Cyclic Hoist Scheduling Problems in Classical and Sustainabl

Weidong Lei

SPIM

Thèse de Doctorat

école doctorale sciences pour l'ingénieur et microtechniques
UNIVERSITÉ DE TECHNOLOGIE BELFORT-MONTBÉLIARD

N° d'ordre :244

Cyclic Hoist Scheduling Problems in Classical and Sustainable Contexts

Ordonnancement cyclique des ressources de transport dans les ateliers de traitement de surface, dans des contextes traditionnel et durable

■ Weidong LEI

SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UNIVERSITÉ DE TECHNOLOGIE BELFORT-MONTBÉLIARD

Order number : 244

PhD THESIS

to obtain the degree

Doctor of Université de Technologie de Belfort-Montbéliard

Speciality : Automatic

Cyclic Hoist Scheduling Problems in Classical and Sustainable Contexts

by **Weidong LEI**

Laboratory OPERA-Université de Technologie de Belfort-Montbéliard (France)

School of Management - Northwestern Polytechnical University (China)

Defended on december 8, 2014

Jury :

M. Aziz MOUKRIM	Professor-UTC-France	Reviewer
M. Feng WU	Professor-XJTU-China	Reviewer
Mme Rong DU	Professor-XDU-China	President
Mme Marie-Ange MANIER	Associate Professor HDR-UTBM-France	Director
M. Ada CHE	Professeur-NPU-Chine	Director
M. Hervé MANIER	Associate Professor -UTBM-France	Co-supervisor

SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UNIVERSITÉ DE TECHNOLOGIE BELFORT-MONTBÉLIARD

N° d'ordre : 244

THÈSE

Pour l'obtention du grade de

Docteur de l'Université de Technologie de Belfort-Montbéliard

Spécialité : Automatique

Ordonnancement cyclique des ressources de transport dans les ateliers de traitement de surface, dans des contextes traditionnel et durable

Présentée par **Weidong LEI**

Laboratoire OPERA – Université de Technologie de Belfort-Montbéliard (France)

School of Management - Northwestern Polytechnical University (Chine)

Soutenue le 8 Décembre 2014 devant le jury composé de :

M. Aziz MOUKRIM	Professeur-UTC-France	Rapporteur
M. Feng WU	Professeur-XJTU-Chine	Rapporteur
Mme Rong DU	Professeur-XDU-Chine	Présidente
Mme Marie-Ange MANIER	Maître de Conférences HDR- UTBM-France	Directeur de thèse
M. Ada CHE	Professeur-NPU-Chine	Directeur de thèse
M. Hervé MANIER	Maître de Conférences-UTBM-France	Co-encadrant

ACKNOWLEDGEMENTS

I would like to thank those who supported and accompanied me with my heartfelt gratitude during the past three years.

This thesis was supported by a co-tutelage program between Université de Technologie de Belfort-Montbéliard (UTBM, France) and Northwestern Polytechnical University (NPU, China). I would like to thank the Laboratoire Optimisation Et ReséAux of UTBM and the School of Management of NPU, for providing me with good working conditions. I acknowledge the China Scholarship Council (CSC) for providing me with a two-year study funds at UTBM.

I would like to express my sincere gratitude to the jury members of my PhD committee, to Professor Rong DU for serving as the chairman; to Professors Aziz MOUKRIM and Feng WU for having accepted to review this thesis and their helpful advices and suggestions; to my supervisors for revising this thesis carefully and correcting the errors word by word.

I will forever be thankful to my two advisors, Associate Professor Marie-Ange MANIER and Professor Ada CHE, for teaching me the research skills and ethics, and for helping me in many aspects during my doctoral studies. I have been most indebted to their professional guidance, great patience and massive help. I am also very grateful to my co-supervisor Dr. Hervé MANIER for his many good discussions and suggestions. Without them, I could not have achieved anything.

I also own my sincere gratitude to my friends, and my labmates at UTBM and NPU who offered their kind helps and supports to me. They added a lot of colors to my life over the past years. Special thanks go to Qiao ZHANG and Frédéric LASSABE for their many helps when I lived in Belfort.

At last, I want to express my sincere thanks to my family for their endless love and selfless support.

ABSTRACT

Automated surface treatment facilities, which employ computer-controlled hoists for part transportation, have been extensively established in various kinds of industrial companies, because of its numerous advantages over manual system, such as higher productivity, better product quality, and reduced labor intensity. This research investigates three typical hoist scheduling problems with processing time windows in surface treatment facilities, which are (I) cyclic single-hoist scheduling problem to minimize the cycle time; (II) cyclic single-hoist scheduling problem to minimize the cycle time and processing resource consumption (and consequently production cost); and (III) cyclic multi-hoist scheduling problem to minimize the cycle time.

Due to the NP-completeness of the studied problems and numerous advantages of quantum-inspired evolutionary algorithm (QEA), we first propose a hybrid QEA with improved decoding mechanism and repairing procedure to find the best cycle time for the first problem. After that, to enhance with both the economic and environmental performance, which constitute two of the three pillars of the sustainable strategy nowadays deployed in many industries, we formulate a bi-objective mathematical model for the second problem by using the method of prohibited interval (MPI). Then we propose a bi-objective QEA with local search procedure to simultaneously minimize the cycle time and the production cost, and we find a set of Pareto-optimal solutions for this problem. As for the third problem, we find that most existing approaches, such as mixed integer programming (MIP) approach, may identify a non-optimal solution to be an optimal one due to an assumption related to the loaded hoist moves which is made in many existing researches. Consequently, we propose an improved MIP approach for this problem by relaxing the above-mentioned assumption. Our approach can guarantee the optimality of its obtained solutions.

For each problem, experimental study on industrial instances and random instances has been conducted. Computational results demonstrate that the proposed scheduling algorithms are effective and justify the choices we made.

Keywords: cyclic hoist scheduling problem; processing time windows; bi-objective optimization; quantum-inspired evolutionary algorithm; mixed integer programming approach

RÉSUMÉ

Les ateliers de traitement de surface automatisés, qui utilisent des robots de manutention commandés par ordinateur pour le transport de la pièce, ont été largement mis en place dans différents types d'entreprises industrielles, en raison de ses nombreux avantages par rapport à un mode de production manuel, tels que: une plus grande productivité, une meilleure qualité des produits, et l'impact sur les rythmes de travail. Notre recherche porte sur trois types de problèmes d'ordonnancement associés à ces systèmes, appelés hoist scheduling problems, caractérisés par des contraintes de fenêtres de temps de traitement: (I) un problème à une seule ressource de transport où l'objectif est de minimiser le temps de cycle; (II) un problème bi-objectif avec une seule ressource de transport où il faut minimiser le temps de cycle et la consommation de ressources de traitement (et par conséquent le coût de production); et (III) un problème d'ordonnancement cyclique mono-objectif mais multi-robots.

En raison de la NP-complétude des problèmes étudiés et de nombreux avantages de les outils de type quantum-inspired evolutionary algorithm (QEA), nous proposons d'abord un QEA hybride comprenant un mécanisme de décodage amélioré et une procédure réparation dédiée pour trouver le meilleur temps de cycle pour le premier problème. Après cela, afin d'améliorer à la fois la performance économique et environnementale qui constituent deux des trois piliers de la stratégie de développement durable de nos jours déployée dans de nombreuses industries, nous formulons un modèle mathématique bi-objectif pour le deuxième problem en utilisant la méthode de l'intervalle interdit. Ensuite, nous proposons un QEA bi-objectif couplé avec une procédure de recherche locale pour minimiser simultanément le temps de cycle et les coûts de production, en générant un ensemble de solutions Pareto-optimales pour ce problème. Quant au troisième problème, nous constatons que la plupart des approaches utilisées dans les recherches actuelles, telles que la programmation entière mixte (MIP), peuvent conduire à l'obtention d'une solution non optimale en raison de la prise en compte courante d'une hypothèse limitant l'exploration de l'espace de recherche et relative aux mouvements en charge des robots. Par conséquent, nous proposons une approche de MIP améliorée qui peut garantir l'optimalité des solutions obtenues pour ce problème, en relaxant l'hypothèse mentionnée ci-dessus.

Pour chaque problème, une étude expérimentale a été menée sur des cas

industriels ainsi que sur des instances générées aléatoirement. Les résultats obtenus montrent que l’efficacité des algorithmes d'ordonnancement proposés, ce qui justifie les choix que nous avons faits.

Mots-clés: ordonnancement cyclique des ateliers de traitement de surface, fenêtres de temps de traitement; optimisation bi-objectif; algorithme évolutionnaire quantique; approche de programmation mixte en nombres entiers.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

1.1 Research Background

In today's fiercely competitive market, to maximize the production capacity and reduce the labor costs, automated production lines have been widely used in many industries, such as the automotive industry, the aerospace industry and more particularly the surface treatment industry. Meanwhile, with the ongoing development in automation technologies and scheduling theories, automated production lines become more and more reliable and efficient.

In modern surface treatment facilities, production lines are often equipped with computer-controlled material handling tools (usually called hoists or robots in different industries) for moving jobs or parts between tanks or machines (Crama *et al*., 2000; Manier and Bloch, 2003). That is to say, all the transportation tasks during the process are performed by hoists instead of workers. Obviously, highly automated production system gains several unique advantages over manual production system. Firstly, both the productivity and product quality are effectively improved since hoists generally have less variability compared to human beings (suppose that hoists never break down). In other words, hoists are not only easy to control and implement but also very stable (i.e., hoists can exactly and timely perform each transportation task assigned to it). Secondly, hoists can replace workers in high-temperature or hazardous environments (or workplaces), since worker safety is one of most important issues that each factory cares about. The last but not the least advantage is that the process line generally has plenty of high-frequency and repetitive transportation jobs, which are generally very boring for workers but relatively suitable for hoists.

Because of its wide applications, electroplating plant has been extensively established in many surface treatment companies, which produce tens of thousands of products each year. According to Schlesinger and Paunovic (2010), electroplating is the coating of an electrically conductive object with a layer of metal using electrical current resulting in a thin, smooth of metal on the object. A representative example is the Printed Circuit Boards (PCBs) electroplating plant. More precisely, a PCB electroplating process line typically consists of a sequence of tanks (containing various kinds of chemical solutions or freshwater) arranged in a row and a number of computer-controlled material handling hoists mounted on a single track above the tanks, as shown in Figure 1.1. Each tank contains special chemicals for a specific

production step, such as depositing, degreasing, and pickling. Besides, multiple hoists are generally used to move PCBs from tank to tank due to its higher productivity. Once a PCB is introduced into the line from the input station, it must be continuously processed in each of the tanks one after another until it is transported to the output station.

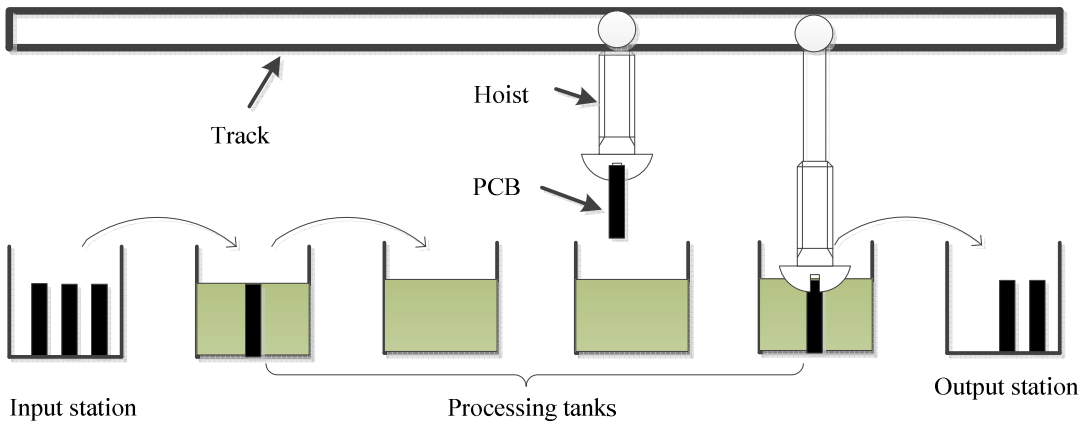

Figure 1.1 A typical automated PCB electroplating line with two hoists.

For automated electroplating process lines, scheduling of hoists' transportation tasks efficiently is very critical because the productivity and the product quality extremely depend on it. Therefore, the decision generally concerns how to sequence the hoists' movements without collision happened among hoists and determine the start time of each hoist move such that the productivity is maximized. It is well known in the literature as Hoist Scheduling Problem (HSP, Manier and Bloch, 2003). It also has some other appellations called in different industrials, such as Robotic Cells (Levner *et al*., 2007) or Robotic flow-shop Scheduling Problems (Crama *et al*., 2000), etc. Similar to the classic flow shop or job shop scheduling problems, Livshits *et al.* (1974) and Lei and Wang (1989) respectively proved that the simple HSP (i.e., cyclic HSP with a single part-type and a single hoist) is *NP*-complete. Note that *NP* means non-deterministic polynomial.

Moreover, in today's fast-changing and competitive market, one most important goal for electroplating plant is to maximize its productivity, so as to timely provide required products to customers. This is very important for company to get good reputation from partners. On the other hand, resource consumption greatly affects the production cost. As the costs of resources increase, the product profit is generally

reduced. The traditional way that only maximizes the productivity cannot effectively respond to the rising production costs. Therefore, minimizing the production cost plays a key role in enhancing the company's competitive ability and profits. It also joins the sustainable development strategies of many industrials because this effort to reduce resource responds to both economic and environmental concerns. At last but not least, the defective part rate must be minimized during the production, which has a negative impact on the company's profits.

Until now, a number of scheduling approaches have been suggested for various HSP to maximize the productivity, for example, please see the works by Phillips and Unger (1976), Shapiro and Nuttle (1988), Lei and Wang (1994), Chen *et al*. (1998), Manier *et al*. (2000), Che and Chu (2007), and Lei *et al*. (2014). But study on multi-objective HSP has not received much attention from researchers, except for a few works, such as Xu and Huang (2004), Kuntay *et al.* (2006), and Feng *et al*. (2014). As a result, research for HSP with simultaneously achieving various goals from different expectations becomes urgent due to its great significance in theory and application. This research will focus on this area.

1.2 Problem Description

During the manufacture of many products, including electronic ones, electroplating is an essential process for making some special treatments on part surface, such as anti-corrosive, abrasion resistance, and improved electrical conductivity. In a typical automated electroplating process line (Figure 1.1), a series of tanks which contain different chemical solutions or freshwater are arranged in a row. The input device and the output device are located at the both ends of the line. Each tank corresponds to a specific process stage, such as degreasing, silver or copper coating, drying, cleaning and rinsing. Since hoist is often the bottleneck resource in the process line, multiple hoists are widely used to balance the line. During the process, parts are transported by a hoist from one tank to the other. For a hoist travel among tanks without carrying a part, it is called an empty move. On contrast, it is a loaded move. All hoists often move on a shared track, so hoist collisions must be avoided. This is called collision avoidance constraint. Due to the processing limitation, each tank can process only one part at any time. So if a tank is occupied by a part, then it must be emptied before processing another part. This is called tank capacity constraint. Similarly, each hoist can only transport one part at any time, and must have

enough time to move empty between any two consecutive loaded moves, which are called hoist capacity constraint.

Once a part is introduced into the process line, it is soaked in tanks to receive its processing operations according to its processing routine until it is removed from the line. According to the processing technology, the soak or processing time in each tank must be within a time window [minimum dwell time, maximum dwell time], called time window constraint (Lei and Wang, 1991). By the way, in this thesis, when we mention HSP, it refers to HSP with processing time windows. If each processing time falls into its time window, then part quality would be guaranteed; otherwise, defective parts would be produced. Besides, no buffer exists among tanks. In other words, once a part finishes its processing operation in a tank, it must be moved out of the current tank and then transported to the next one by a hoist. From this, we can know that each part is either in a tank or being transported by a hoist without any pause allowed.

From above descriptions, we can know that a hoist schedule is said to be feasible for HSP only if it simultaneously satisfies the previously mentioned four families of constraints, i.e., (1) *collision avoidance constraint*, if multiple hoists are used; (2) *tank capacity constraint*; (3) *hoist capacity constraint*; (4) *time window constraint*.

Because of its easy implementation in a mass production environment, cyclic production mode is usually adopted in the electroplating line. This leads to a repetitive schedule performed by hoists in every certain time. The duration of performing the repetitive schedule is called the cycle time (Chen *et al*., 1998). In each cycle, one part is introduced into the line, and one part (note that the two parts are not necessary the same one) is removed from the line after all its processing operations are finished. Obviously, line productivity heavily depends on how to schedule the hoists' transportation tasks, since the more frequently the hoist picks a part from the input station, the higher the line productivity. As a result, in most studies, the objective of HSP is to minimize the cycle time. On the other hand, due to the high treatment costs of hazardous wastes (such as chemical sludge and wastewater) in electroplating plant, the more resource used for processing parts, the higher the operating costs. Therefore, how to optimize the actual processing time in each tank while satisfying the time window constraint is crucial in reducing the production cost.

Since the 1970s, many researchers have dedicated to solve various variants of HSP motivated by automated electroplating process lines. Most studies are relevant with minimizing the cycle time for HSP, e.g. Phillips and Unger (1976), Shapiro and

Nuttle (1988), Lei and Wang (1994), Ng (1996), Chen *et al*. (1998) and Che and Chu (2007). Due to its great significance in theory and practice, several works about multi-hoist scheduling have been published especially in recently years, such as Zhou and Liu (2008), Zhou and Li (2009), Chtourou *et al*. (2013), Jiang and Liu (2014), and Li and Fung (2014). As far as the single-objective HSP is concerned, it is far from meeting the various expectations from the real-world production. To reduce the complexity of multi-objective HSP, a few studies (such as Xu and Huang, 2004, Kuntay *et al.*, 2006, and Subaï *et al.*, 2006) have been conducted on the HSP with dual objectives, which are optimized in a sequential manner, i.e., one objective is considered in the first step, and the other is considered in the second step. Obviously, such separate and sequential optimization approaches are not sufficient in practice. Therefore, simultaneously optimizing different and sometimes conflicting objectives from different aspects for HSP is very necessary and important.

To address the considered problems, we have chosen to use a rather new tool called Quantum-inspired Evolutionary Algorithm (QEA). Since 1990s, QEA has been received much attention and successfully applied to solve travelling salesman problem (Narayanan and Moore, 1996), knapsack problem (Han and Kim, 2002), flow shop/job shop scheduling problems (Li and Wang, 2007; Gu *et al*., 2009), etc. In the following section, we briefly describe its main principles.

1.3 Quantum-inspired evolutionary algorithm

Quantum-inspired Evolutionary Algorithm (QEA) is formed according to the concepts and principles of quantum computation (Deutsch, 1985; Hey, 1999), in which Q-bit is the smallest unit of information in a quantum computer. Each Q-bit may be in "0" state, "1" state, or in any superposition of the two. The following equation is usually used to define a Q-bit (Han and Kim, 2002; Li and Wang, 2007):

$$|\psi\rangle=\alpha|0\rangle+\beta|1\rangle, \text{ where } |\alpha|^2+|\beta|^2=1. \tag{1.1}$$

In (1.1), α and β are two complex numbers, which represent the probability amplitudes of states 0 and 1, respectively. As a result, $|\alpha|^2$ and $|\beta|^2$ represent the probabilities that the Q-bit would be found in state "0" and state "1", respectively. However, each Q-bit collapses to a single state by using a random-key observation way. That is, a random number r is generated from the uniform distribution [0, 1). If $r>|\alpha|^2$, then Q-bit is in state "1"; else, Q-bit is in state "0". So QEA can be seen as a

probabilistic algorithm. Moreover, Q-gate is often employed to change the values of α and β so as to influence the state of Q-bit. Until now, several Q-gates have been proposed in the literature, such as NOT gate, controlled NOT gate, and rotation gate (Hey, 1999).

$$\Psi_m = \begin{bmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_m \\ \beta_1 & \beta_2 & \dots & \beta_m \end{bmatrix}, \text{ where } |\alpha_i|^2+|\beta_i|^2=1, \ 1 \le i \le m. \tag{1.2}$$

Suppose that a quantum individual Ψ_m is a string of m Q-bits, as shown in (1.2), this individual can represent 2^m states at the same time, i.e., a linear superposition of states. For instance, consider a quantum individual with three Q-bits and their amplitudes as the following:

$$\Psi_3 = \begin{bmatrix} \frac{\sqrt{4}}{3} & \frac{\sqrt{3}}{3} & \frac{\sqrt{7}}{3} \\ \frac{-\sqrt{5}}{3} & \frac{-\sqrt{6}}{3} & \frac{\sqrt{2}}{3} \end{bmatrix}, \tag{1.3}$$

In (1.3), Ψ_3 includes the information of eight states, i.e., $|000\rangle$, $|001\rangle$, $|010\rangle$, $|011\rangle$, $|100\rangle$, $|101\rangle$, $|110\rangle$, $|111\rangle$, and their probabilities are respectively 84/729, 24/729, 168/729, 48/729, 105/729, 30/729, 210/729, 60/729. Indeed if we consider the state $|010\rangle$ as an example, the associated probability is $|\alpha_1|^2 \times |\beta_2|^2 \times |\alpha_3|^2$ which equals (4/9)×(6/9)×(7/9)=168/729. From this example, we can know that Q-bit representation has a better characteristic of population diversity than other representations, since it potentially maps to a larger phenotype space than other binary representation based Evolutionary algorithms (EAs).

Like other EAs (such as genetic algorithm and annealing evolution algorithm), QEA generally has a similar evolution paradigm. It begins with an initial population, in which each individual is encoded by Q-bits. After evaluating the population fitness, it applies Q-gate to update individuals for generating new offspring and guiding the individual towards better solutions, and then evaluates the new population. When the stop condition is satisfied, it ends and outputs the best solution. Figure 1.2 illustrates this process in details, where $Q(t)$, $P(t)$ and $B(t)$ are quantum chromosome, problem solution and best solution respectively.

```
Begin: t←0
    (1) Initialize Q(t), i.e. Quantum individual at generation t;
    (2) Make P(t) by observing the states of Q(t);
    (3) Evaluate P(t);
    (4) Store the best solution among P(t) into B(t);
        While (not termination condition) do
            t←t+1;
            Make P(t) by observing the states of Q(t–1);
            Evaluate P(t);
            Store the best solution among P(t) and B(t–1) into B(t);
            Apply Q-gates to update Q(t);
        End
End: Output the best solution
```

Figure 1.2 Pseudocode algorithm for QEA (Han and Kim, 2002).

1.4 Contributions

In this thesis, we investigate three types of HSP motivated by automated electroplating process lines. They are respectively: (I) cyclic single-hoist scheduling problem to minimize the cycle time, (II) cyclic single-hoist scheduling problem to minimize the cycle time and the production cost, and (III) cyclic multi-hoist scheduling problem to minimize the cycle time.

Due to the NP-completeness of HSP, the computation time spent by exact methods usually increases exponentially with its size. Thus, it is a wise choice to adopt meta-heuristic methods to find reasonably good schedules in a reasonable time for HSP. Because of its unique advantages, such as better population diversity and rapid convergence, QEA has gained great success in solving many different optimization problems, but it was not used yet for solving HSP. Therefore, this research tries to connect this gap. The main contributions of this research are summarized as follows.

Firstly, we propose a hybrid QEA with improved decoding scheme for the first problem. More precisely, we elaborate three different decoding procedures to convert Q-bit individual into hoist move sequences. Moreover, we develop a more effective repairing procedure than the existing one. Both quantum rotation-gate and adaptive genetic operators as variant operators are applied to evolve the population towards better solutions.

Secondly, we propose an efficient QEA algorithm with local search procedure for

the second problem. More precisely, based on a full analysis of the studied problem, a bi-objective mathematical model is formulated by using the method of prohibited intervals (MPI). After that, we use a double-decoding procedure to convert Q-bit individuals into problem solutions. All solutions are evaluated by the famous Pareto-dominance technique. A chaotic quantum-rotation gate is designed for updating Q-bit individuals. To increase the individual diversity, mutation operator is implanted into the proposed algorithm. Moreover, external archive is used to store the obtained non-dominated solutions. Local search procedure is applied for further improving the solution quality.

Finally, we propose an improved mixed integer programming (MIP) approach for the last problem. In most existing studies, such as Lei and Wang (1991), Armstrong *et al*. (1996), Leung and Zhang (2003), Leung *et al*. (2004), Che and Chu (2004), Zhou and Liu (2008), Zhou and Li (2009), Chtourou *et al*. (2013) and Jiang and Liu (2014), all loaded moves are implicitly or explicitly assumed to start and end within the same cycle. In this research, we give a counterexample to demonstrate that this assumption should be relaxed, since approaches based on it may identify a non-optimal solution to be an optimal one. In other words, loaded hoist moves are allowed to start in the current cycle and end in the next one if necessary in our research. Consequently, we propose an improved MIP approach for the third problem by relaxing the above-mentioned assumption. Our approach can guarantee the optimality of its obtained solutions.

1.5 Thesis Outline

This thesis is arranged as follows.

Chapter 2 provides a literature review of HSP and quantum-inspired evolutionary algorithm (QEA) most related to this research. The research trends on HSP and the research gap between HSP and QEA are also pointed out.

Chapter 3 mainly develops an effective QEA for solving the cyclic single-hoist scheduling problem with time window constraints in automated electroplating lines. The objective is to minimize the cycle time. The problem formulation and the proposed QEA are presented. Comparison experiments are conducted between the proposed algorithm and the existing approaches.

Chapter 4 first formulates a bi-objective mathematical model by MPI approach

for the studied problem, and then develops a multi-objective QEA with local search procedure to find a set of Pareto-optimal solutions for the problem. The objective of the problem is to minimize both the cycle time and the production cost. At last, a real electroplating instance is used to test the effectiveness of the proposed algorithm.

Chapter 5 focuses on the development of an improved MIP model for the cyclic multiple hoists scheduling problem. In contrast with most previous approaches, our MIP approach can always find a global optimal hoist schedule with the maximum productivity. Experimental study is conducted on both benchmark instances and randomly generated instances.

Chapter 6 makes some concluding remarks of this research, and suggests some directions for future research.

Chapter 2 Literature Review

In this chapter, we perform a literature review related to this research. As mentioned in Chapter 1, part of our research focuses on the development of effective QEAs for solving two kinds of HSP. Therefore, we first review relevant works on the HSP, and then give a literature review on QEA related to our research.

The whole literature is rich of works related to hoist scheduling problems or near problems. Manier and Bloch (2003) proposed a notation and classification allowing to identify the various kinds of HSPs. The following paragraph is directly extracted from (Manier and Lamrous, 2008), and it sums up this notation:

"This one considers some of the main physical and logical parameters found in the literature related to the HSP. The complete notation is expressed in the form:

XHSP|nl, ntransfer, synchro, (*mh, mt, ct*) *i*=1 *to nl/nc, circ, ret, empty/ load-unload | nparts/nps, nop, clean, recrc | criteria.*

It is worth noting that the use of default values makes the expression of this notation not so complex when it was applied to most of the instances studied in literature.

