U}(a, b)$ denote the uniform distribution with lower bound a and upper bound b. The parameters used to generate the first set of random instances are: $L_i = 40 + \mathbf{U}(0, 140)$, $U_i = [1.5L_i]$, $E_{i,i+1} = 1 + \mathbf{U}(0, 4)$,

Table 3.4: Computational results for the randomly generated instances

	W	12	14	16	18	20	22
R^U	Set 1	24.5	23.9	23.6	24.4	24.1	22.3
	Set 2	49.5	48.3	47.6	49.2	48.5	45.1
	Set 3	74.0	72.2	71.2	73.6	72.6	67.4
$R = 0$	Set 1	365.20	451.80	579.70	704.30	945.10	1187.80
	Set 2	303.80	352.20	431.40	504.60	623.00	744.00
	Set 3	285.90	328.30	431.40	446.90	538.50	614.20
$R = 1(\%)$	Set 1	6.63	3.21	10.30	6.28	3.96	8.07
	Set 2	4.34	6.70	7.23	8.22	7.95	6.98
	Set 3	5.00	5.39	7.23	5.24	5.66	6.09
$R = 2(\%)$	Set 1	13.06	13.63	17.79	22.04	10.26	17.20
	Set 2	9.28	12.81	12.63	13.99	12.84	10.15
	Set 3	9.86	11.12	12.63	11.10	12.05	10.73
$R = 3(\%)$	Set 1	20.89	17.40	30.33	25.71	26.27	27.77
	Set 2	17.31	17.58	19.98	19.68	18.80	15.50
	Set 3	14.94	16.24	19.98	18.42	17.38	17.29
$R = 4(\%)$	Set 1	29.71	26.60	42.02	33.55	30.93	33.38
	Set 2	21.79	23.59	24.97	26.42	23.88	24.29
	Set 3	20.22	22.81	24.97	25.15	22.92	24.68
$R = 5(\%)$	Set 1	39.07	34.93	45.32	46.51	39.64	36.63
	Set 2	26.14	29.27	28.86	32.96	28.70	30.39
	Set 3	26.09	28.45	28.86	31.01	27.58	30.58
$R = 6(\%)$	Set 1	43.18	40.55	53.23	53.57	55.68	40.65
	Set 2	34.63	35.32	36.56	37.99	31.89	33.53
	Set 3	31.20	33.32	36.56	36.90	33.13	36.75
$R = 7(\%)$	Set 1	52.44	45.42	66.22	59.18	58.72	61.26
	Set 2	40.26	43.47	43.42	45.03	37.05	37.26
	Set 3	36.03	39.26	43.42	43.68	40.07	43.88
$R = 8(\%)$	Set 1	64.07	54.69	76.51	63.60	74.12	78.67
	Set 2	45.46	47.90	50.76	50.42	45.43	45.35
	Set 3	41.97	46.36	50.76	50.57	46.89	48.75
$R = 9(\%)$	Set 1	72.84	63.35	81.27	77.33	77.60	80.27
	Set 2	50.43	53.44	55.96	55.65	54.90	53.00
	Set 3	46.76	50.99	55.96	55.99	53.20	53.13
$R = 10(\%)$	Set 1	93.32	75.08	83.04	91.88	86.97	100.96
	Set 2	56.48	58.72	60.80	62.17	62.01	62.31
	Set 3	52.99	56.38	60.80	61.58	58.01	58.27

$E_{i,j} = E_{j,i} = \sum_{k=i}^{j-1} E_{k,k+1}$, $D_i = E_{i,i+1} + 12$, $0 \le i \le W$, $W \in \{12, 14, 16, 18, 20, 22\}$, where $[x]$ represents the nearest integer of x. The parameters of the second and the third sets of instances are identical to these for the first set, except that the upper bounds of time-windows are set as $U_i = 2L_i$ and $U_i = [2.5L_i]$, respectively. The three sets of instances correspond to different widths of time-windows. For each given value of W, ten instances are generated according to the above method.

For the randomly generated instances, we also set the actual upper bound of R as 10. We solve a set of problems $P(R)$ successively for $R = 0, 1, \ldots, 10$ and obtain 11 Pareto optimal solutions for each instance. The computational results are summarized in Table 3.4. In the table, the rows for $R = 0$ give the average cycle time over ten instances for each pair of parameters W and R, while the rows for $R = 1, 2, \ldots, 10$ are the average increasing ratio of the optimal cycle time for $R = 1, 2, \ldots, 10$ with respect to its corresponding cycle time for $R = 0$, respectively.

We see from Table 3.4 that for each set of instances, the average cycle time increases with robustness R due to Proposition 3.1. Furthermore, after a careful comparison of average increasing ratio of the optimal cycle time of instances of set 2 (resp. set 3) with that of set 1 (resp. set 2), we can see that the average increasing ratio of the cycle time shows a

Table 3.5: Average computation times for the randomly generated instances

W		12	14	16	18	20	22
$R = 0$	Set 1	0.279	0.488	0.740	1.100	0.964	1.011
	Set 2	0.569	2.329	5.593	21.380	17.297	34.864
	Set 3	0.923	5.840	45.508	227.263	889.323	2638.370
$R = 1$	Set 1	0.263	0.490	0.635	0.993	0.766	0.781
	Set 2	0.659	2.226	6.844	20.597	13.921	30.239
	Set 3	1.186	5.889	51.150	340.654	679.715	1004.780
$R = 2$	Set 1	0.283	0.431	0.587	0.862	0.667	0.767
	Set 2	0.739	2.282	8.418	18.174	10.757	17.011
	Set 3	1.189	7.508	56.668	214.219	540.886	698.569
$R = 3$	Set 1	0.259	0.381	0.573	0.734	0.635	0.707
	Set 2	0.642	2.306	5.149	14.952	8.645	15.196
	Set 3	1.259	8.244	56.168	320.760	257.497	885.726
$R = 4$	Set 1	0.239	0.346	0.543	0.571	0.561	0.575
	Set 2	0.615	2.363	4.427	9.274	6.906	13.030
	Set 3	1.544	9.927	41.799	339.184	312.053	663.675
$R = 5$	Set 1	0.243	0.324	0.438	0.609	0.477	0.494
	Set 2	0.581	2.128	3.824	10.126	5.743	8.671
	Set 3	1.471	9.978	44.934	181.326	162.793	355.366
$R = 6$	Set 1	0.218	0.318	0.418	0.511	0.473	0.447
	Set 2	0.622	1.949	3.461	9.248	5.136	7.105
	Set 3	1.556	10.535	39.850	185.375	145.695	491.090
$R = 7$	Set 1	0.226	0.292	0.372	0.453	0.426	0.455
	Set 2	0.540	1.365	2.510	5.040	3.962	4.832
	Set 3	1.553	7.553	31.970	119.587	123.577	221.715
$R = 8$	Set 1	0.215	0.299	0.351	0.403	0.394	0.399
	Set 2	0.574	1.603	2.483	3.974	3.491	5.219
	Set 3	1.456	8.631	28.783	120.531	85.100	243.728
$R = 9$	Set 1	0.214	0.291	0.355	0.365	0.364	0.408
	Set 2	0.509	1.251	1.993	4.767	3.507	4.242
	Set 3	1.388	6.249	20.359	105.166	72.015	128.297
$R = 10$	Set 1	0.204	0.277	0.340	0.394	0.345	0.331
	Set 2	0.436	1.093	1.648	2.939	2.883	3.235
	Set 3	1.484	7.098	21.250	77.338	75.931	83.134

decreasing trend as the time-windows become wider. That is, the wider the time-window is, generally the smaller the average increasing ratio of the cycle time will be. This suggests that when time-windows are wide, the impact on the cycle time resulting from the increase of the robustness is relatively small. This observation can be explained as follows. When the time-windows are wider, it is apt to obtain a smaller cycle time due to the flexibility in processing times. Thus, a smaller increase of the cycle time can be expected as the robustness R increases. As a result, it increases the possibility to achieve a smaller average increasing ratio of the cycle time.

The computation times for the randomly generated instances are given in Table 3.5. The results in this table suggest that the average computation time has an obvious increasing trend as W increases. Furthermore, for each pair of W and R, the average computation time of the instances in each set has an obvious increasing trend as the time-windows become wider. This is because wider time-windows means looser constraints on the processing times in the tanks. It turns a large number of sequences of hoist moves that is infeasible for narrow time-windows into feasible ones. As a result, a larger solution space is available and finding an optimal solution in it is become time-consuming. However, it seems that there exists no clear relationship between the computation time and the robustness. We also see from Table 3.5 that even for the instances with 22 tanks, the optimal cycle time for any given R can be obtained

within one hour.

3.7 Conclusions

This chapter studied the cyclic hoist scheduling problem considering both cycle time and robustness. We proposed a method to measure the robustness of a cyclic hoist schedule in terms of free slacks in hoist travel times. A bi-objective MILP model was developed for the problem, which aims to minimize the cycle time and maximize the robustness. We proved that the optimal cycle time strictly increases with the robustness, thus the problem has an infinite number of Pareto optimal solutions. Furthermore, we derived lower and upper bounds for the two objectives. A Pareto optimal solution can be obtained by solving a single-objective model by minimizing the cycle time for a given value of robustness or by maximizing the robustness for a specific cycle time. Computational results indicate that the optimal cycle time is a piecewise linear function of the robustness and the increase in cycle time caused by robustness is relatively small when time-windows are wide.

4

Dynamic Jobshop Hoist Scheduling in Extended Lines

Contents

4.1 Introduction

Different from cyclic hoist scheduling, which aims to obtain a cyclic schedule that will run periodically, the objective of non-cyclic hoist scheduling is to obtain a non-cyclic schedule so

that the makespan is minimized. Non-cyclic hoist scheduling where operations are rescheduled as soon as one part or more enter the line is referred to as DHSP (Manier & Bloch, 2003).

Existing studies related to dynamic hoist scheduling usually consider a single new part in the rescheduling procedure (Chauvet et al., 2000; Ge & Yih, 1995; Yih, 1994; Yih & Yin, 1992). With this assumption, it is assumed that each time only one new part is involved in rescheduling. This restriction applies to cases where a new part arrives at the input station only after another one has left, or multiple parts may arrive at the input station at a time but their entering sequence into the production line is known in advance. Consequently, the problem to sequence the parts vanishes. On the other hand, these works usually consider basic lines (Lamothe et al., 1994; Lamothe et al., 1995; Lamothe et al., 1996). Only a few studies for DHSP with multi-function and/or multi-capacity tanks have been reported in the literature. One interesting work is the MILP model developed by Zhao et al. (2013b), which extended Lamothe et al.'s work (Lamothe et al., 1995) by introducing multi-capacity tanks and multiple new arriving parts. This problem is more complicated because it concerns determining the parts' entering sequence and their associated entering times as well as the start times of transportation moves. Tian et al. (2013) reformulated the hoist capacity constraints of Zhao et al.'s model and obtained a more compact MILP model. In the above studies, it was also assumed that a multi-function tank cannot be multi-capacity.

In this study, we develop an MILP model for a DHSP with multi-function tanks, multi-capacity tanks and multiple new arriving parts. We improve the model proposed by Zhao et al. (2013b) in the following aspects. Firstly, we study a more general DHSP with multi-function tanks, which was not considered in Zhao et al. (2013b). It can deal with a more general case where a multi-function tank can also be multi-capacity. Secondly, we demonstrate that Zhao et al.'s model may lead to sub-optimal solutions or infeasibility due to some flaws in formulating tank capacity constraints. To handle this problem, an improved formulation is established. Thirdly, we present a more compact MILP model than Zhao et al.'s in terms of the numbers of constraints and variables. Computational results indicate that the proposed model is much more effective both in solution quality and in computation time.

The remainder of this chapter is organized as follows. we give a formal description of the considered problem in Section 4.2. In Section 4.3, the flawed formulation of Zhao et al.'s model is analyzed and demonstrated by a counterexample. An improved MILP model is developed in Section 4.4. Computational results are presented and analyzed in Section 4.5. Finally, this study is concluded in Section 4.6.

4.2 Problem description and notation

The DHSP considered in this study is similar to those in Lamothe et al. (1995) and Zhao et al. (2013b). The difference is that multi-function tanks were not considered in Zhao et al. (2013b) and neither multi-function tanks nor multi-capacity tanks were addressed by Lamothe et al.

(1995). Similar description for the problem without multi-function tanks or multi-capacity tanks can be found in the above cited references. However, for the sake of self-consistency, we recall this description. Furthermore, the notation given in Zhao et al.'s model cannot be used to formulate the constraints related to multi-function tanks. Hence, we define our own notation used throughout the chapter.

The considered DHSP can be described as follows. The production line is composed of W processing tanks and a computer-controlled hoist. The W tanks are indexed as M_1, M_2, ..., and M_W, respectively. Let M_0 and M_{W+1} denote the input station and output station, respectively. The parts to be processed are of different types. The difference between parts is characterized by their processing routes and processing time requirements in each tank. That is, the considered system is a jobshop. Parts randomly arrive at the input station. The type and the number of the arriving parts are not known until their arrival at the input station. The processing procedure for a part can be described as follows. The part is first unloaded from the input station, and then treated successively in a set of processing tanks according to its specified sequence, and finally moved out of the line into the output station. The tanks may have single-part or multi-part processing capacity. A multi-capacity tank has multiple processing *slots* and each slot can process at most one part at a time. Thus, if the processing capacity of M_w is C_w, the tank can handle at most C_w parts simultaneously. A part cannot be processed in more than one tank or slot at the same time. The hoist is used to transport parts between the tanks and stations. The hoist can hold at most one part at a time.

Let $\mathbf{P}$ be the set of all parts associated with the rescheduling. $\mathbf{P}$ consists of the set of parts waiting at the input station, denoted by $\mathbf{P}_I$, and the set of parts being processed in the line at the start of the reschedule (also called rescheduling point hereafter), denoted by $\mathbf{P}_L$. That is, $\mathbf{P} = \mathbf{P}_I \cup \mathbf{P}_L$. Let N_r be the number of (remaining) processing stages for part $r \in \mathbf{P}$. The i^{th} processing stage of part r is represented by $S_{r,i}$. The index of the tank in which the processing of stage $S_{r,i}$ is executed is denoted by $V_{r,i}$. Without loss of generality, stages $S_{r,0}$ and S_{r,N_r+1} are assumed to be performed on M_0 and M_{W+1}, respectively. That is, we always have $V_{r,0} = 0$ and $V_{r,N_r+1} = W + 1$.

Due to the presence of multi-function tanks, a part may visit some tanks more than once. This means we may have $V_{r,i} = V_{r,j}$ for some parts $r \in \mathbf{P}$, $1 \leq i < j \leq N_r$. Let $\mathbf{K}$ be the set of all tanks, excluding M_0 and M_{W+1}. Set $\mathbf{K}$ consists of two subsets: the set of single-capacity tanks, denoted by $\mathbf{K}_S$, and the set of multi-capacity tanks, denoted by $\mathbf{K}_M$, i.e., $\mathbf{K} = \mathbf{K}_S \cup \mathbf{K}_M$. The actual processing time at stage $S_{r,i}$, $r \in \mathbf{P}$, $1 \leq i \leq N_r$, is bounded from below by a given constant $L_{r,i}$ and from above by a given constant $U_{r,i}$. If the actual processing time for any stage is less than its lower bound or greater than its upper bound, the associated part will be identified as defective. Once the processing at a stage is completed, the corresponding part must be unloaded so that the requirement of its processing time is respected. After unloading a part, the hoist should transport it into the next tank for its subsequent processing without any delay. That is, any wait is not allowed during the execution of a transportation operation.

However, the hoist can wait if it is not holding a part. The operation of transporting part r from tank $V_{r,i}$ to tank $V_{r,i+1}$ is called a loaded move, and denoted as $O_{r,i}$, $r \in \mathbf{P}$, $1 \leq i \leq N_r$. Move $O_{r,i}$ is composed of three operations: (1) unloading part r from $V_{r,i}$; (2) transporting the part to $V_{r,i+1}$; and (3) loading it into $V_{r,i+1}$. The time required for the hoist to execute move $O_{r,i}$ is $D_{r,i}$, $r \in \mathbf{P}$, $1 \leq i \leq N_r$. The hoist operation of traveling from M_i to M_j, $i, j \in \mathbf{K} \cup \{0, W+1\}$, without holding a part is called an unloaded move, and denotes as $G_{i,j}$. The time required for executing $G_{i,j}$ is $E_{i,j}$. The loaded and unloaded move times satisfy the triangular inequality. Note that we assume the traveling time between any two processing slots of a multi-capacity tank is negligible, i.e., the times required for both loaded and unloaded moves between any two tanks do not vary with particular slots.

As parts randomly arrive at the system, their characteristics cannot be known until they arrive at the input station. Once new parts arrive, a rescheduling is triggered and an optimal reschedule should be established according to the current state of the system, including the current status of parts, tanks and hoist (Zhao et al., 2013b). Without loss of generality, the rescheduling point is set to 0. At the rescheduling point, some stages of part r, $r \in \mathbf{P}_L$, are completed, and the part is being processed in some tank. Note that we do not consider the case where the hoist is moving a part from one tank to the next one, since this transportation cannot be rescheduled. Any reschedule cannot start until this move is completed. In other words, the rescheduling point can start only when the hoist is performing an unloaded move or waiting for the completion of some processing stage without loss of generality. Let I_r be the index of the stage at which part r is being processed at the rescheduling point; that is, part r, $r \in \mathbf{P}_L$, is being processed at stage S_{r,I_r} in tank V_{r,I_r}. Note that $I_r = 0$ for any part r such that $r \in \mathbf{P}_I$, without loss of generality.

With the above problem, the objective is to obtain an optimal hoist reschedule for all involved parts so that the makespan is minimized. During the rescheduling, all the processing stages for parts at the input station, and the remaining processing stages for parts being processed in the line should be scheduled. A feasible schedule should satisfy the following three families of constraints (Tian et al., 2013; Zhao et al., 2013b):

- Hoist capacity constraints. The start times of all related hoist moves are well defined so that the hoist is not required to handle more than one part at any time.
- Tank capacity constraints. At any time, a tank cannot handle more parts than its capacity.
- Processing time constraints. The processing time at each stage should be not less than its lower bound and not greater than its upper bound.

As all transportation operations are executed by the hoist, a reschedule can be uniquely represented by the sequence of all moves $O_{r,i}$, $r \in \mathbf{P}$, $I_r \leq i \leq N_r$ and their associated start times. Hence, the following decision variables are defined.

T: the makespan. It equals to the time instant when the last part arrives at the output station.

$s_{r,i}$: the start time of move $O_{r,i}$ in the reschedule, $r \in \mathbf{P}$, $I_r \leq i \leq N_r$. To simplify the expression, the start time of a reschedule is set to 0. The start time of each move is calculated relative to the related rescheduling point.

$y_{r,i;u,j}$: binary variable. $y_{r,i;u,j} = 1$ if and only if $s_{r,i} < s_{u,j}$, for all $r, u \in \mathbf{P}$, $I_r \leq i \leq N_r$, $I_u \leq j \leq N_u$. That is, for any pair of moves $O_{r,i}$ and $O_{u,j}$, if move $O_{r,i}$ is executed before move $O_{u,j}$, then $y_{r,i;u,j} = 1$; otherwise, $y_{r,i;u,j} = 0$.

4.3 Illustration and analysis of a counterexample

Zhao et al. (2013b) developed an MILP model for a similar problem without multi-function tanks. In this section, we will use a counterexample to illustrate that the solution obtained with Zhao et al.'s model is not necessarily optimal. The counterexample is derived from the "Case 4" given in Zhao et al. (2013b) with the following modification: (1) the lower bound of the processing time for parts of type A in tank M_3 is decreased from 75s to 30s; (2) the lower bound of the processing time for parts of type B in tank M_3 is increased from 75s to 120s. For this example, tank M_3 is the only multi-capacity tank with $C_3 = 2$, while the other tanks are all single-capacity ones. With Zhao et al.'s model, the optimal makespan for this example is 425s. The obtained optimal reschedule is illustrated in Figure 4.1. To distinguish different parts of the same type, each part is denoted by its type and an integer in the figure. For example, A1 represents the first part of type A. At the rescheduling point, part A1, part A2 and part B1 have already been processed for 61s, 25s and 36s in M_5, M_4 and M_3, respectively. Figure 4.2 illustrates another feasible reschedule for this example. Note that the reschedule in Figure 4.2 is obtained with our model described in the next section, and its makespan is 424s which is shorter than that of the reschedule in Figure 4.1.

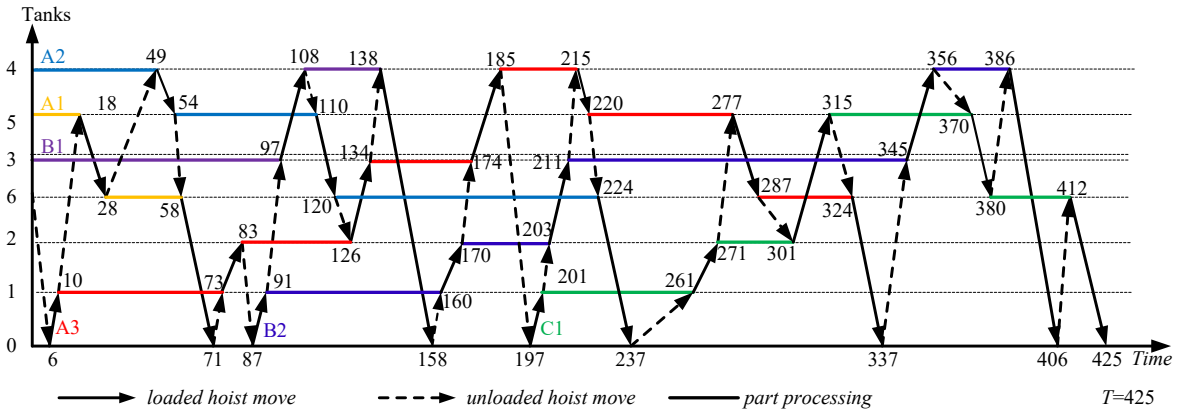

Figure 4.1: The reschedule obtained with Zhao et al.'s model

As both reschedules in Figure 4.1 and Figure 4.2 are feasible, the reschedule in Figure 4.1 is actually not an optimal solution for the example. We now analyze why the solution obtained with Zhao et al.'s model is not optimal. In Figure 4.2, part B1 is being processed in tank M_3 at the rescheduling point. During its processing, part A3 and part B2 are processed in tank

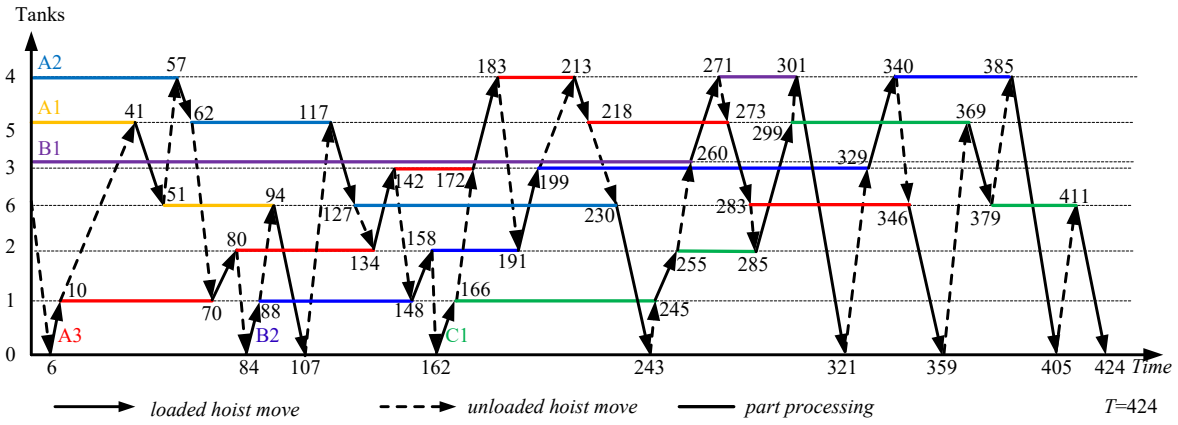

Figure 4.2: A feasible reschedule with smaller makespan

3 sequentially. We can see that the number of parts simultaneously processed in tank M_3 never exceeds its capacity ($C_3 = 2$) during the production. However, such a situation will be identified as infeasible by Zhao et al.'s formulation as explained in the following.

In the following analysis, we will follow the same notation given by Zhao et al. (2013b) when the related notation is not defined in this study. To formulate tank capacity constraints, Zhao et al. first defined a binary variable $x_{i,r,u}$ as follows: if part u ever stays with part r in tank M_i, then $x_{i,r,u} = 1$, and $x_{i,r,u} = 0$ otherwise. With such a definition, the following constraint is formulated to attempt to ensure that the total number of parts simultaneously processed in tank M_i does not exceed its capacity C_i at any time:

$$\sum_{u \in \mathbf{P}} x_{i,r,u} + 1 \leq C_i, \ \forall i \in SI_r^A; \ i \in SI_u^A; \ r, u \in \mathbf{P}, r \neq u \tag{4.1}$$

where $\mathbf{P}$ is the set of parts, SI_r^A and SI_u^A are the set of processing tank for part r and u respectively, not including M_0 and M_{W+1}, C_i is the processing capacity of tank M_i. From the above constraint, we can see that the maximum number of parts processed simultaneously in tank M_i is expressed as the following formula:

$$\max_{r \in \mathbf{P}} \left(\sum_{u \in \mathbf{P}} x_{i,r,u} + 1 \right) \tag{4.2}$$

In what follows, we show that the maximum number of parts simultaneously processed in tank M_i may be over-counted using formula (4.2). In Figure 4.2, both parts A3 and B2 sometime stay with part B1 in tank M_3. By the definition of $x_{i,r,u}$, we can have $x_{3,\mathrm{B1,A3}} = 1$ and $x_{3,\mathrm{B1,B2}} = 1$. With formula (4.2), the maximum number of parts simultaneously processed in tank M_3 is three, instead of two as shown in Figure 4.2. That is, the maximum number of parts simultaneously processed in tank M_3 is over-counted in this case. For this reason, the reschedule in Figure 4.2 will be identified as infeasible by Zhao et al.'s model as tank M_3 only has two-part processing capacity.

In fact, the maximum number of parts simultaneously processed in tank M_i can be correctly calculated with formula (4.2) if all parts to be processed in the tank are sometime being

simultaneously processed in the tank, as illustrated in Figure 4.3(a). However, if not all parts are simultaneously processed in a tank (i.e., some part enters the tank after another one has left), as illustrated in Figure 4.3(b), the maximum number of parts processed simultaneously in the tank is over-counted with formula (4.2). For this reason, a feasible solution may be identified as infeasible with Zhao et al.'s model. It is worth noting that the scenario illustrated in Figure 4.3(b) very likely to happen when the processing time of some part is much longer than those of other parts in a multi-capacity tank. This suggests that it is more likely that Zhao et al.'s model cannot obtain an optimal solution in such situations.

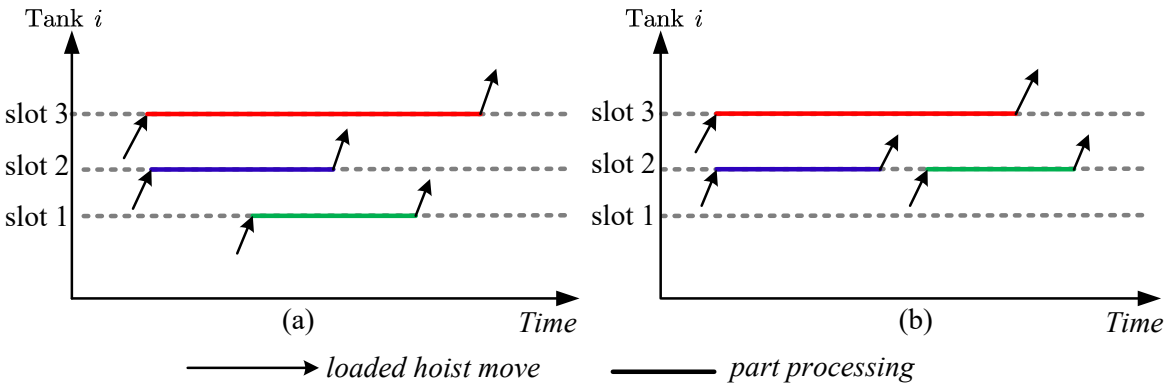

Figure 4.3: Illustration of tank capacity constraints

To sum up, a feasible reschedule can be identified as infeasible with Zhao et al.'s model due to the flawed formulation of tank capacity constraints. For this reason, the solution obtained with Zhao et al.'s model is not necessarily optimal. To fix this problem, we propose an improved MILP model in the next section, which can correctly handle multi-capacity tanks. Furthermore, we generalize the model by considering multi-function tanks.

4.4 Problem formulation

In this section, we propose a new scheme to formulate the tank capacity constraints. The formulation can handle the situation where a multi-function can also be multi-capacity and can obtain an optimal solution for the problem. Note that the brief formulation for the processing time constraints given below are borrowed from Lamothe et al. (1995) and Zhao et al. (2013b), while similar formulation for the hoist capacity constraints can be found in Tian et al. (2013). For self-consistency, we give a complete formulation in what follows.

4.4.1 Time-window constraints

Processing time constraints require that the actual processing time at each stage be not less than its lower bound and not greater than its upper bound. For any part $r \in \mathbf{P}$, and its processing stage $S_{r,i}$, $I_r + 1 \leq i \leq N_r$, the part enters tank $V_{r,i}$ at time instant $s_{r,i-1} + D_{r,i-1}$,

which is the completion time of move $O_{r,i-1}$. After the processing of $S_{r,i}$ is completed, it will be unloaded from tank $V_{r,i}$ at time instant $s_{r,i}$, which is the start time of move $O_{r,i}$. Thus, the actual processing time of stage $S_{r,i}$ is $s_{r,i} - s_{r,i-1} - D_{r,i-1}$. Note that at the rescheduling point, for any part r in set $\mathbf{P}_L$, some of its processing stages are completed and it is being processed at stage S_{r,I_r}. The processing time constraints can be formulated as (4.3) and (4.4) (Lamothe et al., 1995; Zhao et al., 2013b).

$$L_{r,i} \le s_{r,i} - s_{r,i-1} - D_{r,i-1} \le U_{r,i},\ \forall r \in \mathbf{P}, I_r + 1 \le i \le N_r. \tag{4.3}$$

$$L_{r,I_r} \le T^s + s_{r,I_r} - s^0_{r,I_r-1} - D_{r,I_r-1} \le U_{r,I_r},\ \forall r \in \mathbf{P}_L. \tag{4.4}$$

where T^s is the estimated rescheduling point of the current schedule, at which the current schedule is switched to the reschedule, and $s^0_{r,i-1}$ is the start time of move $O_{r,i-1}$ in the current schedule. Note that the meaning of T^s is two-fold: on the one hand, its value is determined according to the start point of the current schedule; on the other hand, it also denotes the start point of the reschedule. In other words, the rescheduling point may denotes T^s when referring to the current schedule, or the start point of the reschedule (which is set as 0) otherwise. With this point, the processing of stage S_{r,I_r} is divided into two parts: the processing executed in the current schedule and the processing to be executed in the reschedule. The first part can be formulated as $T^s - s^0_{r,I_r} - D_{r,I_r-1}$ while the second part is s_{r,I_r}. The value of T^s can be determined according to the arrival time of new parts, the hoist status and the estimated computation time of the reschedule with the method suggested by Zhao et al. (2013b).

4.4.2 Hoist capacity constraints

Hoist movement constraints ensure that the hoist is not required to handle more than one part at any time. To this end, the start times of all moves in the reschedule should be such that for any pair of moves $O_{r,i}$ and $O_{u,j}$, either move $O_{r,i}$ starts after move $O_{u,j}$ is completed or reversely. Furthermore, after executing a loaded move, there should be enough time for the hoist to travel from its current position to the tank where the next move starts. Thus the hoist capacity constraints can be formulated as (4.5) and (4.6) (Tian et al., 2013).

$$\begin{aligned} s_{u,j} - s_{r,i} \ge D_{r,i} + E_{V_{r,i+1},V_{u,j}} - M\left(1 - y_{r,i;u,j}\right), \\ \forall r, u \le \mathbf{P}, I_r \le i \le N_r, I_u \le j \le N_u, r \ne u \text{ or } i \ne j. \end{aligned} \tag{4.5}$$

$$y_{r,i;u,j} + y_{u,j;r,i} = 1,\ \forall r, u \le \mathbf{P}, I_r \le i \le N_r, I_u \le j \le N_u, r \ne u \text{ or } i \ne j. \tag{4.6}$$

where M is a very large positive number. Similar hoist capacity formulation can also be found in cyclic hoist scheduling (Leung & Zhang, 2003; Liu et al., 2002; Zhou et al., 2012). In particular, constraint (4.5) ensures that if $s_{r,i} < s_{u,j}$ (i.e. $y_{r,i;u,j} = 1$), then $s_{u,j} - s_{r,i} \ge D_{r,i} + E_{V_{r,i+1},V_{u,j}}$, which guarantees that the empty hoist should have sufficient amount of time to travel to tank $V_{u,j}$ so as to execute move $O_{u,j}$. Constraint (4.6) ensures the consistency of variables $y_{r,i;u,j}$ and $y_{u,j;r,i}$; that is, if $y_{r,i;u,j} = 1$, then $y_{u,j;r,i} = 0$ and vice versa.

Furthermore, there should be sufficient time for the hoist to travel from its current position to the tank at which the first move in the reschedule is executed. As any other move can start after the first move is completed, they can start after the hoist arrives at its associated tank. Let h be position of the hoist at the rescheduling point, the requirements can be formulated as (4.7) (Zhao et al., 2013b).

$$s_{r,i} \geq E_{h,V_{r,i}},\ \forall r \leq \mathbf{P}, I_r \leq i \leq N_r. \tag{4.7}$$

4.4.3 Tank capacity constraints

Tank capacity constraints ensure that any tank is not required to handle more parts than its processing capacity at any time. This requires that all processing stages processed in the same tank be well scheduled so that no conflict occurs in the use of the tank. For a multi-capacity tank, each stage can use any processing slot of the tank. Thus the specific slot in which a stage is processed should be determined as well. Furthermore, the stages sharing the same slot should also be well scheduled to avoid any conflict in the use of the slot. For this reason, the capacity constraint for multi-capacity tanks is more complicated than that for single-capacity tanks. Thus we separately formulate the tank capacity constraints for single-capacity and multi-capacity tanks. Due to this separation, the model will be more compact in terms of numbers of constraints and variables.

4.4.3.1 Single-capacity tanks

For any pair of stages $S_{r,i}$ and $S_{u,j}$ such that $V_{r,i} = V_{u,j}$, they are not allowed to use the shared tank at the same time. That is, either stage $S_{r,i}$ is processed before stage $S_{u,j}$ with sufficient amount of time, or stage $S_{u,j}$ is processed before stage $S_{r,i}$ with sufficient amount of time. As the transportation moves are executed by the hoist, the following requirements must be satisfied: 1) either move $O_{u,j-1}$, by which part u is moved into the shared tank, is executed after move $O_{r,i}$, by which part r is moved away from the tank; 2) or move $O_{r,i-1}$, by which part r is moved into the shared tank, is executed after move $O_{u,j}$, by which part u is moved away from the tank. The above requirement can be formulated as the below logical relationship:

$$\text{either } s_{u,j-1} \geq s_{r,i} \text{ or } s_{r,i-1} \geq s_{u,j},\ \forall r, u \in \mathbf{P},\ \text{if } V_{r,i} = V_{u,j}.$$

The above relation can be equivalently written as:

$$\begin{gathered} y_{r,i;u,j-1} + y_{u,j;r,i-1} = 1, \\ \forall r, u \in \mathbf{P}, I_r + 1 \leq i \leq N_r, I_u + 1 \leq j \leq N_u,\ \text{if } V_{r,i} = V_{u,j}, C_{V_{r,i}} = 1. \end{gathered} \tag{4.8}$$

Note that any part r such that $r \in \mathbf{P}_L$ is being processed in tank V_{r,I_r} at the rescheduling point. Thus any other stages to be processed in the tank can start only after stage S_{r,I_r} is completed. This suggests that any part to be processed in tank V_{r,I_r} can enter the tank after part r has been moved away by the hoist. Such a requirement can be formulated as equation (4.9).

$$y_{r,I_r;u,j-1} = 1,\ \forall r \in \mathbf{P}_L, u \in \mathbf{P}, I_u + 1 \leq j \leq N_u,\ \text{if } V_{r,I_r} = V_{u,j}, C_{V_{r,I_r}} = 1. \tag{4.9}$$

4.4.3.2 Multi-capacity tanks

For a multi-capacity tank M_i, there are C_i identical processing slots. Without loss of generality, the C_i processing slots are indexed as slot 1, slot 2, . . . , slot C_i in arbitrary order. To formulate the capacity constraints, the following binary variable is defined.

$x_{r,i,k}$: If stage $S_{r,i}$, $r \in \mathbf{P}$, $I_r + 1 \leq i \leq N_r$, is executed in slot k of tank $V_{r,i}$, $1 \leq k \leq C_{V_{r,i}}$, then $x_{r,i,k} = 1$; otherwise, $x_{r,i,k} = 0$.

For a pair of stages $S_{r,i}$ and $S_{u,j}$ such that $V_{r,i} = V_{u,j}$, two cases are possible: 1) they are processed in different processing slots of the tank; 2) they are processed in the same processing slot, say slot k, of the tank. In the first case, there is certainly no conflict as they do not use the same slot. However, in the second case, the conflict between the stages has to be avoid as they are both handled in slot k of the tank, i.e., $x_{r,i,k} = 1$ and $x_{u,j,k} = 1$. Such a requirement can be formulated as the following constraints.

$$y_{r,i;u,j-1} + y_{u,j;r,i-1} \geq x_{r,i,k} + x_{u,j,k} - 1,$$
$$\forall r, u \in \mathbf{P}, I_r + 1 \leq i \leq N_r, I_u + 1 \leq j \leq N_u, V_{r,i} = V_{u,j}, 1 \leq k \leq C_{V_{r,i}}. \tag{4.10}$$

$$y_{r,i;u,j-1} + y_{u,j;r,i-1} \leq 3 - x_{r,i,k} - x_{u,j,k},$$
$$\forall r, u \in \mathbf{P}, I_r + 1 \leq i \leq N_r, I_u + 1 \leq j \leq N_u, V_{r,i} = V_{u,j}, 1 \leq k \leq C_{V_{r,i}}. \tag{4.11}$$

$$\sum_{k=1}^{C_{V_{r,i}}} x_{r,i,k} = 1, \ \forall r \leq \mathbf{P}, I_r + 1 \leq i \leq N_r. \tag{4.12}$$

In particular, if $S_{r,i}$ and $S_{u,j}$ use the same slot k of tank M_i (i.e., $x_{r,i,k} = 1$ and $x_{u,j,k} = 1$), equations (4.10) and (4.11) would be reduced to equation (4.8). It ensures that no conflict in the use of the same slot occurs. On the other hand, the constraints become redundant if the stages use different slots of tank $V_{r,i}$. Equation (4.12) ensures that exactly one processing slot of tank $V_{r,i}$ is used to handle stage $S_{r,i}$.

Note that it may lead to symmetric solutions if more than two slots of a multi-capacity tank are empty at the rescheduling point, because different allocations of stages among the slots may represent the same solution. Knowing that all processing slots are identical and the travel time between any two slots of a multi-capacity tank is negligible, we can change one solution to another by renumbering the empty slots. To illustrate this phenomenon, we use the following example to explain how this can be achieved. Suppose that there are two empty slots at the rescheduling point and three parts denoted by part 1, part 2 and part 3, respectively are processed in the two slots during the reschedule. If part 1 and part 2 are processed in the same slot while part 3 is processed in the other slot. The following two solutions are possible: 1) part 1 and part 2 are processed in slot 1 and part 3 is processed in slot 2; 2) part 3 is processed in slot 1, while part 1 and part 2 are processed in slot 2. The other part of the two solutions are identical. However, the above two situations represent the same solution due to their symmetry. By swapping the two slots, we can change the former solution to the latter one and vice versa. To reduce the number of symmetric solutions, it is assumed that the number

of parts processed in the slot with smaller index is greater than or equal to that in empty slot with larger index. Let $f(w)$ be the set of the empty slots of tank M_w at the rescheduling point and $|f(w)|$ the number of elements in set $f(w)$. This requirement can be formulated as the following constraint.

$$\sum_{r\in\mathbf{P},I_r+1\le i\le N_r,V_{r,i}=w} x_{r,i,k} \ge \sum_{r\in\mathbf{P},I_r+1\le i\le N_r,V_{r,i}=w} x_{r,i,k'},$$
$$\forall w \in \mathbf{K}_M; k, k' \in f(w), k < k'; \text{if } |f(w)| \ge 2. \tag{4.13}$$

Similar to the formulation for single-capacity tanks, if part r is being processed in slot v_r of tank V_{r,I_r} at the rescheduling point, i.e., $x_{r,I_r,v_r} = 1$, any other stages to be processed in this slot can start only after stage S_{r,I_r} is completed. Note that v_r can be determined as soon as the rescheduling point T^s is determined. This requirement can be formulated as constraint (4.14).

$$y_{r,I_r;u,j-1} \ge x_{u,j,v_r},\ \forall r \in \mathbf{P}_L, u \in \mathbf{P}, I_u + 1 \le j \le N_u. \tag{4.14}$$

Furthermore, the processing sequence of a part must respect its processing route. That is, for any pair of stages $S_{r,i}$ and $S_{r,j}$ of part r such that $i < j$, constraint (4.15) should be satisfied.

$$y_{r,i;r,j} = 1,\ \forall r \in \mathbf{P}, I_r \le i \le j \le N_r. \tag{4.15}$$

As T is defined as the makespan, we have:

$$T \ge s_{r,N_r} + D_{r,N_r},\ \forall r \in \mathbf{P}. \tag{4.16}$$

In addition, the binary variables $x_{r,i,k}$ and $y_{r,i,u,j}$ must satisfy the following relations according to their definitions:

$$y_{r,i;u,j} \in \{0,1\},\ \forall r, u \le \mathbf{P}, I_r \le i \le N_r, I_u \le j \le N_u. \tag{4.17}$$
$$x_{r,i,k} \in \{0,1\},\ \forall r \le \mathbf{P}, I_r \le i \le N_r, 1 \le k \le C_{V_{r,i}}. \tag{4.18}$$

With the constraints given above, the problem addressed in this study can be formulated as:
P:

$$\text{Minimize} \quad T \tag{4.19}$$

subject to: (4.3)–(4.18).

4.5 Computational results

In this section, we first use instances with both multi-capacity and multi-function tanks to test the proposed model. Then, we compare our model with Tian et al.'s (Tian et al., 2013) using instances without multi-function tanks. As mentioned above, Tian et al. (2013) improved the

formulation of hoist capacity constraints proposed by Zhao et al. (2013b), while the other part of the model remains the same. That is, Tian et al.'s model represents an improved version of Zhao et al.'s. Thus we compare our model with Tian et al.'s rather than Zhao et al.'s. Both Our and Tian et al.'s models are solved by ILOG CPLEX (Version 12.6) with default parameters. All experiments are conducted on a personal computer with 2.27 GHZ CPU and 4.0GB RAM.

4.5.1 Instances with multi-function tanks

In this subsection, we first use an instance with multi-capacity multi-function tanks to test the proposed model. The considered production line is composed of $W = 7$ processing tanks. The input station and the output station share the same physical tank (i.e. M_0) placed at the left end of the line. The tanks are arranged in the line from left to right with the following sequence: tank M_0, tank M_1, tank M_3, tank M_2, tank M_4, tank M_6, tank M_5, and then tank M_7. The capacities of the tanks are set as follows: $C_1 = C_3 = C_4 = C_5 = C_7 = 1$, $C_2 = 4$, $C_6 = 2$. The time required for the hoist to travel between adjacent tanks is $E_{i,i+1} = 2$, $0 \leq i \leq 6$. The time required for the hoist to execute move $O_{r,i}$ is $D_{r,i} = E_{V_{r,i},V_{r,i+1}} + 1$. There are three types of parts to be processed in the line. The processing route and associated processing time requirement for each type are given in Table 4.1, Table 4.2, and Table 4.3, respectively. Note that both tank M_2 and tank M_6 are multi-function tanks and multi-capacity tanks simultaneously.

Table 4.1: Processing routine for the parts of type A

Stage	0/10	1	2	3	4	5	6	7	8	9
Tank	0	1	2	3	4	2	5	6	7	6
$L_{r,i}$	–	30	200	60	60	200	30	35	90	70
$U_{r,i}$	–	60	400	90	120	400	120	75	160	200

Table 4.2: Processing routine for the parts of type B

Stage	0/7	1	2	3	4	5	6
Tank	0	1	2	3	4	2	6
$L_{r,i}$	–	20	220	70	90	200	70
$U_{r,i}$	–	50	550	120	160	400	200

Table 4.3: Processing routine for the parts of type C

Stage	0/7	1	2	3	4	5	6
Tank	0	1	2	5	6	7	2
$L_{r,i}$	–	25	200	80	70	90	200
$U_{r,i}$	–	55	550	150	200	160	400

Suppose that, two new parts, one of type A and the other of type C, arrive at the input station at time instant 179s in the initial schedule. The system needs to be rescheduled so as to minimize the makespan for all parts. Assume that the rescheduling point T^s is set as 180s, which is the start point of the reschedule. At this point, the processing status for the parts are given in Table 4.4, including the indexes of tanks in which the parts are being processed and the start times of the moves by which these parts are moved to the tanks during the initial schedule. Furthermore, the hoist is at the input station, i.e., $h = 0$, at the rescheduling point.