The notation can be divided in four fields:

kind of HSP | physical parameters | logical parameters | criteria.

Each one consists in several parameters:

– Kind of HSP (*XHSP*): a hoist scheduling problem can be static (cyclic (CHSP) or not (PHSP)), or dynamic (dynamic problems (DHSP), or reactive ones (RHSP) for real time cases);

– Physical parameters: this field respectively includes the number of basic lines (*nl*), the number of transfer systems connecting these lines (*ntransfer*), the need of synchronization between hoists and transfer systems (*synchro*). It also provides, for each basic line *i* of the facility (*i*=1 to *nl*), the number of hoists (*mh*), tanks (*mt*) and available carriers (*nc*), the maximal capacity of tanks (*ct*), the constraints involved by the characteristics of carriers (circulation of products (*circ*), dedicated transport system to ensure the return of empty carriers from the unloading station to the loading one (*ret*), empty carriers remaining on the line if there is no storage place near the

facility (*empty*)), and finally the configuration of the loading and unloading stations: associated or dissociated stations (*load–unload*);

– Logical parameters: they describe the production environment to be considered: the total number of parts to be treated (*nparts*), the number of processing sequences (*nps*), the maximal number of operations among those processing sequences (*nop*), the possible cleaning of empty carriers after the unloading operation (*clean*) (one or several operations included in *nop*), and finally the recirculation constraint (*recrc*) for reentrant problems;

– Criteria: this field expresses one or several objectives to reach. For HSP, they may be several criteria to optimize, for example: minimize the cycle time for the cyclic HSP (*Cmin*), or minimize the makespan (*Cmax*) in dynamic cases."

Among the various kinds of HSPs studied in the literature and possible to identify via this notation, we have chosen to focus on three of them. Then, this chapter is arranged as follows. Section 2.1 divides the HSP into three parts: (2.1.1) Basic HSP; (2.1.2) multiple objectives HSP; (2.1.3) HSP with multiple hoists, which respectively correspond to the contribution points of our research. Section 2.2 gives a briefly literature review on the QEA. Finally, Section 2.3 summaries this chapter.

2.1 Literature review on HSP

2.1.1 Basic hoist scheduling problem (BHSP)

Over the past decades, HSP has gained great attentions from many researchers due to its significance in many real-world applications. As a result, there is a vast literature about it. Most of the works considered the basic (i.e., a single hoist and a single part type) HSP, called BHSP. The objective of BHSP is usually to minimize the cycle time or the makespan. Before 1970, hoist schedules were usually developed by experienced schedulers. The first work on computerized scheduling approach was provided by Phillips and Unger (1976). They formulated the first Mixed Integer Programming (MIP) model to find the optimal hoist schedule for BHSP. In the experimental study, a real life numerical example was used to testify the effectiveness of the proposed MIP model. The example was chosen from Western Electric Plant and became a well-known benchmark (P&U) instance in the later research.

Almost ten years later, Shapiro and Nuttle (1988) proposed a branch-and-bound

(B&B) procedure to find the optimal cycle time for BHSP. The proposed approach was verified by four practical instances, i.e., P&U instance, Black Oxide1 instance, Black Oxide2 instance and Zinc instance. Computational results on those instances demonstrated that the proposed approach had a better performance than experienced schedulers in terms of solution quality and CPU time.

Moreover, Armstrong *et al.* (1994) also proposed a B&B search procedure based on calculating a sequence-dependent parameter (called minimal time span) for the basic hoist scheduling problem. The performance of the proposed B&B algorithm was evaluated on four benchmark instances and 360 randomly generated instances, and experimental results on those instances spent less CPU times than the LP procedure.

Lim (1997) was the first to propose genetic algorithm (GA) to solve BHSP. In his work, a mathematic model based on hoist move sequence was formulated, and the objective is to find the optimal hoist cyclic schedules with minimum cycle time. Specifically, hoist move sequences are encoded as chromosomes. In other words, each chromosome directly represents a possible hoist move sequence. Note that for such a representation way, the search ability of GA is generally reduced as the problem size increases. Besides, Linear Order Crossover (LOX) and two-gene mutation operator were adopted in the proposed GA. Computational results on benchmark instance P&U with different parameter settings were reported and indicated that the proposed GA can find the optimal hoist schedule for instance P&U.

Chen *et al.* (1998) first formulated a mathematical model and then proposed a B&B algorithm for BHSP. The proposed algorithm includes two branch-and-bound trees *A* and *B*. In particular, tree *A* is responsible for enumerating all possible initial part distributions at the beginning of a cycle, while tree *B* is responsible for generating the hoist schedules for each determined initial part distribution. Besides, to reduce the solution space, an upper bound of the number of parts which can be processed in the line within a cycle was derived from the formulated model. The proposed algorithm was evaluated on five benchmark instances: P&U, Ligne1, Ligne2, Black Oxide1 and Black Oxide2. Computational results on those instances indicated that the proposed B&B algorithm can find the optimal solution for each instance in less than 1s.

Recently, Yan *et al.* (2010) applied the method of prohibited intervals (MPI) to solve the BHSP. Specifically, if all the actual processing times in the processing tanks can be known, then the studied problem can be formulated by using the MPI approach (Levner *et al.*, 1997). Due to this fact, the studied problem was further transformed to

find all the non-prohibited intervals for the cycle time, which is done by a specific B&B algorithm. Computational results on benchmark instances and 1800 random instances demonstrated that the proposed method is effective for solving the problem. Moreover, due to the high performance of Tabu search (TS) algorithm, Yan *et al.* (2012) proposed a specific TS algorithm with a repairing procedure and solution space partition approach for the problem. In their work, to reduce the solution space and increase the search speed, the maximum number K of the work-in-process (WIP) parts was used to divide the solution space into K subspaces. Three rules based on the value of K were used to generate the initial population, i.e. hoist move sequences. Note that the proposed algorithm used the real-coded representation, that is, hoist move sequence is directly encoded as chromosome which does not require a decoding mechanism. Finally, the proposed TS algorithm was compared with GA proposed by (Lim, 1997) using both benchmark instances and random instances. Comparison results demonstrated that TS algorithm performs better than GA in terms of solution quality and computation time.

To reduce the complexity of hoist scheduling problem, some researchers studied the problem with given hoist move sequences. For instance, Lei (1993) proposed a simple algebraic procedure to minimize the cycle time and find the optimal start times of hoist operations for the scheduling problem with given hoist move sequences. The proposed procedure solves the studied problem in $O(N^2\log(N)\log(M))$ time, where N and M represent the tank numbers and the number of integer points between the lower bound and the upper bound on the cycle time, respectively. Besides, Ng and Leung (1997) proposed a binary search procedure to determine the optimal execution times of hoist moves for the similar problem.

All the works mentioned above treated the HSP from simple production line, in which each tank corresponds to a specific processing step. However, duplicated tanks and multi-function tanks are often used in practice. The representative works on HSP with duplicated tanks or multi-function tanks are Ng (1995) with MIP approach, Ng (1996) with B&B approach, Liu *et al.* (2002) with MIP approach, Zhou and Li (2003) with MIP approach, and Che and Chu (2007) with B&B approach.

Since a higher degree of cyclic schedule would generally improve the system productivity, several works have been published on this area. Note that a higher degree means that *at least two parts* enter and leave the line within a cycle. Some of the relevant works dealt with the single part type, and which can be found in the work

by Lei and Wang (1994), Spacek *et al.* (1999), Che *et al.* (2011), Kats and Levner (2011a and 2011b), Zhou *et al.* (2012), and Li and Fung (2014). Moreover, various exact or heuristic approaches have been proposed for HSP with multiple distinct parts: B&B approach (Lei and Liu, 2001; Lei *et al.*, 2014), MIP approach (El Amraoui *et al.*, 2008; Zhao *et al*., 2013a; El Amraoui *et al.*, 2013a), Polynomial algorithm (Kats *et al.*, 2008), and GA approach (El Amraoui *et al.*, 2013b).

Although the cyclic HSP is the theme of our research, several researchers have studied various variants of non-cyclic HSP due to its significance both in academic field and industrial practice. To date, much attention has been gained in this area, for examples, please see the work by Yih (1994), Lamothe *et al.* (1995), Ge and Yih (1995), Chauvet *et al.* (2000), Fleury *et al.* (2001), Hindi and Fleszar (2004), Paul *et al.* (2007), Kujawski and Świątek (2011), Zhao *et al*. (2013b), Tian *et al*. (2013), Yan *et al.* (2014), and Zhang *et al.* (2014).

2.1.2 Multiple objectives hoist scheduling problem (MOHSP)

In previous section, all mentioned works treated HSP with single objective, which minimizes either the cycle time or the makespan. This is far from meeting the various expectations from real-world applications. In other words, considering HSP with multiple objectives are more realistic, such as minimize the production cost or wastewater, maximize the productivity and minimize the defective part rate. Since 2000, multi-objective HSP has been studied, and a number of scheduling approaches have been proposed. In what follows, the relevant works are reviewed in details.

Firstly, Fargier and Lamothe (2001) proposed a decision support approach for the dynamic hoist scheduling problem with bi-objective, which is to minimize the makespan and maximize the processing quality. All parts are supposed to be randomly arrived and a single hoist for moving parts from tank to tank. The problem was formulated by a linear programming model to generate the best hoist schedules and a fuzzy model was used to evaluate the part processing operations.

Later, Mak *et al.* (2002) proposed a knowledge-based simulation system to solve the multiple hoists real time scheduling problem, in which multi-function tanks and duplicated tanks are used. The objectives of the problem are to maximize the productivity and minimize the defective rate. To avoid producing defective parts, the time of a new part entering into the line is controlled and determined by a heuristic rule. In the proposed simulation system, there are seven hoist dispatching rules, which

are Nearest Hoist First (NHF), Average Tank Assignment (ATA), Average Hoist Assignment (AHA), Boundary Shift by Job Allocation (BSJA), Modified Average Tank Assignment (MATA), Modified Average Hoist Assignment (MAHA), and Modified Boundary Shift by Job Allocation (MBSJA), respectively. Computational results on several real electroplating lines with different hoist speeds and hoist safe distances were reported and discussed. The results indicated that the two new rules MAHA and MBSJL perform better than all other dispatching rules. Besides, higher hoist speed and shorter hoist safety distances are verified to have higher productivity.

Xu and Huang (2004) designed a graph-assisted search algorithm for the single hoist cyclic scheduling problem with single part type to minimize both the cycle time and the wastewater. Specifically, a two-stage algorithm was proposed to optimize the two studied objectives. The first stage was responsible for finding the optimal hoist schedules with minimum cycle time, while the second stage was responsible for looking for the minimum wastewater for each determined hoist schedule. Moreover, part of infeasible hoist move sequences is eliminated during the search process. At last, a numerical example was used to evaluate the proposed two-stage optimization algorithm.

Jegou *et al*. (2006) proposed a multi-agent system for the reactive multi-hoist scheduling problem, where the objectives are to minimize the defective parts rate and maximize the productivity. In their model, two different agents called input date decision system (IDDS) and hoist assignment system (HAS) were respectively used to determine the time of a new part loading into the process line and to find the optimal schedules for multiple hoists. In HAS, auction operation was applied to assign transportation tasks to hoists and also optimize the hoist schedules. The proposed multi-agent system was compared with the existing hoist assignment heuristics (i.e. NFR, ARA and BSJL) in the literatures and showed better performance.

Kuntay *et al.* (2006) proposed a two-step optimization algorithm for solving the bi-objective single-hoist cyclic scheduling problem. In the proposed algorithm, the first step was responsible for finding an optimal hoist schedule with maximum productivity, while the second optimization step was to minimize the wastewater without reducing the production rate obtained in the first step. Finally, an example from real electroplating facility was used to evaluate the proposed two-step algorithm. Besides, Subaï *et al.* (2006) also proposed a similar two-step optimization algorithm for a bi-objective single-hoist cyclic scheduling problem, in which cycle time and

production cost are minimized in two sequential steps.

Zhang *et al.* (2012) studied the multiple hoists job shop scheduling problem with duplicated tanks and inter-storages between tanks, in which the objectives are to minimize both the makespan and the total waiting times in inter-storages. It should be noted that the solutions found with no waiting times correspond to feasible solutions for HSP. Firstly, a mathematical model was formulated for the problem, and then a genetic algorithm with tabu local search heuristic was proposed to find the optimal solutions. Computational results on several instances from different industry backgrounds demonstrated that the proposed approach is efficient.

Very recently, Feng *et al.* (2014) proposed an iterative epsilon-constraint method to solve a bi-objective HSP with non-Euclidean travel-time metric, which means that an empty move from tank *i* to tank *j* may need longer time than passing by an intermediate tank *k*. The objective is to minimize the cycle time and the total hoist travel times simultaneously. Firstly, an initial MIP model was formulated for the problem and then was further tightened by adding some valid inequalities. Secondly, an iterative epsilon-constraint method was proposed to find the complete Pareto optimal solutions for the problem. Finally, both benchmark instances and randomly generated instances were used to evaluate the effectiveness of the proposed method. Computational results showed that the proposed method can obtain Pareto optimal solutions in reasonable time.

Most above mentioned works (such as Xu and Huang, 2004, Kuntay *et al.*, 2006, and Subaï *et al.*, 2006) examined HSP with dual objectives, which are optimized in a separate way, i.e., one objective is optimized in the first step, and the other is considered in the next step while maintaining the optimized results obtained in the first step. Obviously, such separate and sequential optimization approaches can not necessarily find the global Pareto-optimal solutions for MOHSP. So it becomes urgent to develop efficient scheduling approaches for simultaneously achieving different objectives for HSP.

2.1.3 Cyclic multiple hoists scheduling problem (CMHSP)

Besides above, researchers have also worked on the problem with multiple hoists that generally lead to higher productivity compared to the single hoist system. In a multi-hoist system, the hoist usually move the part either in a unidirectional way or a bidirectional way. To be more specific, the unidirectional way means that the hoist

moves parts from left to right, i.e., the part processing sequence is exactly identical to the tanks layout, while the bidirectional way means that the hoist can move parts from left to right and from right to left, i.e., the part processing sequence is not necessarily identical to the tanks layout. To avoid hoist collisions, various scheduling approaches have been proposed, and they can be generally classified into two classes: (I) zone-partitioned based approaches and (II) overlapped based approaches. For class (I), the production line is divided into several non-overlapping zones according to the number of the hoists, and each hoist is exclusively assigned to one of zones for moving parts. Thus, overlapping the coverage ranges of the hoists is forbidden. In contrast, the production line is not divided and thus hoists can overlap with each other in class (II).

(I) CMHSP with zone-partitioned approach

Lei and Wang (1991) were the first to propose heuristic algorithm that is called Minimum Common-Cycle (MCC) algorithm, to find the optimal move schedules for a two-hoist cyclic scheduling problem. The proposed algorithm used a zone-partition approach to avoid two hoists conflicting with each other when they moved on a single track. More precisely, the production line is divided into two sections and each section is exclusively assigned to a single hoist. Finally, the proposed algorithm was verified by benchmark instance and random instances.

Armstrong *et al.* (1996) proposed a local optimization algorithm based on the greedy zone-partition approach for the multiple hoists scheduling problem with given cycle times, where overlapping the coverage ranges of the hoists are forbidden. The objective is to minimize the number of hoists used in the line. To avoid hoist collisions, the production line was divided into several non-overlapping zones, and each hoist was exclusively assigned to one of zones for moving parts. A local optimization algorithm was proposed to maximize the size of each zone, which is equivalent to minimize the number of hoists used in the system. Finally, computational results on both benchmark instances and random instances showed that the proposed approach is efficient for solving the problem.

Riera and Yorke-Smith (2002) proposed an improved hybrid model combining CLP with MIP to solve the generic cyclic scheduling problem with unidirectional multiple hoists. The proposed hybrid model adopted two different approaches to deal with hoist collisions, which are zone-partitioned (i.e. non-overlapped) approach and collision-based approach, respectively. Computational results on P&U instance and

randomly generated instances demonstrated that the proposed model is robust and scalable compared with the existing approaches.

Alcaide *et al*. (2007) proposed a parametric algorithm for a multiple hoists cyclic scheduling problem with given hoist move sequence. To prevent hoist collisions, all hoists are supposed to run on a circuit line in a carousel mode. Besides, all loaded or empty hoist moving times are not given specifically but within the pre-defined time intervals. The objective is to determine the values for actual processing times, loaded and empty hoist moving times so that the cycle time is minimized. The proposed parametric algorithm was verified by a numerical example.

Manier and Lamrous (2008) applied an evolutionary algorithm with a repairing procedure to solve the cyclic scheduling problem with multiple hoists running on parallel tracks, which means that each hoist has its own track and no collision happens between hoists. The objective is to minimize both the cycle time and the number of hoists since it is not given in advance. In their algorithm, chromosome is represented by empty hoist moves. An MIP approach was proposed to evaluate the feasibility of generated solutions. Moreover, a repairing procedure was designed to repair infeasible sequences. Computation results were reported and discussed with benchmark instances.

Besides, Zhou and Li (2009) proposed an MIP approach for the multi-hoists cyclic scheduling problem with duplicated tanks. In their work, the line was divided into several non-overlapping areas according to the number of hoists. That is, each hoist is assigned to an exclusive area and collisions only happen when two adjacent hoists meet at the boundary tank. An MIP model was first proposed to find the optimal hoist schedules. Then, the model was extended to solve the problem with duplicated tanks. The proposed model was solved by commercial software CPLEX. Computational results on three numerical examples with two and three hoists implied that the proposed approach is effective for solving the studied problem.

(II) CMHSP with overlapped approach

Baptiste *et al.* (1993) proposed a Constraint Logic Programming (CLP) method with depth-first search procedure to find the minimum cycle time for the hoist scheduling problem with different line configuration. The optimal cycle times obtained with the proposed approach for the P&U instance with one degree and single/two hoists as well as two degrees single hoist were reported. Finally,

advantages and disadvantages of the CLP languages as well as the comparison between the two different implementation languages (i.e. PROLOG III and CHIP) were also presented.

Moreover, Varnier *et al.* (1997) proposed a CLP based heuristic approach to obtain the optimal hoist schedules for a multi-hoist cyclic scheduling problem, where coverage ranges of the two neighboring hoists are allowed to overlap. That is, adjacent hoists can share several common tanks of the production line. The proposed approach consists of two specific procedures. In particular, procedure *A* used a heuristic rule to assign transportation tasks for each hoist. Then, procedure *B* used an exact method based on CLP to determine the optimal hoist schedules for the problem. Computational results on benchmark instances and random instances indicated that the multi-hoist system has larger productivity than the single hoist system.

Manier *et al.* (2000) developed a resolution procedure to solve the cyclic scheduling problem with bidirectional multiple hoists allowed to overlap on a single line, which includes duplicated tanks and multi-function tanks. Firstly, a mathematical model was formulated for the problem with disjunctive constraints (i.e. mutually exclusive inequalities). Then, the proposed model was implemented using CLP language. Based on the above works, a resolution procedure using branch-and-bound tree with depth-first search strategy was developed to find the optimal hoist schedules. Note that a node of the search tree represents a disjunctive constraint (i.e. a pair of operations), and when a leaf node is reached, an entire hoist schedule is obtained. Finally, computational results on benchmark instances and 35 randomly generated instances with no more than 3 hoists were given and showed that multi-hoists system improves the line productivity compared to the single hoist system.

Leung and Zhang (2003) formulated the first MIP model for the bidirectional multiple hoists cyclic scheduling problem. All hoists are supposed to be run on a single track and the production line is not partitioned according to the number of hoists. That is, two adjacent hoists may overlap in a common segment of the line. A branch-and-cut procedure with depth-first search strategy was used to solve the formulated MIP model. Computational results on six benchmark instances with no more than three hoists were reported and analyzed.

Che and Chu (2004) first formulated an analytical mathematical model and then proposed a B&B algorithm for the single track multiple hoists cyclic scheduling problem. The production line is supposed to be unidirectional. In their paper, two

collision-checking properties were derived to indentify the hoist collisions. The proposed B&B algorithm consists of two nest procedures *A* and *B*. In particular, procedure *A* is used to enumerate all possible tank state distributions at time zero, while procedure *B* is responsible for finding an optimal cyclic schedule for each given tank state distribution. The proposed algorithm was compared with the existing approaches by using both benchmark instances and random instances. Comparison results showed that the proposed B&B algorithm can find a smaller cycle times than the existing approaches.

Besides above, Leung *et al.* (2004) formulated the first MIP model for the cyclic scheduling problem with multiple hoists moving parts on a single track, in which the part processing sequence is exactly identical to the tanks layout. The objective of the problem is to minimize the cycle time for a given number of hoists. The authors first tighten the MIP model proposed by Phillips and Unger (1976) with new valid constraints. After that, by identifying all possible hoists-collision situations, they formulated an MIP model for the studied problem. In the experimental study, six benchmark instances with no more than three hoists were used to evaluate the performance of the proposed model, which is solved by the commercial optimization software CPLEX 6.5. Computational results on those instances were given and discussed.

Later, Zhou and Liu (2008) proposed a heuristic algorithm based on enumerating trial processing times for solving the cyclic scheduling problem with two hoists running on a single track. More precisely, actual processing time in each tank was randomly generated within their corresponding time intervals. Then, a simple algebraic method was proposed to determine the hoist move sequence according to the generated actual processing times. In their work, the production line was divided into three areas from left to right. For each given move sequence, all moves located at the left area (resp. right area) is exclusively assigned to hoist 1(resp. hoist 2). Hoist 1 and hoist 2 together take charge of performing all moves located at the middle area. Thus, collisions only happen in the middle area. Based on the above works, a linear programming (LP) approach was proposed to find the best schedule for each given hoist assignment. Finally, benchmark instance P&U and randomly generated instances were used to evaluate the performance of the proposed algorithm. Computational results on those instances demonstrated that the proposed heuristic algorithm can obtain near-optimal cycle time in a short time.

Chtourou *et al.* (2013) proposed a heuristic algorithm for the single track two hoists cyclic scheduling problem, where overlapping the coverage ranges of the hoists are allowed. Thus, hoist collisions in common segments must be avoided. In particular, the same method that presented in Zhou and Liu (2008) was used to generate hoist move sequences. Then, a heuristic algorithm was proposed for dispatching moves to hoist. Besides, to save the computation time, an MIP model without hoist collision constraints was formulated for determining the start time of each hoist move, and a test procedure was proposed for checking the collision constraints. The best solution is chosen from all the verified feasible solutions. Computational results were reported and analyzed with benchmark instances and random instances.

Very recently, Jiang and Liu (2014) formulated an MIP model and then proposed a B&B algorithm for the cyclic scheduling problem with bidirectional multiple hoists moving parts on a single line. For such a problem, identifying possible situations of hoist collisions are very crucial since that is a main part of the problem formulation. Based on a full analysis of the studied problem, an MIP model was first formulated, and then a B&B algorithm was proposed. The proposed algorithm was compared with Leung and Zhang's MIP approach (Leung and Zhang, 2003) and optimization software CPLEX (11.11) using P&U instance and random instances with different parameter settings (such as hoist numbers, problem size and time window width). Comparison results presented that the proposed B&B algorithm is more efficient than the two competitors in terms of CPU time.

2.2 Literature review on QEA

In this section, we review some works on QEA related to this research. In recent years, QEA has been received considerable attention from researchers because of its excellent optimization performance. It can be seen as a probability optimization algorithm based on the concepts and principles of quantum computation, such as Q-bits representation, observation process and various quantum gates (Deutsch, 1985). It has achieved great success in several well known optimization problems, such as travelling salesman problem (Narayanan and Moore, 1996), knapsack problem (Han and Kim, 2002), production scheduling problem (Li and Wang, 2007), and economic dispatch problem (Neto *et al*., 2011).

To our knowledge, Narayanan and Moore (1996) firstly introduced QEA to solve

the travelling salesman problem (TSP) and gained significant performance compared to classical method. Talbi *et al*. (2004) proposed a new QEA for TSP, and comparison results showed that QEA performs better than GA. Besides above, Han and Kim (2002) were the first to apply QEA to solve the knapsack problem. Moreover, Han and Kim (2004) proposed a new termination criterion and a novel quantum gate for QEA to solve the knapsack problem. Zhao *et al*. (2006) proposed a hybrid QEA that combines QEA with constraint handling method for knapsack problem. Zhang and Gao (2007) proposed an improved QEA (IQEA) with new rotation gate for knapsack problem. Comparison results indicated that IQEA is superior to basic QEA.

Due to its excellent performance, several researchers have also proposed various variants of QEA for production scheduling problems. For instance, Li and Wang (2007) employed QEA to solve the multi-objective flow shop scheduling problem. In their proposed QEA, chromosome is encoded by Q-bits, which are transformed into job sequence by a binary-decimal decoding scheme. Computational results showed that QEA is efficient and robust to obtain Pareto-optimal solutions with good diversity and proximity. Later, Gu *et al*. (2009) proposed a parallel QEA which also uses Q-bits encoding and binary-decimal decoding scheme for the stochastic job shop scheduling problem. Moreover, Gu *et al*. (2010) proposed a co-evolutionary QEA with same encoding and decoding scheme for the same problem as the one studied in Gu *et al*. (2009). Besides, Niu *et al*. (2009) proposed a hybrid algorithm called QIA that combines QEA with immune algorithm for the hybrid flow shop scheduling problem. Experimental results indicated that QIA is better than Immune algorithm in solution quality. Zheng and Yamashiro (2010) proposed a novel heuristic algorithm called QDEA that combines QEA with differential evolution for the permutation flow shop scheduling problem to minimize the total flowtime, makespan, and maximum lateness of jobs. In their proposed QDEA, chromosome is encoded by rotation angles, which are further used to order the job sequence.

2.3 Synthesis

In above sections, more than 60 articles about HSP are reviewed and analyzed in details. We judged that they are significant of the researches in the field, even if they still remain a part of the whole literature dealing with HSP and near problems. Figure 2.1 demonstrates the trend of those publications. We can see from it that the number of articles has been gradually increased in time, which implies that HSP has become a

hot research topic in the operations research area. A pie chart given in Figure 2.2 shows the ratios according to the approaches proposed in the literature. As can be seen from Figure 2.2, the most proposed approaches are Heuristic algorithm, MIP approach and B&B algorithm. Moreover, Table 2.1 presents a brief summary of the existing works on QEA related to our research. We can see from it that QEA has been applied in many research fields except for HSP. Based on the above works, we make the following remarks:

(I) By analyzing the publications about HSP in recent years, two research trends can be observed. One is to develop efficient approaches for solving various HSPs with multiple objectives, because optimizing a single objective is not enough to deal with the practical applications. The other is to study the HSP with multiple hoists since it is often encountered in many industrials.

(II) Due to the NP-completeness of HSP, it is a wise choice to adopt heuristic or meta-heuristic methods to find reasonably good schedules in a reasonable time, instead of obtaining an optimal one. To the best of our knowledge, no work was reported for using QEA to solve any types of HSP. This research tries to connect this gap as described in previous section.