Table 4.4: The initial state at the rescheduling point

Parts (r)	A2	A1	B2	B1	C2	C1
I_r	0	4	2	6	0	6
V_{r,I_r}	0	4	2	6	0	2
s^0_{r,I_r}	–	145	75	120	–	152

Note that such a hoist rescheduling problem with multi-capacity multi-function tanks cannot be solved by any models or approaches in the literature. With the MILP proposed in this study, we obtain an optimal reschedule with $T = 825$ for this instance. The schedule is illustrated by the time-way diagram shown in Figure 4.4. Note that the start time of the reschedule is set to 0 in order to facilitate the presentation.

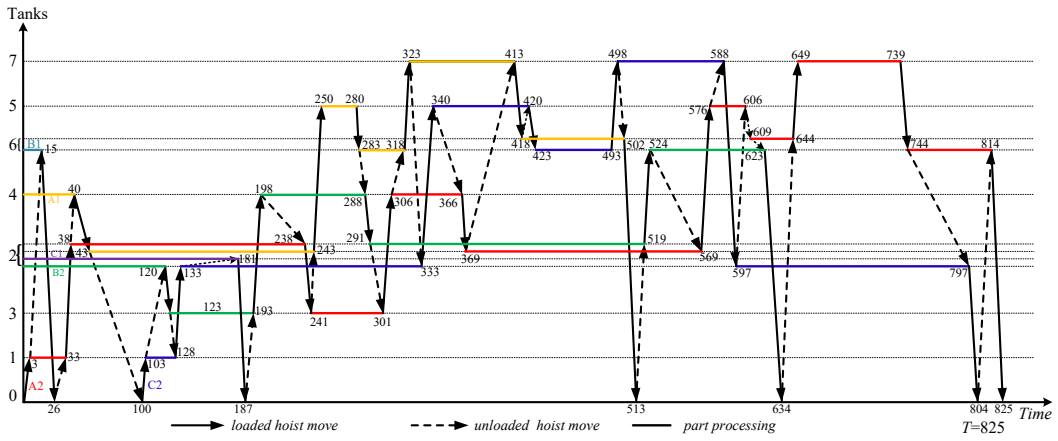

Figure 4.4: The obtained optimal reschedule for the instance

From Figure 4.4, we can see that parts B2 and C1 are being processed on two different processing slots of tank M_2 at the start of the reschedule. Parts A2 and A1 enter tank M_2 at time instants 38 and 43, respectively. Thus, starting at time instant 43, tank M_2 will handle 4 parts simultaneously. This status will last until part B2 leaves the tank at time instant 120. Tank M_2 treats 4 parts simultaneously again after part C2 arrives at the tank at time instant 133. In the remainder of the schedule, tank M_2 handles at most 3 parts after part C1 has left at time instant 181. Similarly, we can also see that tank M_6 handles at most two parts simultaneously at any time. No tank is required to process more parts than its capacity during the schedule.

To further evaluate the proposed model, a set of randomly generated instances, denoted by Set-1, are constructed. For each value of $W \in \{8, 10, 12\}$, 30 instances are generated. Let $\mathbf{U}(a, b)$ represent an integer random distribution drawn from interval $[a, b]$. Each instance is generated in the following way: the number of new parts is generated by $\mathbf{U}(1, W/2)$, $N_r = \mathbf{U}(2, W)$ for $r \in \mathbf{P}$, $V_{r,i} = \mathbf{U}(1, W)$ for $r \in \mathbf{P}, 1 \leq i \leq N_r$ such that $V_{r,i} \neq V_{r,i+1}$, $C_w = \mathbf{U}(1, 2)$ for $1 \leq w \leq W$, $E_{i,i+1} = \mathbf{U}(1, 2)$ and $E_{i,j} = \sum_{k=i}^{j-1} E_{k,k+1}$ such that $E_{i,j} = E_{j,i}$ for $i, j \in \mathbf{K} \cup \{0, W + 1\}$, and $D_{r,i} = E_{V_{r,i},V_{r,i+1}} + 5$ for $r \in \mathbf{P}$, $1 \leq i \leq N_r$; the processing time requirement for each stage is generated by randomly selecting one of the following three methods: 1) $L_{r,i} = \mathbf{U}(90, 150)$, $U_{r,i} = L_{r,i} + \mathbf{U}(0, 60)$; 2) $L_{r,i} = \mathbf{U}(30, 90)$, $U_{r,i} = L_{r,i} + \mathbf{U}(0, 90)$; and 3) $L_{r,i} = \mathbf{U}(150, 270)$, $U_{r,i} = L_{r,i} + \mathbf{U}(0, 150)$. The rescheduling point is set as $T^s = \mathbf{U}(30, T_0)$, where T_0 is the makespan of the current schedule. To evaluate the effects of parallel processing slots, we also generate another set of instances, denoted by Set-2. The instances in Set-2 are generated with the above parameters except that the capacity of all processing tanks are set to $C_w = 1$, $1 \leq w \leq W$.

The computational results for the two sets of instances, Set-1 and Set-2, are summarized in Table 4.5. Note that the results are the average values of the 30 tested instances. In the table, #Moves denotes the average number of moves in the reschedule; #Cons and #Vars represent the average number of constraints and the average number of variables, respectively; #Makespan is the average makespan; and #Time is the average computation time measured in CPU seconds.

Table 4.5: Computational results for random instances with multi-function tanks

W	Set	#Moves	#Cons	#Vars	#Makespan	#Time (s)
8	Set-1	21.03	1180.57	521.83	875.6	0.4274
	Set-2	23.5	1413.5	640.5	1105.4	1.0695
10	Set-1	33.73	3011.43	1350.2	1279.77	2.8813
	Set-2	32.13	2821.8	1298.6	1328.37	4.5391
12	Set-1	44.17	5157.97	2348.63	1565.83	14.5236
	Set-2	42.4	4717.03	2185.93	1689.3	28.4573

As shown in Table 4.5, all instances are optimally solved in half a minute. By comparing the makespan of the two sets, we can see that the instances with multi-capacity tanks usually report shorter makespan than those without multi-capacity tanks. The average difference ranges from 48.6 to 229.8, which is equivalent to about 3.8%–26.2% reduction in makespan by using multi-capacity tanks. On the other hand, it shows a positive correlation between the numbers of constraints or variables and the number of moves needed to be scheduled. In spite of more moves to be scheduled in Set-1, the presence of multi-capacity tanks usually requires shorter computation time as shown in the table. This phenomenon can be explained as follows. The multiple processing slots of a multi-capacity tank improve the flexibility in assigning stages among the slots. It implies greater opportunity to obtain a better schedule,

which may provide better bounds in the solution procedure.

4.5.2 Instances without multi-function tanks

Note that multi-function tanks are not addressed in Zhao et al.'s model and Tian et al.'s model. To compare the performance of our model and Tian et al.'s, we test them on four instances given in the literature and a set of randomly generated instances in this subsection.

We first compare our model with Tian et al.'s using four numerical instances (called Case 1, Case 2, Case 3, and Case 4, respectively) given in Zhao et al. (2013b) and the counterexample given above (called Case 5 below). For the five instances, there is a single multi-capacity tank and no multi-function tank in the system. At the rescheduling point, there are five, four and five parts being processed in the system for Case 1, Case 2 and Case 3, respectively, each including one part at the input station. For Cases 4 and 5, six parts are being processed in the system, including three parts at the input station.

For the five cases, Table 4.6 gives the computational results of the two models. Note that the results for Tian et al.'s model were also reported in Tian et al. (2013). We can see that the two models report the same optimal makespan for Cases 1–4. Zhao et al.'s model obtain actual optimal solutions for the four instances in spite of the flawed formulation of tank capacity constraints. However, the solution obtained with Zhao et al.'s model for Case 5 is not optimal. For all the five instances, our model can obtain a genuine optimal solution.

Table 4.6: Computational results for the numerical instances

Case	Model	#Cons	#Vars	#Makespan	#Time
Case 1	Tian et al.	795	368	283	0.359
	Our	670	297	283	0.281
Case 2	Tian et al.	396	174	235	0.296
	Our	337	148	235	0.281
Case 3	Tian et al.	481	216	248	0.234
	Our	392	173	248	0.187
Case 4	Tian et al.	1497	714	426	1.56
	Our	1295	584	426	0.421
Case 5	Tian et al.	1497	714	425	1.435
	Our	1295	584	424	0.437

To further compare our model with Tian et al.'s, a set of randomly generated instances are used to test these two models. For each value of $W \in \{8, 10, 12\}$, 30 instances are generated. The parameters used to generate these instances are the same with those for Set-1 except that any part can visit each tank at most once during its processing, i.e., $V_{r,i} \neq V_{r,j}$ for all $r \in \mathbf{P}$, $1 \leq i, j \leq N_r$.

The computational results for the randomly generated instances are summarized in Table 4.7. In the table, #Feas represents the number of instances that are reported feasible by the model,

while the other columns have the same meaning as those in Table 4.6. The results in Table 4.7 are the average results for all feasible instances.

Table 4.7: Computational results for random instances without multi-function tanks

W	Model	#Feas	#Moves	#Cons	#Vars	#Makespan	#Time (s)
8	Tian et al.	30	21.67	1338.53	778.93	948.2	0.911
	Ours	30	21.67	1225.87	549.4	947.93	0.3918
10	Tian et al.	30	27.17	2320.37	1327.07	1162.8	6.8368
	Ours	30	27.17	2130.97	958.7	1154.33	1.0434
12	Tian et al.	29	48.45	6430.83	3494.97	1597.38	60.2148
	Ours	30	48.45	5649.31	2577.14	1590.28	9.403

Note that for $W = 12$, 29 out of all 30 instances are reported feasible by Tian et al.'s model, while all 30 instances can be solved to optimality by our model. This can be explained as follows. If all feasible solutions for an instance contains similar situations with the one illustrated in Figure 4.3(b), Tian et al.'s model (as well as Zhao et al's model) may identify all of them as infeasible and consequently leads to infeasibility. On the other hand, our model can correctly deal with such situations and obtain an optimal solution.

From Table 4.6 and Table 4.7, we can also see that our model always includes fewer constraints and variables. This suggests that our model is more compact than Tian et al.'s. Consequently, our model is more effective in solving these instances than Tian et al.'s. This phenomenon becomes even more obvious as the value of W increases. This can be explained as follows. First, the tank capacity constraints for the single-capacity and multi-capacity tanks are separately formulated, which reduces the number of constraints and variables. Second, Tian et al.'s model used Zhao et al.'s formulation of tank capacity constraints which introduce additional binary variables.

As for the computational performance, our model performs better than Tian et al's in terms of makespan and computation time. First, our model can handle all instances correctly, while one instance is reported infeasible by Tian et al.'s for the reason explained above. Second, for three groups of instances, our model always reports shorter average makespan than Tian et al.'s as the latter one can identify feasible instance as infeasible. Third, our model is more effective than Tian et al.'s as our model always requires shorter average computation time. In summary, our model is more effective than Tian et al.'s in terms of both computational efficiency and solution quality.

4.6 Conclusions

This chapter proposed an improved MIP model for DHSP with multi-function tanks and multi-capacity tanks. We demonstrated that an existing model for a similar problem without multi-function tanks may obtain sub-optimal solution or identify feasible instance as infeasible.

To handle the tank capacity constraint correctly, a new MILP model was developed. It was shown that the proposed model can deal with a more general and complicated case where a multi-function tank can also be multi-capacity. The computational results indicated that the improved model can guarantee the optimality of the obtained solutions and is more efficient than an existing model in the literature.

5

Cyclic Jobshop Hoist Scheduling in Extended Lines

Contents

5.1 Introduction

As discussed in Chapter 2, few works have addressed cyclic jobshop hoist scheduling in extended lines due to its complexity. On the one hand, the formulation of tank capacity constraints is more complicated due to the following reasons. First, some tanks are not visited

by some parts in a jobshop. Furthermore, a multi-function tank will be visited by a part more than once. Thus the processing for parts will not follow the first-in-first-out rule. Second, the processing time in a multi-capacity tank can be longer than the cycle time, which is more complicated than that in non-cyclic scheduling. On the other hand, the tank capacity constraints and the processing time-window constraints should be consistent in order to guarantee all possible scenarios can be handled correctly. A prior work is the model proposed by Zhao et al. (2013a). The authors developed an MILP model for cyclic jobshop hoist scheduling in an extend line with multi-capacity tanks. The model is formulated by separately addressing the hoist capacity constraint, the tank capacity constraints, and the processing time-window constraints. However, their model can lead to sub-optimality or infeasibility due to some flawed formulation of tank capacity constraints and processing time-window constraints.

In this chapter, we consider the cyclic jobshop hoist scheduling in extended lines. In particular, we consider a more general cyclic hoist scheduling problem in a jobshop with both multi-function tanks and multi-capacity tanks. It involves the scenario where a multi-function tank can also be multi-capacity. We develop an MILP model for the problem. In particular, we extend the formulation of time-window constraints for cyclic hoist scheduling with a single part-type and multi-capacity tanks in the literature to the jobshop scenario. This is done by first examining the relationship between the actual processing time of a stage and the cycle time in different cases, and then formulating the general constraints with binary variables. As for tank capacity constraints, we transformed them into two equivalent families of constraints which can be separately addressed. For tank capacity constraints, we investigate the properties related to single-capacity tanks, with which the above general constraints can be tightened. Computational experiments are conducted to test the proposed model on an illustrative instance and a set of randomly generated instances.

The remainder of this chapter is organized as follows. Section 5.2 presents a formal description of the problem and defines the notation used throughout the chapter. In Section 5.3, an MILP model is formulated and analyzed. Section 5.4 is devoted to computational results and analysis. Finally, the work is concluded in Section 5.5.

5.2 Problem description and notation

The studied cyclic jobshop hoist scheduling problem can be described as follows. The jobshop is composed of a material handling hoist, W processing tanks denoted as $M_1, M_2, \ldots, M_W$, respectively, and an input station and an output station denoted as M_0 and M_{W+1}, respectively. Tank M_w, $1 \leq w \leq W$, contains C_w identical processing slots each of which can handle at most one part at a time. Thus the value of C_w defines the processing capacity of M_w, i.e., the maximum number of parts it can handle simultaneously. Tank M_w, $1 \leq w \leq W$, is called a single-capacity tank if $C_w = 1$ and a multi-capacity one otherwise. M_0 and M_{W+1} are assumed

to have infinite capacity; thus they can handle any number of parts. Parts of different types are processed in the jobshop. Each part-type is characterized by its specific processing routing, including the tanks to be visited, the order of visits and the required processing time-window in each tank. Note that some parts may visit certain (multi-function) tanks more than once while some parts will never visit certain tanks. Furthermore, it is assumed that any two consecutive stages of the same part cannot use the same tank, as the two stages can be treated as a single one in such a situation. To simplify the expression, the notation used throughout the chapter is summarized in Table 5.1.

Table 5.1: Notation and decision variables

Input data:	
R:	The number of parts in an Minimal Part Set (MPS).
W:	The number of processing tanks.
N_r:	The number of stages for part r in an MPS (Minimal Part Set), $1 \leq r \leq R$.
C_w:	The number of processing slots in tank M_w, $1 \leq w \leq W$.
$S_{r,i}$:	The i^{th} processing stage of a part of class r, $1 \leq r \leq R, 0 \leq i \leq N_r + 1$.
$V_{r,i}$:	The tank in which stage $S_{r,i}$ is processed, $1 \leq r \leq R, 0 \leq i \leq N_r + 1$.
$O_{r,i}$:	The hoist move that transports a part of class r from tank $V_{r,i}$ to tank $V_{r,i+1}$, $1 \leq r \leq R, 0 \leq i \leq N_r$.
Q_w:	The set of all stages processed in tank M_w, i.e., $Q_w = \{S_{r,i}$ such that $V_{r,i} = w, 1 \leq r \leq R, 0 \leq i \leq N_r + 1\}$, $1 \leq w \leq W$.
$L_{r,i}$:	The lower bound of the processing time-window of $S_{r,i}$, $1 \leq r \leq R$, $1 \leq i \leq N_r$.
$U_{r,i}$:	The upper bound of the processing time-window of $S_{r,i}$, $1 \leq r \leq R$, $1 \leq i \leq N_r$.
$D_{r,i}$:	The time required to execute move $O_{r,i}$, for all $1 \leq r \leq R, 0 \leq i \leq N_r$.
$E_{w,v}$:	The time required to execute unloaded move $G_{w,v}$, $0 \leq w, v \leq W + 1$.
M:	A very big positive number.
Decision variables:	
T:	The cycle time.
$s_{r,i}$:	The start time of move $O_{r,i}$ within a cycle, $1 \leq r \leq R, 0 \leq i \leq N_r$.
$t_{r,i}$:	The actual processing time of stage $S_{r,i}$, $1 \leq r \leq R$, $1 \leq i \leq N_r$.
$z_{r,i,k}$:	$0-1$ variable. If $t_{r,i}$ is within time interval $[kT, (k+1)T)$ then $z_{r,i,k} = 1$; otherwise, $z_{r,i,k} = 0$, $1 \leq r \leq R$, $1 \leq i \leq N_r$, $0 \leq k \leq C_{V_{r,i}} - 1$.
$y_{r,i,u,j}$:	$0-1$ variable. For any pair of moves $O_{r,i}$ and $O_{u,j}$, $1 \leq r, u \leq R, 0 \leq i \leq N_r$, $0 \leq j \leq N_u$, $y_{r,i,u,j} = 1$ if and only if move $O_{r,i}$ is executed before move $O_{u,j}$ within a cycle; otherwise, $y_{r,i,u,j} = 0$.

The production runs in a cyclic mode. During each cycle, an associated MPS composed of R parts is unloaded from M_0 to start its processing and a not-necessarily-the-same MPS is loaded into M_{W+1} after its processing is completed. Note that some of the R parts in an MPS

may be identical. To distinguish the parts, we say two parts are of the same *class* if and only if the difference between the moments they are unloaded from M_0 is a multiple of the cycle time T. Therefore, the R parts in each MPS, denoted as part 1, part 2, ..., part R, respectively, are of different classes one from another. Thus, all parts can be classified into R classes, denoted as class 1, class 2, ..., class R, respectively. To simplify the presentation, it is assumed that part r in each MPS is of class r, $r = 1, 2, \ldots, R$.

For part r in each MPS, let N_r be the number of required processing stages and $V_{r,i}$ the tank in which its i^{th} stage, $1 \leq i \leq N_r$, is processed. Furthermore, we assume that M_0 and M_{W+1} represent two dummy stages, the 0^{th} stage and the $(N_r + 1)^{\text{th}}$ stage, respectively; that is, we have $V_{r,0} = M_0$ and $V_{r,N_r+1} = M_{W+1}$ for all $r = 1, 2, \ldots, R$. Thus the processing flow for any part of class r can be described as follows. After the part is unloaded from M_0, it is successively processed in $V_{r,1}, V_{r,2}, \ldots, V_{r,N_r}$, and finally loaded into M_{W+1}. The processing time at its i^{th} stage, $1 \leq i \leq N_r$, must fall within a prescribed time-window $[L_{r,i}, U_{r,i}]$. The hoist is responsible for transporting the parts from a tank to another. Once a part of class r completes its processing in $V_{r,i}$, it must be moved to $V_{r,i+1}$, $0 \leq i \leq N_r$.

The operation that the hoist transports a part of class r from $V_{r,i}$ to $V_{r,i+1}$ is called a (loaded) move, and denoted as $O_{r,i}$, $1 \leq r \leq R$, $0 \leq i \leq N_r$; and the hoist operation of traveling from M_w to M_v without holding a part is called an unloaded move, and denoted as $G_{w,v}$, $0 \leq w, v \leq W + 1$. The hoist can hold at most one part at a time, and it is not allowed to wait during the execution of a loaded move. The time required for executing move $O_{r,i}$ is $D_{r,i}$, $1 \leq r \leq R$, $0 \leq i \leq N_r$, and the time required for executing unloaded move $G_{w,v}$ is $E_{w,v}$, $0 \leq w, v \leq W + 1$. It is worth noting that the hoist travel times, whether loaded or unloaded, only depend on the related tanks and are independent of the specific processing slots, as it is commonly treated in the literature (Liu et al., 2002; Li et al., 2015). Furthermore, $D_{r,i}$ and $E_{w,v}$ satisfy the triangular inequalities (5.1) and (5.2), which is usually the case in practice.

$$D_{r,i} > E_{V_{r,i},V_{r,i+1}}, \ \forall 1 \leq r \leq R, 0 \leq i \leq N_r. \tag{5.1}$$

$$E_{w,v} \leq E_{w,u} + E_{u,v}, \ \forall 0 \leq w, u, v \leq W + 1. \tag{5.2}$$

It is worth nothing that during each cycle, each of the R parts (in an MPS), after being unloaded from M_0 and until it is loaded onto M_{W+1}, is being either processed in some tank or held by the hoist. Furthermore, the processing of a part may span several cycles, depending on its processing time at each stage (Phillips & Unger, 1976; Liu et al., 2002). Due to the presence of multi-capacity tanks, several not-necessarily-identical parts may be simultaneously processed in the same tank during a cycle. We study the cyclic schedule when the system has been in steady state. In this situation, the hoist repeatedly executes a fixed sequence of moves during each cycle; thus a cyclic schedule can be represented by a permutation over the set of moves $\{O_{r,i} \mid 1 \leq r \leq R, 0 \leq i \leq N_r\}$ and the start time $s_{r,i}$ for each move $O_{r,i}$. To simplify the description and without loss of generality, the start time of move $O_{1,0}$, i.e., $s_{1,0}$, is set as the start point of a cycle.

With the problem described above, the objective is to obtain an optimal cyclic hoist schedule so that the cycle time is minimized; i.e., the throughput rate is maximized. A feasible cyclic hoist schedule should satisfy the following three categories of constraints (Phillips & Unger, 1976; Liu et al., 2002; Zhou et al., 2012; Zhao et al., 2013a):

- Hoist capacity constraints. Because the hoist can handle at most one part at a time, the start and end times of hoist moves should be well defined so that no conflict exists in the use of the hoist at any time.
- Time-window constraints. The actual processing time of each part at each stage should fall within its prescribed time-window.
- Tank capacity constraints. Each tank cannot handle more parts than its capacity at any time.

Figure 5.1 illustrates the time-way diagram of a cyclic schedule for a jobshop with $W = 6$ tanks and $R = 2$ classes of parts. For this example, M_2 is a multi-capacity tank with $C_2 = 3$ processing slots while the others are with single-capacity. At the start of a cycle, two parts of class 2 are being processed in M_2, and one part of class 1 is being processed in M_4. During each cycle, the hoist successively executes the moves in the following order: $O_{1,0}$, $O_{1,4}$, $O_{1,1}$, $O_{2,2}$, $O_{1,5}$, $O_{2,0}$, $O_{2,3}$, $O_{1,6}$, $O_{2,1}$, $O_{1,2}$, $O_{2,4}$, and $O_{1,3}$. Note that with move $O_{2,2}$, the hoist will unload a part of class 2 that is being processed at the start of the cycle. Note also that this part was being processed in the tank over the entire previous cycle and was unloaded from M_0 in an even earlier cycle. With move $O_{2,1}$, the hoist unloads the part (of class 2) from M_1 and transports it to M_2. Later on, this part will be processed in M_2 until the end of the current cycle, and for the entire next cycle. During each cycle, one part of class 2 is processed in M_2 during the entire cycle. This part is unloaded from M_0 in the previous cycle and will be loaded into M_7 in the next cycle. Among all the stages, the processing of any part of class 2 in M_2 (i.e., stage $S_{2,2}$) is longer than the cycle time T, while that for any other stage is less than the cycle time.

5.3 Problem formulation

In this section, we formulate an MILP model for the studied cyclic jobshop hoist scheduling problem by addressing its hoist capacity constraints, time-window constraints and tank capacity constraints, respectively.

5.3.1 Hoist capacity constraints

Since the hoist can handle at most one part at a time, it should not be required to handle another part when it is holding one. In other words, for any pair of moves $O_{r,i}$, $1 \le r \le R$, $0 \le i \le N_r$, and $O_{u,j}$, $1 \le u \le R$, $0 \le j \le N_u$, either move $O_{r,i}$ is executed before move $O_{u,j}$,

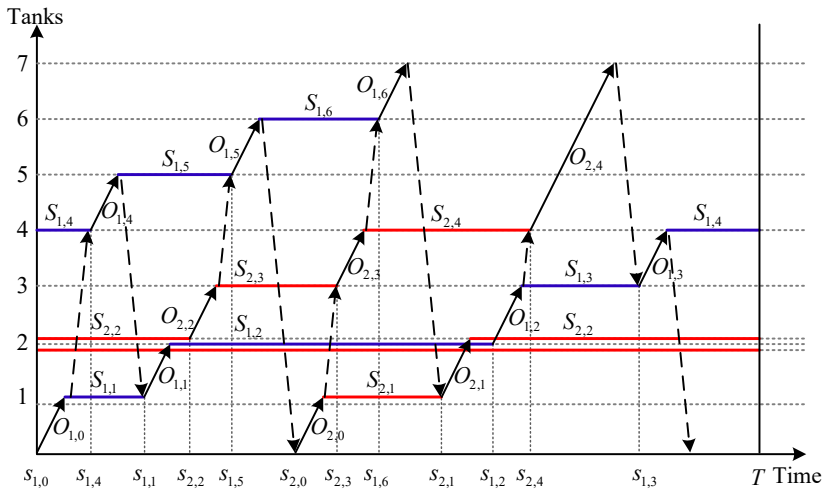

Figure 5.1: Illustration of a schedule with two classes of parts

or the opposite. Furthermore, after a move is completed, the hoist should have enough time to travel from the tank where it loads the part to the tank at which the next move will start. The requirements can be formulated as constraint (5.3) and (5.4) (Phillips & Unger, 1976; Liu et al., 2002; Leung et al., 2004; Zhou et al., 2012).

$$s_{u,j} - s_{r,i} \geq D_{r,i} + E_{V_{r,i+1},V_{u,j}} - M\left(1 - y_{r,i,u,j}\right),$$
$$\forall 1 \leq r,u \leq R, 0 \leq i \leq N_r, 0 \leq j \leq N_u; r \neq u \text{ or } i \neq j. \tag{5.3}$$

$$y_{r,i,u,j} + y_{u,j,r,i} = 1,$$
$$\forall 1 \leq r,u \leq R, 0 \leq i \leq N_r, 0 \leq j \leq N_u; r \neq u \text{ or } i \neq j. \tag{5.4}$$

As mentioned before, the start time of move $O_{1,0}$ is set as the start point of the cycle. Furthermore, any other move in the cycle must be executed after $O_{1,0}$ is completed. In addition, after the final move in the cycle is completed, the hoist should have enough time to travel back to the input station M_0 to execute the first move of the next cycle. These requirements can be represented by constraints (5.5)–(5.7), respectively.

$$s_{1,0} = 0. \tag{5.5}$$

$$s_{r,i} \geq D_{1,0} + E_{V_{1,1},V_{r,i}},\ \forall 1 \leq r \leq R, 0 \leq i \leq N_r; r \neq 1 \text{ or } i \neq 0. \tag{5.6}$$

$$T \geq s_{r,i} + D_{r,i} + E_{V_{r,i+1},0},\ \forall 1 \leq r \leq R, 0 \leq i \leq N_r. \tag{5.7}$$

5.3.2 Time-window constraints

Time-window constraints ensure that the actual processing time of each part at each stage is within its prescribed time-window. For any single-capacity tank, all the stages processed in it will share its single processing slot. However, for a multi-capacity tank, each stage may be processed in any of its processing slots, and multiple stages may be simultaneously processed in it during a cycle. Accordingly, the processing of a stage may start and end during the same cycle or in different cycles; and its processing time may vary depending on these situations.

The main difficulty comes from the fact that the actual processing time of any stage $S_{r,i}$, denoted as $t_{r,i}$, $1 \leq r \leq R$, $0 \leq i \leq N_r$, may be longer than the cycle time T. We introduce a binary variable $z_{r,i,k}$ which is equal to 1 if and only if $t_{r,i}$ is within time interval $[kT, (k+1)T)$ and is equal to 0 otherwise, where k is an integer.

We first determine the range of k. It is obvious that $k \geq 0$. Consider now its upper limit. If $z_{r,i,k} = 1$, $t_{r,i} \in [kT, (k+1)T)$. When move $O_{r,i-1}$ loads a part (of class r) into $V_{r,i}$ for its i^{th} stage (stage $S_{r,i}$) during a cycle, this stage will be still uncompleted k cycles later. Conversely, due to the cyclic nature, at time $s_{r,i-1}$ during the cycle, the parts of class r loaded into $V_{r,i}$ by move $O_{r,i-1}$ up to k cycles earlier (i.e., those loaded after time $s_{r,i-1} - kT$) are still in process for their i^{th} stages. There are k such parts, since exactly one such part is loaded into $V_{r,i}$ in each cycle, and each of them occupies a processing slot. Therefore, when move $O_{r,i-1}$ starts in a cycle, exactly k processing slots are occupied by parts of class r for their i^{th} stages. On the other hand, for $O_{r,i-1}$ to be able to start, there must be at least one processing slot available. As a consequence, we must have $k \leq C_{V_{r,i}} - 1$. The following constraint (5.8) is the result of the range of k and the definition of $z_{r,i,k}$'s.

$$\sum_{k=0}^{C_{V_{r,i}}-1} z_{r,i,k} = 1, \ \forall 1 \leq r \leq R, 1 \leq i \leq N_r. \tag{5.8}$$

In particular, equation (5.8) is a consistency constraint ensuring that $t_{r,i}$ is within exactly one of the disjoint intervals $\left\{[kT, (k+1)\,T) \mid k = 0, 1, \ldots, \left(C_{V_{r,i}} - 1\right)\right\}$.

We now establish the relation between the processing time of a stage and the decision variables. Before proceeding, let h be such that $t_{r,i} \in [hT, (h+1)T)$. In other words, relation (5.9) must hold.

$$h = \left\lfloor \frac{t_{r,i}}{T} \right\rfloor = \sum_{k=0}^{C_{V_{r,i}}-1} k z_{r,i,k}. \tag{5.9}$$

By definition, stage $S_{r,i}$ of a part starts at time $s_{r,i-1} + D_{r,i-1}$ and finishes at time $s_{r,i} + lT$ for some nonnegative integer l. In other words, $s_{r,i-1} + D_{r,i-1} + t_{r,i} = s_{r,i} + lT$ or equivalently,

$$t_{r,i} = s_{r,i} - \left(s_{r,i-1} + D_{r,i-1}\right) + lT. \tag{5.10}$$

According to equality (5.10) and the definition of h, we can obtain relation (5.11).

$$h = \left\lfloor \frac{t_{r,i}}{T} \right\rfloor = l + \left\lfloor \frac{s_{r,i} - \left(s_{r,i-1} + D_{r,i-1}\right)}{T} \right\rfloor. \tag{5.11}$$

By definition, if $y_{r,i-1,r,i} = 1$, from constraints (5.3), (5.6) and (5.7), we can obtain relation (5.12).

$$0 \leq s_{r,i} - \left(s_{r,i-1} + D_{r,i-1}\right) < s_{r,i} < T. \tag{5.12}$$

Therefore, relations (5.4), (5.11) and (5.12) can lead to relation (5.13).

$$h = \left\lfloor \frac{t_{t,i}}{T} \right\rfloor = l + \left\lfloor \frac{s_{r,i} - \left(s_{r,i-1} + D_{r,i-1}\right)}{T} \right\rfloor = l = l + y_{r,i-1,r,i} - 1 = l - y_{r,i,r,i-1}. \tag{5.13}$$

Coupling relations (5.9) and (5.13), we can obtain equality (5.14).

$$l = \sum_{k=0}^{C_{V_{r,i}}-1} k z_{r,i,k} - y_{r,i-1,r,i} + 1 = \sum_{k=0}^{C_{V_{r,i}}-1} k z_{r,i,k} + y_{r,i,r,i-1}. \tag{5.14}$$

Similarly, we can show that equality (5.14) also holds if $y_{r,i-1,r,i} = 0$. Relations (5.10) and (5.14) imply that the time-window constraints are equivalent to the following nonlinear constraint (5.15).

$$L_{r,i} \le s_{r,i} - \left(s_{r,i-1} + D_{r,i-1}\right) + \left(\sum_{k=0}^{C_{V_{r,i}}-1} k z_{r,i,k} - y_{r,i-1,r,i} + 1\right) T \le U_{r,i}. \tag{5.15}$$

With linearizing methods, constraint (5.15) can be formulated as linear constraints (5.16)–(5.19).

$$s_{r,i} - s_{r,i-1} - D_{r,i-1} + kT \ge L_{r,i} - M\left(2 - y_{r,i-1,r,i} - z_{r,i,k}\right), \quad \forall 1 \le r \le R, 1 \le i \le N_r, 0 \le k \le C_{V_{r,i}} - 1. \tag{5.16}$$

$$s_{r,i} - s_{r,i-1} - D_{r,i-1} + kT \le U_{r,i} + M\left(2 - y_{r,i-1,r,i} - z_{r,i,k}\right), \quad \forall 1 \le r \le R, 1 \le i \le N_r, 0 \le k \le C_{V_{r,i}} - 1. \tag{5.17}$$

$$s_{r,i} - s_{r,i-1} - D_{r,i-1} + (k+1)T \ge L_{r,i} - M\left(1 + y_{r,i-1,r,i} - z_{r,i,k}\right), \quad \forall 1 \le r \le R, 1 \le i \le N_r, 0 \le k \le C_{V_{r,i}} - 1. \tag{5.18}$$

$$s_{r,i} - s_{r,i-1} - D_{r,i-1} + (k+1)T \le U_{r,i} + M\left(1 + y_{r,i-1,r,i} - z_{r,i,k}\right), \quad \forall 1 \le r \le R, 1 \le i \le N_r, 0 \le k \le C_{V_{r,i}} - 1. \tag{5.19}$$

In particular, $t_{r,i}$ is correctly defined for the case $y_{r,i-1,r,i} = 1$ by constraints (5.16) and (5.17), and for the case $y_{r,i-1,r,i} = 0$ by constraints (5.18) and (5.19).

5.3.3 Tank capacity constraints

As for tank capacity constraints, each tank cannot handle more parts than its capacity at any time. As the start and end times of the stages are determined by their associated moves, the hoist moves should be well scheduled so that the parts processed in a tank should never exceeds its capacity at any time. This requirement is equivalent to the following two families of constraints:

*WCC*1*:* For each tank M_w, $1 \le w \le W$, at most C_w parts are being processed in it at the start/end point of each cycle.

*WCC*2*:* During each cycle, at the moment any move $O_{r,i-1}$, $1 \le r \le R$, $1 \le i \le N_r$, loads a part into $V_{r,i}$, at least one empty processing slot of $V_{r,i}$ is available.

The sufficiency of the above two families of constraints for ensuring tank capacities to be respected is obvious because they ensure there are enough processing slots to handle parts at the beginning of a cycle (the first family of constraints *WCC*1) and at any moment a part is loaded till the end of the cycle (the second family of constraints *WCC*2).

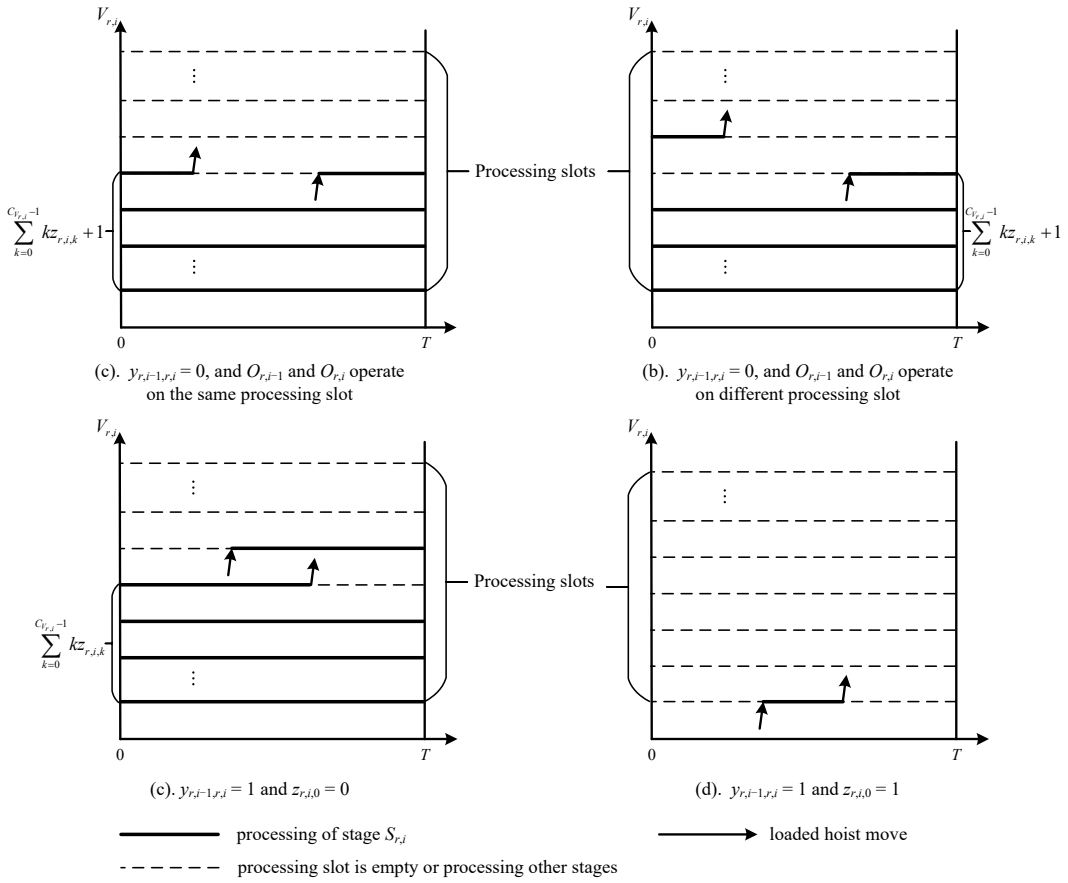

Figure 5.2: The use of processing slots of $V_{r,i}$ by stage $S_{r,i}$

We first deal with constraints $WCC1$. To formulate these constraints, we should know the number of parts being processed (i.e, the number of processing slots occupied by parts) in each tank at the beginning of a cycle. Let φ_w be the number of parts being processed in M_w at the beginning of a cycle. It is obvious that $\varphi_w \geq 0$. Thus, the first family of constraints can be formulated as (5.20).

$$0 \leq \varphi_w \leq C_w, \ \forall 1 \leq w \leq W. \tag{5.20}$$

The value of φ_w can be calculated by first determining the number of parts of each class r being processed in M_w and then summing these numbers up. As multi-function tanks are visited by some parts more than once, the parts of each class r, $1 \leq r \leq R$, may be processed in the same tank for different stages. In a tank, some parts of class r are processed for their i^{th} stage while some other parts of the same class are processed for stages. Thus in order to calculate φ_w, we first determine how many parts of each class r, $1 \leq r \leq R$, are being processed in M_w for their i^{th} stage at the start point of a cycle, and then obtain φ_w by considering all stages of the parts of each class.

For any part of class r, $1 \leq r \leq R$, and any stage index i, $1 \leq i \leq N_r$, such that $V_{r,i} = M_w$, we should determine how many parts of class r are being processed in $V_{r,i}$ for their i^{th} stages at the start point of a cycle. In fact, the number l defined in relation (5.14) gives this number, as illustrated in Figure 5.2. As analyzed above, if $t_{r,i}$ is within $[kT, (k+1)T)$ for some k, i.e.,

$z_{r,i,k} = 1$, k parts of class r are being processed in $V_{r,i}$ for their i^{th} stages at the moment move $O_{r,i-1}$ loads a part in $V_{r,i}$ during a cycle.

If $y_{r,i-1,r,i} = 0$, move $O_{r,i}$ is executed before this $O_{r,i-1}$ and has moved a part (of class r) away from $V_{r,i}$ after the its i^{th} stage is completed, thus there are $k+1$ (which is also equal to $k - y_{r,i-1,r,i} + 1$) parts of class r being processed for their i^{th} stages at the beginning of the cycle. Note that moves $O_{r,i-1}$ and $O_{r,i}$ may operate on the same processing slot (see Figure 5.2(a)) or different processing slots (see Figure 5.2(b)).

Similarly, if $y_{r,i-1,r,i} = 1$, no part of class r being processed for its i^{th} stage is removed from $V_{r,i}$ in a cycle before $O_{r,i-1}$ because move $O_{r,i}$ is executed after this $O_{r,i-1}$, thus there are also k (which is also equal to $k - y_{r,i-1,r,i} + 1$) parts of class r being processed for their i^{th} stages at the beginning of the cycle. Note that moves $O_{r,i-1}$ and $O_{r,i}$ necessarily operate on different processing slots if $z_{r,i,0} = 0$ (see Figure 5.2(c)) and the same processing slot if $z_{r,i,0} = 1$ (see Figure 5.2(d)).

In either cases, relation (5.14) correctly defines the number of parts (of class r) being processed in $V_{r,i}$ for their i^{th} stages at the start point of a cycle. Therefore, φ_w can be calculated with equality (5.21) by considering all stages of the parts of each class processed in M_w.

$$\varphi_w = \sum_{S_{r,i} \in Q_w} \left(\sum_{k=0}^{C_w - 1} k z_{r,i,k} - y_{r,i-1,r,i} + 1 \right), \ \forall 1 \le w \le W. \tag{5.21}$$

By substituting (5.21) into (5.20), the first family of tank capacity constraints $WCC1$ can be formulated as (5.22).

$$0 \le \sum_{S_{r,i} \in Q_w} \left(\sum_{k=0}^{C_w - 1} k z_{r,i,k} - y_{r,i-1,r,i} + 1 \right) \le C_w, \ \forall 1 \le w \le W. \tag{5.22}$$

We now deal with the second family of constraints $WCC2$ that guarantees at least one empty processing slot of $V_{r,i}$ is available for the part loaded by each move $O_{r,i-1}$, $1 \le r \le R$, $0 \le i \le N_r$. To this end, we should know how many processing slots of $V_{r,i}$ are occupied at this moment. Let $\varphi^1_{r,i}$ and $\varphi^2_{r,i}$ denote the number of parts that have been loaded into $V_{r,i}$ before move $O_{r,i-1}$ within a cycle and the number of parts unloaded from $V_{r,i}$ before move $O_{r,i-1}$ within a cycle, respectively. Recall that the number of parts being processed in $V_{r,i}$ at the beginning of a cycle, $\varphi_{V_{r,i}}$, can be calculated by equation (5.21). Thus at the moment move $O_{r,i-1}$ loads a part (of class r) into $V_{r,i}$, the number of occupied processing slots can be formulated as $\varphi_{V_{r,i}} + \varphi^1_{r,i} - \varphi^2_{r,i}$. It is worth noting that no other parts can be unloaded from or loaded into $V_{r,i}$ when the hoist is performing move $O_{r,i-1}$, i.e., between the start and end times of move $O_{r,i-1}$. The number of of processing slots occupied must be less than or equal to $C_{V_{r,i}} - 1$. Of course, the number of processing slots occupied must be nonnegative. Thus, the second family of tank capacity constraints $WCC2$ can be formulated as (5.23).

$$0 \le \varphi_{V_{r,i}} + \varphi^1_{r,i} - \varphi^2_{r,i} \le C_{V_{r,i}} - 1, \ \forall 1 \le r \le R, 1 \le i \le N_r. \tag{5.23}$$

To implement constraint (5.23), the values of $\varphi^1_{r,i}$ and $\varphi^2_{r,i}$ should be determined. We first derive the value of $\varphi^1_{r,i}$. For this purpose, we are interested in the parts loaded into $V_{r,i}$ before move $O_{r,i-1}$. Any move $O_{u,j-1}$, $1 \le u \le R$, $1 \le j \le N_u$, such that $V_{u,j} = V_{r,i}$ and $u \ne r$ or $j \ne i$, loads a part (of class u) into $V_{r,i}$ before move $O_{r,i-1}$ if and only if $y_{u,j-1,r,i-1} = 1$. As a result, $\varphi^1_{r,i}$ can be calculated by equation (5.24).

$$\varphi^1_{r,i} = \sum_{S_{u,j} \in Q_{V_{r,i}} \setminus \{S_{r,i}\}} y_{u,j-1,r,i-1}, \ \forall 1 \le r \le R, 1 \le i \le N_r. \tag{5.24}$$

Similarly, $\varphi^2_{r,i}$ can be calculated by equation (5.25). Note that equation (5.25) is achieved under the assumption that any two consecutive stages of the same part cannot be processed in the same tank.

$$\varphi^2_{r,i} = \sum_{S_{u,j} \in Q_{V_{r,i}}} y_{u,j,r,i-1}, \ \forall 1 \le r \le R, 1 \le i \le N_r. \tag{5.25}$$

By substituting equations (5.21), (5.24) and (5.25) into inequality (5.23), the second family of tank capacity constraints can be formulated as (5.26).

$$0 \le \sum_{S_{u,j} \in Q_{V_{r,i}}} \left(\sum_{k=0}^{C_{V_{r,i}}-1} k z_{u,j,k} + y_{u,j,u,j-1} - y_{u,j,r,i-1} \right) + \sum_{S_{u,j} \in Q_{V_{r,i}} \setminus \{S_{r,i}\}} \left(y_{u,j-1,r,i-1} \right) \le C_{V_{r,i}} - 1,$$
$$\forall 1 \le r \le R, 1 \le i \le N_r. \tag{5.26}$$

We now demonstrate that with constraint (5.26), the first family of tank capacity constraints (5.22) is respected automatically (i.e., becomes redundant). With constraint (5.26), the number of parts being processed at the start of a cycle is no more than C_w for any tank M_w. This can be explained by the fact that the same number of parts are processed in this tank at the start and end points of a cycle. At the moment the last loaded move in a cycle loads a part into tank M_w, constraint (5.26) ensures that at most $C_w - 1$ parts are processed in M_w. After this loaded move is completed, at most C_w parts are processed in M_w. No other part will be loaded into M_w in the remaining of the cycle; thus at most C_w parts are being processed in M_w at the end (i.e., start) point of each cycle.