(III) In most existing studies on the cyclic multiple hoists scheduling problem (CMHSP), such as Lei and Wang (1991), Armstrong *et al.* (1996), Leung *et al.* (2004), Zhou and Liu (2008), Chtourou *et al.* (2013), Jiang and Liu (2014), loaded hoist moves are implicitly or explicitly assumed to start and end within the same cycle. We think that scheduling approach under such an assumption may identify a non-optimal solution to be an optimal one, which can be verified by a counterexample. To find a global optimal solution, the above-mentioned assumption should be relaxed. In other words, a loaded hoist move is allowed to start in one cycle and end in the next one if necessary. Therefore, this research focuses on the development of an improved MIP approach for the CMHSP with relaxing the above-mentioned assumption.

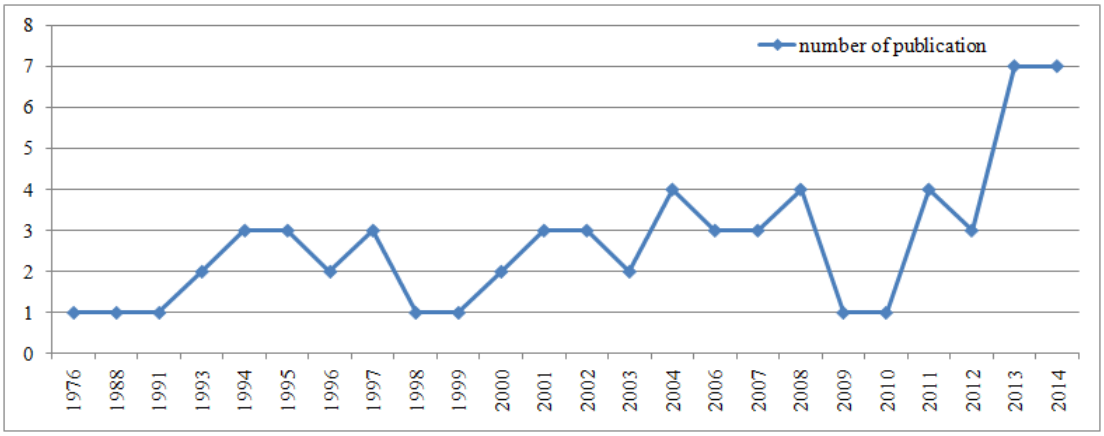

Figure 2.1 The trend of publications about HSP from 1976 to 2014.

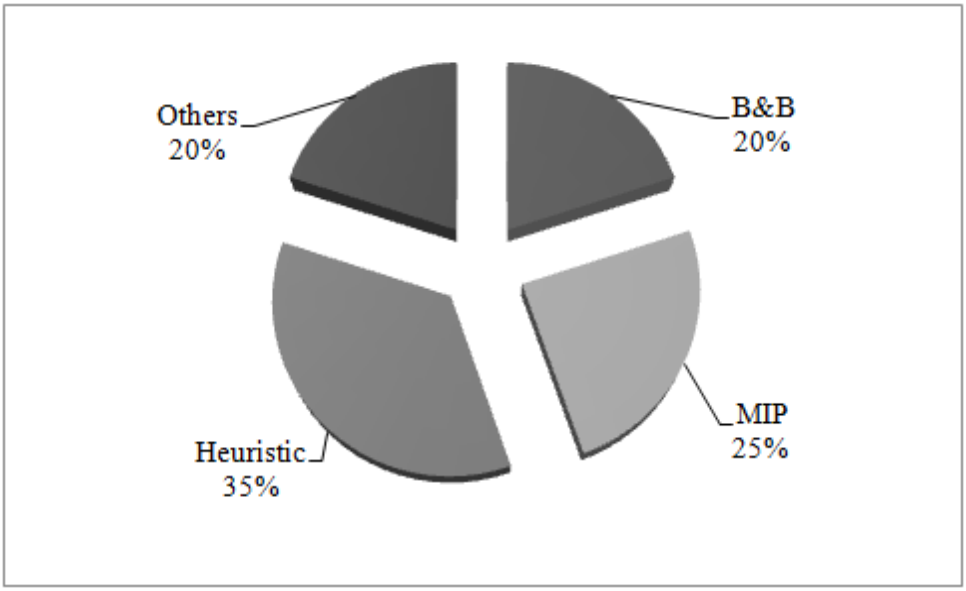

Figure 2.2 Ratio of proposed approaches in the reviewed HSP articles.

Table 2.1 Summary of QEA works

Problems	References
TSP	Narayanan and Moore (1996), Talbi *et al*. (2004)
Knapsack problem	Han and Kim (2002), Han and Kim (2004), Zhao *et al*. (2006), Zhang and Gao (2007)
Flow shop/Job shop scheduling	Li and Wang (2007), Gu *et al*. (2009), Niu *et al*. (2009), Gu *et al*. (2010), Zheng and Yamashiro (2010)
HSP	**Our contribution**

Chapter 3 A Hybrid Quantum Evolutionary Algorithm with Improved Decoding Scheme for HSP

3.1 Introduction

With the development of automation technologies, computer-controlled hoists instead of workers have been gradually used in many manufacturing industries to perform high frequency or dangerous transportation jobs. The advantages of robotic or automated manufacturing systems include higher productivity, better product quality, more efficient use of materials, improved safety and reduced labor intensity. Besides, highly robotic or automated manufacturing systems can effectively meet the requirement of mass production and respond to global competition.

In modern surface treatment facilities, the production line usually consists of several processing tanks arranged in a line and one or more hoists for transporting parts from tank to tank, as shown in Figure 1.1. Due to the industrial applications (Armstrong *et al*., 1996), the part processing time in each tank is usually limited to a pair of minimum and maximum time intervals, which is called time window constraints. The cyclic production mode is usually adopted in the automated manufacturing systems because of easy implementation in a mass production environment. This leads to a repetitive schedule performed by the hoist in every certain time. The duration of performing the repetitive schedule is called the cycle time or cycle length (Chen *et al*., 1998).

As mentioned in Chapter 2, Lei and Wang (1989) has proved that the simple HSP is NP-complete, but many researchers have constantly dedicated to this area and proposed various efficient methods for solving the relevant problems (Phillips and Unger 1976; Baptiste *et al*., 1993; Lei and Wang, 1994; Ng, 1996; Chen *et al*., 1998; Yan *et al*., 2010; Yan *et al*., 2012).

Since 1990s, QEA has been successfully applied to solve several well-known optimization problems, such as travelling salesman problem (Narayanan and Moore, 1996), knapsack problems (Han and Kim, 2002; Zhang and Gao, 2007), flow shop/job shop scheduling problems (Li and Wang, 2007; Gu *et al*., 2009; Gu *et al*., 2010), etc. Due to the NP-completeness of the studied problem, the computation time spent by exact methods usually increases exponentially with its size. Thus, it is a wise choice to use meta-heuristics to find sufficiently good schedules within a reasonable time.

Because of its unique advantages, such as better population diversity, rapid convergence, and very well global search ability, QEA has gained great success in many different optimization problems. Up to now, there is no work reported on using QEA to solve any types of HSP. So in this chapter, we propose a new scheduling algorithm based on QEA and genetic operators for the single-hoist cyclic scheduling problem with processing time windows.

The main contribution of this chapter is summarized as follows. Firstly, we propose a new decoding scheme with three different conversion procedures. Secondly, we propose a more effective repairing procedure than the one in Yan *et al*. (2012) to overcome the problem of unfeasibility of generated sequences which are often encountered in HSP. Note that in Yan *et al*. (2012), for each infeasible sequence, the reparation is conducted by randomly swapping any two moves. In this chapter, we first identify the move segment that causes infeasibility of the entire move sequence and then repair it. Finally, to increase the population diversity, crossover and mutation operators with adaptive probabilities are also implanted into our algorithm.

The rest of this chapter is arranged as follows. In the next section, we introduce the problem description and show an illustrative example of the problem as well as the problem formulation. The proposed algorithm with a repairing procedure is the subject of the Section 3.3. The experimental results and comparisons of the proposed algorithm with the existing approaches are given in Section 3.4. And finally, we conclude this chapter in Section 3.5.

3.2 Problem statement and mathematical model

3.2.1 Problem statement

As the problem has been studied in the literature, e.g. Phillips and Unger (1976), Lei and Wang (1994), Ng (1996), Chen *et al*. (1998), Leung *et al*. (2004), and Che and Chu (2007), we briefly give a problem description and notation, which are similar to those existing in the literature. Given n processing tanks (i.e., M_1, M_2,…, M_n) in a production line and a single hoist for part transportation. Both tanks and hoist are single capacity resources. Besides, tank 0 (i.e. M_0) and tank n+1 (i.e. M_{n+1}) are the input station and the output station, respectively. After a part is unloaded from M_0, it is to be successively processed through M_1 to M_n. The hoist moves a part from M_i to M_{i+1}, $0 \le i \le n$, which is called (loaded) move i. Each (loaded) movement includes three

sub-operations: 1) unloading a part from a tank; 2) carrying the part to the next tank; 3) loading the part into the tank. The hoist without carrying a part travels between two tanks, which is called empty move.

Moreover, the part processing time at each tank is said to be processing time windows, as it is confined within a pair of minimum and maximum time bounds. If the actual processing time violates the time limits, defective parts would be produced. Furthermore, at any time, each tank can process only one part. When a processing operation in a tank is finished, the part must be moved by the hoist to the next one without delay, which includes no pause of the loaded hoist. The production lines usually run in a cyclic mode since it is easy to implement. In each cycle, each tank is emptied exactly one time during a cycle, which involves cyclic schedules with one-degree. This chapter studies the one-cyclic scheduling problem with a single hoist, and the decision concerns how to optimize the hoist move sequences so as to maximize the productivity.

To facilitate the problem formulation, we define the following notations and variables in this chapter, which are similar to Leung *et al*. (2004):

$[L_i, U_i]$: the minimum and maximum bounds of the part processing time in M_i, respectively, $1 \le i \le n$.

d_i: the time needed to perform move i, $0 \le i \le n$.

$e_{i,j}$: the travel time for empty hoist from M_i to M_j, note that $e_{i,i} = 0$ and $e_{i,j} = e_{j,i}$, $0 \le i, j \le n+1$. The values of $e_{i,j}$ satisfy the well-known triangular inequality (Chen *et al*., 1998): $e_{i,j} \le e_{i,k} + e_{k,j}$, $k \notin \{i, j\}$, $i \ne j$, $0 \le i, j, k \le n+1$.

The decision variables are the following ones:

C: cycle time. It is the duration of a cycle.

t_i: the start time of (loaded) move i within a cycle, $0 \le i \le n$. Without loss of generality, move 0 is supposed to be the first move of a cycle, thus $t_0 = 0$.

To facilitate the formulation, we define the following intermediate variables:

s_i: if $s_i = 0$, then M_i is empty at the beginning of a cycle; else $s_i = 1$, then M_i is occupied by a part, $0 \le i \le n$. Define $S_n = \{s_0, s_1, \ldots, s_n\}$, which is called the initial part distribution at the beginning of a cycle. Without loss of generality, we let $s_0 = 1$ and $s_1 = 0$, since M_0 is always occupied by part at the beginning of a cycle and move 0 is

the first move of a cycle.

$r[i]$: the i+1th move performed by the hoist within a cycle, $0 \le i \le n$. As mentioned above, we have $r[0]=0$. Define $R_n=<r[0], r[1], r[2],\ldots, r[n]>$, which represents the sequence of moves during a cycle. An example of R_n with $n=3$ is $R_3=<0, 2, 3, 1>$, where $r[1]=2$, $r[2]=3$, and $r[3]=1$, as shown in Figure 3.1. Here, $r[1]=2$ means that the second move transfers a part from M_2 to M_3.

Figure 3.1 shows an illustrative example of the studied problem with $n=3$. In this example, there are three processing tanks (i.e., M_1, M_2 and M_3) with a single hoist for part transportation as well as the loading station (i.e. M_0) and the unloading station (i.e. M_4). In Figure 3.1, the inclined solid arrows and the broken arrows represent the loaded moves and the empty moves, respectively. The start point and end point of an inclined solid arrow (resp. a broken arrow) represent the start time and the end time of corresponding loaded (resp. empty) move, respectively. Furthermore, the horizontal solid line represents the duration of the part processing operation. The production line is supposed to be in steady-state. As can be seen from Figure 3.1, at time 0, M_2 is the only tank to be occupied (and implicitly M_0). So the initial part distribution is S_3 = {1, 0, 1, 0}. For this distribution, the optimal hoist move sequence is $R_3=<0, 2, 3, 1>$ (i.e., $t_0<t_2<t_3<t_1$). When move 1 finishes, the hoist comes back to M_0 and performs move 0 of the next cycle. We can also see that the hoist performs the same loaded (or empty) move sequence in time interval [C, 2C] as those ones in time interval [0, C]. This is called cyclic production mode. The duration of the repetitive sequence (i.e. R_3) is the cycle time C.

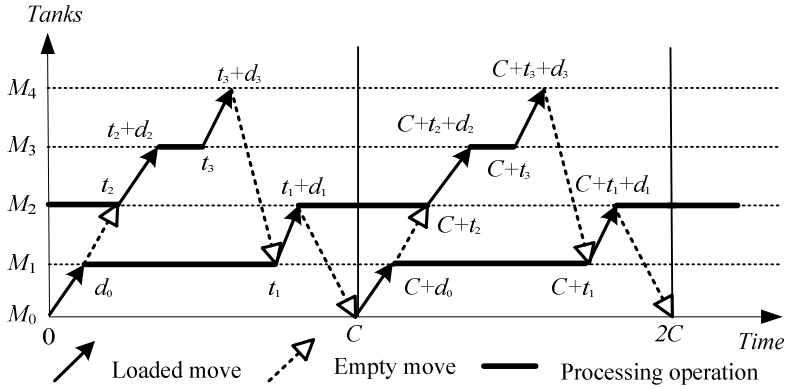

Figure 3.1 An example of cyclic scheduling problem with a single hoist.

According to the notation in (Manier and Bloch, 2003) dedicated to hoist

scheduling problems, the problem studied in this chapter can be expressed in the following form:

$$CHSP \mid n \,//\, diss \mid /n+2 \mid Cmin$$

which means the single hoist cyclic scheduling problem with n tanks, n+2 operations per part, dissociated loading and unloaded stations, and minimization of cycle time C as the objective.

3.2.2 Mathematical model

As mentioned above, move 0 is supposed to start at time 0, then the start times of other moves are all greater than 0. Thus, we have (Lei, 1993; Ng, 1995):

$$t_0=0,\ t_i>0, \text{ for } 1\le i\le n, \tag{3.1}$$

In Figure 3.1, we notice that the start time of processing operation i is the same as the end time of loaded move $i-1$(i.e. $t_{i-1}+d_{i-1}$); the end time of processing operation i is the same as the start time of loaded move i (i.e. t_i). Moreover, there are in total two possible states (empty or occupied) for each tank at the beginning of a cycle. Based on the above observations, the actual processing time in M_i can be represented as $t_i-(t_{i-1}+d_{i-1})$ for $s_i=0$ (like tank M_1 in Figure 3.1) and $C+t_i-(t_{i-1}+d_{i-1})$ for $s_i=1$ (like tank M_2 in Figure 3.1), respectively. Considering the processing time requirements, we have (Chen *et al*., 1998):

$$L_i\le s_iC+t_i-(t_{i-1}+d_{i-1})\le U_i,\ 1\le i\le n, \tag{3.2}$$

Furthermore, the hoist must have enough time to perform any two successive moves (i.e. $r[i]$ and $r[i+1]$), thus, the following relation holds (Chen *et al*., 1998):

$$t_{r[i]}+d_{r[i]}+e_{r[i]+1,\,r[i+1]}\le t_{r[i+1]},\ 0\le i\le n-1, \tag{3.3}$$

It should be noted that constraint (3.3) also implicitly guarantees the satisfaction of tank capacity constraint. For instance, as shown in Figure 3.1, we have $r[1]=2$, $r[3]=1$, and $c_2=1$. By the definition of tank capacity constraints (i.e., an occupied tank must be emptied before processing a new part), move 2 must performs before move 1, and thus we have: $t_2+d_2+e_{3,1}\le t_1$, which must hold. From constraint (3.3), we can have: $t_2+d_2+e_{3,3}\le t_3$; $t_3+d_3+e_{4,1}\le t_1$, which leads to $t_2+d_2+e_{3,3}+d_3+e_{4,1}\le t_1$. Since $d_3+e_{4,1}>e_{3,1}$, the inequality $t_2+d_2+e_{3,1}<t_1$ holds. Therefore, we see that tank capacity constraint is implicitly ensured by constraint (3.3).

Once the last move (i.e. $r[n]$) finishes, the hoist must come back to M_0 for

executing move 0 of the next cycle. Hence, we have (Chen *et al*., 1998):

$$t_{r[n]}+d_{r[n]}+e_{r[n]+1,\,0}\le C,\ 1\le r[n]\le n. \tag{3.4}$$

Based on the above works, the mathematical model for the single-hoist one-degree cyclic scheduling problem with processing time windows can be formulated as (Chen *et al*., 1998):

Min. C

s.t. (3.1)–(3.4).

3.3 Hybrid Method

In what follows, we present a specific hybrid QEA (labeled HQEA in the following) for the studied problem. More precisely, in Section 3.3.1, we introduce the traditional solution representation and decoding schemes; in Section 3.3.2, we present the Q-bits representation; in Section 3.3.3, we determine the states of Q-bits in each individual; in Section 3.3.4, we present the decoding procedures; in Sections 3.3.5 and 3.3.6, we describe the fitness evaluation function and the repairing procedure, respectively; in Section 3.3.7, we introduce the rotation gate and the genetic operators to update individuals; finally, in Section 3.3.8, we present the flowchart of the proposed hybrid algorithm.

3.3.1 Introduction

In QEA or GA models, a solution (also called chromosome) is usually represented by a permutation of job input sequence in classic flow shop or job shop scheduling problems. However, a chromosome is encoded by Q-bits in QEA, which is then converted into a binary chromosome. That is, QEA is generally based on a binary encoding. For this reason, a key issue in the development of QEA for production scheduling problems is to design an efficient decoding mechanism to convert a binary representation into a permutation-based representation. Typically, there are mainly two decoding schemes used in QEAs in the literature for solving various scheduling problems: binary-decimal decoding and shifting decoding. For the binary-decimal decoding, it first uses a binary segment for each job and then converts it into a decimal number. After that, all jobs are sequenced based on their corresponding converted decimal numbers. It is understandable that the chromosome under such a scheme is usually very long, especially when the problem size is large. As a result, the

search efficiency of the algorithm may be reduced. As for shifting decoding, it uses a permutation chromosome as a parent pattern and shifts its genes with the direction of a binary chromosome so as to generate a new permutation chromosome. Such a decoding usually has a better computational efficiency than binary-decimal decoding. But it cannot make full use of the advantage of QEA due to its permutation-based representation.

To overcome the above drawbacks, we propose a new decoding scheme in this study. In our scheme, a binary chromosome is directly converted into permutation chromosome (i.e. a hoist move sequence) using several different decoding procedures. Our decoding scheme can efficiently exploit the solution diversity due to Q-bits chromosome compared to shifting decoding, and has a shorter chromosome than binary-decimal decoding. In the following, we present the Q-bits representation.

3.3.2 Representation

Indeed, we notice that tank state and Q-bit state have the same characteristics. That is, they both are either 0 or 1. Since precedence relations need to be determined between n moves in this chapter, we let Q-bit i corresponding to tank i, for $1 \le i \le n$, and use Rule 1 and Rule 2 introduced in the following section to determine each Q-bit state. If Q-bit i is in state "0" (i.e., $s_i = 0$), which represents that move $i-1$ is performed before move i during a cycle; otherwise (i.e., $s_i = 1$), move i is performed before move $i-1$ during a cycle. Hence, an individual Ψ containing n Q-bits is used to represent n tank states, and is defined as follows:

$$\Psi = \begin{bmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \beta_1 & \beta_2 & \dots & \beta_n \end{bmatrix} \tag{3.5}$$

where $|\alpha_i|^2+|\beta_i|^2=1, 1 \le i \le n$. Note that in the initialization step, all Q-bits in Ψ are initialized as the equal probability (i.e. 1/2) of being 0 or 1. From above, we can know that each quantum individual corresponds to a complete part distribution S_n, more precisely the state of each M_i (i.e. empty or occupied).

In more classical and direct representations for the studied problem, each individual represents a moving sequence, so the value of gene j gives the index of the tank from which the j^{th} move starts during one cycle. In such representations, the solution space contains $n!$ individuals. With our representation, we handle in a first step only 2^{n-1} individual (and not 2^n, because s_1 is always equal to 0, it is not use making it explicitly appear in the representation). This number may be further reduced for some instances with Rule 1, as explained in the following. Moreover,

each Q-bits individual generally corresponds to several moving sequences, which we consider in a second step. Each time Rule 1 enables us to determine that an individual is not good, then all the associated moving sequences are unfeasible ones and it is no use evaluating them.

3.3.3 Initialization

For each specific instance, some tank states may be directly determined by the following method. Specifically, we first suppose that s_i=1, therefore move i occurs before move i–1 within a cycle. Moreover, let us suppose that move i–1 and move i are the last move and the second move of a cycle, respectively. Correspondingly, the minimum processing time of a part in M_i with s_i =1 is $e_{i,0}+d_0+e_{1,i}$. As an example, if we consider move 1 and move 2 in Figure 3.1, the processing time of a part in M_2 is equal to $e_{2,0}+d_0+e_{1,2.}$ Indeed, move i would be the first move to be performed after move 0, and move i–1would be the last move of the cycle. Else, the processing time in M_i would be greater than $e_{i,0}+d_0+e_{1,i}$, which would make the following assertion even more true. Then we can compare this processing time with U_i which is the maximum authorized time in M_i.

1) If $U_i<e_{i,0}+d_0+e_{1,i}$ (hereafter called Rule 1) happens, then we can know that the processing time requirement in M_i is violated. Consequently, all sequences relevant with s_i =1 are infeasible ones. So s_i must be 0.
2) Else, s_i may be 0 or 1.

Note that Rule 1 can be used to reduce the enumerating space of S_n and thus improve the search efficiency. Indeed, if Rule 1 enables us to fix 0 to the values of p variables s_i, then the search space of S_n can be reduced to 2^{n-p-1}individuals.

For the state of Q-bit i in a quantum individual Ψ that is not determined by Rule 1, a random number rd_i is generated from the uniform distribution [0, 1). If $rd_i>|\alpha_i|^2$, then Q-bit i is in state “1” (i.e. s_i =1); else, Q-bit i is in state “0” (i.e. s_i =0). This method is called Rule 2. Based on the above, the states of all Q-bits in one individual can be easily determined by Rule 1 and Rule 2, that is to say the initial part distribution S_n.

3.3.4 Decoding Scheme

In what follows, we present how we derive the hoist move sequence from a quantum individual. For a better diversification, three different decoding procedures described in the following are used to convert a quantum individual into possible hoist

move sequences, providing that the states of all Q-bits (i.e. S_n) in a quantum individual are already determined.

3.3.4.1 Decoding procedure 1

For ease of description, we first define λ_i be a copy of s_i and $\lambda_i = s_i$. Let Φ be a set that records the performed moves. It should be noted that λ_i can be seen as an indicator that indicates the state (i.e., empty or occupied) of M_i in the process. Thus, the value of λ_i is dynamically modified in the process. That is, when move i finishes, both the states of M_i and M_{i+1} are changed, i.e., M_i becomes empty and M_{i+1} is occupied by a part. Thus, we set λ_i =0, λ_{i+1} =1 and put move i into set Φ .

Procedure 1 mainly depends on the probability sizes of Q-bits in Ψ to derive the hoist move sequence, for $1 \le i \le n$. In particular, for given S_n, when move $r[k]$ finishes, for $0 \le k \le n$, we first calculate the number (labeled with cnt) of λ_i=1 under condition λ_{i+1}=0 (note that if $i=n$, the output station can be seen as always be empty) and $i \notin \Phi$. Then, we successively assign i with above condition to Ω_m (i.e. $\Omega_m=i$) in set $\Omega=\{\Omega_1,\ldots\Omega_{cnt}\}$, which is defined to record the possible moves for the next step, for $1 \le m \le cnt$. Thus, each step has in total cnt possibilities. Finally, we choose move j with the highest probability (i.e. $|\alpha_j|^2$) in set Ω as move $r[k+1]$, and let λ_j=0, λ_{j+1}=1(for $j \ne n$),$\Phi=\Phi \cup \{j\}$. In the next step, we update both cnt and Ω, and use a similar way to derive the following move (i.e. $r[k+2]$). When the whole hoist move sequence (i.e. R_n) is determined, this procedure stops.

For example, a complete part distribution (corresponding to a quantum individual) S_n with n=5 is S_5 ={1, 0, 1, 0, 1, 0}. When the first move (i.e. $r[0]$) finishes, by definitions, we have λ_1=1, λ_2=1, λ_3=0, λ_4=1, λ_5=0 and Φ={0}, from which we can know that M_1, M_2 and M_4 are currently occupied by a part. As the hoist cannot unload a part from an empty tank and also cannot load a part into an occupied tank, we have Ω={2,4}. Finally, according to the selection rule, if $|\alpha_2|^2 \ge |\alpha_4|^2$, we have $r[1]$=2;else, $r[1]$=4. The similar ways are used to update λ_i, Φ, Ω and then determine $r[k]$, $2 \le k \le 5$.

3.3.4.2 Decoding procedure 2

For ease of description, we keep the intermediate parameters λ_i, Φ and Ω defined in procedure 1. Furthermore, we define st_i be the start time of move i in the process of deriving the whole sequence and let st_0=0, for $0 \le i \le n$. To derive a move sequence from given S_n, procedure 2 mainly depends on the rule of minimal time unit increment between $st_{r[k]}$ and $st_{r[k+1]}$, for $1 \le k < n$, while respecting the processing time windows,

since the objective of the problem is to minimize the cycle time C. In other words, in each step, we have a set of several moves and choose one move with the earliest starting time as move $r[k+1]$ from the set.

In particular, on one hand, when move $r[k]$ finishes, as similarly done in procedure 1, we derive the values of cnt and Ω from each given S_n. On the other hand, we design a different strategy to determine move $r[k+1]$ compared with the procedure 1. At first, we calculate each st_j (supposing $j=\Omega_m$) in set Ω, that is, $st_j=st_{r[i]}+d_{r[i]}+e_{r[i]+1,\ j}$, $1\le m\le cnt$. Then, for each move j in set Ω, we check whether move $j-1$ exists in the partial determined sequence $<r[0], r[1],\ldots, r[i]>$. If it exists and $st_j-st_{j-1}-d_{j-1}<L_j$ happens, then we update $st_j=st_{j-1}+d_{j-1}+L_j$ so as to meet the minimal processing time requirement. Then it involves a waiting time of the empty hoist above tank j until the minimal processing time in tank j is completed. Finally, we choose move j (supposing $j=\Omega_m$) with the smallest value of st_j in set Ω as move $r[k+1]$, and let $\lambda_j=0$, $\lambda_{j+1}=1$ (for $j\neq n$),$\Phi=\Phi\cup\{j\}$. In the next step, we update both cnt and Ω so as to derive move $r[k+2]$. When the whole sequence (i.e. R_n) is determined, this procedure stops.