As a consequence, the function of constraint (5.26) is two-fold. First, it ensures that at least one empty processing slot of $V_{r,i}$ is available at the moment when move $O_{r,i-1}$ loads a part into the tank during each cycle. Second, it also ensures that at most $C_{V_{r,i}}$ parts are being processed in $V_{r,i}$ at the start point of each cycle. This means the second family of tank capacity constraints is a *necessary and sufficient* condition for ensuring the tank capacity to be respected. In spite of this, constraint (5.22) can still be used to tighten the model.

Note that we did not distinguish single-capacity and multi-capacity tanks in the above analysis. Constraints (5.22) and (5.26) are obviously valid for single-capacity tanks. Nevertheless, we further analyze the properties of single-capacity tanks, and formulate a set of valid constraints.

For any single-capacity tank, all stages processed in it use the unique processing slot. No potential conflict in the use of the tank is equivalent to saying that any pair of stages processed

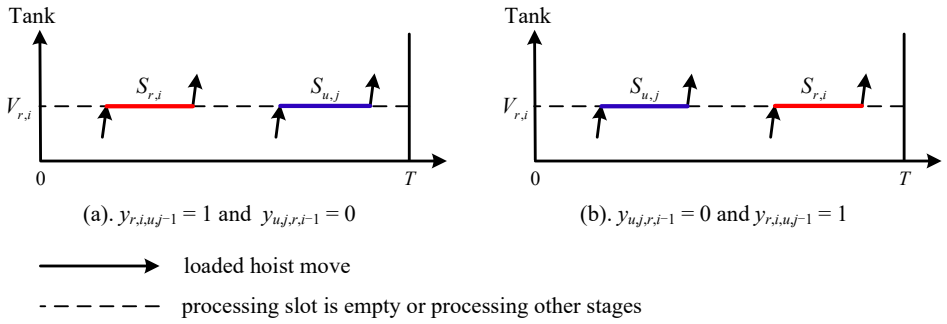

Figure 5.3: The use of workstation $V_{r,i}$ when $y_{r,i-1,r,i} = y_{u,j-1,u,j} = 1$

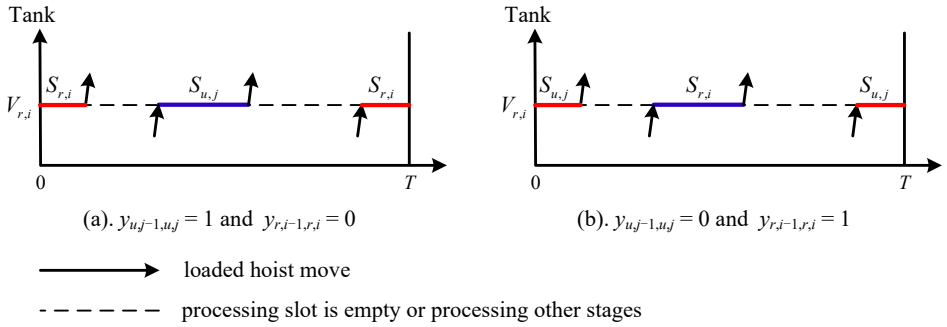

Figure 5.4: The use of workstation $V_{r,i}$ when $y_{r,i-1,r,i} \neq y_{u,j-1,u,j}$

in it have no overlap during their processing durations, which can be formulated as constraint (5.27).

$$y_{r,i-1,r,i} + y_{u,j-1,u,j} + y_{r,i,u,j-1} + y_{u,j,r,i-1} = 3,$$
$$\forall 1 \leq r, u \leq R, 1 \leq i \leq N_r, 1 \leq j \leq N_u; r \neq u \text{ or } i \neq j; V_{r,i} = V_{u,j}. \tag{5.27}$$

From constraint (5.22), for any pair of stages $S_{u,j}$ and $S_{r,i}$ such that $V_{r,i} = V_{u,j}$ and $C_{V_{r,i}} = 1$, at least one of $y_{u,j-1,u,j}$ and $y_{r,i-1,r,i}$ must be equal to 1.

If $y_{u,j-1,u,j} = y_{r,i-1,r,i} = 1$, both stages $S_{u,j}$ and $S_{r,i}$ start and end at the same cycle. For there to be no overlap, either stage $S_{u,j}$ starts after the completion of stage $S_{r,i}$ (i.e., $y_{r,i,u,j-1} = 1$ and $y_{u,j,r,i-1} = 0$, see Figure 3(a)), or $S_{r,i}$ starts after the completion of stage $S_{u,j}$ (i.e., $y_{u,j,r,i-1} = 1$ and $y_{r,i,u,j-1} = 0$, see Figure 3(b)). In other words, equality (5.27) must hold for this case.

If either $y_{u,j-1,u,j}$ or $y_{r,i-1,r,i}$ is equal to 1, we consider only the case where $y_{u,j-1,u,j} = 1$ and $y_{r,i-1,r,i} = 0$ (see Figure 4(a)), since the opposite case where $y_{u,j-1,u,j} = 0$ and $y_{r,i-1,r,i} = 1$ (see Figure 4(b)) is symmetric and can be analyzed similarly. In this case, stage $S_{u,j}$ starts and ends at the same cycle but stage $S_{r,i}$ starts at the previous cycle. As a consequence, stage $S_{u,j}$ has to start after the completion of stage $S_{r,i}$ (i.e., $y_{r,i,u,j-1} = 1$) but end before the start of the i^{th} stage of the next part of class r (i.e., $y_{u,j,r,i-1} = 1$). Once again, equality (5.27) must hold in this case.

Note that constraint (5.27) is similar to the formulation for cyclic hoist scheduling problem with a single part-type (Liu et al., 2002). Nevertheless, equation (5.27) is more compact than the inequality established in Liu et al. (2002).

Furthermore, the decision variables should satisfy their definitions (5.28)–(5.30).

$$s_{r,i} \geq 0,\ \forall 1 \leq r \leq R, 0 \leq i \leq N_r. \tag{5.28}$$

$$y_{r,i,u,j} \in \{0,1\},\ \forall 1 \leq r,u \leq R, 0 \leq i \leq N_r, 0 \leq j \leq N_u. \tag{5.29}$$

$$z_{r,i,k} \in \{0,1\},\ \forall 1 \leq r \leq R, 1 \leq i \leq N_r, 0 \leq k \leq C_{V_{r,i}} - 1. \tag{5.30}$$

With the above analysis, the cyclic jobshop hoist scheduling with multi-function and multi-capacity tanks can be formulated as follows.
P:

$$\text{Minimize} \quad T \tag{5.31}$$

Subject to:
Hoist capacity constraints: (5.3)–(5.7).
Time-window constraints: (5.8) and (5.16)–(5.19).
Tank capacity constraints: (5.22) and (5.26).
Valid inequality for single-capacity tanks: (5.27).
Variable definitions: (5.28)–(5.30).

5.4 Computational results

In this section, the proposed MILP model is evaluated on a series of numerical instances. We first give an illustrative instance to demonstrate how it is handled by the model. The time-way diagram of an optimal schedule for the instance is presented. To further validate the model, a set of randomly generated instances are constructed. The effects of the number of parts in an MPS, the number of multi-capacity tanks, and the hoist speed are evaluated. All instances are solved by ILOG CPLEX (Version 12.6.2) with default parameters. All experiments are conducted on a personal computer with 3.2 GHz CPU and 8.0 GB RAM.

5.4.1 An illustrative instance

The production line consists of $W = 12$ processing tanks, an input station and an output station. Each MPS contains $R = 3$ different parts, denoted by part 1, part 2, and part 3, with $N_1 = 12$, $N_2 = 8$ and $N_3 = 8$ stages, respectively. The other problem data is given in Table 5.2. Tanks M_2, M_7, M_8 are multi-capacity ones with $C_2 = 2$, $C_7 = 3$, $C_8 = 2$, respectively, while the other tanks are with single-capacity. The hoist travel times and move times are given as follows, $E_{i,i+1} = 2$, and $E_{i,j} = E_{j,i} = \sum_{k=i}^{j-1} E_{k,k+1}$ for $0 \leq i < j \leq W + 1$; $D_{r,i} = E_{V_{r,i},V_{r,i+1}} + 20$ for all $1 \leq r \leq R$, $0 \leq i \leq N_r$.

For this instance, the required processing times for the stages processed in M_2, M_7, M_8 are longer than that in other tanks, especially the three stages processed in tank M_7. To handle these stages effectively, the corresponding tanks are with multi-capacity ones as given before.

Table 5.2: Input data for the illustrative instance

i	1	2	3	4	5	6	7	8	9	10	11	12
$V_{1,i}$	1	2	3	4	5	6	7	8	9	10	11	12
$L_{1,i}$	30	150	60	60	30	40	350	90	70	45	60	30
$U_{1,i}$	60	350	90	120	75	120	800	160	200	90	120	80
$V_{2,i}$	1	2	4	5	7	8	11	12	–	–	–	–
$L_{2,i}$	30	130	30	40	420	50	90	70	–	–	–	–
$U_{2,i}$	50	280	70	90	850	120	160	200	–	–	–	–
$V_{3,i}$	1	3	5	7	8	9	10	12	–	–	–	–
$L_{3,i}$	30	80	40	300	50	90	90	70	–	–	–	–
$U_{3,i}$	50	220	90	550	120	150	160	200	–	–	–	–

Solving this instance with the proposed MILP model, we can obtain the optimal cycle time $T = 1005$ and its corresponding cyclic schedule as illustrated in Figure 5.5.

In Figure 5.5, the start and end times of a hoist move are labeled at its two ends of the corresponding solid arrow line, respectively. We can see that for each multi-capacity tank, multiple parts are processed simultaneously at some time instants, and the number of these parts does not exceed its capacity at any time. We demonstrate this by taking tank M_7 for an example. At the start point of a cycle, there are two parts being processed in M_7, and an empty processing slot is available. This empty processing slot is occupied since part 3 is loaded into it at time instant 372. From now on, as all three processing slots of M_7 are occupied, and no part can enter it unless some part is unloaded from it. In the remaining of the cycle, two parts are unloaded from M_7 at time instants 452 and 476, respectively; and two parts are loaded into the tank at time instants 704 and 742, respectively. Note that the two unloaded parts arrived at M_7 in previous cycle, and the two loaded parts will be unloaded from the tank in the subsequent cycle. After the part loaded at time 372 is unloaded at time 843, tank M_7 will keep this state until the end of the current cycle.

Note that the optimal cycle time for the problem without parallel processing slots is 1222. This shows that applying parallel processing slots leads to 17.76% reduction of the cycle time. This can be partially explained by the processing of the stages in tank M_7 as demonstrated above. The three stages processed in tank M_7, $S_{1,7}$, $S_{2,5}$, and $S_{3,4}$, have long processing times compared to other stages, and the sum of the lower bounds of the time-windows of the three stages, which is $L_{1,7} + L_{2,5} + L_{3,4} = 1070$, is longer the optimal cycle time $T = 1005$. With a multi-capacity tank, three processing slots are used to handle these stages. This allows multiple stages to be processed simultaneously during the cycle. As a results, a hoist schedule with a shorter cycle time is obtained.

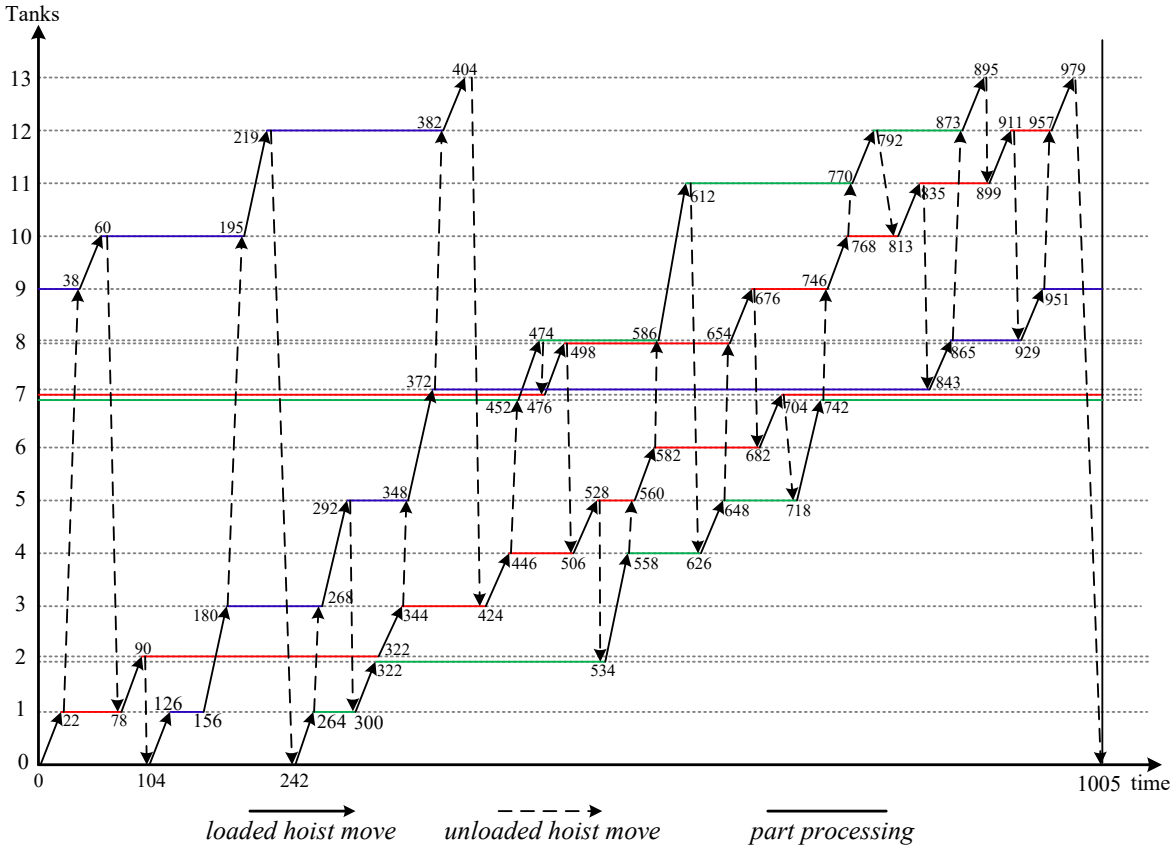

Figure 5.5: An optimal schedule for the illustrative example

5.4.2 Randomly generated instances

Furthermore, a set of random instances are generated to evaluate the proposed model. Let $\mathbf{U}(a, b)$ represents an integer located in the closed interval $[a, b]$ and derived by a uniform distribution.

We first generate a production line with W processing tanks, an input station and an output station. The unloaded hoist travel time between two adjacent tanks is set as $E_{i,i+1} \in \{2, 4\}$, and $E_{i,j} = E_{j,i} = \sum_{k=i}^{j-1} E_{k,k+1}$, $0 \leq i < j \leq W + 1$. For each part r, $1 \leq r \leq R$, the number of stages is set as $N_r = \mathbf{U}(\frac{W}{2}, W)$. For each stage $S_{r,i}$, its associated tank is set as $V_{r,i} = \mathbf{U}(1, W)$ such that $V_{r,i} \neq V_{r,i+1}$. The time required for executing move $O_{r,i}$ is $D_{r,i} = E_{V_{r,i},V_{r,i+1}} + 20$. The time-window for stage $S_{r,i}$ is generated by randomly selecting one of the following three sets: (1) $L_{r,i} = \mathbf{U}(30, 70)$ and $U_{r,i} = \mathbf{U}\left(1.2L_{r,i}, 1.5L_{r,i}\right)$; (2) $L_{r,i} = \mathbf{U}(50, 90)$ and $U_{r,i} = \mathbf{U}(1.5L_{r,i}, 2L_{r,i})$; and (3) $L_r, i = \mathbf{U}(90, 150)$ and $U_{r,i} = \mathbf{U}\left(2L_{r,i}, 4L_{r,i}\right)$.

Let P denote the number of multi-capacity tanks. To balance the workloads among the tanks, we choose P tanks with the heaviest workload as multi-capacity tanks. The workload of tank M_w is evaluated by the following equation.

$$WL_w = \sum_{S_{r,i} \in Q_w} \left(D_{r,i-1} + D_{r,i} + L_{r,i} + U_{r,i}\right)$$

The workload is set by taking into account the number of stages processed in a tank,

measured by item $(D_{r,i-1} + D_{r,i})$, and the required processing time for the stages, measured by item $(L_{r,i} + U_{r,i})$.

For each $W \in \{6, 8, 10, 12, 14, 16, 18, 20\}$, $R \in \{2, 3, 4\}$ and $P \in \{1, 2, 3\}$, the instances are generated by the following way. For each multi-capacity tank M_w, its capacity is determined by $C_w = \mathbf{U}(2, 3)$. To evaluate the effects of C_w, let T_0 be the cycle time for the corresponding instances with $C_w = 1$ for all $1 \leq w \leq W$. For each combination of W, R, P, denoted as (W, R, P), and $E_{i,i+1}$, ten instances are generated. There are in total 1440 instances for the test.

The computational results of the random tests are summarized in Table 5.3. The presented results are the average values of the ten instances for each combination of W, R, P and $E_{i,i+1}$. In this table, #Move represents the average number of moves to be scheduled, and #Cons and #Vars stand for the average numbers of constraints and variables, respectively. They measure the size of the instances. For the computational performance, T gives the average cycle time of the ten instances, I presents the ratio of the average cycle times for instances with and without multi-capacity tanks ($I = \frac{T}{T_0} \times 100\%$), #Node represents the average number of nodes explored in the solution procedure, and #Time gives the average CPU computation time measured in seconds.

Table 5.3: Computational results for random instances

(W, R, P)	#Move	#Cons	#Vars	$E_{i,i+1} = 2$				$E_{i,i+1} = 4$			
				T	I(%)	#Node	#Time	T	I (%)	#Node	#Time
(6, 2, 1)	10.6	315.4	128.9	355.0	85.05%	140.4	0.17	483.7	89.03%	108.3	0.23
(6, 2, 2)	10.6	325.8	131.5	344.6	82.56%	393.2	0.20	471.1	86.71%	223.5	0.19
(6, 2, 3)	10.6	335.0	133.8	325.4	77.96%	1305.6	0.27	460.7	84.80%	627.6	0.22
(6, 3, 1)	16.4	669.8	291.7	597.4	86.78%	897.9	0.49	809.4	89.90%	757.0	0.46
(6, 3, 2)	16.4	693.0	297.5	536.8	77.98%	2727.6	0.85	762.5	84.69%	2245.5	0.67
(6, 3, 3)	16.4	700.6	299.4	523.6	76.06%	3413.6	1.10	736.1	81.76%	2865.6	0.88
(6, 4, 1)	21.1	1055.5	473.1	717.7	82.04%	3002.9	1.77	937.0	86.80%	2639.1	1.73
(6, 4, 2)	21.1	1073.1	477.5	655.1	74.89%	8097.9	3.61	893.4	82.76%	6482.6	3.15
(6, 4, 3)	21.1	1097.1	483.5	630.3	72.05%	59157.8	15.81	851.1	78.84%	15541.7	6.67
(8, 2, 1)	14.0	510.0	215.0	495.6	82.92%	610.0	0.75	724.3	87.76%	626.4	0.75
(8, 2, 2)	14.0	525.6	218.9	469.6	78.57%	2049.0	1.02	696.0	84.33%	1403.8	0.85
(8, 2, 3)	14.0	535.2	221.3	464.6	77.73%	3500.9	1.30	689.3	83.52%	1891.1	0.92
(8, 3, 1)	22.2	1158.0	519.3	844.7	85.73%	3111.7	1.67	1193.5	92.86%	1927.9	1.26
(8, 3, 2)	22.2	1186.4	526.4	774.7	78.63%	5949.4	2.88	1122.3	87.32%	4581.7	2.00
(8, 3, 3)	22.2	1202.0	530.3	758.8	77.01%	8828.9	4.01	1092.2	84.98%	4239.5	2.24
(8, 4, 1)	26.9	1658.9	758.3	981.7	86.18%	5633.8	4.59	1419.6	89.99%	5828.6	3.08
(8, 4, 2)	26.9	1687.3	765.4	934.3	82.02%	26717.0	10.77	1366.6	86.63%	7631.6	4.77
(8, 4, 3)	26.9	1696.9	767.8	909.9	79.88%	98076.0	37.60	1308.0	82.92%	12859.8	9.43
(10, 2, 1)	16.1	658.5	282.0	575.8	90.62%	2004.9	0.96	961.9	98.18%	938.1	0.82
(10, 2, 2)	16.1	670.9	285.1	565.5	89.00%	3932.0	1.46	945.8	96.54%	1625.0	1.10
(10, 2, 3)	16.1	682.5	288.0	563.7	88.72%	6125.9	1.88	909.8	92.87%	1768.8	1.09
(10, 3, 1)	25.2	1480.6	670.5	965.2	93.09%	9207.1	4.34	1466.0	92.64%	5019.0	2.70
(10, 3, 2)	25.2	1498.6	675.0	939.7	90.63%	14591.1	6.19	1435.4	90.71%	6957.8	3.52
(10, 3, 3)	25.2	1519.4	680.2	916.3	88.38%	52668.4	15.26	1393.5	88.06%	8249.8	4.21

Table 5.3: Computational results for random instances (continued)

(W, R, P)	#Move	#Cons	#Vars	$E_{i,i+1} = 2$				$E_{i,i+1} = 4$			
				T	I(%)	#Node	#Time	T	I (%)	#Node	#Time
(10, 4, 1)	34.1	2603.7	1210.0	1378.4	95.54%	21908.3	16.90	2137.4	97.06%	8878.0	7.81
(10, 4, 2)	34.1	2631.3	1216.9	1331.7	92.31%	59331.2	35.56	2065.4	93.79%	11240.5	9.66
(10, 4, 3)	34.1	2655.3	1222.9	1289.7	89.39%	513994.0	380.35	2031.0	92.23%	29517.9	22.29
(12, 2, 1)	19.7	951.7	416.7	827.6	96.55%	3717.9	1.54	1328.2	98.61%	1544.1	1.00
(12, 2, 2)	19.7	961.3	419.1	811.7	94.69%	5415.1	2.03	1302.5	96.70%	1937.9	1.09
(12, 2, 3)	19.7	971.7	421.7	807.9	94.25%	6299.2	2.26	1298.9	96.44%	2826.0	1.30
(12, 3, 1)	31.7	2281.1	1050.9	1393.9	93.68%	11809.3	10.07	2265.3	97.00%	5216.4	4.52
(12, 3, 2)	31.7	2301.5	1056.0	1346.0	90.46%	21125.9	17.08	2162.8	92.61%	6643.5	5.88
(12, 3, 3)	31.7	2317.9	1060.1	1307.6	87.88%	59854.4	38.36	2123.0	90.91%	8738.5	7.83
(12, 4, 1)	41.1	3725.5	1752.2	1635.2	93.93%	73278.4	117.40	2585.1	94.10%	12082.7	27.85
(12, 4, 2)	41.1	3761.1	1761.1	1596.3	91.69%	195779.0	186.82	2562.5	93.27%	12469.6	29.80
(12, 4, 3)	41.1	3783.1	1766.6	1571.7	90.28%	253659.0	414.00	2497.3	90.90%	19062.1	49.80
(14, 2, 1)	23.7	1350.5	605.1	1038.5	93.63%	5264.1	2.97	1800.8	99.57%	2239.8	1.58
(14, 2, 2)	23.7	1366.5	609.1	1011.5	91.20%	6949.9	3.38	1769.8	97.85%	2788.0	1.44
(14, 2, 3)	23.7	1378.5	612.1	999.0	90.07%	7217.2	3.65	1749.2	96.72%	3056.7	1.68
(14, 3, 1)	35.7	2846.7	1324.1	1646.1	94.41%	22685.4	22.67	2978.4	99.02%	4795.0	5.38
(14, 3, 2)	35.7	2865.1	1328.7	1623.1	93.09%	78118.0	49.32	2911.3	96.79%	7431.7	6.18
(14, 3, 3)	35.7	2894.3	1336.0	1583.5	90.82%	146645.0	96.17	2821.7	93.81%	9545.9	7.22
(14, 4, 1)	46.7	4733.5	2240.7	2123.1	98.09%	47498.6	153.00	3619.7	98.42%	10700.4	29.70
(14, 4, 2)	46.7	4751.5	2245.2	2074.2	95.83%	210635.0	317.28	3600.1	97.89%	11406.0	38.84
(14, 4, 3)	46.7	4781.1	2252.6	2028.4	93.72%	224323.0	463.17	3514.3	95.55%	12937.7	50.15
(16, 2, 1)	24.1	1371.7	612.9	1201.7	97.80%	3744.7	1.95	2178.5	96.94%	850.4	1.10
(16, 2, 2)	24.1	1386.9	616.7	1188.8	96.75%	4303.2	2.18	2178.5	96.94%	712.0	1.17
(16, 2, 3)	24.1	1400.9	620.2	1185.2	96.46%	5619.5	2.88	2148.9	95.63%	1186.7	1.24
(16, 3, 1)	38.9	3350.3	1564.5	1874.6	97.81%	78734.5	63.30	3278.4	96.64%	7712.4	10.88
(16, 3, 2)	38.9	3370.3	1569.5	1831.8	95.58%	51514.8	61.52	3251.4	95.84%	9396.7	12.60
(16, 3, 3)	38.9	3377.5	1571.3	1816.3	94.77%	64659.3	73.34	3235.2	95.36%	9366.2	10.77
(16, 4, 1)	52.1	5856.9	2782.9	2550.2	96.84%	137367.0	451.27	4389.8	96.59%	14313.5	68.91
(16, 4, 2)	52.1	5877.7	2788.1	2498.0	94.85%	143292.0	633.82	4344.7	95.59%	15415.7	75.22
(16, 4, 3)	52.1	5911.3	2796.5	2438.7	92.60%	426549.0	1237.40	4299.6	94.60%	23967.5	160.11
(18, 2, 1)	32.5	2396.1	1101.0	1812.4	97.11%	7290.6	5.23	3223.4	100.00%	1449.9	1.88
(18, 2, 2)	32.5	2408.9	1104.2	1775.1	95.11%	8944.3	6.04	3223.4	100.00%	1451.8	1.74
(18, 2, 3)	32.5	2426.5	1108.6	1721.6	92.24%	12067.9	7.57	3196.7	99.17%	1896.6	2.22
(18, 3, 1)	42.6	4059.6	1908.1	2207.0	98.76%	49605.9	120.43	3882.3	100.00%	5152.3	13.46
(18, 3, 2)	42.6	4070.8	1910.9	2196.0	98.27%	119411.0	399.04	3857.7	99.37%	6232.9	19.52
(18, 3, 3)	42.6	4099.6	1918.1	2180.8	97.59%	105355.0	334.33	3807.7	98.08%	8539.3	20.83
(18, 4, 1)	60.1	7736.9	3703.1	3186.7	96.51%	65514.5	558.97	5599.5	97.17%	13351.9	109.92
(18, 4, 2)	60.1	7764.5	3710.0	3122.3	94.56%	97491.8	830.16	5523.9	95.86%	10142.1	98.41
(18, 4, 3)	60.1	7797.7	3718.3	3059.6	92.66%	125042.0	971.23	5471.4	94.95%	20796.4	190.40
(20, 2, 1)	30.2	2088.2	950.9	1686.2	98.61%	6849.8	4.67	3003.5	100.00%	1803.6	1.69
(20, 2, 2)	30.2	2099.4	953.7	1659.4	97.05%	6388.7	4.92	2945.1	98.06%	1612.9	1.75
(20, 2, 3)	30.2	2107.8	955.8	1657.9	96.96%	8893.4	5.82	2945.1	98.06%	2486.0	1.92

Table 5.3: Computational results for random instances (continued)

(W, R, P)	#Move	#Cons	#Vars	$E_{i,i+1} = 2$				$E_{i,i+1} = 4$			
				T	I(%)	#Node	#Time	T	I (%)	#Node	#Time
(20, 3, 1)	47.6	4938.6	2332.4	2809.2	98.12%	50663.7	105.16	4905.1	100.00%	8641.1	14.23
(20, 3, 2)	47.6	4957.4	2337.1	2744.6	95.86%	89069.8	152.34	4834.2	98.55%	8037.6	14.34
(20, 3, 3)	47.6	4979.8	2342.7	2728.5	95.30%	78767.7	221.00	4833.5	98.54%	9619.6	15.44
(20, 4, 1)	64.8	8968.0	4305.7	3659.0	97.78%	132682.0	1724.76	6454.2	98.13%	16765.2	288.30
(20, 4, 2)	64.8	9002.8	4314.4	3606.6	96.38%	213538.0	2647.62	6361.4	96.72%	17722.7	391.62
(20, 4, 3)	64.8	9018.8	4318.4	3572.0	95.45%	382829.0	4689.31	6322.7	96.13%	22505.9	305.05

From Table 5.3, we can see that the average cycle time per each part (T/R) benefits from multi-capacity tanks. The increase of both R and P can lead to a certain degree of reduction of the average cycle time per part. In particular, as the value of P increases, which means the number of multi-capacity tanks increases, it usually yields a shorter cycle time. For the instances with $E_{i,i+1} = 4$, the increase of P usually leads to a large reduction of the cycle time when W is less than or equal to 10. However, when W becomes larger, the reduction becomes less significant. For some cases (see the instances with $I = 100\%$), the increase of P even has no effect on the cycle time. When $E_{i,i+1}$ changes from 4 to 2, which means a much faster hoist, the situation is improved to some extent. This is because the hoist is becoming more critical as the number of tanks (W) and the number of moves to be scheduled (#Move) increase, and a much faster hoist can overcome this bottleneck in some degree. However, such an improvement is limited as a too fast hoist may reduce the stability of the system in practice.

In particular, as the value of R increases, which usually leads to a larger number of moves to be scheduled (#Move), it may yields a higher productivity measured by the average cycle time. A careful analysis demonstrates that the increase of cycle time when R changes from 3 to 4 is relatively small compared with that observed when R changes from 2 to 3. A reasonable explanation for this is that when R increases, more parts and stages are involved, there is less probability for the resources to get idle.

For the instances with a high-speed hoist ($E_{i,i+1} = 2$), the number of nodes explored and the computation time increase accordingly, and this trend becomes more obvious as the number of moves (#Move) increases. This is because the sequences of the hoist moves that are infeasible when the hoist is slow become feasible for a faster hoist. As the number of moves (#Move) increases, the number of feasible sequences of the hoist moves increases sharply. Note that when W becomes large, which implies a longer production line, the number of nodes explored and the computation time decrease for some instances. This can be partially explained as follows. In these situations, the hoist has to serve a large number of tanks and perform loaded and unloaded moves that span over a long distance. Many sequences of the hoist moves become infeasible due to the time-window constraints. This means that the hoist is becoming the bottleneck of the system. However, when the number of moves (#Move) increases, the solution space becomes larger accordingly. The computation time will be the

results of the trade-off of the two factors.

Due to the NP-hardness of the problem, the computation time sharply increases as the size of the problem increases such as the number of parts in an MPS R, the number of moves (#Move), the number of tanks (W), and the number of multi-capacity tanks (P). However, the proposed model can, within a reasonable amount of time, solve the scheduling problem with $W = 20$ tanks and $R = 4$ classes of parts, which is very realistic for lines with a single hoist.

Table 5.4: Comparison of the results obtained with and without valid inequalities

(W, R, P)	$E_{i,i+1} = 2$		$E_{i,i+1} = 4$		(W, R, P)	$E_{i,i+1} = 2$		$E_{i,i+1} = 4$	
	%Node	%Time	%Node	%Time		%Node	%Time	%Node	%Time
(6, 2, 1)	81.34%	93.59%	61.22%	75.36%	(14, 2, 1)	106.78%	95.69%	96.36%	88.60%
(6, 2, 2)	101.70%	95.19%	80.72%	92.07%	(14, 2, 2)	103.71%	108.97%	90.75%	103.30%
(6, 2, 3)	90.14%	95.99%	99.25%	95.76%	(14, 2, 3)	113.58%	107.19%	93.15%	97.85%
(6, 3, 1)	65.06%	84.06%	91.62%	97.87%	(14, 3, 1)	154.73%	108.06%	107.88%	89.45%
(6, 3, 2)	108.42%	114.60%	90.06%	98.33%	(14, 3, 2)	91.64%	91.45%	108.25%	106.84%
(6, 3, 3)	102.19%	91.57%	89.54%	87.96%	(14, 3, 3)	78.10%	83.31%	76.09%	98.37%
(6, 4, 1)	59.00%	67.22%	74.16%	76.51%	(14, 4, 1)	216.91%	112.85%	135.76%	112.59%
(6, 4, 2)	102.54%	92.49%	94.89%	88.20%	(14, 4, 2)	39.07%	69.78%	103.27%	72.18%
(6, 4, 3)	86.82%	86.82%	83.42%	83.33%	(14, 4, 3)	127.50%	98.30%	96.36%	78.07%
(8, 2, 1)	84.90%	95.70%	82.01%	93.61%	(16, 2, 1)	98.53%	99.92%	73.58%	93.98%
(8, 2, 2)	104.42%	87.71%	95.10%	100.21%	(16, 2, 2)	110.58%	106.05%	89.94%	92.26%
(8, 2, 3)	105.52%	104.39%	90.75%	97.98%	(16, 2, 3)	108.23%	95.34%	104.52%	114.50%
(8, 3, 1)	62.75%	78.31%	72.37%	90.79%	(16, 3, 1)	114.92%	100.32%	98.42%	71.59%
(8, 3, 2)	84.28%	87.01%	72.73%	77.01%	(16, 3, 2)	174.21%	118.79%	94.86%	92.77%
(8, 3, 3)	125.73%	114.76%	106.94%	102.54%	(16, 3, 3)	171.94%	155.40%	86.27%	101.35%
(8, 4, 1)	88.32%	74.03%	86.99%	92.04%	(16, 4, 1)	88.22%	71.39%	78.74%	55.08%
(8, 4, 2)	90.85%	93.59%	92.06%	96.03%	(16, 4, 2)	136.61%	100.71%	78.39%	60.87%
(8, 4, 3)	78.06%	104.97%	122.30%	91.01%	(16, 4, 3)	272.36%	162.60%	67.39%	51.08%
(10, 2, 1)	104.32%	96.20%	110.84%	97.72%	(18, 2, 1)	109.38%	101.67%	101.15%	90.63%
(10, 2, 2)	104.79%	99.41%	94.83%	100.75%	(18, 2, 2)	84.08%	88.65%	101.65%	103.41%
(10, 2, 3)	101.48%	97.42%	87.91%	95.80%	(18, 2, 3)	97.49%	122.11%	107.90%	94.11%
(10, 3, 1)	90.13%	85.14%	93.83%	96.41%	(18, 3, 1)	122.47%	117.71%	106.91%	105.82%
(10, 3, 2)	111.61%	105.17%	94.79%	94.31%	(18, 3, 2)	89.43%	87.25%	81.45%	86.95%
(10, 3, 3)	117.66%	117.80%	103.25%	106.91%	(18, 3, 3)	97.62%	133.17%	79.88%	102.98%
(10, 4, 1)	94.92%	79.16%	101.23%	84.64%	(18, 4, 1)	102.77%	84.76%	113.78%	65.29%
(10, 4, 2)	124.13%	109.77%	98.97%	95.27%	(18, 4, 2)	97.51%	95.62%	162.48%	63.97%
(10, 4, 3)	106.91%	117.41%	60.89%	60.65%	(18, 4, 3)	223.75%	110.54%	103.07%	52.85%
(12, 2, 1)	104.57%	98.28%	106.48%	100.29%	(20, 2, 1)	96.68%	91.63%	95.02%	103.42%
(12, 2, 2)	92.80%	95.99%	96.85%	96.18%	(20, 2, 2)	121.80%	95.37%	112.59%	105.06%
(12, 2, 3)	96.96%	95.83%	82.82%	93.83%	(20, 2, 3)	75.95%	92.22%	107.59%	103.34%
(12, 3, 1)	144.38%	103.86%	97.67%	83.04%	(20, 3, 1)	98.15%	87.44%	66.53%	83.83%
(12, 3, 2)	137.58%	102.62%	114.04%	88.71%	(20, 3, 2)	52.45%	85.15%	88.46%	85.14%
(12, 3, 3)	93.49%	89.41%	115.05%	103.25%	(20, 3, 3)	143.27%	138.98%	106.80%	140.70%

Table 5.4: Comparison of the results obtained with and without valid inequalities (continued)

(W, R, P)	$E_{i,i+1} = 2$		$E_{i,i+1} = 4$		(W, R, P)	$E_{i,i+1} = 2$		$E_{i,i+1} = 4$	
	%Node	%Time	%Node	%Time		%Node	%Time	%Node	%Time
(12, 4, 1)	176.60%	87.19%	93.18%	74.12%	(20, 4, 1)	81.64%	79.47%	92.18%	77.34%
(12, 4, 2)	77.80%	77.71%	123.84%	100.29%	(20, 4, 2)	82.40%	77.96%	166.67%	79.82%
(12, 4, 3)	210.86%	124.01%	193.94%	100.40%	(20, 4, 3)	116.39%	82.27%	122.53%	138.91%

To evaluate the influence of the valid inequality (5.27), we also test the above instances with the model without inequality (5.27). Table 5.4 presents the comparison of the results obtained with and without valid inequality (5.27). The presented results are the average values of the ten instances for each combination of (W, R, P) and $E_{i,i+1}$. In this table, %Node gives the ratio of the average numbers of nodes explored during solution procedure of the models with and without inequality (5.27), and %Time represents the ratio of the average computation times for the solution procedure of the models with and without inequality (5.27).

Note that in Table 5.4, if the ratio is larger than one, it means that the number of the nodes or the amount of computation time related to the model without inequality (5.27) is less than these related to the model with inequality (5.27). For the instances with $E_{i,i+1} = 2$, more than half of the instances explore a larger number of nodes if inequality (5.27) is involved, and as for computation time the proportion is about $1/3$. This means that the solution procedure with inequality (5.27) explores more nodes in a shorter time for some instances while in a longer time for some other instances. Note that a similar phenomenon occurs for the instances with $E_{i,i+1} = 4$. This can be partially explained as follows. On the one hand, the solver may be able to generate tighter linear relaxations by implementing inequality (5.27). On the other hand, with inequality (5.27) the solver has a trend to explore more nodes during the solution because of these extra constraints. Thus for the instances that more nodes are explored in a shorter time, the tighter linear relaxations will dominate the total computation time while the increased number of nodes to explore will play a leading role for the others. Furthermore, a careful examination demonstrates that the model with inequality (5.27) performs more effective on the instances with $E_{i,i+1} = 4$ than those with $E_{i,i+1} = 2$. This can be partially explained as follows. As mentioned above, a high-speed hoist usually implies more feasible sequences of hoist moves, and the solution space become large as a result. To search for an optimal solution for the instances with $E_{i,i+1} = 2$, the solver usually explore a large number of nodes. Therefore, it also has a larger possibility to be affected by the extra constraints introduced by inequality (5.27).

5.5 Conclusions

In this chapter, we studied the cyclic jobshop hoist scheduling with multi-function and multi-capacity tanks. An MILP model is developed for the problem. The presence of multiple

part-types, multi-function tanks and multi-capacity tanks makes the problem more complicated. To model the problem, we separately addressed its time-window constraints, hoist capacity constraints and tank capacity constraints. The properties of single-capacity tanks were also investigated to simplify and tighten the model. An illustrative instance and a set of randomly generated instances are used to test the effectiveness and efficiency of the proposed model.

6

Conclusions and Future Research

Contents

Hoist scheduling problems arise in the surface treating. In this scenario, the material handling device is a hoist. As the hoist executes all transportation operations between tanks, it plays an important role in the production. A well planned schedule of the operations can improve the productivity of the system. The thesis studies several hoist scheduling problems in basic and extended electroplating lines. We developed MILP models for these problems to improve the throughput. In this chapter, we first conclude our main research work. Then, we discuss the limitations of the present research and suggest potentially promising directions for future research.

6.1 Conclusions

In this research, three types of hoist scheduling problems are investigated: robust optimization for cyclic hoist scheduling problems, a dynamic jobshop hoist scheduling problem in extended lines and a cyclic jobshop hoist scheduling problem in extended lines. The main contributions are summarized as follows.

Chapter 3 studied the robust optimization for a cyclic hoist scheduling problem. The robustness of a cyclic hoist schedule is defined as its ability to remain feasible in the presence of perturbations or variations of certain degree in the hoist transportation times. With such

a definition, we proposed a method to measure the robustness in terms of the free slacks in both loaded and unloaded hoist moves. A bi-objective MILP model was developed, which aims to simultaneously optimize the cycle time and the robustness. It was proved that the optimal cycle time strictly increases with the value of the robustness. With this property, the problem has an infinite number of Pareto optimal solutions. Furthermore, we examined the lower and upper bounds of these objectives, and derived the ideal and nadir points which define the region containing the Pareto front. A Pareto optimal solution can be obtained by solving a single-objective hoist scheduling problem that aims to minimize the cycle time for a given value of robustness or maximize the robustness for a specific cycle time. It was also demonstrated that the single-objective HSP that aims to minimize the cycle time for a given value of robustness can be transformed into a classical HSP with some substitutions. Computational results on several benchmark instances and randomly generated instances indicate that the proposed approach can effectively handle the problem.

Chapter 4 examined a dynamic jobshop hoist scheduling problem with multi-function and multi-capacity tanks. In a general case, a multi-function tank can also be multi-capacity. The parts to be processed dynamically arrive at the input station. Upon arrival of new parts, a rescheduling may be triggered to schedule all parts in the system, including the waiting ones and the uncompleted ones, such that the makespan is minimized. A counterexample demonstrates that an existing MILP model for a similar problem but without multi-function tanks may lead to sub-optimality due to the flawed formulation of tank capacity constraints. To deal with this issue, a new MILP model was developed to generate an optimal reschedule. The key point is the formulation of capacity constraints for multi-capacity tanks. This was handled by considering the collision-avoidance constraints for each processing slot of the tanks. Computational experiments on instances with and without multi-function tanks were carried out to evaluate the proposed model and compare it with an existing model. The results indicate that the proposed model can always obtain an optimal solution, while the existing model may lead to infeasibility for some cases. Furthermore, the proposed model is more compact and effective than the existing model in terms of both solution quality and computation time.

Chapter 5 investigated a cyclic jobshop hoist scheduling problem with multi-function and multi-capacity tanks. Multiple parts of different types are to be processed during a cycle. An MILP model was developed for the problem by addressing the time-window constraints, hoist capacity constraints and tank capacity constraints. As for tank capacity constraints, we equivalently transformed them into two families of constraints which can be separately addressed. We showed that when one of the families of constraints are satisfied, the other family of constraints are automatically satisfied. We showed also that the above general constraints can be simplified and tightened for single-capacity tanks. Computational results on an illustrative instance and a set of randomly generated instances indicate the effectiveness and efficiency of the proposed model.

6.2 Limitations and future research

This thesis focuses on modeling and optimization of three types of hoist scheduling problems arising in electroplating industry. In what follows, we discuss the limitations of this research and suggest some further work which seems to be relevant and promising.

(1). The hoist scheduling problems studied in this thesis consider production lines with a single hoist. However, many production systems, especially large ones, may be quipped with multiple hoists running on a single track or parallel tracks. With multiple hoists, several material handling tasks can be executed at the same time. Thus, it is desirable to consider multi-hoist scheduling in both situations with a single track or parallel tracks.