For instance, an example of S_n with $n=5$ is $S_5=\{1, 0, 0, 1, 0, 1\}$. When the first move (i.e. $r[0]$) finishes, by definitions, we have $\lambda_1=1$, $\lambda_2=0$, $\lambda_3=1$, $\lambda_4=0$, $\lambda_5=1$ and $\Phi=\{0\}$ as well as $\Omega=\{1, 3, 5\}$. We first calculate st_1 (note that if $st_1-d_0<L_1$, then $st_1=d_0+L_1$), st_3 and st_5 by st_0+d_0 plus $e_{1,\ 1}$, $e_{1,\ 3}$, $e_{1,\ 5}$, respectively, then choose the move with the smallest starting time among the three candidates as $r[1]$. The similar ways are used to update λ_i, Φ, Ω and then determine $r[k]$, $2\le k\le 5$.

3.3.4.3 Decoding procedure 3

Procedure 3 mainly depends on the precedence relationship between move $i-1$ and move i (i.e., the value of s_i) to derive the move sequence. For each given S_n and R_n (i.e. quantum individual), if $s_i=1$, then move i is set before move $i-1$in R_n; else, move i is set after move $i-1$ in R_n. For instance, an examples of S_n and R_n with $n=5$ are respectively $S_5=\{1,0,1,1,0,1\}$ and $R_5=<0, 2, 1, 4, 3, 5>$, from which we can easily derive a possible sequence that is $R_5=<0, 3, 2, 1, 5, 4>$. Note that at the initial step, we set $r[i]= i$, $0\le i\le n$.

Based on the above descriptions, we first apply the three proposed decoding procedures to each quantum individual and then select the best sequence (i.e. the best fitness) from the three generated sequences to represent this individual.

3.3.5 Fitness evaluation

To facilitate the description, *fit*(X) is defined to represent the fitness value of each individual X, and it can be computed as follows: *fit*(X)=*F*/*C*, in which *F* is a parameter and set as 2000 in this chapter. From this definition, we see that the smaller the cycle time *C* (*C*>0), the greater the fitness value. For each individual relevant with a hoist move sequence, it is evaluated by using the graph-based polynomial procedure (Chen *et al*., 1998). In particular, if the sequence is proved to be feasible, then the procedure returns a positive value for the cycle time *C* and the individual fitness can be calculated; Otherwise, the individual fitness is set to be 0. For more details about the graph-based polynomial procedure, please see Chen *et al*. (1998).

3.3.6 Repairing procedure

It should be noted that constraints (3.2) ~ (3.4) formulated in subsection 3.2.2 can be regarded as two classes. One is flexible processing time constraints and the other is hoist transportation capacity constraints, which are (3.2) and (3.3), (3.4), respectively. Generally, if a sequence R_n is infeasible, the following cases happen:

(C1) the flexible processing time constraint is violated;

(C2) the hoist transportation capacity constraint is violated;

Due to the characteristics of the HSPs in terms of constraints, it is well known that very few feasible solutions exist among the numerous possible moving sequences. Long before searching the optimal solution, the first challenge is to find feasible sequences. So some repairing procedures are often required to transform the unfeasible solutions into feasible ones. In what follows, we present the repairing procedure based on the above cases. For an individual with an associated hoist move sequence R_n, we identify each partial sequence in a whole hoist move sequence R_n which is either in sequence of $i-1\rightarrow\bullet\rightarrow\bullet\rightarrow i$ (which means move $i-1$ is performed before move i within a cycle) or of $i\rightarrow\bullet\rightarrow\bullet\rightarrow i-1$(which means move i is performed before move $i-1$ within a cycle). That is to say, a complete hoist move sequence R_n consists of n pieces of such a partial sequence. For ease of description, we define the following parameters:

$z_{i-1,\,i}$: the duration between the finish time of move $i-1$ and the start time of move i for a partial sequence $i-1\rightarrow\bullet\rightarrow\bullet\rightarrow i$, for $1\le i\le n$. Note that $z_{i-1,\,i}$ generally equals to the sum of all loaded move (denoted by •) times and relevant empty move times. If there exists a pair of moves $j-1$ and j in the sequence, that is $i-1\rightarrow\bullet\rightarrow j-1\rightarrow\bullet\rightarrow j\rightarrow i$, and $z_{j-1,\,j}<L_j$, then we let $z_{i-1,\,i}= z_{i-1,\,i}+L_j-z_{j-1,\,j}$. Note that $z_{i-1,\,i}$ may span the cycle or be

within a cycle. For example, in Figure 3.1, the two consecutive sequences are $0\rightarrow2\rightarrow3\rightarrow1\rightarrow0\rightarrow2\rightarrow3\rightarrow1$. From it, we can see that $z_{0,1}$ and $z_{2,3}$ are within a cycle, but $z_{1,2}$ spans the cycle. Therefore, $z_{i-1,i}$ can be used to check the satisfaction of flexible processing requirements no matter s_i=0 or s_i=1.

d: the mean time of all loaded move times, $d=\sum_{i=0}^{n} d_i/(n+1)$.

e: the mean time of all empty move times, $e=2(\sum_{i=0}^{n+1}\sum_{j=i}^{n+1} e_{i,j}-e_{0,1})/(n^2+5n)$. Note that the possible number of empty moves is $(n^2+5n+6)/2$. Since the empty moves between M_0 and M_0, M_0 and M_1, M_{n+1} and M_{n+1} do not actually happen, the number is reduced to $(n^2+5n)/2$.

For an infeasible sequence R_n, we first use parameters $z_{i-1,i}$ to check the sequence R_n.

1) If $z_{i-1,i}$ is verified to be greater than its upper bound U_i, then we remove one or more move(s) from the corresponding partial sequence, so as to make the partial sequence to be feasible; else if $z_{i-1,i}$ is verified to be smaller than its lower bound L_i, and the time gap between L_i and $z_{i-1,i}$ is greater than the sum of *d* and 2*e*, then we insert possible moves into the partial sequence.

2) Then, we identify the violated hoist capacity constraints by the start times of all moves (i.e., t_i, $1\le i\le n$) given by the evaluation process. For ease of description, let moves *i* and *j* be the identified two moves violating the hoist capacity constraints, that is, $t_i+d_i+e_{i+1,j}>t_j$, with $t_i<t_j$. If these two moves are two consecutive moves, we set move *j* before move *i* in sequence R_n so as to make the sequence be feasible; else, we remove one or more moves between moves *i* and *j* so as to make the two moves satisfy the hoist capacity constraints.

3.3.7 Updating individuals

3.3.6.1 Rotation gate

In this chapter, the rotation gate $U(\Delta\omega_i)$ is adopted as the variation operator to update the Q-bits in (3.5). ω_0 is set to be as the initial rotation angle. For individual X, the Q-bit *i* in it can be updated as the following way (Han and Kim, 2002; Li and Wang, 2007):

$$\begin{bmatrix} \alpha_i' \\ \beta_i' \end{bmatrix} = \begin{bmatrix} \cos\Delta\omega_i & -\sin\Delta\omega_i \\ \sin\Delta\omega_i & \cos\Delta\omega_i \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} \tag{3.6}$$

We define *fit_b* be the fitness of the best individual found in population. The rotation angle $\Delta\omega_i$ is defined according to the respective values of the corresponding parameter s_i in the individual X (labeled s_{i-X}) and in the best one (labeled s_{i-best}). If the condition *fit*(X)<*fit_b* holds, then consider the following conditions (Han and Kim, 2002):

Case A: If Q-bit *i* is in the 1st or the 3rd quadrant, then consider the following:

Case (A.1): if s_{i-best}=1 and s_{i-X}=0, then $\Delta\omega_i$=($-\omega_0$), here the rotation angle $\Delta\omega_i$ is set negative so as to increase the probability that Q-bit *i* is in state "1";

Case (A.2): if s_{i-best}=0 and s_{i-X}=1, then $\Delta\omega_i$=ω_0, the rotation angle $\Delta\omega_i$ is set positive so as to increase the probability that Q-bit *i* is in state "0";

Case (A.3): else, $\Delta\omega_i$ =0;

Case B: If Q-bit *i* is in the 2nd or the 4th quadrant, then consider the following:

Case (B.1): if s_{i-best}=1 and s_{i-X}=0, then $\Delta\omega_i$=ω_0, here the rotation angle $\Delta\omega_i$ is set positive so as to increase the probability that Q-bit *i* is in state "1";

Case (B.2): if s_{i-best}=0 and s_{i-X}=1, then $\Delta\omega_i$=($-\omega_0$), the rotation angle $\Delta\omega_i$ is set negative so as to increase the probability that Q-bit *i* is in state "0";

Case (B.3): else, $\Delta\omega_i$ =0;

Besides, since the probability of a Q-bit *i* in state "0" may be equal to 1 or 0, the updated Q-bit *i* may be trapped in state "0" or "1", which may lead to the premature convergence of population. Thus, a small constant μ is applied to ensure that the probabilities of the two states are both belonged to the range [μ, 1−μ]. As a result, the following equation must be considered (Han and Kim, 2004):

$$\begin{bmatrix} \alpha_i' \\ \beta_i' \end{bmatrix} = \begin{cases} \left[\sqrt{\mu} \quad \sqrt{1-\mu}\right]^T & if \quad \alpha_i' < \sqrt{\mu} \\ \left[\sqrt{1-\mu} \quad \sqrt{\mu}\right]^T & if \quad \alpha_i' > \sqrt{\mu} \\ \left[\alpha_i' \quad \beta_i'\right]^T & else \end{cases} \tag{3.7}$$

By applying the decoding procedures given in Section 3.3.4 to each updated quantum individual, hoist move sequences can be generated from it.

3.3.6.2 Genetic operators

In this subsection, selection, crossover and mutation operators (Akpinar and Bayhan, 2011) are applied to further evolve the population. To facilitate the description, the following notations are given:

c_p, m_p: crossover and mutation probabilities, respectively.

fit_a: the average fitness of the entire population.

*fit*_0: the maximum fitness of a specific instance, which is computed as follows: *fit*_0=2000/C_L. C_L is the lower bound on cycle time C for the instance. It can be obtained by the following way, which is taken from Chen *et al*. (1998):

$$C_L \geq max(L_i+d_i+d_{i-1}+e_{i+1,\,i-1}),\ 1\leq i\leq n. \tag{3.8}$$

According to Srinivas and Patnaik (1994), c_p and m_p are defined respectively in a similar way:

$$c_p = 0.7\times[fit\_0 - fit\_b]/[fit\_0 - fit\_a]. \tag{3.9}$$

$$m_p = 0.5\times[fit\_0 - fit(\mathrm{X})]/[fit\_0 - fit\_a]. \tag{3.10}$$

Adaptively adjusting c_p and m_p (i.e., (3.9) and (3.10)) can prevent divergence and escape from the local optimal, since (3.9) and (3.10) can dynamically reduce c_p and m_p for individuals with high fitness, or increase c_p and m_p for individuals with low fitness.

In this chapter, two-point crossover operator is applied to generate the offspring. First, two individuals are chosen by the binary tournament method as parents 1 and 2; then, for parent 1, two different positions p and q are randomly chosen, $p, q\in[1, n]$. For $i\in[1, p)$ and $(q, n]$, the values of $r[i]$ for the new offspring1 inherits from parent 1. For $i\in[p, q]$, the new $r[i]$ is sequentially chosen from parent 2, on condition that its value was not already chosen from parent 1. The same operations are done, starting with parent 2 and then parent 1, to generate offspring2. This operation is depicted as Figure 3.2(a), in which | is the chosen position.

Besides, a mutation operator is adopted to prevent a solution falling into a local optimum of a specific instance, which is designed as follows. For a chosen individual $R_n = <r[0], r[1], r[2],\ldots, r[n]>$, first, we randomly choose a position p, $p\in[1, n]$, then randomly reorders the move sequence in $<r[p+1], r[p+2],\ldots, r[n]>$. This operation is depicted as Figure 3.2(b), in which | is the chosen position.

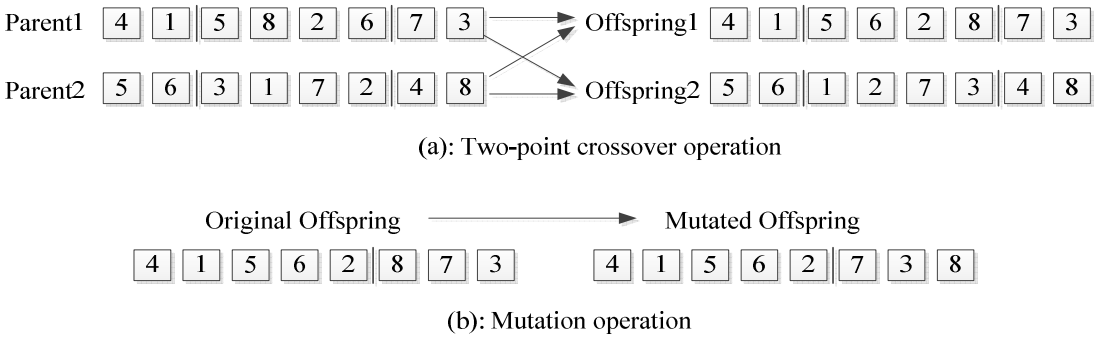

Figure 3.2 Crossover and mutation operators.

3.3.8 The procedure of hybrid QEA(HQEA)

Based on the above works presented in sections 3.3.1~3.3.7, the procedure of HQEA for solving the considered problem can be depicted as Figure 3.3. From this flowchart, we can see that the proposed algorithm uses two mechanisms to update the population: Q-gate and genetic operators.

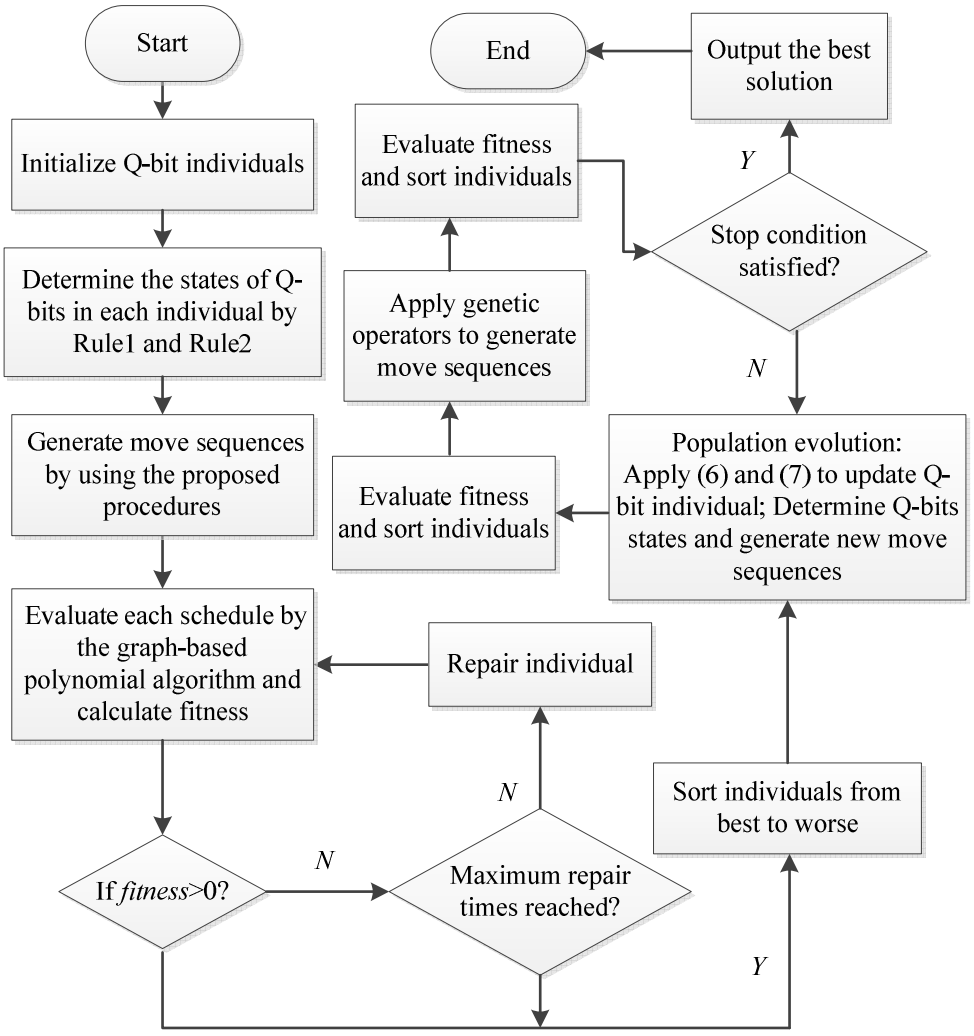

Figure 3.3 The flowchart of the proposed HQEA.

3.4 Experimental results

To verify the feasibility and applicability of the proposed HQEA, both benchmark and randomly generated instances were used in the experimental study. All computational experiments were conducted on an ASUS Laptop with an Intel Core i5-3210M Processor 2.50GHZ and on a windows 8 environment. The parameters were set as follows: population size: Popsize=50; the maximum number of generations: MaxIter=200; Initial rotation angle ω_0=0.05π; μ= 0.008. The maximum repairing times were set as 6. For evaluating the quality of the solution obtained with our HQEA, the same problem was also formulated by the mixed integer programming (MIP) approach and solved by the ILOG CPLEX (Version12.4).

3.4.1 Experimental results on benchmark instances

The proposed algorithm was verified by using five well known benchmark instances in the literature: Mini Phillips (Mini, n=8), Black and Oxide2 (BO2, n=11), Phillips and Unger (P&U, n=12), Ligne1 (n=12) and Ligne2 (n=14), which are taken from Leung *et al*. (2004), Phillips and Unger (1976) and Manier (1994), respectively.

Table 3.1 gives the experimental results for five benchmark instances obtained with our algorithm and CPLEX software, in terms of the number of remaining possible S_n after applying Rule 1 (Nb. S_n after Rule 1 for short), the Convergence generation(Con.gen. for short), the Best cycle times and the CPU times (measured in seconds). The "Con.gen." refers to how many generations are needed for our algorithm to find the best solution and no improvement on the solution in the later. Consequently, the sub-column "Con. time" represents the time needed by the "Con.gen." and is computed as: Con. time= Con.gen. × (Our CPU time/MaxIter).

Table 3.1 Results for the benchmark instances

Instances	Nb. S_n after Rule 1	Con.gen.	Best cycle times			CPU times(In seconds)			
			Our	CPLEX	SD	Our	Con. time	CPLEX	Gap
Mini	$\mathbf{2^6}$	2	287	287	0	4.75	0.048	0.16	–0.112
BO2	2^{10}	13	279.3	279.3	0	5.26	0.342	0.25	+0.095
P&U	$\mathbf{2^{10}}$	29	521	521	0	5.65	0.819	0.47	+0.349
Ligne1	2^{11}	24	411	392	4.84%	7.35	0.882	0.72	+0.162
Ligne2	2^{13}	26	712	712	0	6.71	0.872	0.48	+0.392

In Table 3.1, we can see that Rule 1 works well on two benchmark instances (i.e., Mini and P&U) as shown in column "Nb. S_n after Rule 1", as the enumerating space of S_n is respectively reduced 50% for the two instances (Note that there are in total 2^{n-1} individuals for each instance with given value of n.). In column "Best cycle times", our algorithm finds the same solutions as the optimal ones obtained with CPLEX (see "Our" and "CPLEX"), except for Ligne1. The standard deviation of the best cycle time obtained with our algorithm from the optimal cycle time obtained with CPLEX for Ligne1 is less than 5%, see sub-column "SD", which is computed as: SD=(Our−CPLEX)/CPLEX×100%. Although the CPU times spent by our algorithm are generally longer than those spent by CPLEX (see column "CPU times"), we can also see in column "Con.gen." that our algorithm finds the optimal solutions for most cases in very early generations (the spent time is given in sub-column "Con.time"). Note that the time gaps (i.e. sub-column "Gap") between Con. time and CPLEX are

very narrow, less than 1s. Due to this very small amount of gaps, the difference in CPU times between CPLEX and our algorithm is meaningless and can be negligible. In summary, our algorithm is an effective method for solving the benchmark instances in terms of solution quality and CPU times.

3.4.2 Experimental results on randomly generated instances

In this subsection, random instances are generated to further test the performance of the proposed algorithm. We compare our algorithm with the QEA with shifting decoding scheme to demonstrate the effectiveness of our decoding scheme. We also compare it with commercial software CPLEX and Tabu search (TS) algorithm (Yan *et al*., 2012). The random instances are generated as follows. We set n belongs to {10, 15, 18, 20, 22}, and let $U(c1, c2)$ be a uniform distribution between parameters $c1$ and $c2$. The random tests were set as two different groups. One (called Group1) was defined as the following way: the time windows were set as $L_i=U(30, 120)$ and $U_i=L_i+U(10, 750)$, $1\le i\le n$; the time of empty and loaded moves were respectively computed as the followings: $e_{i,\,i+1}=U(3, 6)$, $e_{i,\,j}=\sum_{k=i}^{j-1} e_{k,k+1}$, $0\le i, j\le n+1$, and $d_i=20+e_{i,\,i+1}$, $0\le i\le n$. The other (called Group2) was defined as the following: $L_i=U(40, 120)$, $U_i=30+U(1, 8)\times L_i$, for $1\le i\le n$, $e_{i,\,i+1}=U(2, 5)$, $e_{i,\,j}=\sum_{k=i}^{j-1} e_{k,k+1}$, for $0\le i, j\le n+1$, and $d_i=15+e_{i,\,i+1}$, for $0\le i\le n$. These defined parameters were based on the magnitude of the data from real production lines (Phillips and Unger, 1976; Manier, 1994). For each given n, five instances were randomly generated.

Table 3.2 reports the remaining number of S_n for each randomly generated instance after applying Rule 1. As mentioned before, there are in total 2^{n-1} individuals for each instance with a given value of n. As presented in Table 3.2, Rule 1 is efficient on 22 random instances (i.e. the numbers in bold font). We can also see in Table 3.2 that the enumerating space of S_n for each instance among the 22 instances is reduced at least 50% and at most 87.5% after applying Rule 1. Based on these results, Rule 1 seems efficient for the studied problem.

Firstly, we compare our algorithm with the QEA with shifting decoding scheme (i.e. SQEA). Table 3.3 presents the comparison results between our decoding scheme and shifting decoding scheme on Group1 and Group2. We can see that our decoding scheme generally outperforms than the shifting decoding scheme in terms of solution

quality and CPU times for all random instances. In particular, the deviations (i.e. AD) of our algorithm from that with shifting decoding generally decrease with the problem size. Besides, our algorithm spent less time than that with shifting decoding for all random instances.

Table 3.2 Results for the remaining number of S_n for each instance after applying Rule 1

n	Group1					Group2				
	1	2	3	4	5	1	2	3	4	5
10	2^9	$\mathbf{2^8}$	2^9	2^9	2^9	2^9	2^9	2^9	2^9	2^9
15	2^{14}	2^{14}	2^{14}	$\mathbf{2^{13}}$	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}
18	$\mathbf{2^{16}}$	2^{17}	2^{17}	$\mathbf{2^{16}}$	2^{17}	$\mathbf{2^{16}}$	2^{17}	2^{17}	$\mathbf{2^{16}}$	$\mathbf{2^{15}}$
20	$\mathbf{2^{18}}$	2^{19}	2^{19}	$\mathbf{2^{17}}$	$\mathbf{2^{18}}$	$\mathbf{2^{18}}$	$\mathbf{2^{18}}$	$\mathbf{2^{18}}$	$\mathbf{2^{18}}$	2^{19}
22	$\mathbf{2^{18}}$	2^{21}	$\mathbf{2^{20}}$	$\mathbf{2^{20}}$	$\mathbf{2^{18}}$	$\mathbf{2^{19}}$	$\mathbf{2^{20}}$	2^{21}	$\mathbf{2^{19}}$	$\mathbf{2^{19}}$

Table 3.3 Comparison results between our decoding scheme and shifting decoding scheme on Group1 and Group2

n	Group1					Group2				
	Average cycle times			Average CPU times		Average cycle times			Average CPU times	
	Our	SQEA	AD	Our	SQEA	Our	SQEA	AD	Our	SQEA
10	400.4	401.2	–0.20%	6.74	10.12	318.4	318.4	0	6.83	17.8
15	607.2	628	–3.31%	24.56	51.79	470.6	470.8	–0.04%	37.44	146.68
18	808.8	817.4	–1.05%	54.88	286.55	627.4	638.2	–1.69%	49.46	267.57
20	897.2	927.8	–3.30%	117.53	360.14	678.6	690.2	–1.68%	141.02	275.59
22	1058.6	1351.2	–21.65%	274.43	315.62	802.6	878.2	–8.61%	190.16	373.43

Secondly, Tables 3.4 and 3.5 reports the comparison results for randomly generated instances using our algorithm, Yan's algorithm (Yan *et al.*, 2012) and commercial software CPLEX. Columns AD^1 and AD^2 represent the standard deviation of our solution from those obtained with CPLEX and Yan's algorithm, respectively. They are computed as: AD^1=(Our−CPLEX)/CPLEX×100%, and AD^2= (Our−Yan)/Yan×100%). As presented in Tables 3.4 and 3.5, our algorithm and Yan's algorithm find the same solutions as the optimal ones obtained with CPLEX for random instances with n=10. For the remaining random instances, the average cycle times obtained with our algorithm are smaller than those obtained with Yan's algorithm. As a result, the deviations (i.e. AD^2) of our algorithm from Yan's algorithm are all negative, which range from −5.89% to −1.9% in Table 3.4 and from −3.93% to −0.42% in Table 3.5. Note that the smaller the AD^2, the better solution quality our algorithm obtained over Yan's algorithm. Therefore, our algorithm has a better

solution quality than Yan's algorithm. We also notice that CPLEX has a better solution quality than our algorithm and Yan's algorithm but it spent much longer CPU times, which will be discussed later. Moreover, the values of AD^1 in Table 3.4 and Table 3.5 both increase with the problem size, but are less than 4% and 3%, respectively, which are generally small and acceptable.

Table 3.4 Comparison results for the randomly generated instances Group1

n	Average cycle times					Average CPU times (In seconds)		
	Our	Yan	CPLEX	AD^1	AD^2	Our	Yan	CPLEX
10	400.4	400.4	400.4	0	0	6.74	2.7	1.44
15	607.2	624.6	602.4	0.8%	–2.79%	24.56	19.95	42.95
18	808.8	859.4	797.6	1.4%	–5.89%	54.88	32.16	1351.53
20	897.2	914.6	865.8	3.63%	–1.90%	117.53	114.51	1692.12
22	1058.6	1122.4	1025	3.28%	–5.68%	274.43	211.34	2712.38

Table 3.5 Comparison results for the randomly generated instances Group2

n	Average cycle times					Average CPU times (In seconds)		
	Our	Yan	CPLEX	AD^1	AD^2	Our	Yan	CPLEX
10	318.4	318.4	318.4	0	0	6.83	4.58	1.38
15	470.6	472.6	466.4	0.9%	–0.42%	37.44	62.35	51.50
18	627.4	636.4	612.6	2.42%	–1.41%	49.46	92.84	324.24
20	678.6	684	661.8	2.54%	–0.79%	141.02	53.52	1077.9
22	802.6	835.4	779.8	2.92%	–3.93%	190.16	102.62	1897.76

For the average CPU times, we can see from Tables 3.4 and 3.5 that both our algorithm and Yan's algorithm performs much better than CPLEX for each value of *n*, except for *n*=10. We also notice that Yan's algorithm has a better performance than our algorithm in terms of CPU times except for *n*=15 and *n*=18 in Table 3.5. But their gaps are not so large. Moreover, although the CPU times spent by the three approaches generally increase with the instance size *n*, the CPU times spent by CPLEX generally have a very sharper growth than those spent by our algorithm and Yan's algorithm, especially for large-size instances. From these results, we can see that our algorithm has a better computational performance than CPLEX.

3.5 Summary

This chapter proposed a hybrid QEA with improved decoding scheme to solve a single-hoist cyclic scheduling problem with processing time windows. In particular,

three different decoding procedures were proposed to convert Q-bit individual into robot move sequences. A repairing procedure was designed to repair the infeasible sequences. Both Q-gate and adaptive genetic operators as variant operators were applied to evolve the population. The effectiveness of the proposed algorithm were demonstrated by solving benchmark instances and randomly generated instances compared with commercial software CPLEX and Yan's algorithm. Experimental results indicate that our decoding scheme outperforms the shifting decoding scheme, and the proposed algorithm can provide high-quality solutions within a reasonable time. The results also imply that the proposed algorithm generally has a shorter computation time than CPLEX, especially for large-size instances, and has a better solution quality than Yan's algorithm.

Chapter 4 Bi-objective QEA with Local Search Procedure for HSP with Simultaneous Productivity Maximization and Production Cost Minimization

4.1 Introduction

In practice, electroplating plant is huge resource (such as electricity and freshwater) consumer due to its specific processing technology. For instance, part may be firstly immersed into an electrolytic degreasing tank containing certain volume of concentrated acids and alkalis solutions at required temperatures, for removing dust and grease from its surface, and then put into a rinsing tank containing certain volume of freshwater for cleaning possible chemical residue on its surface. Obviously, the amount of consumed electricity and freshwater mainly depends on the soaking duration (i.e. actual processing time). In other words, increased soaking durations in processing tanks generally give rise to the resource consumption, resulting in higher production cost.

On the other side, electroplating plant also generates plenty of toxic waste daily, such as sludge and wastewater from treatment, and used acids and other chemicals. Generally, the less resource spent during the process, the less waste generated by electroplating plant. Concerning the environmental pollution as well as the shortage of freshwater and electricity, most countries such as France and China enact legislation to regulate the amount of freshwater and electricity consumed and pollutant emissions daily in electroplating industry. Note that the governments not only severely punish the electroplating plants discharging heavy pollution to the environment, but also charge higher prices of electricity and freshwater for industrial usage. Viewed from these aspects, optimal HSP with production cost minimization has great significance from both theoretical and practical perspectives. It implies more benefits while minimizing the amount of freshwater, electricity and chemicals used, then while limiting the associated costs as well as the pollutant emission and effluent treatment. So scheduling such facilities enhances with both the economic and environmental pillars which are the basis of the sustainable strategy deployed in many industries, due to the double pressure of concurrency and legislation.

In the past decades, a number of efficient scheduling approaches, such as B&B

algorithm (Shapiro and Nuttle, 1988; Ng, 1995; Chen *et al*., 1998; Manier *et al*., 2000; Che and Chu, 2004; Che *et al*., 2011; Lei *et al*., 2014), MIP approach (Phillips and Unger, 1976; Liu *et al*., 2002; Leung *et al*., 2004; Zhou *et al*., 2012), and heuristics or meta-heuristics (Lei and Wang, 1991; Baptiste *et al*., 1993; Zhou and Liu, 2008; Zhang *et al*., 2014), have been suggested for various variants of HSP with productivity maximization (i.e. cycle time or makespan minimization). To reduce the problem complexity, some researchers, such as Kuntay *et al.* (2006) and Subaï *et al.* (2006), proposed various two-step sequential scheduling approaches for bi-objective HSP, where cycle time and wastewater or production cost are minimized. Obviously, such sequential approaches are not sufficient to find the complete Pareto-optimal solutions for the multi-objective HSP.

It is understandable that a hoist schedule is a key factor for improving the productivity. Typically, the more frequently the hoist picks a part from the input station, the higher the productivity. Moreover, efficient hoist scheduling can also plays an important role in decreasing the production cost, since it is inherently determined by the actual processing times, which also affect the production cost. So maximizing the productivity may conflict with minimizing the production cost. This creates the trade-off between the two objectives, since that is hard to determine whether one solution is better than another if it is better on the productivity but is worse on the production cost. Therefore, there is a set of Pareto-optimal solutions for multi-objective optimization problem (MOP), instead of a single optimal one (Miettinen, 1999).

To overcome the solution evaluation issue of MOP, several approaches have been suggested, such as Pareto-dominance (PD) approach, objective aggregation (OA) approach and lexicographic ordering (LO) approach. The PD approach is the most commonly used approach. It is mainly based on the concepts of Pareto-dominance and crowding-distance to evaluate solutions. It has been shown that PD approach is very efficient in optimizing bi-objective or three-objective optimization problems. Besides, by assigning weight to each objective and then summing up all objectives, the OA approach transforms multiple objectives into a single objective. Since determining suitable weight for different objectives plays an important role in the success of this approach, it is not sufficient in practice. In addition to OA approach, some researchers suggested LO approach for MOP. All objectives are sorted based on their importance and optimized alternately. It is also difficult to give orders to different objectives.

As mentioned above, no research has been reported on HSP with simultaneously maximizing productivity and minimizing production cost. Therefore, in this chapter, we study the cyclic HSP with the above mentioned dual objectives. In order to find a set of Pareto-optimal solutions, an efficient QEA with local search procedure is designed for the studied problem. By adopting the well-known concepts of Pareto dominance and crowding distance, the proposed algorithm can optimize the two objectives effectively and simultaneously, and can obtain a set of Pareto-optimal solutions for the problem in very short time. To guide the search direction and generate the offspring population, a chaotic quantum-rotation gate is proposed. For increasing the individual diversity, mutation operator is implanted into the proposed algorithm. As usual, an external archive is used to store the obtained non-dominated solutions, and it is updated at each generation.

The rest of this chapter is arranged as follows. In Section 4.2, we present the problem description and its formulation. Some concepts about the multi-objective optimization problem (MOP) and the Pareto-optimal solutions are given in Section 4.3. Section 4.4 details the proposed bi-objective QEA. The experimental results are given in Section 4.5. Section 4.6 gives some conclusions.

4.2 Problem description and its formulation

4.2.1 Sequence-based bi-objective mathematical model

In this chapter, the studied problem is similar to that in Chapter 3, except for the problem objective. More precisely, two conflicting objectives (i.e., minimization of production cost and maximization of productivity, which equivalents to minimize the cycle time C) are simultaneously considered in this chapter, instead of a single one. The objective “production cost” represents the sum costs of the resource consumed in all processing tanks per cycle. To avoid introducing the problem repeatedly, the problem description is omitted here. Then according to the notation in Manier and Bloch (2003), the studied problem can be written in the following form:

$$CHSP \mid n // diss \mid /n+2 \mid (Cmin, Production\ Cost\ min)$$

In the following, the same notations and variables defined in Chapter 3 are used in this chapter. To facilitate the problem formulation, we assume that the cost of resource consumption in each tank is proportional to the processing times in it. Therefore, the following notation (i.e. w_i) and decision variable (i.e. p_i) are defined:

w_i: the cost of resource consumed per time unit in tank M_i, $1 \le i \le n$. For simplicity, we define $W = (w_1, w_2, w_3 \ldots w_n)$, which will be given by each specific instance.

p_i: the actual processing or soaking time in tank M_i, $1 \le i \le n$. For simplicity, we define $P = (p_1, p_2, p_3 \ldots p_n)$. Furthermore, from constraint (3.2) formulated in Chapter 3, we can know that $p_i = Cs_i + t_i - (t_{i-1} + d_{i-1})$, for $1 \le i \le n$.

Based on the above descriptions and notations, the bi-objective mathematical model for the studied problem can be formulated as:

$$\text{Min } f_1 = C,$$

$$\text{Min } f_2 = \sum_{i=1}^{n} w_i p_i ,$$

subject to (3.1) –(3.4).

In above model, the first objective (i.e. f_1) is set to minimize the cycle time C, which equivalents to maximize the productivity, and the second objective (i.e. f_2) is set to minimize the total production cost of all processing tanks per cycle. As reported in Chapter 3, if a hoist move sequence H satisfies the constraints (3.1)–(3.4), then it is a feasible schedule for HSP with only minimizing the cycle time (i.e. f_1 in this chapter). On the other side, as all values of decision variables (i.e., t_i, C, s_i) can be obtained from a feasible sequence H, the value of P can be easily calculated. In other words, as W is known in advance, the value of the second objective (i.e. f_2) can be easily deduced from a feasible hoist move sequence H, which is a solution for the HSP with only minimizing the cycle time.

From above point of view, it seems that the HQEA proposed in Chapter 3 is also suitable for solving the bi-objective HSP considered in this chapter. But it is not in fact. The reason is two-fold. The first one is that as the value of production cost (denoted by $f_2(C_1)$) obtained from a shorter cycle time (denoted by C_1) may be greater than that (denoted by $f_2(C_2)$) from a longer cycle time (denoted by C_2), i.e., $C_1 < C_2$ and $f_2(C_1) > f_2(C_2)$, it is difficult to say that solution (C_1, $f_2(C_1)$) is better or worse than (C_2, $f_2(C_2)$). For this reason, the fitness evaluation function proposed in HQEA is no longer suitable for bi-objective HSP. The second one is that the feasibility checking procedure used in HQEA only returns the minimum cycle time for a feasible hoist move sequence. It is understandable that a feasible hoist move sequence may have several different cycle times, which consequently may result in different production costs. In other words, a feasible hoist move sequence may generate multiple different

solutions (note that one solution represents a pair of values respectively for f_1 and f_2) for bi-objective HSP. Obviously, the HQEA proposed in Chapter 3 has one main shortcoming in obtaining the Pareto-optimal solutions for bi-objective HSP, i.e., it only returns one feasible solution and inherently drops other potential ones for a feasible hoist move sequence. Based on above the descriptions, a new scheduling approach needs to be developed for bi-objective HSP in this chapter.

4.2.2 Modified bi-objective mathematical model

Inspired by the previous descriptions, we can know that the bi-objective HSP can be reduced to the single-objective HSP (i.e. minimize the cycle time C) if P is given. It should be noted that Levner *et al*. (1997) proposed a method of prohibited intervals (MPI) to formulate the HSP with fixed processing times (i.e., P is given in advance), and developed an efficient polynomial procedure (called Levner's procedure hereafter) to find the optimal cycle time C for their studied problem. The complexity of Levner's procedure is $O(n^3 \log n)$, where n is the number of processing tanks. Inspired by their work, we can use the MPI approach to reformulate our bi-objective optimization problem, and then apply the associated polynomial procedure to obtain the values of cycle time and production cost providing that P can be determined in advance. Similarly to Levner *et al*. (1997), Yan *et al*. (2010), and Wang and Che (2013), the new mathematical model for the studied bi-objective problem providing that P is given can be reformulated as follows:

$$\text{Min } f_1(P)=C,$$

$$\text{Min } f_2(P)=\sum_{i=1}^{n} w_i p_i ,$$

subject to:

$$Z_i=\sum_{j=1}^{i}(d_{j-1}+p_j), \quad \text{for } 1\le i\le n. \tag{4.1}$$

$$C \notin V \equiv \bigcup_{i=1}^{n}(-\infty, Z_i - Z_{i-1} + d_i + e_{i+1,i-1}). \tag{4.2}$$

$$C \notin I \equiv \bigcup_{i=1}^{n}\bigcup_{j=0}^{i-1}\bigcup_{k=1}^{i-j}((Z_i - Z_j - d_j - e_{j+1,i})/k, (Z_i - Z_j + d_i + e_{i+1,j})/k). \tag{4.3}$$

$$L_i \le p_i \le U_i, \quad \text{for } 1\le i\le n. \tag{4.4}$$

In constraint (4.1), Z_i represents the start time of move i of part 0 (suppose that it entered the line at time 0) from M_i, $1 \le i \le n$, i.e., the completion time of part 0's i^{th} processing operation. Moreover, Z_i+mC represents the start time of move i of part m (note that it is introduced into the m^{th} cycle at time mC, as only one part can enter the line within each cycle) from tank M_i, $0 \le i \le n$, and $Z_0=0$. Constraints (4.2) and (4.3) impose a series of prohibited intervals for cycle time C. In particular, if the value (denoted by C') of cycle time falls within the prohibited intervals V (i.e., $C' \in V$) in (4.2), then at least one conflict happens in the use of a same tank by different parts at the same time. Thus, C' is an infeasible solution for the problem since each tank cannot process more than one part at any time. Similarly, if C' belongs to prohibited intervals defined in (4.3) (i.e., $C' \in I$), then C' is also infeasible for the problem since two consecutive moves conflict in the use of the hoist. At last, constraint (4.4) ensures that the processing time window constraints are satisfied.

4.3 Basic concepts of MOP and Pareto-optimal solutions

Multi-objective optimization problem (MOP) is often encountered in many real-world applications. In practice, it involves optimizing at least two objectives simultaneously, which are usually conflicting with each other, i.e., an improvement on one objective may give declination to some others. Due to this reason, MOP is more complex than the single-objective optimization problem. Suppose an optimization problem with minimization of two objectives, which can be expressed as follows:

$$Min \quad F(x) = [f_1(x), f_2(x)],$$

$$\text{s.t. } x \in X.$$

In above definition, $f_i(x)$ is the problem objective, $1 \le i \le 2$; x denotes the decision variables vector; X represents the solution space or the constraints of MOP. Generally, there are multiple optimal solutions for MOP, instead of a single one. They are usually called as Pareto-optimal or non-dominated solutions, which are defined by the Pareto dominance concept. It is explained as follows. For any two solutions $x_1 \in X$ and $x_2 \in X$, if we have $f_1(x_1) \le f_1(x_2)$ and $f_2(x_1) < f_2(x_2)$, or $f_1(x_1) < f_1(x_2)$ and $f_2(x_1) \le f_2(x_2)$, then we say that solution x_1 dominates solution x_2. If a solution x^* is not dominated by any other solutions, then x^* is called non-dominated (i.e. Pareto-optimal) solution. Moreover, the Pareto front (PF) is defined as: $PF=\{F(x) | x \in \Omega\}$, in which Ω denotes the set of

non-dominated solutions. For more details about the MOP, please see the works by Miettinen (1999) and Deb (2001).

4.4 Solution method

In this section, we develop an efficient bi-objective QEA with local search procedure to find a set of Pareto-optimal solutions for the studied problem. Figure 4.1 depicts the main flowchart of our proposed algorithm. We can see from Figure 4.1 that the proposed algorithm includes the encoding and decoding scheme, the individual evaluation procedure based on the Pareto-dominance technique, the chaotic quantum-rotation gate, the mutation operator, the external archive updating mechanism and the local search procedure. The algorithm stops when the maximal number of iterations (i.e. maxgen) is reached. As mentioned above, our bi-objective problem can be solved by Levner's procedure on condition that P can be known. In what follows, we first present how to obtain P with the proposed encoding and decoding scheme and then introduce other components of the algorithm in details.

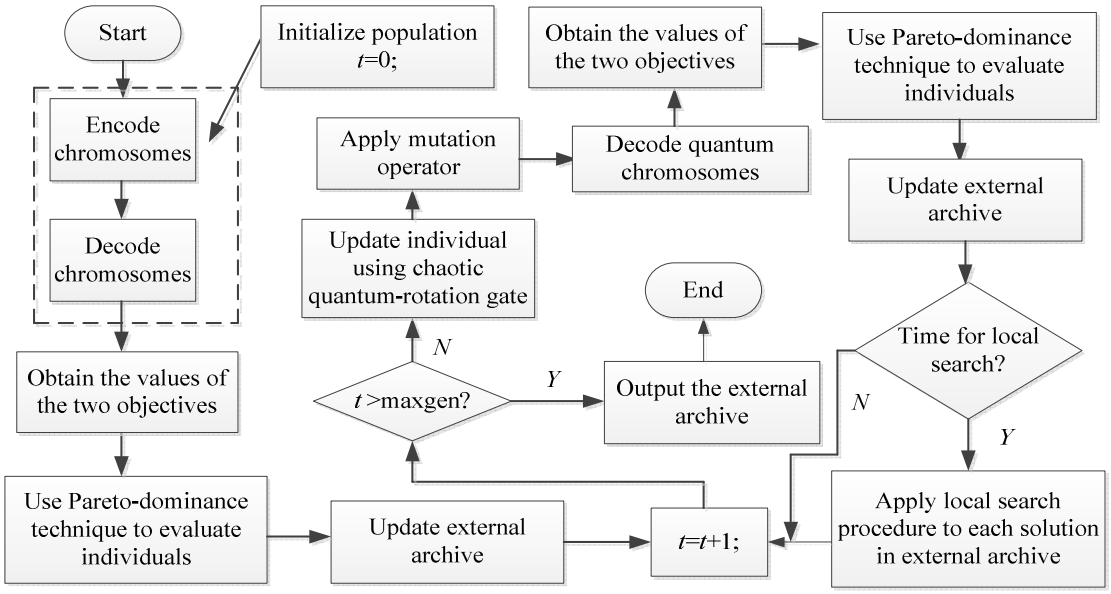

Figure 4.1 The main flowchart of the proposed bi-objective QEA.

4.4.1 Encoding and decoding scheme

As there are n processing operations, each chromosome is encoded as a string consisting of n Q-bits, which are defined as follows:

$$\Psi_n = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \beta_1 & \beta_2 & \cdots & \beta_n \end{bmatrix}, \ 1 \le i \le n. \tag{4.5}$$

where $|\alpha_i|^2+|\beta_i|^2=1$. Since we need to know the value of p_i, $1 \le i \le n$, and it must fall within its corresponding time windows $[L_i, U_i]$, the following two decoding schemes are used to transform each quantum chromosome (i.e. (4.5)) into the actual processing time P(Li and Li, 2008):

$$p_i = 0.5 \times (U_i + L_i + (U_i - L_i) \times \alpha_i), \ \text{for } 1 \le i \le n. \tag{4.6}$$

$$p_i = 0.5 \times (U_i + L_i + (U_i - L_i) \times \beta_i), \ \text{for } 1 \le i \le n. \tag{4.7}$$

In (4.6) and (4.7), we define $\alpha_i=cos(\sigma_i)$, $\beta_i=sin(\sigma_i)$, and $\sigma_i=2\pi \times rd$, where π=3.1415926 and rd is randomly generated between 0 and 1. From this definition, we can see that α_i and β_i fall within the range [−1, 1]. Consequently, each generated processing time p_i is limited by its corresponding lower and upper bounds $[L_i, U_i]$. Therefore, processing time window constraints are ensured. Note that for each quantum chromosome, it is decoded by both (4.6) and (4.7). In other words, two different solutions (such as P and P') are generated from each quantum chromosome. For this reason, such an encoding and decoding scheme can provide a better diversity of population.

4.4.2 Individual evaluation

After the chromosomes decoding, the objective values of each individual can be obtained with Levner's procedure. Thereafter, individual evaluation is an important issue for the studied problem. To fix this issue, the Pareto-dominance approach is adopted to evaluate all individuals. According to Deb *et al*. (2002), the population is first classified into K different frontiers (F_1, F_2, F_3,…, F_K) based on the dominance relationship by a fast sorting procedure. Note that F_1 includes all the non-dominated solutions obtained in each generation. After that, distance metrics are assigned to individuals by a crowing distance computing procedure. In what follows, we first describe the fast non-dominated sorting procedure and then the crowing distance computing procedure, which can be found in Deb *et al*. (2002). To facilitate the descriptions, we let nd_P denote the number of solutions which dominate solution P, and Ω_P denote the set of solutions which are dominated by solution P.

(a)The fast non-dominated sorting procedure:

Step(*I*): *For each solution P, first set* nd_P=0 *and* $\Omega_P = \varnothing$; *then determine* nd_P *and*

Ω_x.

Step(*II*): *For any solution P with* nd_P =0, *first put it into the first frontier* F_1, *and set its rank number to be* 1, *i.e.,* $Rank_P$=1; then *set k*=1.

Step(*III*): *If* $F_k \neq \varnothing$, *then set Q*=∅; *else*, *go to Step*(*VI*).

Step(*IV*): *For* $\forall x \in F_k$, *set* $nd_q = nd_q - 1$ *for* $q \in \Omega_P$; *if* nd_q=0, *put solution q into Q*.

Step(*V*): *Let k*=*k*+1 *and* F_k=*Q*; *For* $\forall q \in F_k$, *set* $Rank_q$=*k*. *And go to Step*(*III*).

Step(*VI*): *Let K*=*k*−1; *End*.

(b)The crowding distances calculation procedure:

Step(*I*): *Order the population according to each objective value in increasing order; for each objective, set infinite distance value (denoted by M) for both the smallest and largest solutions* (*boundary solutions*).

Step(*II*): *For objective i*($i \in \{1,2\}$), *the distance* $Dis_i(P_j)$ *of each non-boundary solution* P_j *is calculated based on the absolute normalized difference in the objective values of two neighbor solutions by the following equation*:

$$Dis_i(P_j) = (f_i(P_{j+1}) - f_i(P_{j-1}))/(f_i^{\max} - f_i^{\min}) \tag{4.8}$$

Step(*III*): *For each solution* P_j, *its overall crowding distance* $CD(P_j)$ *is calculated as the sum of the distance value for all objectives. This is expressed as follows*:

$$CD(P_j) = \sum_{i=1}^{G} Dis_i(P_j) \tag{4.9}$$

where *G* represents the total number of considered objectives. Figure 4.2 illustrates an example of an optimization problem with dual objectives minimization. In Figure 4.2(*a*), the population is divided into 3 frontiers (i.e., F_1, F_2, F_3) by the above described fast non-dominated sorting procedure. Note that F_1 represents the set of all non-dominated solutions (denoted by •), which dominate those in F_2, and solutions in F_2 dominate those in F_3. Moreover, Figure 4.2(*b*) depicts the crowding-distance calculation process of solution P_j. As can be seen from Figure 4.2(*b*), P_1 and P_D denote the two boundary solutions.

After using above two described procedures, each solution *P* has two attributes:

Non-domination rank ($Rank_P$) and crowding distance ($CD(P)$). For any two solutions P and P', if $Rank_P<Rank_{P'}$, then we say that solution P is *better* than solution P', because the former dominates the latter. For solutions with same rank (i.e. $Rank_P=Rank_{P'}$), if $CD(P)>CD(P')$, then we say that solution P is *better* than solution P', because P is located in a lesser crowded area, and it improves the population diversity.

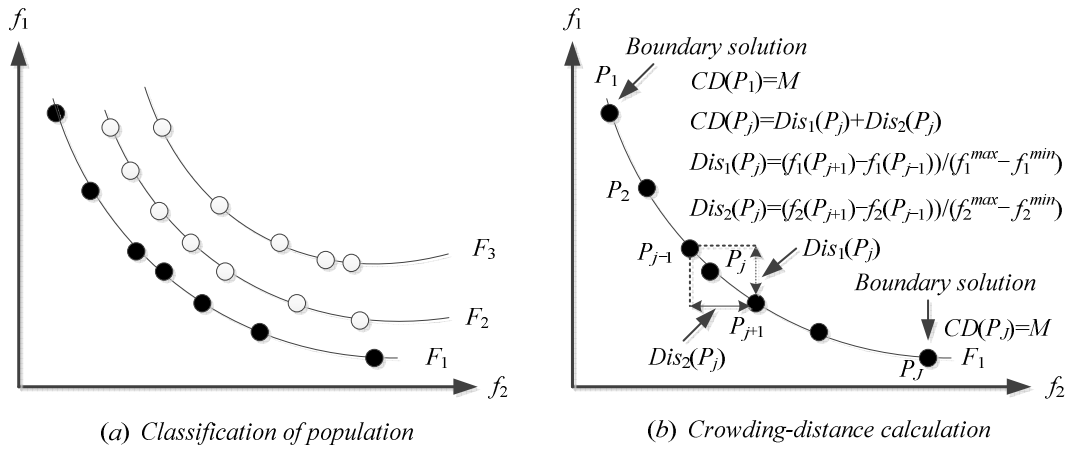

Figure 4.2 Classification of the population (a) and Crowding-distance calculation (b).

4.4.3 Chaotic quantum-rotation gate

In this chapter, for generating new offspring, quantum-rotation gate is adopted to update each Q-bits chromosome. For a Q-bits chromosome Y, its Q-bit i can be updated as follows (Han and Kim, 2002; Li and Wang, 2007):

$$\begin{aligned}\alpha_i^{'} &= \cos(\Delta\omega_i)\times\alpha_i - \sin(\Delta\omega_i)\times\beta_i \\ \beta_i^{'} &= \sin(\Delta\omega_i)\times\alpha_i + \cos(\Delta\omega_i)\times\beta_i\end{aligned}. \tag{4.10}$$

In (4.10), $\Delta\omega_i$ represent the rotation angle, which plays an important role in updating Q-bits chromosome. Generally, the value of $\Delta\omega_i$ is determined by an intuitive reasoning way (Han and Kim, 2002; Li and Wang, 2007). In this section, we propose a different way to determine suitable rotation angle for updating each Q-bit. Firstly, for driving the search direction towards Pareto-optimal solutions, we randomly choose a non-dominated solution P (note that $P=(p_1, p_2, p_3 \ldots p_n)$) from

external archive to guide the updating process of chromosome Y. Then, we assume that each actual processing time p_i of P corresponds to a probability amplitude γ_i of a Q-bit m with $\gamma_i=cos(\eta_i)$. Note that γ_i can be deduced by (4.6) or (4.7), and then η_i can also be known. For ease of description, we let $\varphi=\eta_i-\sigma_i$, where $\alpha_i=cos(\sigma_i)$. From this, we can know that the gap (i.e. φ) between η_i and σ_i can be used as the rotation angle to update Q-bit i. But this may reduce the diversity of Q-bits chromosome, and the solutions may fall into local optimal. For this reason, chaotic sequence is used in the updating process of each Q-bit due to its good ergodicity and regularity. It is produced by the logistic map, which is usually defined as follows (Dettmer, 1993):

$$\mu_g = 4\times\mu_{g-1}\times(1-\mu_{g-1}),\ 1<g. \tag{4.11}$$

where μ_g is generated at generation g. Note that μ_0 is randomly generated from (0, 1) at the initial generation. Finally, we propose a chaotic quantum-rotation gate to update each Q-bits chromosome, i.e., the rotation angle is mainly determined by μ_g and φ. In the following, we explain how to choose the rotation angle according to eight different cases, which are illustrated in Figure 4.3 (case(*I*)–case(*IV*)) and Figure 4.4(case(*V*)–case(*VIII*)). Note that in the two figures, the curved arrow represents our proposed rotation direction for Q-bit i.