(2). In the present research, the processing route for each part is fixed and prescribed in advance. In practice, several alternative processing routes may be available and equivalent from a technological perspective. The final processing route can be determined by considering factors such as ease of management and cost efficiency. The availability of alternative processing routes provides flexibility in the scheduling of the material handling operations. For example, a specific stage of a part can be handled by one of two tanks located on different positions, thus the actual processing route of the part can involve either one of the tanks. This yields two different processing routes for the part.

(3). The main criterion for hoist scheduling problems is the cycle time or makespan. In this thesis, we also consider the robustness of cyclic schedules. Besides these objectives, other factors may be important and need to be considered in scheduling. Production cost is a critical factor that affects the decision-making in production and operations planning. Several costs, such as energy consumption and loss due to defective parts, can be considered. Some of the costs are related to material handling operations. Thus, it is desirable to take these related costs into consideration when dealing with scheduling of hoist operations.

(4). Most works assume that the resources are continuously available, and no breakdown happens during the planning horizon. Obviously, it is not the case in reality. Equipment failures may occur during production. The consequences may be serious if the hoist device breaks down. In this case, all subsequent tasks will be delayed, and defective parts may be produced because of the violation of processing time-windows. Furthermore, if a defective part is detected, it is desirable to be able to remove it from the production as soon as possible. From this point of view, if a failure occurs, a recovery strategy that yields the least influence would be of much interest.

(5). Besides electroplating lines, material handling devices are widely used in many other sectors such as hoists in steel production, cranes in quay scheduling, cluster tools in wafer fabrication. Similar scheduling problems can be found in these systems. Future work can adapt the results obtained in the present research to these systems.

Bibliography

Achterberg, T. (2009). SCIP: solving constraint integer programs. *Mathematical Programming Computation*, *1*(1), 1–41. doi:10.1007/s12532-008-0001-1. (Cited on page 4)

Agnetis, A. (2000). Scheduling no-wait robotic cells with two and three machines. *European Journal of Operational Research*, *123*(2), 303–314. doi:10.1016/s0377-2217(99)00258-1. (Cited on pages 29 and 31)

Armstrong, R., Gu, S., & Lei, L. (1996). A greedy algorithm to determine the number of transporters in a cyclic electroplating process. *IIE Transactions*, *28*(5), 347–355. doi:Doi10.1080/07408179608966281. (Cited on pages 17 and 22)

Armstrong, R., Lei, L., & Gu, S. (1994). A bounding scheme for deriving the minimal cycle time of a single-transporter n-stage process with time-window constraints. *European Journal of Operational Research*, *78*(1), 130–140. doi:http://dx.doi.org/10.1016/0377-2217(94)90127-9. (Cited on pages 14 and 15)

Baptiste, P., Legeard, B., & Varnier, C. (1992). Hoist scheduling problem: an approach based on constraint logic programming. In *Proceedings 1992 IEEE international conference on robotics and automation* (Volume 2, Pages 1139–1144). IEEE. doi:10.1109/ROBOT.1992.220195. (Cited on pages 14, 15 and 17)

Baptiste, P., Legeard, B., Manier, M.-A., & Varnier, C. (1993). Optimization with constraint logic programming: the hoist scheduling problem solved with various solvers. *WIT Transactions on Information and Communication Technologies*, *1*, 599–614. (Cited on pages 14, 15 and 17).

Billaut, J.-C., Moukrim, A., & Sanlaville, E. (2010). *Flexibility and robustness in scheduling*. Wiley. com. (Cited on page 34).

Bloch, C., Varnier, C., & Baptiste, P. (1996). Applying stochastic methods to the real time hoist scheduling problem. In *CESA'96 IMACS Multiconference: computational engineering in systems applications* (Pages 479–484). (Cited on pages 23 and 27).

Branke, J., Deb, K., Miettinen, K., & Slowinski, R. (2008). *Multiobjective optimization: interactive and evolutionary approaches*. Springer. (Cited on page 45).

Brauner, N. (2008). Identical part production in cyclic robotic cells: concepts, overview and open questions. *Discrete Applied Mathematics*, *156*(13), 2480–2492. doi:10.1016/j.dam.2008.03.021. (Cited on page 13)

Briskorn, D., Leung, J., & Pinedo, M. (2011). Robust scheduling on a single machine using time buffers. *IIE Transactions*, *43*(6), 383–398. doi:10.1080/0740817X.2010.505123. (Cited on page 34)

Brucker, P., Burke, E. K., & Groenemeyer, S. (2012). A mixed integer programming model for the cyclic job-shop problem with transportation. *Discrete Applied Mathematics*, *160*(13-14), 1924–1935. doi:10.1016/j.dam.2012.04.001. (Cited on page 34)

Chauvet, F., Levner, E., Meyzin, L. K., & Proth, J.-M. (2000). On-line scheduling in a surface treatment system. *European Journal of Operational Research*, *120*(2), 382–392. (Cited on pages 10, 25, 27 and 54).

Che, A. & Chu, C. (2004). Single-track multi-hoist scheduling problem: a collision-free resolution based on a branch-and-bound approach. *International Journal of Production Research*, *42*(12), 2435–2456. doi:10.1080/00207540410001666288. (Cited on pages 21 and 22)

Che, A. & Chu, C. (2005a). A polynomial algorithm for no-wait cyclic hoist scheduling in an extended electroplating line. *Operations Research Letters*, *33*(3), 274–284. doi:10.1016/j.orl.2003.10.012. (Cited on pages 10, 29 and 31)

Che, A. & Chu, C. (2005b). Multi-degree cyclic scheduling of two robots in a no-wait flowshop. *IEEE Transactions on Automation Science And Engineering*, *2*(2), 173–183. doi:10.1109/tase.2004.835600. (Cited on pages 30 and 31)

Che, A., Chu, C., & Chu, F. (2002). Multicyclic hoist scheduling with constant processing times. *IEEE Transactions on Robotics And Automation*, *18*(1), 69–80. (Cited on pages 29 and 31).

Che, A., Hu, H., Chabrol, M., & Gourgand, M. (2011a). A polynomial algorithm for multi-robot 2-cyclic scheduling in a no-wait robotic cell. *Computers & Operations Research*, *38*(9), 1275–1285. doi:10.1016/j.cor.2010.11.008. (Cited on pages 30 and 31)

Che, A., Chabrol, M., Gourgand, M., & Wang, Y. (2012). Scheduling multiple robots in a no-wait re-entrant robotic flowshop. *International Journal of Production Economics*, *135*(1), 199–208. doi:10.1016/j.ijpe.2011.07.008. (Cited on pages 30 and 31)

Che, A. & Chu, C. (2007a). An efficient solution to cyclic scheduling of a no-wait reentrant serial-parallel PCB production line. In H. Lan (Editor), *Proceedings of 2007 international conference on management science & engineering* (Pages 746–751). doi:10.1109/ICMSE.2007.4421935. (Cited on pages 29 and 31)

Che, A. & Chu, C. (2007b). Cyclic hoist scheduling in large real-life electroplating lines. *OR Spectrum*, *29*(3), 445–470. doi:10.1007/s00291-006-0040-9. (Cited on pages 8, 10, 16, 17, 19 and 35)

Che, A. & Chu, C. (2008). Optimal scheduling of material handling devices in a pcb production line: problem formulation and a polynomial algorithm. *Mathematical Problems in Engineering*, *2008*(0), 1–21. doi:10.1155/2008/364279. (Cited on pages 29 and 31)

Che, A. & Chu, C. (2009). Multi-degree cyclic scheduling of a no-wait robotic cell with multiple robots. *European Journal of Operational Research*, *199*(1), 77–88. doi:10.1016/j.ejor.2008.10.035. (Cited on pages 11, 30 and 31)

Che, A., Chu, C., & Levner, E. (2003). A polynomial algorithm for 2-degree cyclic robot scheduling. *European Journal of Operational Research*, *145*(1), 31–44. doi:10.1016/S0377-2217(02)00175-3. (Cited on pages 28 and 31)

Che, A., Lei, W., Feng, J., & Chu, C. (2014). An improved mixed integer programming approach for multi-hoist cyclic scheduling problem. *IEEE Transactions on Automation Science and Engineering*, *11*(1), 302–309. doi:Doi10.1109/Tase.2013.2254713. (Cited on pages 21 and 22)

Che, A., Yan, P., Yang, N. D., & Chu, C. (2010). Optimal cyclic scheduling of a hoist and multi-type parts with fixed processing times. *International Journal of Production Research*, *48*(5), 1225–1243. doi:10.1080/00207540802552659. (Cited on pages 29 and 31)

Che, A., Zhou, Z., Chu, C., & Chen, H. (2011b). Multi-degree cyclic hoist scheduling with time window constraints. *International Journal of Production Research*, *49*(19), 5679–5693. doi:10.1080/00207543.2010.503200. (Cited on pages 11, 16 and 17)

Chen, H., Chu, C., & Proth, J. M. (1998). Cyclic scheduling of a hoist with time window constraints. *IEEE Transactions on Robotics And Automation*, *14*(1), 144–152. (Cited on pages 10, 14, 15, 17, 35, 41, 42 and 46).

Cheng, C. C. & Smith, S. F. (1997). Applying constraint satisfaction techniques to job shop scheduling. *Annals of Operations Research*, *70*, 327–357. doi:10.1023/a:1018934507395. (Cited on pages 25 and 27)

Chtourou, S., Manier, M.-A., & Loukil, T. (2013). A hybrid algorithm for the cyclic hoist scheduling problem with two transportation resources. *Computers & Industrial Engineering*, *65*(3), 426–437. doi:http://dx.doi.org/10.1016/j.cie.2013.03.013. (Cited on pages 21 and 22)

Chu, C. (2006). A faster polynomial algorithm for 2-cyclic robotic scheduling. *Journal of Scheduling*, *9*(5), 453–468. doi:10.1007/s10951-006-8501-1. (Cited on pages 28 and 31)

Cornuéjols, G. (2008). Valid inequalities for mixed integer linear programs. *Mathematical Programming*, *112*(1), 3–44. doi:10.1007/s10107-006-0086-0. (Cited on page 3)

Crama, Y., Kats, V., van de Klundert, J., & Levner, E. (2000). Cyclic scheduling in robotic flowshops. *Annals of Operations Research*, *96*, 97–124. doi:10.1023/a:1018995317468. (Cited on page 13)

Dawande, M. W., Geismar, H. N., Sethi, S. P., & Sriskandarajah, C. (2007). *Throughput optimization in robotic cells*. Springer Science & Business Media. (Cited on pages 11 and 13).

Dawande, M., Geismar, H. N., Pinedo, M., & Sriskandarajah, C. (2010). Throughput optimization in dual-gripper interval robotic cells. *IIE Transactions*, *42*(1), 1–15. doi:10.1080/07408170902789092. (Cited on pages 3 and 13)

Dooley, K. J. & Mahmoodi, F. (1992). Identification of robust scheduling heuristics: application of taguchi methods in simulation studies. *Computers & Industrial Engineering*, *22*(4), 359–368. doi:http://dx.doi.org/10.1016/0360-8352(92)90012-9. (Cited on page 34)

E. G., C., Jr, Garey, M. R., & Johnson, D. S. (1978). An application of bin-packing to multiprocessor scheduling. *SIAM Journal on Computing*, *7*(1), 1–17. doi:doi:10.1137/0207001. (Cited on page 25)

El Amraoui, A., Manier, M.-A., El Moudni, A., & Benrejeb, M. (2011). Heuristic for the resolution of the cyclic hoist scheduling problem with multi-items. In *18th ifac world congress, august 28, 2011 - september 2, 2011* (Volume 44, *1*, Pages 8195–8200). IFAC Proceedings Volumes. IFAC Secretariat. doi:10.3182/20110828-6-IT-1002.03372. (Cited on pages 16 and 17)

El Amraoui, A., Manier, M.-A., El Moudni, A., & Benrejeb, M. (2008). A mixed linear program for a multi-part cyclic hoist scheduling problem. *International Journal of Sciences and Techniques of Automatic control and computer engineering (IJ-STA), Special issue on CEM*, *2*, 612–623. (Cited on pages 16 and 17).

El Amraoui, A., Manier, M.-A., El Moudni, A., & Benrejeb, M. (2012). Resolution of the two-part cyclic hoist scheduling problem with bounded processing times in complex lines' configuration. *European Journal of Industrial Engineering*, *6*(4), 454–473. doi:10.1504/ejie.2012.047661. (Cited on pages 16 and 17)

El Amraoui, A., Manier, M.-A., El Moudni, A., & Benrejeb, M. (2013a). A genetic algorithm approach for a single hoist scheduling problem with time windows constraints. *Engineering Applications of Artificial Intelligence*, *26*(7), 1761–1771. doi:10.1016/j.engappai.2013.02.004. (Cited on pages 16 and 17)

El Amraoui, A., Manier, M.-A., El Moudni, A., & Benrejeb, M. (2013b). A linear optimization approach to the heterogeneous r-cyclic hoist scheduling problem. *Computers & Industrial Engineering*, *65*(3), 360–369. doi:10.1016/j.cie.2013.03.007. (Cited on pages 10, 16 and 17)

Fargier, H. & Lamothe, J. (2001). Handling soft constraints in hoist scheduling problems: the fuzzy approach. *Engineering Applications of Artificial Intelligence*, *14*(3), 387–399. doi:10.1016/S0952-1976(01)00008-2. (Cited on page 26)

Feng, J., Che, A., & Wang, N. (2014). Bi-objective cyclic scheduling in a robotic cell with processing time windows and non-Euclidean travel times. *International Journal of Production Research*, *52*(9), 2505–2518. doi:10.1080/00207543.2013.849015. (Cited on page 11)

Fleury, G., Gourgand, M., & Lacomme, P. (2001). Metaheuristics for the stochastic hoist scheduling problem (SHSP). *International Journal of Production Research*, *39*(15), 3419–3457. doi:10.1080/00207540110058331. (Cited on pages 23, 27 and 34)

Fu, J., Zhao, C. Y., Xu, Q., & Ho, T. C. (2013). Debottleneck of multistage material-handling processes via simultaneous hoist scheduling and production line retrofit. *Industrial & Engineering Chemistry Research*, *52*(1), 123–133. doi:10.1021/ie300550a. (Cited on page 19)

Ge, Y. & Yih, Y. (1995). Crane scheduling with time windows in circuit board production lines. *International Journal of Production Research*, *33*(5), 1187–1199. doi:10.1080/00207549508930203. (Cited on pages 25, 27 and 54)

Ginsberg, M. L. (1993). Dynamic backtracking. *Journal of Artificial Intelligence Research*, *1*, 25–46. (Cited on page 26).

Goujon, J.-Y., Lacomme, P., Norre, S., & Traoré, M. K. (1996). Computerized industrial surface treatment line control: resolution of hoist scheduling problem. *Simulation in Industry, ESS*, *96*, 377–382. (Cited on pages 24 and 27).

Gurobi Optimization, I. (2016). *Gurobi optimizer reference manual*. Houston, Texas: Gurobi Optimization, Inc. (Cited on page 4).

Herroelen, W. & Leus, R. (2004). Robust and reactive project scheduling: a review and classification of procedures. *International Journal of Production Research*, *42*(8), 1599–1620. doi:10.1080/00207540310001638055. (Cited on page 34)

Al-Hinai, N. & ElMekkawy, T. Y. (2011). Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. *International Journal of Production Economics*, *132*(2), 279–291. doi:10.1016/j.ijpe.2011.04.020. (Cited on page 34)

Hindi, K. S. & Fleszar, K. (2004). A constraint propagation heuristic for the single-hoist, multiple-products scheduling problem. *Computers & Industrial Engineering*, *47*(1), 91–101. doi:10.1016/j.cie.2004.03.002. (Cited on pages 25 and 27)

IBM ILOG, C. (2009). IBM ILOG CPLEX optimization studio: CPLEX user's manual. (Cited on page 4).

Jegou, D., Kim, D. W., Baptiste, P., & Lee, K. H. (2006). A contract net based intelligent agent system for solving the reactive hoist scheduling problem. *Expert Systems with Applications*, *30*(2), 156–167. doi:10.1016/j.eswa.2005.06.019. (Cited on pages 24 and 27)

Jiang, Y. & Liu, J. (2007). Multihoist cyclic scheduling with fixed processing and transfer times. *IEEE Transactions on Automation Science and Engineering*, *4*(3), 435–450. doi:10.1109/TASE.2006.884057. (Cited on pages 30 and 31)

Jiang, Y. & Liu, J. (2014). A new model and an efficient branch-and-bound solution for cyclic multi-hoist scheduling. *IIE Transactions*, *46*(3), 249–262. (Cited on pages 21 and 22).

Kats, V., Lei, L., & Levner, E. (2008). Minimizing the cycle time of multiple-product processing networks with a fixed operation sequence, setups, and time-window constraints. *European Journal of Operational Research*, *187*(3), 1196–1211. doi:10.1016/j.ejor.2006.07.030. (Cited on page 14)

Kats, V. & Levner, E. (1997a). A strongly polynomial algorithm for no-wait cyclic robotic flowshop scheduling. *Operations Research Letters*, *21*(4), 171–179. doi:Doi10.1016/S0167-6377(97)00036-9. (Cited on pages 28 and 31)

Kats, V. & Levner, E. (1997b). Minimizing the number of robots to meet a given cyclic schedule. *Annals of Operations Research*, *69*, 209–226. doi:10.1023/a:1018980928352. (Cited on pages 29 and 31)

Kats, V. & Levner, E. (1998). Minimizing the number of vehicles in periodic scheduling: the non-Euclidean case. *European Journal of Operational Research*, *107*(2), 371–377. doi:Doi10.1016/S0377-2217(97)00339-1. (Cited on page 11)

Kats, V. & Levner, E. (2002). Cyclic scheduling in a robotic production line. *Journal of Scheduling*, *5*(1), 23–41. doi:10.1002/jos.92. (Cited on pages 29 and 31)

Kats, V. & Levner, E. (2009). A polynomial algorithm for 2-cyclic robotic scheduling: a non-Euclidean case. *Discrete Applied Mathematics*, *157*(2), 339–355. doi:10.1016/j.dam.2008.03.025. (Cited on pages 11, 28 and 31)

Kats, V. & Levner, E. (2011a). A faster algorithm for 2-cyclic robotic scheduling with a fixed robot route and interval processing times. *European Journal of Operational Research*, *209*(1), 51–56. doi:10.1016/j.ejor.2010.10.002. (Cited on pages 14 and 15)

Kats, V. & Levner, E. (2011b). Parametric algorithms for 2-cyclic robot scheduling with interval processing times. *Journal of Scheduling*, *14*(3), 267–279. doi:10.1007/s10951-010-0166-0. (Cited on page 14)

Kats, V. & Levner, E. (2012). Cyclic flowshop scheduling with operators and robots: Vyacheslav Tanaev's vision and lasting contributions. *Journal of Scheduling*, *15*(4), 419–425. doi:10.1007/s10951-010-0221-x. (Cited on page 13)

Kats, V., Levner, E., & Meyzin, L. (1999). Multiple-part cyclic hoist scheduling using a sieve method. *IEEE Transactions on Robotics and Automation*, *15*(4), 704–713. doi:10.1109/70.781993. (Cited on pages 15, 28 and 31)

Kats, V. & Mikhailetskii, Z. (1980). Exact solution of a cyclic scheduling problem. *Automation and Remote Control*, *4*, 187–190. (Cited on pages 28 and 31).

Kats, V. & Levner, E. (2006). A polynomial algorithm for 2-cyclic robotic scheduling. In A. Gelbukh & C. A. Reyes-Garcia (Editors), *Micai 2006: advances in artificial intelligence: 5th mexican international conference on artificial intelligence, apizaco, mexico, november 13-17, 2006. proceedings* (Pages 439–449). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/11925231_41. (Cited on pages 28 and 31)

Kirlik, G. & Sayın, S. (2014). A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. *European Journal of Operational Research*, *232*(3), 479–488. doi:http://dx.doi.org/10.1016/j.ejor.2013.08.001. (Cited on page 46)

Kujawski, K. & Swiatek, J. (2011). Electroplating production scheduling by cyclogram unfolding in dynamic hoist scheduling problem. *International Journal of Production Research*, *49*(17), 5355–5371. doi:10.1080/00207543.2010.519733. (Cited on pages 26 and 27)

Kumar, P. R. (1994). Scheduling semiconductor manufacturing plants. *IEEE Control Systems*, *14*(6), 33–40. doi:10.1109/37.334413. (Cited on page 3)

Lamothe, J., Correge, M., & Delmas, J. (1994). Hoist scheduling problem in a real time context. In G. Cohen & J.-P. Quadrat (Editors), *11th international conference on analysis and optimization of systems discrete event systems* (Chapter 65, Volume 199, Pages 586–592). Lecture Notes in Control and Information Sciences. Springer Berlin Heidelberg. doi:10.1007/BFb0033591. (Cited on pages 10, 26, 27 and 54)

Lamothe, J., Correge, M., & Delmas, J. (1995). A dynamic heuristic for the real time hoist scheduling problem. In *Emerging technologies and factory automation, 1995. etfa '95, proceedings., 1995 inria/ieee symposium on* (Volume 2, Pages 161–168). doi:10.1109/ETFA.1995.496655. (Cited on pages 26, 27, 54, 59 and 60)

Lamothe, J., Thierry, C., & Delmas, J. (1996). A multihoist model for the real time hoist scheduling problem. In *CESA'96 IMACS Multiconference: computational engineering in systems applications: symposium on discrete events and manufacturing systems* (Pages 461–466). (Cited on pages 26, 27 and 54).

Lee, C. Y., Lei, L., & Pinedo, M. (1997). Current trends in deterministic scheduling. *Annals of Operations Research*, *70*, 1–41. doi:10.1023/a:1018909801944. (Cited on pages 8 and 13)

Lee, T. E. (2008). A review of scheduling theory and methods for semiconductor manufacturing cluster tools. In *2008 winter simulation conference* (Pages 2127–2135). doi:10.1109/WSC.2008.4736310. (Cited on page 13)

Lei, L. (1993). Determining the optimal starting times in a cyclic schedule with a given route. *Computers & Operations Research*, *20*(8), 807–816. doi:Doi10.1016/0305-0548(93)90102-O. (Cited on page 14)

Lei, L. & Liu, Q. (2001). Optimal cyclic scheduling of a robotic processing line with two-product and time-window constraints. *INFOR: information systems and operational research*, *39*(2), 185–199. (Cited on pages 16 and 17).

Lei, L. & Wang, T. J. (1989). A proof: the cyclic hoist scheduling problem is np-complete. *Graduate School of Management, Rutgers University, Working Paper*, 89–0016. (Cited on page 3).