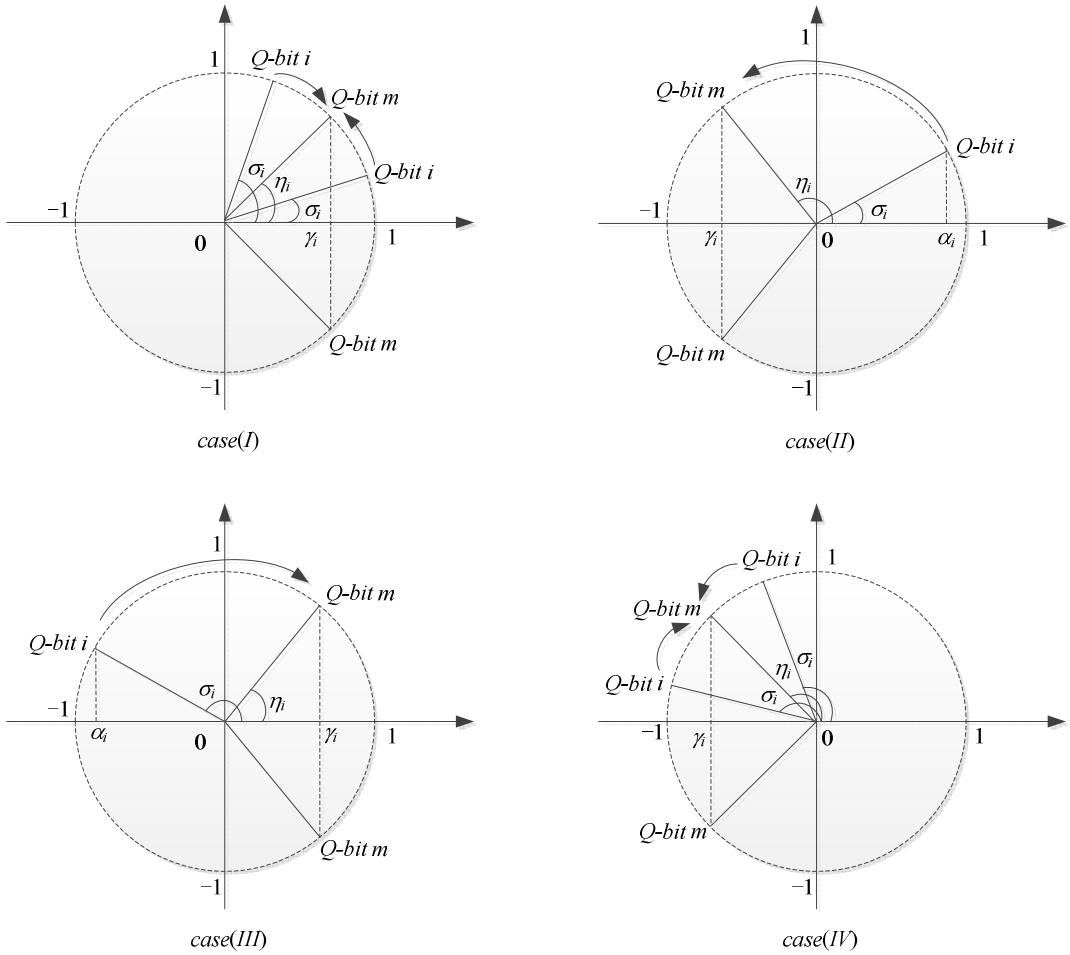

Figure 4.3 The updating processes for Q-bit *i* in the 1st and 2nd quadrants.

If Q-bit *i* is located in the first quadrant, then consider the following cases:

Case (*I*): For $\gamma_i \geq 0$, as case (*I*) illustrated in Figure 4.3, to simplify the updating process, if $1.5\pi < \eta_i \leq 2\pi$ (i.e., Q-bit *m* is in the fourth quadrant), then we let $\eta_i = 2\pi - \eta_i$ (i.e., let Q-bit *m* in the first quadrant). After that, we set $\Delta\omega_i = \mu_g \times \varphi$ ($\varphi = \eta_i - \sigma_i$), which implies that the value of $\Delta\omega_i$ is positive if $\varphi > 0$ and negative if $\varphi < 0$. This makes Q-bit *i* closer to Q-bit *m*. Moreover, if $\varphi = 0$, both small negative and positive values are acceptable for $\Delta\omega_i$, so as to search the neighborhood area.

Case (*II*): For $\gamma_i < 0$, as illustrated in Figure 4.3, we know that Q-bit *m* is located either in the second or the third quadrant, so the value of $\Delta\omega_i$ is set to be $0.5\pi \times \mu_g$, which is a relatively "big jump" to drive Q-bit *i* towards the location area of Q-bit *m*.

If Q-bit *i* is located in the second quadrant, then consider the following cases:

Case (*III*): For $\gamma_i \geq 0$, we set $\Delta\omega_i = (-0.5\pi) \times \mu_g$, in order to drive Q-bit *i* towards the location area of Q-bit *m*.

Case (*IV*): For $\gamma_i<0$, we first let $\eta_i = 2\pi - \eta_i$ if $\pi < \eta_i \le 1.5\pi$, and then set $\Delta\omega_i = \mu_g \times \varphi$. It implies that the value of $\Delta\omega_i$ is positive if $\varphi>0$ and negative if $\varphi<0$, which makes Q-bit *i* closer to Q-bit *m*. Moreover, if $\varphi=0$, both small negative and positive values are acceptable for $\Delta\omega_i$, so as to search the neighborhood space.

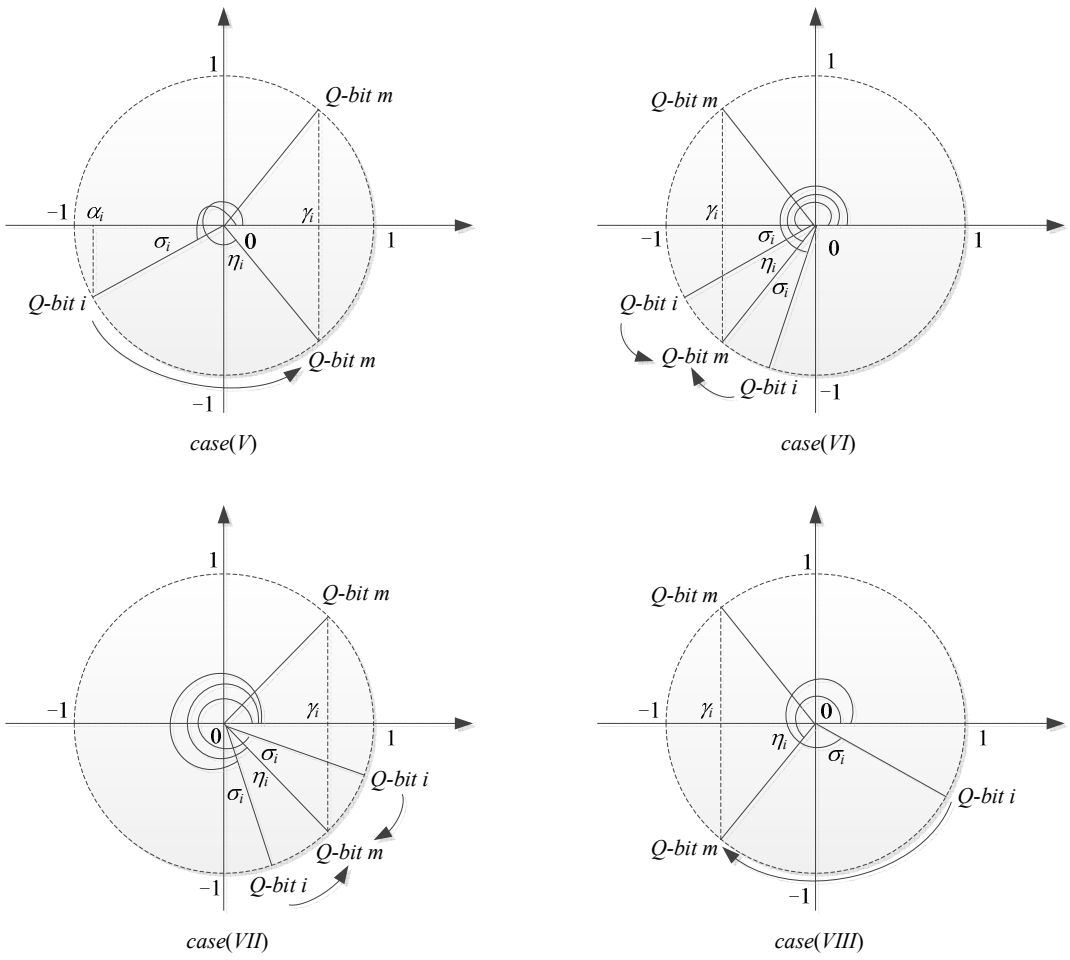

Figure 4.4 The updating processes for Q-bit *i* in the 3rd and 4th quadrants.

Furthermore, similar analyses have been performed for Q-bit *i* in the third and fourth quadrants, i.e., case(*V*)–case(*VIII*) shown in Figure 4.4. Based on the above analysis, Table 4.1 presents the lookup table for choosing suitable rotation angle to update Q-bits chromosome. By using the above described chaotic quantum-rotation gate, different rotation angle is determined for different cases. Consequently, each chromosome has an evolutionary diversification, and it is updated towards the non-dominated solution space by a diverse way.

Table 4.1 Lookup table of rotation angle

	$\gamma_i \geq 0,\ \varphi = \eta_i - \sigma_i$	$\gamma_i < 0,\ \varphi = \eta_i - \sigma_i$
$\alpha_i > 0,\ \beta_i \geq 0$	If $\varphi \neq 0$, $\Delta\omega_i = \mu_g \times \varphi$; else, $\Delta\omega_i = \pm 0.008\pi$;	$\Delta\omega_i = 0.5\pi \times \mu_g$;
$\alpha_i \leq 0,\ \beta_i > 0$	$\Delta\omega_i = (-0.5\pi) \times \mu_g$;	If $\varphi \neq 0$, $\Delta\omega_i = \mu_g \times \varphi$; else, $\Delta\omega_i = \pm 0.008\pi$;
$\alpha_i < 0,\ \beta_i \leq 0$	$\Delta\omega_i = 0.5\pi \times \mu_g$;	If $\varphi \neq 0$, $\Delta\omega_i = \mu_g \times \varphi$; else, $\Delta\omega_i = \pm 0.008\pi$;
$\alpha_i \geq 0,\ \beta_i < 0$	If $\varphi \neq 0$, $\Delta\omega_i = \mu_g \times \varphi$; else, $\Delta\omega_i = \pm 0.008\pi$;	$\Delta\omega_i = (-0.5\pi) \times \mu_g$;

4.4.4 Mutation operator

Although the proposed decoding scheme and updating scheme has a strong ability to provide a better diversity of population, it still has some room to increase the population diversity, so as to prevent the algorithm falling into local optimal as far as possible. Thus, mutation is applied to each chosen chromosome according to the mutation rate. More precisely, two positions x and y are randomly generated for each chosen chromosome, $1< x, y<n$. For each Q-bit i between positions x and y, we swap the values of α_i and β_i. If x equals to y, then just swap the values of α_x and β_x.

4.4.5 Updating external archive

The external archive (EA) is initialized to be empty. It is updated at each generation. For simplicity, let ND_{g-1} be the set of non-dominated solutions stored in EA updated at generation $g-1$and F_1 be the set of non-dominated solutions obtained at generation g. We first let $ND_g=ND_{g-1}\cup F_1$, and then calculate the crowding-distance for each solution in ND_g. For any two solutions $P1$ and $P2$ in ND_g, consider the following: (a) if $P1$ is the same as $P2$ (i.e., $f_1(P1)=f_1(P2)$ and $f_2(P1)=f_2(P2)$), then remove one of them from ND_g; (b) if $P1$ dominates $P2$, then remove $P2$ from ND_g and vice versa. If the size of ND_g exceeds the pre-defined maximum size, then we remove the individual with the smallest crowding distance from ND_g until the size equals to the maximum size. Finally, EA is updated and ND_g contains the final non-dominated solutions. The above described updating process is depicted in Figure 4.5.

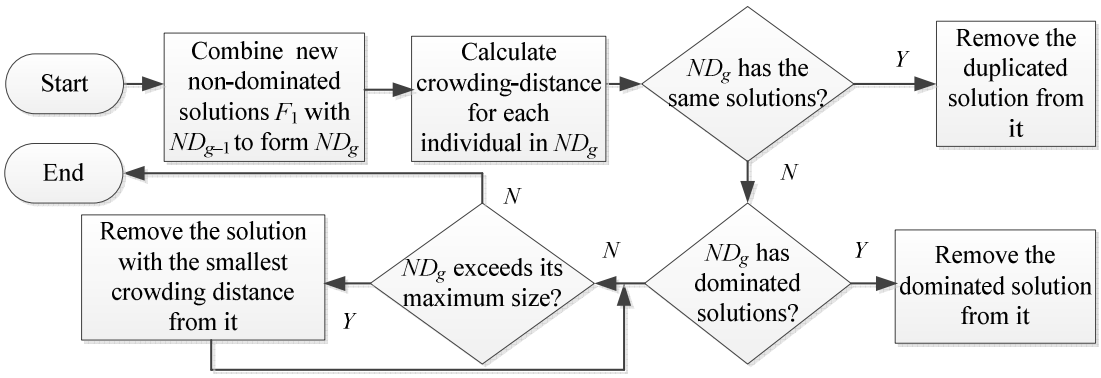

Figure 4.5 The process of updating external archive.

4.4.6 Local search (LS) procedure

As mentioned above, as soon as the actual processing time P (note that P= (p_1, p_2, p_3…p_n)) is determined, Levner's procedure can be applied to find its corresponding optimal cycle time C_b (i.e. C_b= $f_1(P)$). After that, the associated hoist move sequence H and value of the production cost (i.e. $f_2(P)$) can be known for P. Due to the special characteristic of hoist scheduling problem, it is understandable that a feasible hoist move sequence H may has several different feasible cycle times, which are denoted by {C_1, C_2, C_3, …, C_m}, corresponding to diverse processing times for each tank. Obviously, the optimal cycle time C_b for P obtained with Levner's procedure is one of the cycle times {C_1, C_2, C_3, …, C_m} related to H. This implies that there probably exists a better cycle time in {C_1, C_2, C_3, …, C_m} than C_b for H. Besides, it should be noted that different feasible hoist move sequences may have the same cycle time C.

For the above reasons, a local search (LS) procedure is needed for H so as to further search other possibly better cycle times related to it. To save the computation time, LS procedure is applied to the non-dominated individuals from External Archive at every χ generation, where χ is a parameter to be set in the experimental section. Due to its high efficiency in finding the best cycle time for each given H (Wang and Che, 2013), in this chapter, the graph-based polynomial procedure proposed by Chen *et al*. (1998) is used as the LS procedure to find the optimal cycle time C^* for each H (it corresponds to a non-dominated solution P with objective values ($f_1(P)$, $f_2(P)$)) in External Archive. Thereafter, the new processing times spent in all tanks (i.e. P') can be determined according to the newly found C^*(C^*=$f_1(P')$), and the value (i.e. $f_2(P')$) of the second objective can be calculated for H according to P'. As a result, a new solution P' with objective values (C^*, $f_2(P')$) for H is obtained with our LS procedure.

At last, we update the External Archive with the newly found solutions. The above described LS procedure is depicted in Figure 4.6.

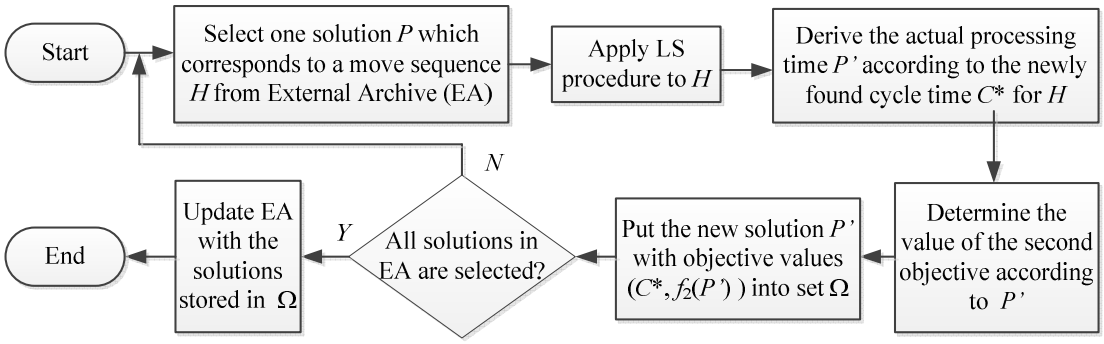

Figure 4.6 The process of the proposed LS procedure.

To better understand our above observation, Figures 4.7~4.8 illustrate two different feasible cycle times with the same hoist move sequence for a HSP example. The data for the example is given in Table 4.2, which was generated via our experiment. Note that the travel times of empty hoist moves for the presented move sequence are given as: $e_{1,5}$=12s, $e_{6,3}$=9s, $e_{4,2}$= 5s, $e_{3,1}$=7s, $e_{2,4}$=5s, $e_{5,0}$=16s. As illustrated in Figures 4.7~4.8, M_1~ M_5 are processing tanks, M_0 and M_6 are input station and output station, respectively. The hoist move sequences illustrated in the two figures are the same, i.e., 0–5–3–2–1–4. But the cycle times given in the two figures are different, i.e., C=170s and C=220s, which are all feasible ones. To our knowledge, the value C=170s given in Figure 4.7 is the optimal cycle time for the given example. Note that the numbers around an inclined solid arrow (resp. a broken arrow) in Figures 4.7 and 4.8 represent the start and end times of a loaded move (resp. an empty move). Moreover, we can derive the actual processing times P= (90s, 124s, 128s, 56s, 48s) from Figure 4.7 and P= (140s, 174s, 137s, 97s, 48s) from Figure 4.8. From these values, we can see that two different actual processing times are given by the same hoist move sequence for each tank except M_5.

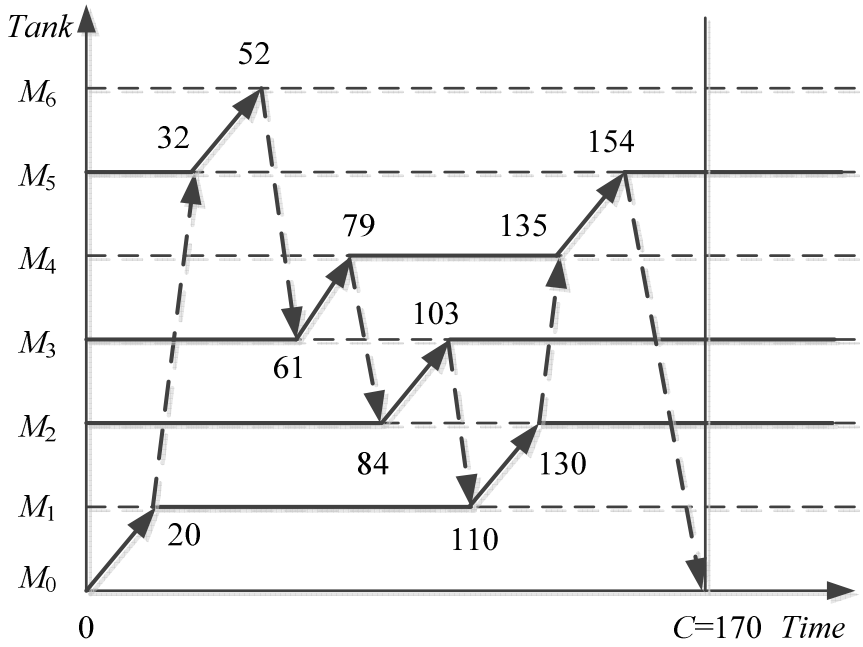

Figure 4.7 Hoist move sequence 0–5–3–2–1–4 with *C*=170.

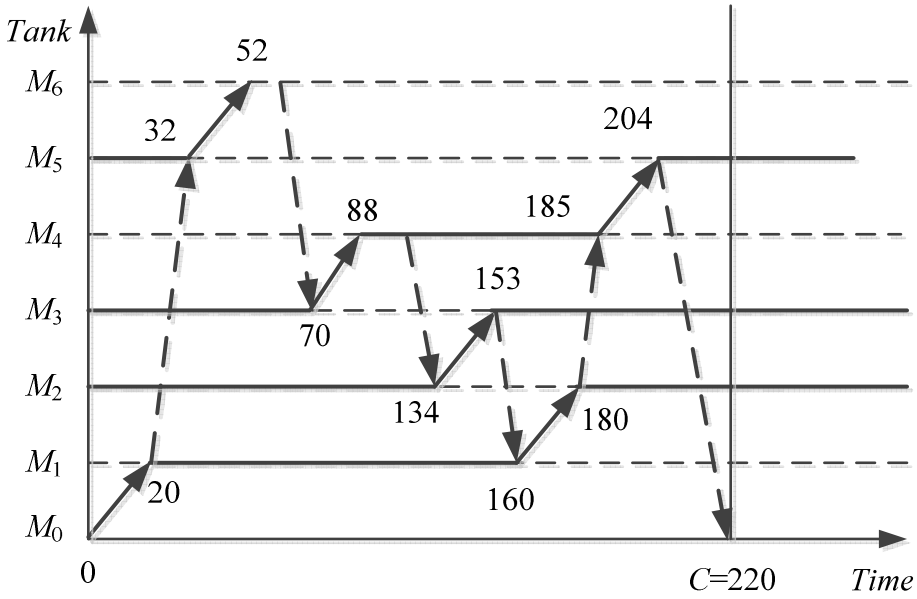

Figure 4.8 Hoist move sequence 0–5–3–2–1–4 with *C*=220.

Table 4.2 Data for the example

Tank i	0	1	2	3	4	5
L_i	–	71s	81s	45s	40s	30s
U_i	–	187s	188s	137s	97s	63s
d_i	20s	20s	19s	18s	19s	20s

Furthermore, Figure 4.9 illustrates a different feasible hoist move sequence for C=220s. The travelling times of empty hoist moves related to the presented move sequence are: $e_{1,3}$=7s, $e_{4,4}$=$e_{5,5}$=0, $e_{6,2}$=12s, $e_{3,1}$=7s, $e_{2,0}$=8s. As can be seen from Figure 4.9, the hoist move sequence is 0–3–4–5–2–1. As verified by Figures 4.8 and 4.9, different hoist move sequences can have the same cycle time.

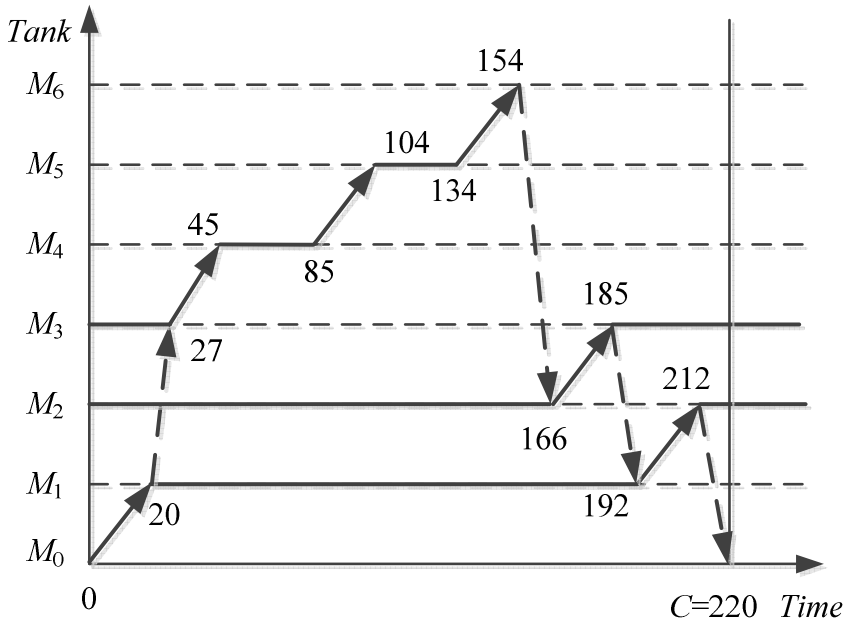

Figure 4.9 Hoist move sequence 0–3–4–5–2–1 with C=220.

4.4.7 Steps of the proposed algorithm

Input: *Np* (*size of the quantum chromosomes*); *Maxgen* (*maximum number of iterations*); *MaxEA* (*maximum size of external archive*); m_p (*probability of mutation*); χ (*LS period*); ND_0=$\varnothing$ (*external archive, which is set to be empty at the initial step*).

Output: *ND* (*the set of non-dominated solutions*).

Step(*I*) *Initialization: First encode an initial population with Np quantum individuals, and then decode each quantum chromosome into 2 problem solutions (i.e. P) using* (4.6) *and* (4.7); *set g*=0.

Step(*II*) *Determine objective values: First use Levner's procedure to find the optimal value of the first objective (i.e. cycle time), and then calculate the value of the second objective (i.e., production cost) according to each solution P.*

Step(*III*) *Individual evaluation: classify the population into K different frontiers* F_1, F_2, F_3,…, F_K, *and calculate the crowding-distance for each individual.*

Step(*IV*) *Update the external archive*: $ND_0 = ND_0 \cup F_1$.

Step(*V*) *Let* $g=g+1$.

Step(*VI*) if $g>Maxgen$, *then go to Stop and output the external archive; else, go to Step*(*VII*).

Step(*VII*) *Update quantum individuals: apply the proposed chaotic rotation gate to update each quantum individual*.

Step(*VIII*) *Apply mutation operator to each chosen quantum individual.*

Step(*IX*) *Decode the quantum individuals using conversion procedures* (4.6) *and* (4.7).

Step(*X*) *Obtain objective values and evaluate solutions*.

Step(*XI*) *Update the external archive*: $ND_g = ND_{g-1} \cup F_1$.

Step(*XII*) *At every* χ *generation, apply the LS procedure to improve the solutions in external archive*. *After that, Go to Step*(*V*).

4.5 Experimental study

In this section, the performance of the proposed bi-objective optimization algorithm QEA with local search procedure is evaluated on a practical electroplating problem selected from an automated zinc plating plant in China (Ni, 2010). In what follows, we first describe the selected real industrial instance, and then present the computational results as well as some analysis and discussions on the obtained results.

4.5.1 Industrial instance

Due to its wide application, zinc plating has existed for a long time. It is mainly for providing corrosion-resistance or decorative layers to metal objects, such as steel plates and nuts. As shown in Figure 4.10, the selected zinc electroplating process has 20 processing stages, each of which corresponds to a specific tank containing special solutions. A steel plate with double-surface area $5m^2$ is processed through M_1 to M_{20} for achieving a uniform zinc layer on its surface. More precisely, as steel plate is generally contaminated with dust, grease lubricants and metal fines, M_1~ M_{12} (usually

called pre-treatment step) are used to remove these residues from its surface. This is a prerequisite for achieving better adhesion of zinc layer to be deposited on the steel part in later stages. Thereafter, steel part is placed in the plating tank M_{13} containing alkaline-type electrolytes for zinc electroplating process. After that, bright dipping and passivating tanks (usually called post-treatment step) containing concentrated acid are used to further improve the corrosion-resistance of the treated steel part. Moreover, after each chemical tank, at least one rinsing tank is used, which is designed for cleaning the chemical solution adsorbed on the part surface as well as other processing purposes. The process technology of the selected electroplating problem is given in Table 4.3.

In this study, for each rinsing tank i (i.e., M_2, M_3, M_4, M_6, M_9, M_{10}, M_{12}, M_{14}, M_{15}, M_{17}, M_{19}), its cost coefficient w_i is computed as: $w_i=q_i\times 0.006$RMB/L, where q_i denotes the water flow rate per second, and 0.006 RMB is the water price per liter, i.e. 6RMB/tonnes. For each electricity-based tank i (i.e. M_5, M_8, M_{11}, M_{13}), its cost coefficient w_i can be computed as follows: $w_i=(100\times I_i\times V_i\times SA)\times 4.17\times 10^{-7}$RMB/Watt, where $100\times I_i\times V_i\times SA$ denotes the amount of electricity consumed per second, and 4.17×10^{-7} RMB is the electricity price per Watt, i.e. 1.5 RMB/kWh. More precisely, $100\times I_i$ represents the current density per square meters. V_i denotes the voltage, and SA denotes the double surface areas of the treated steel part. Note that both the water and the electricity prices are obtained from the Price Bureau of Xi'an, China. For the rest tanks (i.e. M_1, M_7, M_{16}, M_{18}, M_{20}), their cost coefficients are set to be 0 due to the difficulties of obtaining the resource consumption amount during the process. Based on the above descriptions, Table 4.4 reports the cost coefficient of each tank and the execution times of loaded moves. Moreover, the move 0's execution time is given as: d_0=15s. The travel time between tanks i and j is computed as: $e_{i,j}=|i-j|\times 2$s.

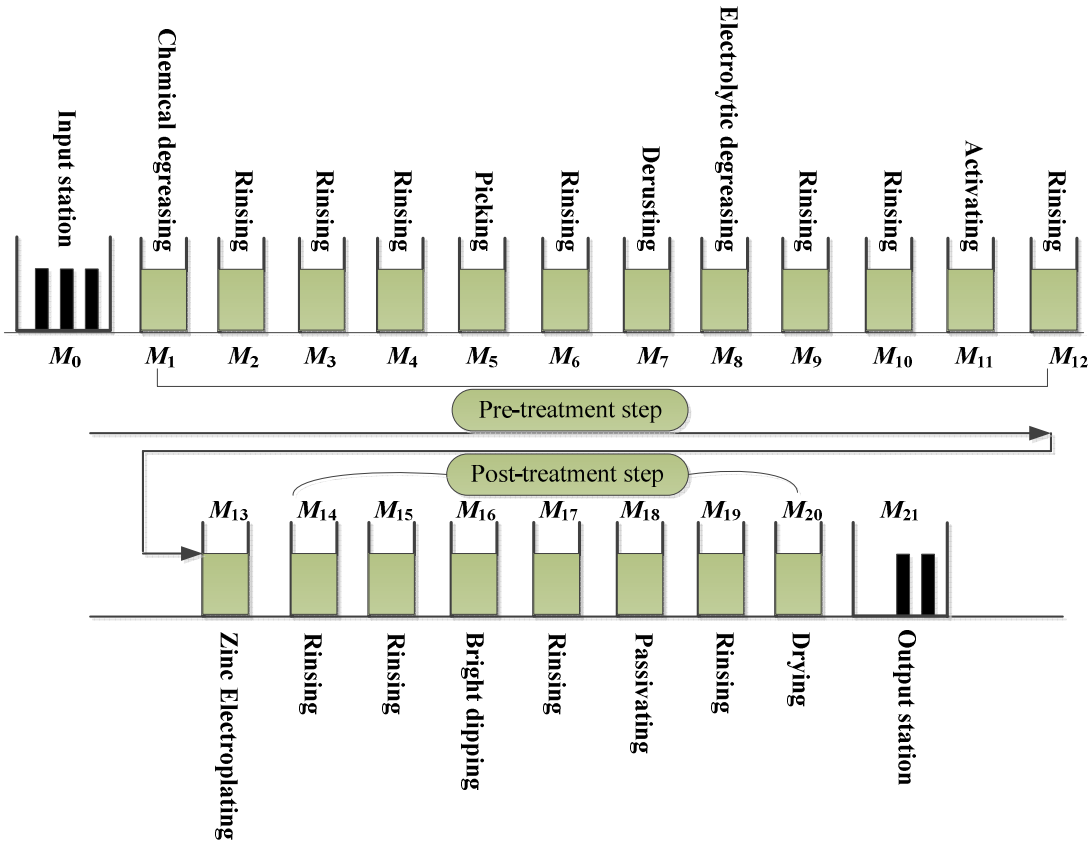

Figure 4.10 Zinc electroplating process for the selected problem.

Table 4.3 Process technology of a steel plate for Zinc-electroplating

Tank	Processing stage	Solutions	Processing time windows (s)	Current density I (A/dm^2)	Water flow rate q
1	Chemical degreasing	NaOH, Na_3PO_4	300~450		
2	Rinsing	Hot water	30~90		0.3L/s
3, 4	Rinsing	Purified water	60~120, 30~90		0.4L/s, 0.3L/s
5	Pickling	HCI	600~900	2~10(9V~12V)	
6	Rinsing	Purified water	30~120		0.4L/s
7	Derusting	CrO_3, H_3PO_4	60~300		
8	Electrolytic degreasing	NaOH, Na_3PO_4, Na_2CO_3	30~120	3~10(9V~12V)	
9	Rinsing	Hot water	30~90		0.3L/s
10	Rinsing	Purified water	60~120		0.5L/s
11	Activating	H_2SO_4, H_3PO_4	30~60	3~5 (1V~18V)	
12	Rinsing	Purified water	20~80		0.4L/s
13	Zinc-plating	ZnO, NaOH, JZ04	660~1350	1~12(6V~16V)	
14, 15	Rinsing	Purified water	30~60, 30~90		0.5L/s, 0.4L/s
16	Bright dipping	HNO_3	10~30		
17	Rinsing	Purified water	30~90		0.2L/s
18	Color Passivating	CrO_3, $NaNO_3$, $NisO_4$·6H2O	120~480		
19	Rinsing	Purified water	20~30		0.4L/s
20	Drying	—	15~35		

Table 4.4 Data for the selected Zinc-electroplating problem

Tank i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
d_i	22	15	15	20	21	20	19	20	15	20	19	15	25	20	21	15	20	22	15	15
w_i	0	0.0018	0.0024	0.0018	0.012	0.0024	0	0.0165	0.0018	0.003	0.0075	0.0024	0.21	0.003	0.0024	0	0.0012	0	0.0024	0

4.5.2 Computational results

In this section, the proposed bi-objective QEA with LS procedure is implemented in C programming language and evaluated by the above described instance. It is solved on an ASUS Laptop with an Intel Core i5-3210M Processor 2.50GHZ and on a windows 8 environment. The parameters are set as follows: maximum generations, *Maxgen*=1000; maximum size of external archive, *MaxEA*=20; local search period, χ =100. As evolutionary algorithm is generally sensitive to the value of initial population size *Np* and mutation probability m_p, we set $Np \in \{50, 100, 150, 200, 250\}$ and $m_p \in \{0.2, 0.5, 0.7, 0.9\}$ in our experimental study to investigate the performance of our proposed algorithm.

Table 4.5 gives the computational results for $Np \in \{50, 100, 150, 200, 250\}$ obtained with the proposed algorithm. Note that for each given *Np*, the proposed algorithm with four different mutation probabilities has been tested. From Table 4.5, we can see that the proposed algorithm with *Np*=100 (by m_p=0.5) and *Np*=250 (by m_p=0.2) generally has a better solution quality than other parameter settings. Besides, we observe that as the population size increases, some new non-dominated solutions are identified. Note that for ease of description here, each solution is represented by its objective values (i.e., *cycle time* and *production cost*) instead of the processing time *P* used before. For instance, solutions (783, 152.7117), (801, 148.6116) and (843, 147.6519) are found by setting *Np*=100 with m_p=0.5. As for *Np*=150, we can see that another new solution (823,147.9924) is identified by the algorithm with m_p=0.9, and it is not dominated by any other solutions reported in Table 4.5. Moreover, a better solution (801, 148.2918) is produced by setting *Np*=200 and 250. As we can see, none of the reported solutions can dominate the solution (801, 148.2918), which dominates the solution (801, 148.6116) produced by setting *Np*=100 and m_p=0.5, since the former gives a smaller (i.e. better) value of production cost than the latter. We also notice that the two solutions have a same value of cycle time (i.e. *C*=801) but have different values of production cost. This is because different actual processing times or hoist move sequences may have the same cycle time.

Table 4.5 Computational results obtained with the proposed algorithm

Np		Non-dominated solution (*Cycle Time*, *Production Cost*)	Computational time (s)
50	m_p=0.2	(787, 154.1709), (883, 152.0364), (964, 148.1961), (1389, 148.0755), (1402, 147.4062), (1449, 147.372)	8.14
	m_p=0.5	(863, 147.765), (1402, 147.4062), (1449, 147.372)	8.26
	m_p=0.7	(782, 153.6855), (964, 148.1961), (1389, 148.0755), (1402, 147.4062), (1449, 147.372)	8.29
	m_p=0.9	(843, 148. 9065), (1389, 148.0755), (1402, 147.4062), (1449, 147.372)	8.32
100	m_p=0.2	(782, 153.6855), (964, 148.1961), (1005, 149.469), (1415, 148.224), (1449, 147.372)	16.91
	m_p=0.5	(782, 153.6855), (783, 152.7117), (801, 148. 6116), (843, 147.6519), (1372,147.4212), (1402, 147.4062), (1449, 147.372)	16.22
	m_p=0.7	(787, 154.1709), (843, 148.9065), (863, 147.7649), (1402, 147.4062), (1449, 147.372)	16.30
	m_p=0.9	(787, 154.1709), (964, 148.1961), (1402, 147.4062), (1449, 147.372)	15.87
150	m_p=0.2	(782, 153.6855), (801, 148.6116), (891, 148.1592), (1402, 147.4062), (1449, 147.372)	23.53
	m_p=0.5	(782, 153.6855), (843, 147.6519), (1402, 147.4062), (1449, 147.372)	23.34
	m_p=0.7	(863, 147.7649), (1402, 147.4062), (1449, 147.372)	23.23
	m_p=0.9	(782, 153.6855), (823, 147.9924), (1402, 147.4062), (1449, 147.372)	23.49
200	m_p=0.2	(782, 153.6855), (801, 148.2918), (843, 147.6519), (1402, 147.4062), (1449, 147.372)	30.9
	m_p=0.5	(787, 154.1709), (801, 148.2918), (843, 147.6519), (1402, 147.4062), (1449, 147.372)	31.04
	m_p=0.7	(782, 153.6855), (843, 147.6519), (1402, 147.4062), (1449, 147.372)	31.02
	m_p=0.9	(813, 171.45), (816, 149.224), (843, 148.9065), (863, 147.7649), (1372, 147.4212), (1402, 147.4062), (1449, 147.372)	30.97
250	m_p=0.2	(782, 153.6855), (801, 148.2918), (843, 147.6519), (1372, 147.4212), (1402, 147.4062), (1449, 147.372)	38.44
	m_p=0.5	(843, 147.6519), (1372, 147.4212), (1402, 147.4062), (1449, 147.372)	38.52
	m_p=0.7	(787, 154.1709), (843, 147.6519), (1372, 147.4212), (1402, 147.4062), (1449, 147.372)	38.68
	m_p=0.9	(782, 153.6855), (816, 148.8456), (1372, 147.4212), (1402, 147.4062), (1449, 147.372)	38.49

Furthermore, we notice from Table 4.5 that all the computational times are less than one minute, and it generally increases with the initial population size *Np*. For each given *Np*, it seems that the computational time has been slightly influenced by the mutation probability. The Pareto frontiers for *Np*=50, 100, 150, 200, and 250 are respectively illustrated in Figure 4.11~Figure 4.15. Note that in each figure, four Pareto frontiers are illustrated, and each one presents the distribution state of the obtained solutions for a given value of m_p. We can see from these figures that as the population size *Np* increases, it seems that the four obtained Pareto frontiers gradually have similar curves. This indicates that the proposed algorithm has a good computational performance.

Finally, to test the performance of the proposed local search (LS) procedure, we also run our proposed bi-objective QEA without LS procedure. Since it has a worse performance than the algorithm with LS procedure for each pair of *Np* and m_p, we do not present the computational results for all values of *Np* and m_p. Instead, we only illustrate the comparison results of *Np*=100 with m_p=0.5 in Figure 4.16. In summary, the computational results show that our proposed bi-objective QEA with LS procedure is efficient in solving the studied dual-objective hoist scheduling problem with processing time windows.

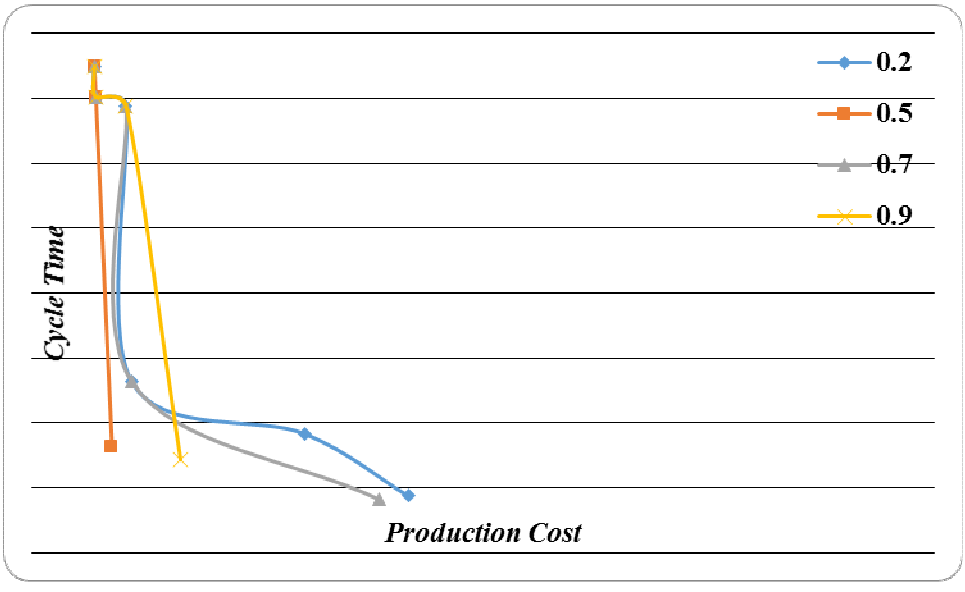

Figure 4.11 Pareto frontiers identified with different m_p for *Np*=50.

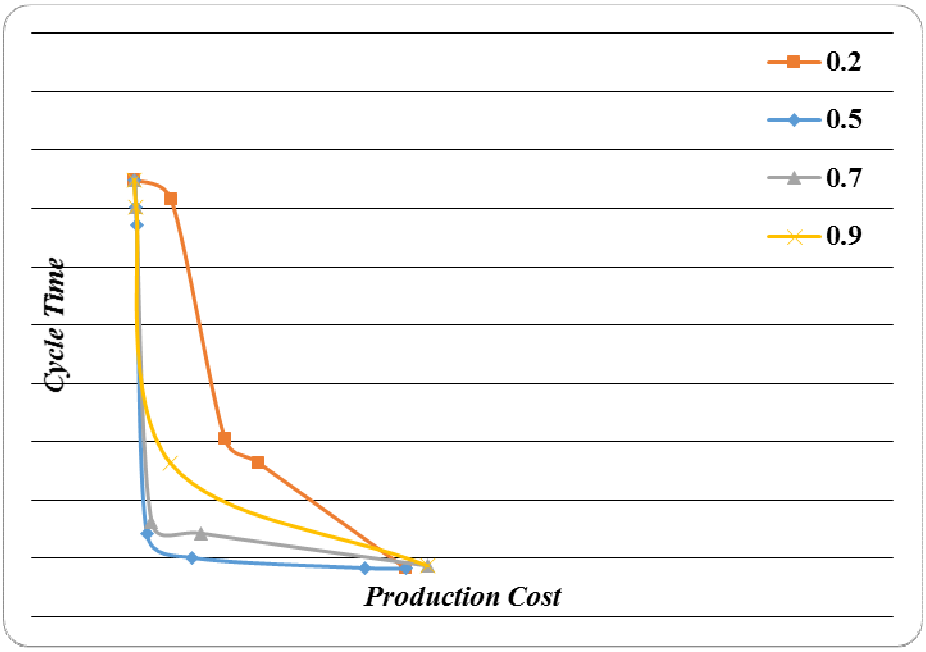

Figure 4.12 Pareto frontiers identified with different m_p for Np=100.

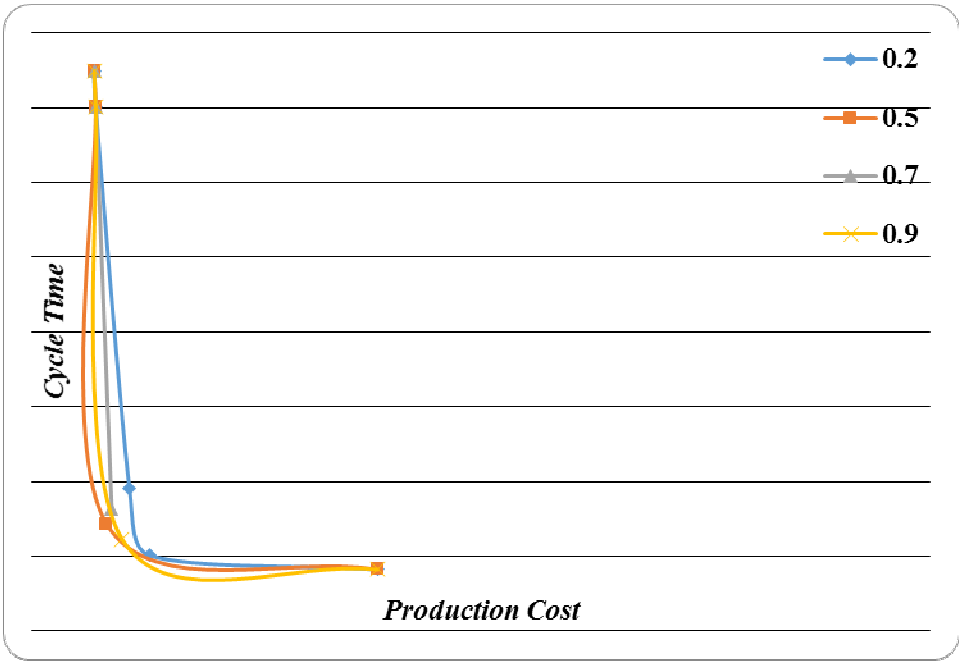

Figure 4.13 Pareto frontiers identified with different m_p for Np=150.

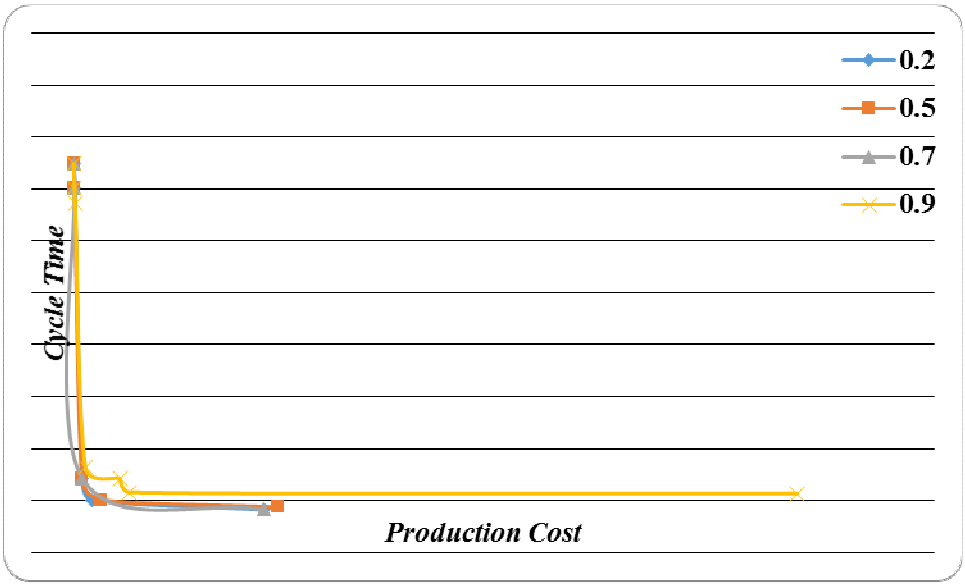

Figure 4.14 Pareto frontiers identified with different m_p for Np=200.

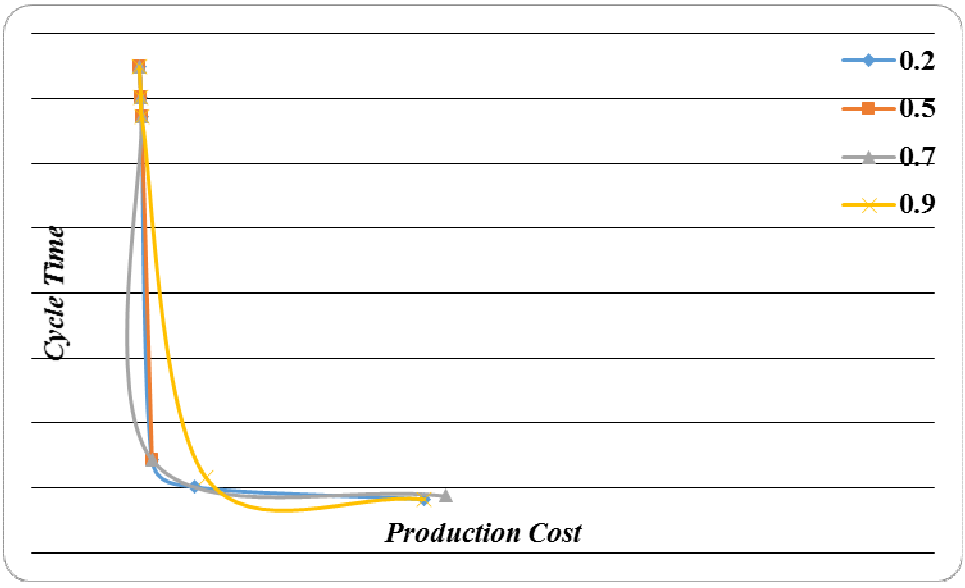

Figure 4.15 Pareto frontiers identified with different m_p for Np=250.

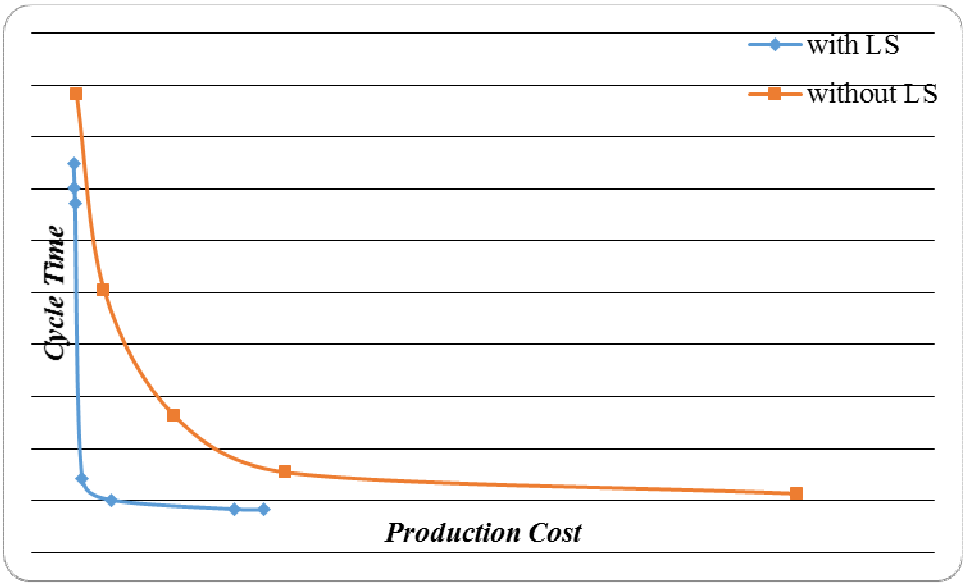

Figure 4.16 Comparison results of the algorithm with and without LS for Np=100 and m_p=0.5.

4.6 Summary

In this chapter, minimizing both cycle time and production cost for a cyclic hoist scheduling problem with processing time windows has been studied. Firstly, by using the MPI approach, a bi-objective mathematical model was formulated for the studied problem supposing that all actual processing times are known (In fact they are decision variables). Thereafter, a Pareto-dominance evaluation based QEA with local search (LS) procedure was proposed for the problem to find a set of Pareto-optimal solutions, which are stored and updated in an external archive. More precisely, each chromosome was encoded by n Q-bits, which were converted into actual processing times by a double-decoding procedure. Then, we proposed a specific chaotic rotation gate to update each Q-bits chromosome. Besides, mutation operator was implanted into the proposed algorithm to increase the individual diversity. All solutions were evaluated by the well-known Pareto-dominance technique. Because of the special solution feature of the studied problem, an efficient LS procedure was proposed for further improving the solution quality. Finally, a real zinc electroplating problem was used to test the performance of our proposed algorithm. Experimental results showed that the proposed algorithm is efficient.

Chapter 5 An Improved Mixed Integer Programming Approach for Multi-hoist Cyclic Scheduling Problem

5.1 Introduction

Multi-hoist cyclic scheduling problems are often encountered in automated electroplating lines for processing printed circuit boards (PCBs) and other electronics (e.g., Lei and Wang, 1991; Leung and Zhang, 2003; Che and Chu, 2004). The key to the multi-hoist cyclic scheduling problem is to determine an executable hoist schedule such that the cycle time is minimized.

In most existing studies on the multi-hoist cyclic scheduling problem, such as Lei and Wang (1991), Armstrong *et al*. (1996), Leung and Zhang (2003), Leung *et al.* (2004), Che and Chu (2004), Zhou and Liu (2008), Zhou and Li (2009), Chtourou *et al.* (2013) and Jiang and Liu (2014), loaded hoist moves are assumed to start and end within the same cycle. In this chapter, we first give a counterexample to demonstrate that the optimal solution obtained under such an assumption is not necessarily the best one among all feasible solutions, which we call hereafter global optimal solution.

To obtain a global optimal solution, the assumption that loaded hoist moves are assumed to start and end within the same cycle should be relaxed. That is, a loaded hoist move is allowed to start in the current cycle and end in the next one if necessary. With the relaxation of the assumption mentioned above, we propose an improved MIP approach for the multi-hoist cyclic scheduling problem with unidirectional part flow, where the part processing sequence is the same as the tanks layout. Since Leung *et al*. (2004) developed the first MIP model for the same problem as the one considered in this chapter, this work can be seen as an extension of their MIP model. Hence, in what follows, we will first present Leung *et al*.'s MIP model and then describe our extension and improvements based on their MIP model.