Lei, L. & Wang, T. J. (1994). Determining optimal cyclic hoist scheduling in a single-hoist electroplating line. *IIE Transactions*, *26*(2), 25–33. doi:10.1080/07408179408966593. (Cited on pages 10, 11, 14, 15, 16 and 17)

Lei, L., Armstrong, R., & Gu, S. (1993). Minimizing the fleet size with dependent time-window and single-track constraints. *Operations Research Letters*, *14*(2), 91–98. (Cited on page 22).

Lei, L. & Wang, T.-J. (1991). The minimum common-cycle algorithm for cyclic scheduling of two material handling hoists with time window constraints. *Management Science*, *37*(12), 1629–1639. doi:doi:10.1287/mnsc.37.12.1629. (Cited on pages 20 and 22)

Lei, W., Che, A., & Chu, C. (2014). Optimal cyclic scheduling of a robotic flowshop with multiple part types and flexible processing times. *European Journal of Industrial Engineering*, *8*(2), 143–167. doi:10.1504/EJIE.2014.060434. (Cited on pages 10, 16 and 17)

LEON, V. J., WU, S. D., & STORER, R. H. (1994). Robustness measures and robust scheduling for job shops. *IIE transactions*, *26*(5), 32–43. doi:10.1080/07408179408966626. (Cited on page 35)

Leung, J. M. Y. & Levner, E. (2006). An efficient algorithm for multi-hoist cyclic scheduling with fixed processing times. *Operations Research Letters*, *34*(4), 465–472. doi:10.1016/j.orl.2005.07.010. (Cited on pages 29 and 31)

Leung, J. M. Y., Zhang, G. Q., Yang, X. G., Mak, R., & Lam, K. (2004). Optimal cyclic multi-hoist scheduling: a mixed integer programming approach. *Operations Research*, *52*(6), 965–976. doi:10.1287/opre.1040.0144. (Cited on pages 10, 11, 21, 22, 35, 47 and 76)

Leung, J. & Zhang, G. (2003). Optimal cyclic scheduling for printed circuit board production lines with multiple hoists and general processing sequence. *IEEE Transactions on Robotics and Automation*, *19*(3), 480–484. (Cited on pages 21, 22 and 60).

Levner, E. & Kats, V. (1998). A parametric critical path problem and an application for cyclic scheduling. *Discrete Applied Mathematics*, *87*(1-3), 149–158. doi:Doi10.1016/S0166-218x(98)00054-7. (Cited on page 14)

Levner, E., Kats, V., & Levit, V. E. (1997). An improved algorithm for cyclic flowshop scheduling in a robotic cell. *European Journal of Operational Research*, *97*(3), 500–508. doi:10.1016/s0377-2217(96)00272-x. (Cited on pages 8, 15, 28, 29, 31, 48 and 49)

Levner, E., Kats, V., Alcaide López de Pablo, D., & Cheng, T. C. E. (2010). Complexity of cyclic scheduling problems: a state-of-the-art survey. *Computers & Industrial Engineering*, *59*(2), 352–361. doi:http://dx.doi.org/10.1016/j.cie.2010.03.013. (Cited on pages 8 and 13)

Levner, E., Kats, V., & De Pablo, D. A. L. (2007). Cyclic scheduling in robotic cells: an extension of basic models in machine scheduling theory. *Multiprocessor scheduling: theory and applications*, 1–20. (Cited on page 14).

Levner, E., Kats, V., & Sriskandarajah, C. (1996). A geometric algorithm for finding two-unit cyclic schedules in no-wait robotic flowshop. In *Proceedings of the international workshop in intelligent scheduling of robots and fms, wisor* (Volume 96, Pages 101–112). (Cited on pages 28 and 31).

Li, X. & Fung, R. Y. K. (2013). A mixed integer linear programming approach for multi-degree cyclic multi-hoist scheduling problems without overlapping. In *2013 ieee international conference on automation science and engineering (case)* (Pages 274–279). IEEE. doi:10.1109/CoASE.2013.6653893. (Cited on pages 21 and 22)

Li, X. & Fung, R. Y. K. (2014). A mixed integer linear programming solution for single hoist multi-degree cyclic scheduling with reentrance. *Engineering Optimization*, *46*(5), 704–723. doi:10.1080/0305215x.2013.795560. (Cited on pages 11, 18 and 19)

Li, X., Chan, F. T. S., & Chung, S. H. (2015). Optimal multi-degree cyclic scheduling of multiple robots without overlapping in robotic flowshops with parallel machines. *Journal of Manufacturing Systems*, *36*, 62–75. doi:http://dx.doi.org/10.1016/j.jmsy.2015.03.003. (Cited on pages 11, 18, 19, 21, 22 and 74)

Lim, J.-M. (1997). A genetic algorithm for a single hoist scheduling in the printed-circuit-board electroplating line. *Computers & Industrial Engineering*, *33*(3), 789–792. (Cited on pages 14, 15, 17 and 46).

Liu, J., Jiang, Y., & Zhou, Z. (2002). Cyclic scheduling of a single hoist in extended electroplating lines: a comprehensive integer programming solution. *IIE Transactions*, *34*(10), 905–914. doi:Doi10.1080/07408170208928921. (Cited on pages 9, 10, 11, 16, 18, 19, 35, 41, 42, 60, 74, 75, 76 and 82)

Lu, C.-C., Lin, S.-W., & Ying, K.-C. (2012). Robust scheduling on a single machine to minimize total flow time. *Computers & Operations Research*, *39*(7), 1682–1691. doi:http://dx.doi.org/10.1016/j.cor.2011.10.003. (Cited on page 34)

Mak, R. W. T., Gupta, S. M., & Lam, K. (2002). Modeling of material handling hoist operations in a pcb manufacturing facility. *Journal of Electronics Manufacturing*, *11*(1), 33–50. doi:10.1142/S0960313102000229. (Cited on pages 24 and 27)

Manier, M.-A. & Bloch, C. (2003). A classification for hoist scheduling problems. *International Journal of Flexible Manufacturing Systems*, *15*(1), 37–55. doi:Doi10.1023/A:1023952906934. (Cited on pages 9, 23 and 54)

Manier, M.-A. & Lamrous, S. (2008). An evolutionary approach for the design and scheduling of electroplating facilities. *Journal of Mathematical Modelling and Algorithms*, *7*(2), 197–215. doi:10.1007/s10852-008-9083-z. (Cited on pages 8, 10, 20, 22 and 47)

Manier, M.-A., Varnier, C., & Baptiste, P. (2000). Constraint-based model for the cyclic multi-hoists scheduling problem. *Production Planning and Control*, *11*(3), 244–257. doi:10.1080/095372800232216. (Cited on pages 20 and 22)

Mateo, M. & Companys, R. (2006). Hoist scheduling in a chemical line to produce batches with identical sizes of different products. In *Sixième conférence francophone de Modélisation et SIMulation, MOSIM* (Volume 6, Pages 677–684). (Cited on pages 16 and 17).

Ng, W. C. (1995). Determining the optimal number of duplicated process tanks in a single-hoist circuit board production line. *Computers and Industrial Engineering*, *28*(4), 681–688. doi:10.1016/0360-8352(95)98201-K. (Cited on pages 17, 18 and 19)

Ng, W. C. (1996). A branch and bound algorithm for hoist scheduling of a circuit board production line. *International Journal of Flexible Manufacturing Systems*, *8*(1), 45–65. doi:Doi10.1007/Bf00167800. (Cited on pages 16, 18 and 19)

Paul, H. J., Bierwirth, C., & Kopfer, H. (2007). A heuristic scheduling procedure for multi-item hoist production lines. *International Journal of Production Economics*, *105*(1), 54–69. doi:10.1016/j.ijpe.2005.11.008. (Cited on pages 25 and 27)

Phillips, L. W. & Unger, P. S. (1976). Mathematical programming solution of a hoist scheduling program. *AIIE transactions*, *8*(2), 219–225. (Cited on pages 3, 8, 10, 14, 16, 17, 19, 26, 35, 41, 42, 47, 74, 75 and 76).

Rahmani, D. & Heydari, M. (2013). Robust and stable flow shop scheduling with unexpected arrivals of new jobs and uncertain processing times. *Journal of Manufacturing Systems*, (0). doi:10.1016/j.jmsy.2013.03.004. (Cited on page 34)

Rangsaritratsamee, R., Ferrell Jr, W. G., & Kurz, M. B. (2004). Dynamic rescheduling that simultaneously considers efficiency and stability. *Computers & Industrial Engineering*, *46*(1), 1–15. doi:http://dx.doi.org/10.1016/j.cie.2003.09.007. (Cited on page 35)

Ranjbar, M., Davari, M., & Leus, R. (2012). Two branch-and-bound algorithms for the robust parallel machine scheduling problem. *Computers & Operations Research*, *39*(7), 1652–1660. doi:http://dx.doi.org/10.1016/j.cor.2011.09.019. (Cited on page 34)

Riera, D. & Yorke-Smith, N. (2002). An improved hybrid model for the generic hoist scheduling problem. *Annals of Operations Research*, *115*(1-4), 173–191. doi:10.1023/a:1021101321339. (Cited on pages 18, 19, 21 and 22)

Rodosek, R., Wallace, M. G., & Hajian, M. T. (1999). A new approach to integrating mixed integer programming and constraint logic programming. *Annals of Operations Research*, *86*(0), 63–87. doi:10.1023/A:1018904229454. (Cited on pages 18 and 19)

Sevaux, M. & Sörensen, K. (2004). A genetic algorithm for robust schedules in a one-machine environment with ready times and due dates. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, *2*(2), 129–147. doi:10.1007/s10288-003-0028-0. (Cited on page 34)

Shapiro, G. W. & Nuttle, H. L. (1988). Hoist scheduling for a pcb electroplating facility. *IIE Transactions*, *20*(2), 157–167. (Cited on pages 11, 14, 17, 19 and 35).

Smith, S. F. & Cheng, C.-C. (1993). Slack-based heuristics for constraint satisfaction scheduling. In *Proceedings of the eleventh national conference on artificial intelligence* (Pages 139–144). AAAI'93. Washington, D.C.: AAAI Press. (Cited on page 25).

Song, W., Zabinsky, Z. B., & Storch, R. L. (1993). An algorithm for scheduling a chemical processing tank line. *Production Planning & Control*, *4*(4), 323–332. doi:10.1080/09537289308919454. (Cited on pages 23, 25, 27, 28 and 31)

Spacek, P., Manier, M.-A., & Moudni, A. E. (1999). Control of an electroplating line in the max and min algebras. *International Journal of Systems Science*, *30*(7), 759–778. doi:10.1080/002077299292065. (Cited on pages 14, 15 and 17)

Storer, R., Wu, S. D., & Vaccari, R. (1992). Local search in problem and heuristic space for job shop scheduling genetic algorithms. In G. Fandel, T. Gulledge, & A. Jones (Editors), *New directions for operations research in manufacturing* (Chapter 9, Pages 149–160). Springer Berlin Heidelberg. doi:10.1007/978-3-642-77537-6_9. (Cited on page 34)

Sun, T. C., Lai, K. K., Lam, K., & So, K. P. (1994). A study of heuristics for bidirectional multi-hoist production scheduling systems. *International Journal of Production Economics*, *33*(1–3), 207–214. (Cited on pages 10, 24 and 27).

Thesen, A. & Lei, L. (1986). An expert system for scheduling robots in a flexible electroplating system with dynamically changing workloads. In *Proceedings of the second orsa/tims conference on flexible manufacturing systems* (Pages 555–566). Elsevier Science Publishers BV, Amsterdam. (Cited on pages 24 and 27).

Thesen, A. & Lei, L. (1990). An expert scheduling system for material handling hoists. *Journal of Manufacturing Systems*, *9*(3), 247–252. doi:http://dx.doi.org/10.1016/0278-6125(90)90055-M. (Cited on pages 24 and 27)

Tian, N., Che, A., & Feng, J. (2013). Real-time hoist scheduling for multistage material handling process under uncertainties. *AIChE Journal*, *59*(4), 1046–1048. doi:10.1002/aic.14048. (Cited on pages 27, 54, 56, 59, 60, 63 and 67)

Tompkins, J. A., White, J. A., Bozer, Y. A., & Tanchoco, J. M. A. (2010). *Facilities planning*. John Wiley & Sons. (Cited on page 3).

Van de Vonder, S., Demeulemeester, E., Herroelen, W., & Leus, R. (2005). The use of buffers in project management: the trade-off between stability and makespan. *International Journal of Production Economics*, *97*(2), 227–240. doi:http://dx.doi.org/10.1016/j.ijpe.2004.08.004. (Cited on page 34)

Varnier, C., Bachelu, A., & Baptiste, P. (1997). Resolution of the cyclic multi-hoists scheduling problem with overlapping partitions. *INFOR: Information Systems and Operational Research*, *35*(4), 309–324. doi:10.1080/03155986.1997.11732338. (Cited on pages 21 and 22)

Varnier, C. & Jeunehomme, N. (2000). A cyclic approach for the multi-product hoist scheduling problem. In *7th international workshop on project management and scheduling-pms (osnabrueck, germany, 2000), euro, ed.* (Cited on pages 16 and 17).

Wu, S. D., Storer, R. H., & Chang, P.-C. (1993). One-machine rescheduling heuristics with efficiency and stability as criteria. *Computers & Operations Research*, *20*(1), 1–14. (Cited on page 35).

Xiong, J., Xing, L., & Chen, Y. (2013). Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns. *International Journal of Production Economics*, *141*(1), 112–126. doi:http://dx.doi.org/10.1016/j.ijpe.2012.04.015. (Cited on page 34)

Yan, P., Che, A., Cai, X., & Tang, X. (2014). Two-phase branch and bound algorithm for robotic cells rescheduling considering limited disturbance. *Computers & Operations Research*, *50*, 128–140. doi:http://dx.doi.org/10.1016/j.cor.2014.04.002. (Cited on pages 10, 25 and 27)

Yan, P., Che, A., Yang, N., & Chu, C. (2012). A tabu search algorithm with solution space partition and repairing procedure for cyclic robotic cell scheduling problem. *International Journal of Production Research*, *50*(22), 6403–6418. doi:10.1080/00207543.2011.645953. (Cited on pages 8, 14, 15, 17 and 46)

Yan, P., Chu, C., Yang, N., & Che, A. (2010). A branch and bound algorithm for optimal cyclic scheduling in a robotic cell with processing time windows. *International Journal of Production Research*, *48*(21), 6461–6480. doi:10.1080/00207540903225205. (Cited on pages 14, 15 and 17)

Yang, G. W., Ju, D. P., Zheng, W. M., & Lam, K. (2001). Solving multiple hoist scheduling problems by use of simulated annealing. *Ruan Jian Xue Bao/Journal of Software*, *12*(1), 11–17. (Cited on pages 20 and 22).

Yih, Y. (1994). An algorithm for hoist scheduling problems. *International Journal of Production Research*, *32*(3), 501–516. (Cited on pages 10, 25, 27 and 54).

Yih, Y. & Yin, N.-C. (1992). Crane scheduling in a flexible electroplating line: a tolerance-based approach. *Journal of Electronics Manufacturing*, *02*(04), 137–144. doi:doi:10.1142/S0960313192000169. (Cited on pages 25, 27 and 54)

Zhang, Q., Manier, H., & Manier, M.-A. (2012). A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. *Computers & Operations Research*, *39*(7), 1713–1723. doi:10.1016/j.cor.2011.10.007. (Cited on pages 23 and 27)

Zhang, Q., Manier, H., & Manier, M.-A. (2014). A modified shifting bottleneck heuristic and disjunctive graph for job shop scheduling problems with transportation constraints. *International Journal of Production Research*, *52*(4), 985–1002. doi:10.1080/00207543.2013.828164. (Cited on pages 23 and 27)

Zhang, W. & Reimann, M. (2014). A simple augmented ϵ-constraint method for multi-objective mathematical integer programming problems. *European Journal of Operational Research*, *234*(1), 15–24. (Cited on page 46).

Zhao, C., Fu, J., & Xu, Q. (2013a). Production-ratio oriented optimization for multi-recipe material handling via simultaneous hoist scheduling and production line arrangement. *Computers & Chemical Engineering*, *50*(0), 28–38. doi:http://dx.doi.org/10.1016/j.compchemeng.2012.10.016. (Cited on pages 10, 19, 72 and 75)

Zhao, C., Fu, J., & Xu, Q. (2013b). Real-time dynamic hoist scheduling for multistage material handling process under uncertainties. *AIChE Journal*, *59*(2), 465–482. doi:10.1002/aic.13852. (Cited on pages 10, 27, 54, 56, 57, 58, 59, 60, 61, 64 and 67)

Zhou, Z., Che, A., & Yan, P. (2012). A mixed integer programming approach for multi-cyclic robotic flowshop scheduling with time window constraints. *Applied Mathematical Modelling*, *36*(8), 3621–3629. (Cited on pages 11, 15, 16, 17, 18, 60, 75 and 76).

Zhou, Z. & Li, H. (2002). A heuristic method for one hoist dynamic scheduling. *Systems Enging-theory Methodology Application*, (2), 136–140. (Cited on pages 25 and 27).

Zhou, Z. & Li, L. (2003). Single hoist cyclic scheduling with multiple tanks: a material handling solution. *Computers & Operations Research*, *30*(6), 811–819. doi:Doi10.1016/S0305-0548(02)00043-6. (Cited on pages 18 and 19)

Zhou, Z. & Li, L. (2009). A solution for cyclic scheduling of multi-hoists without overlapping. *Annals of Operations Research*, *168*(1), 5–21. doi:10.1007/s10479-008-0372-8. (Cited on pages 20 and 22)

Zhou, Z. & Liu, J. (2008). A heuristic algorithm for the two-hoist cyclic scheduling problem with overlapping hoist coverage ranges. *IIE Transactions*, *40*(8), 782–794. doi:10.1080/07408170701748729. (Cited on pages 20, 21 and 22)

université PARIS-SACLAY
ÉCOLE DOCTORALE
INTERFACES
Approches Interdisciplinaires : Fondements, Applications et Innovation

Titre : Modélisation et optimisation des Hoist Scheduling Problems
Mots clés : hoist scheduling, optimisation, programmation linéaire mixte en nombres entiers

Résumé : Dans cette thèse, nous étudions des Hoist Scheduling Problems (HSP) qui se posent fréquemment dans des lignes automatiques de traitement de surface. Dans ces lignes, des ponts roulants sont utilisés pour transporter les pièces entre les bains. Ainsi, les ponts roulants jouent un rôle essentiel dans la performance de ces lignes ; et un ordonnancement optimal de leurs mouvements est un facteur déterminant pour garantir la qualité des produits et maximiser la productivité. Les lignes que nous étudions comportent un seul pont roulant mais peuvent être des lignes de base ou des lignes étendues (où des bains sont à fonctions et/ou capacités multiples). Nous examinons trois Hoist Scheduling Problems : l'optimisation robuste d'un HSP cyclique, l'ordonnancement dynamique d'une ligne étendue de type job shop et l'ordonnancement cyclique d'une telle ligne.
Pour l'optimisation robuste d'un HSP cyclique, nous définissons la robustesse comme la marge dans le temps de déplacement du pont roulant. Nous formulons le problème en programmation linéaire en nombres mixtes à deux objectifs pour optimiser simultanément le temps de cycle et la robustesse. Nous démontrons que le temps de cycle minimal augmente avec la robustesse, et que par conséquent la frontière Pareto est constituée d'une infinité de solutions. Les valeurs minimales et maximales des deux objectifs sont établies. Les résultats expérimentaux à partir de benchmarks et d'instances générées aléatoirement montrent l'efficacité de l'approche proposée.
Nous étudions ensuite un problème d'ordonnancement dynamique dans une ligne étendue de type job shop. Nous mettons en évidence une erreur de formulation dans une un modèle existant pour un problème similaire mais sans bains multi-fonctions. Cette erreur peut rendre l'ordonnancement obtenu sous-optimal voire irréalisable. Nous construisons un nouveau modèle qui corrige cette erreur. De plus il est plus compact et s'applique au cas avec des bains à la fois à capacités et à fonctions multiples. Les résultats expérimentaux menés sur des instances avec ou sans bains multi-fonctions montrent que le modèle proposé conduit toujours à une solution optimale et plus efficace que le modèle existant.
Nous nous focalisons enfin sur l'ordonnancement cyclique d'une ligne étendue de type job shop avec des bains à fonctions et capacités multiples. Nous construisons un modèle mathématique en formulant les contraintes de capacité du pont roulant, les intervalles des durées opératoires, et les contraintes de capacité des bains. Nous établissons également des contraintes valides. Les expériences réalisées sur des instances générées aléatoirement montrent l'efficacité du modèle proposé.

Title: Modeling and Optimization for Hoist Scheduling Problems
Keywords: hoist scheduling, optimization, mixed-integer linear programming

Abstract: This thesis studies hoist scheduling problems (HSPs) arising in automated electroplating lines. In such lines, hoists are often used for material handing between tanks. These hoists play a crucial role in the performance of the lines and an optimal schedule of the hoist operations is a key factor in guaranteeing product quality and maximizing productivity. We focus on extended lines (i.e. with multi-function and/or multi-capacity tanks) with a single hoist. This research investigates three hoist scheduling problems: robust optimization for cyclic HSP, dynamic jobshop HSP in extended lines and cyclic jobshop HSP in extended lines.
We first study the robust optimization for a cyclic HSP. The robustness of a cyclic hoist schedule is defined in terms of the free slacks in hoist traveling times. A bi-objective mixed-integer linear programming (MILP) model is developed to optimize the cycle time and the robustness simultaneously. It is proved that the optimal cycle time strictly increases with the robustness, thus there is an infinite number of Pareto optimal solutions. We established lower and upper bounds of these two objectives. Computational results on several benchmark instances and randomly generated instances indicate that the proposed approach can effectively solve the problem.
We then examine a dynamic jobshop HSP with multi-function and multi-capacity tanks. We demonstrate that an existing model for a similar problem can lead to sub-optimality. To deal with this issue, a new MILP model is developed to generate an optimal reschedule. It can handle the case where a multi-function tank is also multi-capacity. Computational results on instances with and without multi-function tanks indicate that the proposed model always yields optimal solutions, and is more compact and effective than the existing one.
Finally, we investigate a cyclic jobshop HSP with multi-function and multi-capacity tanks. An MILP model is developed for the problem. The key issue is to formulate the time-window constraints and the tank capacity constraints. We adapt the formulation of time-window constraints for a simpler cyclic HSP to the jobshop case. The tank capacity constraints are handled by dealing with the relationships between hoist moves so that there is always an empty processing slot for new parts. Computational experiments on numerical examples and randomly generated instances indicate that the proposed model can effectively solve the problem.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France