The rest of this chapter is arranged as follows. The problem description and Leung *et al*.'s MIP model are given in Section 5.2. In Section 5.3, we give a counterexample to justify our findings. Then, an improved MIP model is proposed in Section 5.4. Computational results are presented and analyzed in Section 5.5. Section 5.6 concludes this chapter.

5.2 Problem definition and Leung *et al.*'s MIP model

For completeness, we give in this section a brief problem description and Leung *et al.*'s MIP model. For ease of comparison between Leung *et al.*'s MIP model and ours, we follow all the assumptions and notations given in Leung *et al.* (2004).

5.2.1 Problem definition

Firstly, we describe the problem involved. Consider an automated electroplating line with n processing tanks and K hoists for material handling between the tanks. Each part to be processed starts at the input station (i.e. tank 0), then successively passes through tank 1, tank 2, …, tank n and is finally unloaded at the output station (i.e. tank n+1). The tanks are arranged in a row according to the processing sequence of the parts. Each tank can process only one part at any time. There is no intermediate buffer between the tanks. After the processing in a tank has been completed, the part must be transported by a hoist to the next tank without any delay.

The K hoists are numbered consecutively with the one closest to tank 0 being hoist 1 and the one closest to tank n+1 being hoist K. The hoists are assumed to have zero width and the same travel speed. The hoist movement of transporting a part from tank i to tank i+1 is called (loaded) move i, which is composed of three simple hoist operations: 1) unload a part from tank i; 2) transport it to tank i+1; and 3) load it into tank i+1.

In a cyclic schedule, the hoists perform a fixed sequence of moves repeatedly. Each repetition of the sequence of hoist moves is called a cycle. The duration of a cycle is the cycle time. The objective is to find an optimal K-hoist schedule such that the cycle time is minimized.

Let N= {1, 2, …, n}, N^0={0, 1, 2, …, n} and $\mathcal{K}$= {1, 2, …, K}. The following parameters are given:

d_i : the time required to execute move i, for $i \in N^0$.

$e_{i,j} = e_{j,i}$: the empty hoist travelling time from tank i to tank j, for i, $j \in N^0 \cup \{n+1\}$.

L_i : the minimum processing time in tank i, for $i \in N$.

U_i : the maximum processing time in tank i, for $i \in N$.

M: a very large positive number.

δ: a small constant.

The following decision variables are involved in this chapter:

t_i : start time of move i, for $i \in N^0$.

y_{ij}: 0-1 variable. If $t_i < t_j$, then $y_{ij}=1$, which means that move j starts after move i; otherwise, $y_{ij}=0$, for $i \neq j$, $i, j \in N$.

$\mathcal{L}_i$: 0-1 variable. If move i is the last move for hoist 1, then $\mathcal{L}_i$ =1; otherwise, $\mathcal{L}_i=0$, for $i \in N^0$.

z_i^k :0-1 variable. If move i is executed by hoist k, then z_i^k =1; otherwise, z_i^k =0, for $i \in N^0$, $k \in \mathcal{K}$,

s_i: 0-1 variable. If a part is in process in tank i at the beginning of a cycle, then $s_i=1$; otherwise, $s_i=0$, for $i \in N$.

C: cycle time.

With above notations and according to Manier and Bloch (2003), the considered problem can be written in the form:

$$CHSP \mid K, n, 1 \,//\, diss \mid /n+2 \mid Cmin$$

which means cyclic hoist scheduling problem with K hoists and n tanks, each tank being a single capacity resource, with dissociated loading and unloaded stations, n+2 operations per part, and minimization of cycle time C as the objective.

5.2.2 Leung *et al.*'s model

Leung *et al*. (2004) developed their MIP model by addressing the following four families of constraints:

1) Hoist assignment and cycle-time definitional constraints. Each hoist move is assigned to one and only one hoist and the cycle time is long enough to allow hoist 1 to return to the input station (i.e. tank 0) for starting move 0 of the next cycle.

2) Time window constraints. The soaking or processing time of a part in a tank must be within its prescribed minimum and maximum processing times. Otherwise,

defective parts would be produced.

3) Hoist capacity constraints. The start-times of the moves executed by the same hoist are determined in such a way that there is sufficient time gap for any hoist to travel between the successive moves assigned to that hoist.

4) Collision avoidance constraints. No collisions happen among the hoists running on a single shared track.

According to the four families of constraints given above, Leung *et al*. (2004) developed the following MIP model for the multi-hoist cyclic scheduling problem:

Minimize C

subject to

Hoist assignment and cycle-time definitional constraints:

$$\sum_{k=1}^{K} z_i^k = 1, \text{ for all } i \in N, \tag{5.1}$$

$$\sum_{i=0}^{n} \mathcal{L}_i = 1, \tag{5.2}$$

$$\mathcal{L}_0 + z_i^1 \le 1, \text{ for all } i \in N, \tag{5.3}$$

$$\mathcal{L}_i \le z_i^1, \text{ for all } i \in N, \tag{5.4}$$

$$z_i^1 + \mathcal{L}_j - y_{ij} \le 1, \text{ for all } i, j \in N, \tag{5.5}$$

$$t_i + d_i + e_{i+1,\,0}\mathcal{L}_i \le C, \text{ for all } i \in N^0, \tag{5.6}$$

$$t_j - (d_0 + e_{1,\,j})\, z_j^1 \ge 0, \text{ for all } j \in N, \tag{5.7}$$

$$t_0 = 0, \tag{5.8}$$

Time window constraints:

$$t_i - (t_{i-1} + d_{i-1}) \le U_i, \text{ for all } i \in N, \tag{5.9}$$

$$t_i - (t_{i-1} + d_{i-1}) + Ms_i \ge L_i, \text{ for all } i \in N, \tag{5.10}$$

$$t_i + C - (t_{i-1} + d_{i-1}) - M(1 - s_i) \le U_i, \text{ for all } i \in N, \tag{5.11}$$

$$t_i+C-(t_{i-1}+d_{i-1})\geq L_i, \text{ for all } i\in N, \tag{5.12}$$

$$t_i-t_{i-1}-d_{i-1}+\delta-(U_i+\delta)(1-s_i)\leq 0, \text{ for all } i\in N, \tag{5.13}$$

Hoist capacity constraints:

$$t_j-t_i\leq My_{ij}, \text{ for all } i,\ j\in N,\ i\neq j, \tag{5.14}$$

$$y_{ij}+y_{ji}=1, \text{ for all } i,\ j\in N,\ i\neq j, \tag{5.15}$$

Collision avoidance constraints:

$$t_i+d_i+e_{i+1,j}-t_j\leq M(3-y_{ij}-z_i^k-\sum_{h=k}^{K}z_j^h), \text{ for all } i, j\in N, j<i, k\in\mathcal{K}, \tag{5.16}$$

$$t_j+d_j+e_{j+1,i}-t_i\leq M(3-y_{ji}-z_i^k-\sum_{h=k}^{K}z_j^h), \text{ for all } i, j\in N, j<i, k\in\mathcal{K}, \tag{5.17}$$

$$t_j+d_j+e_{j+1,i}-t_i\leq M(3-y_{ji}-z_i^k-\sum_{h=1}^{k}z_j^h), \text{ for all } i,\ j\in N,\ i<j,\ k\in\mathcal{K}, \tag{5.18}$$

$$t_i+d_i+e_{i+1,j}-t_j\leq M(3-y_{ij}-z_i^k-\sum_{h=1}^{k}z_j^h), \text{ for all } i,\ j\in N,\ i<j,\ k\in\mathcal{K}, \tag{5.19}$$

$$t_j+d_j+e_{j+1,i}-(C+t_i)\leq M(2-z_i^k-\sum_{h=k}^{K}z_j^h), \text{ for all } i, j\in N, j<i, k\in\mathcal{K}, \tag{5.20}$$

$$t_i+d_i+e_{i+1,j}-(C+t_j)\leq M(2-z_i^k-\sum_{h=k}^{K}z_j^h), \text{ for all } i, j\in N, j<i, k\in\mathcal{K}, \tag{5.21}$$

$$t_j+d_j+e_{j+1,i}-(C+t_i)\leq M(2-z_i^k-\sum_{h=1}^{k}z_j^h), \text{ for all } i, j\in N, i<j, k\in\mathcal{K}, \tag{5.22}$$

$$t_i+d_i+e_{i+1,j}-(C+t_j)\leq M(2-z_i^k-\sum_{h=1}^{k}z_j^h), \text{ for all } i, j\in N, i<j, k\in\mathcal{K}, \tag{5.23}$$

Binary variable definitional constraints:

$$z_i^k\in\{0, 1\}, \text{ for all } i\in N^0, k\in\mathcal{K}, \tag{5.24}$$

$$\mathcal{L}_i\in\{0,\ 1\}, \text{ for all } i\in N^0, \tag{5.25}$$

$$s_i\in\{0, 1\}, \text{ for all } i\in N, \tag{5.26}$$

$$y_{ij}\in\{0, 1\}, \text{ for all } i, j\in N. \tag{5.27}$$

5.3 Illustration of a counterexample

We now use the following counterexample to demonstrate that the optimal solution obtained with Leung *et al.*'s MIP approach is not a global optimal solution. There are 5 processing tanks and 2 hoists for this example (i.e., n = 5, K = 2). The data for the example is given in Table 5.1, which was generated via our experiment. Tank 0 and tank 6 are the input station and the output station, respectively. The travel time between tank i and tank j can be computed as follows: $e_{i,j}=e_{j,i}=\sum_{k=i}^{j-1} e_{k,k+1}$, $i<j$ and $i, j \in N^0 \cup \{n+1\}$. The spent time of loaded move i can be computed as the following way: $d_i=20+e_{i,i+1}$, $i \in N^0$. Without loss of generality, we assume that move 0 is executed by hoist 1 and starts at the beginning of a cycle.

Table 5.1 Data for the counterexample

Tank i	0	1	2	3	4	5
L_i	–	80s	68s	75s	61s	66s
U_i	–	126s	126s	154s	104s	146s
$e_{i,i+1}$	9s	8s	6s	4s	8s	8s
d_i	29s	28s	26s	24s	28s	28s

For this example, the optimal cycle time obtained with Leung *et al.*'s MIP approach is 145s. The time-way diagram for the corresponding optimal cyclic schedule is shown in Figure 5.1. Note that the numbers around a loaded move in Figure 5.1 represent its start and end times. We give in Figure 5.2 a feasible schedule for this example with the cycle time C=142s, which is smaller than the optimal cycle time obtained with Leung *et al.*'s approach. Hence, for this example, the optimal solution obtained with Leung *et al.*'s approach is actually not a global optimal solution.

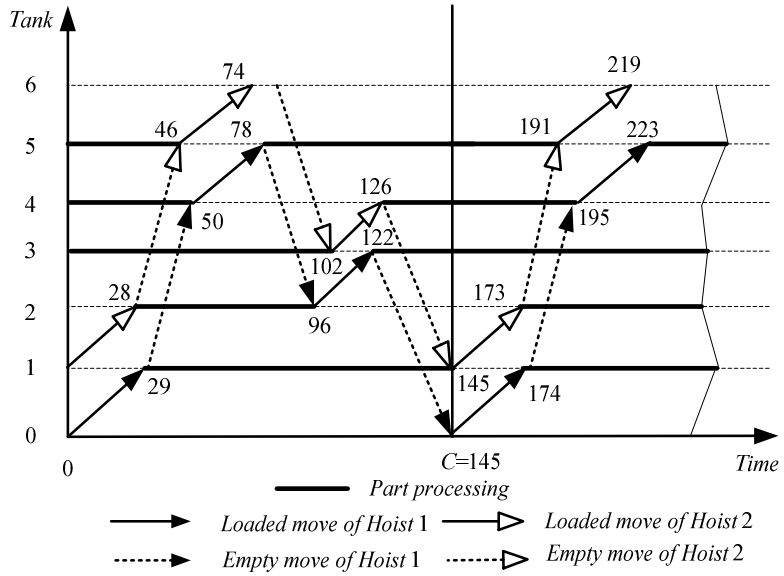

Figure 5.1 Optimal cyclic schedule obtained with Leung *et al.*'s MIP approach.

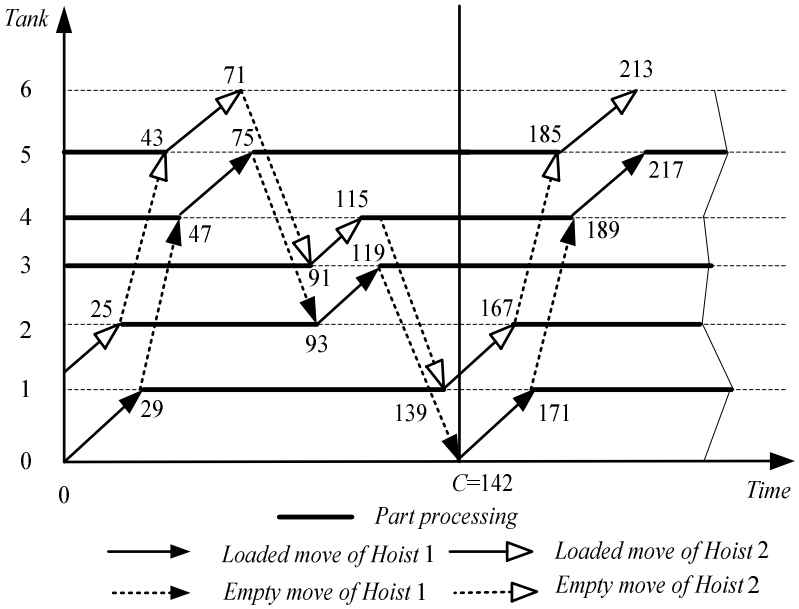

Figure 5.2 A feasible cyclic schedule with shorter cycle time.

We explain the above observation as follows. Note that constraint (5.6) in Leung *et al.*'s model implies that $t_i+d_i \leq C$ holds for all loaded moves. This requires that any loaded move started in the current cycle must be completed within the same cycle. Hence, in their model, Leung *et al.* implicitly assumed that no loaded moves are allowed to go across the cycle (i.e., start in one cycle and end in the next one). Although such an assumption may simplify the formulation of the problem, it may

restrict the possibility of achieving a better feasible solution.

We verify the above observation using the cyclic schedule given in Figure 5.2. We note that move 1 in Figure 5.2 starts at time 139s and ends at time 167s. Recall that the cycle time C is 142s. Thus, move 1 goes across the cycle. We see that a better feasible solution than the one obtained with Leung *et al.*'s MIP approach was obtained by allowing move 1 to go across the cycle. Note that the cyclic schedule with shorter cycle time given in Figure 5.2 was obtained by using our improved MIP approach, which will be presented in section 5.4.

To sum up, no loaded moves are allowed to go across the cycle in Leung *et al.*'s MIP model. For this reason, the optimal solution obtained with Leung *et al.*'s MIP approach is not necessarily a global optimal solution.

5.4 The improved MIP model

5.4.1 Reformulation of the time window constraints

To obtain a global optimal solution, the assumption that no loaded moves are allowed to go across the cycle should be relaxed in the formulation of the problem. To achieve this purpose, constraint (5.6) in Leung *et al.*'s model, which requires that no loaded moves are allowed to go across the cycle, should be replaced with the following formula:

$$t_i+(d_i+e_{i+1,0})\mathcal{L}_i\leq C, \text{ for all } i\in N^0, \tag{5.28}$$

In what follows, we first extend Leung *et al.*'s time window constraints (5.9)–(5.12) by relaxing the assumption that no loaded moves are allowed to go across the cycle. With such a relaxation, four possible cases, as illustrated in Figure 5.3, should be considered when the time window constraints are formulated. In Figure 5.3, Case (a) (resp. Case (b)) corresponds to the case in which tank i is empty (resp. occupied) at the beginning of a cycle and move $i-1$ does not go across the cycle. Cases (c) and (d) correspond to the situations in which tank i is empty and occupied, respectively, at the beginning of a cycle and move $i-1$ goes across the cycle.

In fact, Leung *et al*. (2004) only considered Cases (a) and (b) in their formulation of the time window constraints, which lead to constraints (5.9)–(5.12) in their MIP model. They did not consider Cases (c) and (d) in which move $i-1$ goes across the

cycle.

In what follows, we give a complete formulation of the time window constraints by considering Cases (a), (b), (c) and (d) in Figure 5.3. To facilitate the reformulation, we define a new binary variable w_i to represent whether move i goes across the cycle:

w_i: 0-1 variable. If move i starts and ends within the same cycle, i.e., $t_i<C$ and $t_i+d_i \le C$, then $w_i= 0$; otherwise, $w_i= 1$, i.e., $t_i <C$ and $t_i+d_i>C$, for $i \in N^0$.

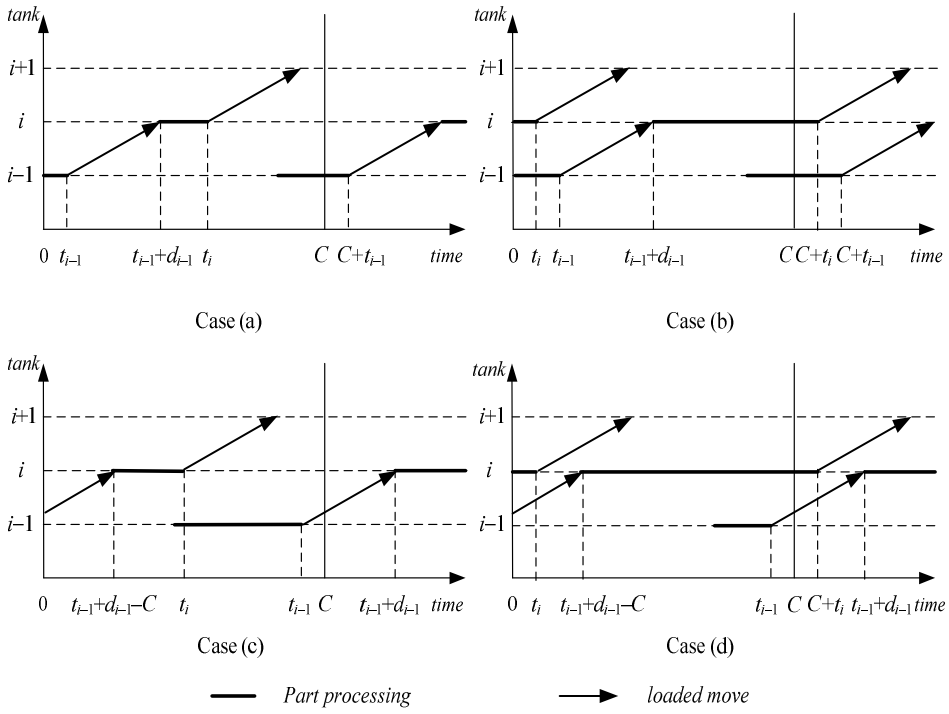

Figure 5.3 Four types of tank states for the time window constraints.

Case (a): $s_i =0$ and $w_{i-1}=0$. It means that tank i is empty at the beginning of a cycle and move i–1 does not go across the cycle. For this case, tank i is still empty until a part enters upon completion of move i–1, which happens at time $t_{i-1}+d_{i-1}$. Note that the part will be unloaded from tank i at time t_i. As shown in Case (a) in Figure 5.3, move i–1 and move i happen within the same cycle. Thus, the actual processing time in tank i is $t_i-(t_{i-1}+d_{i-1})$. Consequently, the time window constraints for tank i can be formulated as:

$$t_i-(t_{i-1}+d_{i-1}) \le U_i+M(s_i +w_{i-1}), \text{ for all } i \in N, \tag{5.29}$$

$$t_i-(t_{i-1}+d_{i-1}) \ge L_i-M(s_i +w_{i-1}), \quad \text{for all } i \in N, \tag{5.30}$$

Case (b): s_i =1 and w_{i-1}=0. It means that a part is in process in tank i at the beginning of a cycle and move i–1 does not go across the cycle. As shown in Case (b) in Figure 5.3, a part is loaded into tank i at time $t_{i-1}+d_{i-1}$ in the current cycle, and it will be unloaded from tank i at time t_i+C in the next cycle. Thus, the actual processing time in tank i is $t_i +C–(t_{i-1}+d_{i-1})$. Based on the above analysis, the time window constraints for tank i can be formulated as:

$$C+t_i–(t_{i-1}+d_{i-1})\leq U_i+M(1–s_i +w_{i-1}), \text{ for all } i\in N, \quad (5.31)$$

$$C+t_i –(t_{i-1}+d_{i-1})\geq L_i–M(1–s_i +w_{i-1}), \text{ for all } i\in N, \quad (5.32)$$

Case (c): s_i =0 and w_{i-1}=1. It means that tank i is empty at the beginning of a cycle and move i–1 goes across the cycle. For this case, move i–1 starts at time t_{i-1} in the current cycle and ends at time $t_{i-1}+d_{i-1}$ in the next cycle, which means that move i–1 goes across the cycle because we have $t_{i-1}<C$ and $t_{i-1}+d_{i-1}>C$. Thus, as shown in Case (c) in Figure 5.3, the actual processing time in tank i is $t_i–(t_{i-1}+d_{i-1}–C)$. Consequently, the time window constraints for tank i can be formulated as:

$$t_i–(t_{i-1}+d_{i-1}–C)\leq U_i+M(1–w_{i-1}+ s_i), \text{ for all } i\in N, \quad (5.33)$$

$$t_i–(t_{i-1}+d_{i-1}–C)\geq L_i–M(1–w_{i-1}+ s_i), \text{ for all } i\in N, \quad (5.34)$$

It is interesting to note that constraints (5.10)–(5.12) can correctly impose the lower and upper bounds on soak time in tank i for this case. To be more specific, constraint (5.12) imposes the lower bound on soak time in tank i. Constraint (5.10) would set s_i to be 1. Consequently, constraint (5.11) would correctly impose the upper bound on soak time tank i. We also note that in this case, the value of s_i being 1 is inconsistent with its definition. By definition, if s_i =1, there should be a part in tank i at the beginning of a cycle. However, we see that for this case, tank i is empty at the beginning of a cycle, as illustrated in Figure 5.2. Hence, if constraints (5.10)–(5.12) are used to formulate the time window constraint for case (c), s_i should be redefined. In our model, constraints (5.33) and (5.34) handle case (c) without such an inconsistency.

Case (d): s_i =1 and w_{i-1}=1. It means that a part is in process in tank i at the beginning of a cycle and move i–1 goes across the cycle. For this case, move i–1 starts at time t_{i-1} in the current cycle and ends at time $t_{i-1}+d_{i-1}$ in the next cycle. Thus, as shown in Case (d) in Figure 5.3, the actual processing time in tank i is $C+t_i$ $–(t_{i-1}+d_{i-1}–C)$. Based on the above analysis, the time window constraints for tank i can be formulated as:

$$C+t_i-(t_{i-1}+d_{i-1}-C)\le U_i+M(2-w_{i-1}-s_i), \text{ for all } i\in N, \quad (5.35)$$

$$C+t_i-(t_{i-1}+d_{i-1}-C)\ge L_i-M(2-w_{i-1}-s_i), \text{ for all } i\in N, \quad (5.36)$$

From the above analysis, constraints (5.29)–(5.36) ensure that the processing time in each tank is within its prescribed lower and upper bounds. Note that if we set $w_{i-1}=0$ for all $i\in N$, as is the case in Leung *et al*.'s formulation of the time window constraints, then constraints (5.29)–(5.32) would be reduced to constraints (5.9)–(5.12) in Leung *et al*.'s model.

We now deal with Leung *et al*.'s time window constraint (5.13). As stated by Leung *et al*. (2004), constraint (5.13) ensures that if tank i is occupied by a part at the beginning of a cycle, then there is a time gap of δ between when the part is unloaded from tank i (at time t_i) and another part is loaded into the tank (at time $t_{i-1}+d_{i-1}$). Below we extend this formulation to handle the case in which a loaded move is allowed to go across the cycle. Hereafter, to facilitate the reformulation, we define:

ε_i: the time required to unload a part from tank i, for all $i\in N$.

ρ_i: the time required to load a part into tank i, for all $i\in N$.

We first consider the case in which move i–1 does not go across the cycle, as illustrated in Case (b) in Figure 5.3. In this case, the unloading operation of the previous part from tank i starts at time t_i and ends at time $t_i+\varepsilon_i$. The loading operation of the next part into tank i starts at time $t_{i-1}+d_{i-1}-\rho_i$ and ends at time $t_{i-1}+d_{i-1}$. To avoid the collision in using tank i, it follows that:

$$(t_i+\varepsilon_i)-(t_{i-1}+d_{i-1}-\rho_i)\le M(1-s_i+w_{i-1}), \text{ for all } i\in N. \quad (5.37)$$

Similarly, if move i–1 goes across the cycle, as illustrated in Cases (c) and (d) in Figure 5.3, we have:

$$(t_i+\varepsilon_i)-(t_{i-1}+d_{i-1}-\rho_i)\le M(1-w_{i-1}+s_i), \text{ for all } i\in N, \quad (5.38)$$

$$(t_i+\varepsilon_i)-(t_{i-1}+d_{i-1}-\rho_i-C)\le M(2-w_{i-1}-s_i), \text{ for all } i\in N. \quad (5.39)$$

Note that Leung *et al*. (2004) only consider Case (b) in Figure 5.3, in which move i–1 does not go across the cycle. If we set $w_{i-1}=0$ for all $i\in N$ and set $\delta=\varepsilon_i+\rho_i$, then constraint (5.37) would be equivalent to constraint (5.13) in Leung *et al*.'s model. Note also that Case (a) is not required to be considered here because in this case, the time window constraint (5.30) guarantees that $t_i\ge t_{i-1}+d_{i-1}+L_i$. As L_i is usually greater than $\varepsilon_i+\rho_i$, there is sufficient time gap between the loading and unloading operations

of the (same) part and no collision would happen between the two hoists executing the loading and unloading operations, respectively.

In addition, in order to ensure variable w_i to be well defined, the following constraints must hold:

$$t_i < C, \text{ for all } i \in N, \tag{5.40}$$

$$t_i + d_i \leq C + M w_i, \text{ for all } i \in N, \tag{5.41}$$

$$t_i + d_i > C - M(1 - w_i), \text{ for all } i \in N, \tag{5.42}$$

$$w_i + z_i^1 \leq 1, \text{ for all } i \in N^0, \tag{5.43}$$

$$w_i \in \{0,1\}, \text{ for all } i \in N^0. \tag{5.44}$$

Constraint (5.40) says that the start time of move i should be less than the cycle time C. Constraints (5.40) and (5.42) guarantee that if $w_i = 1$, then move i starts in the current cycle and ends in the next one. On the other hand, constraints (5.40) and (5.41) ensure that move i starts and ends within the same cycle if $w_i = 0$. Constraint (5.43) ensures that if move i is executed by hoist 1(i.e. $z_i^1 = 1$), then it cannot go across the cycle as explained below. In each cycle, hoist 1 would first execute move 0 and then other moves assigned to it, and finally return to the input station to start move 0 of the next cycle, which happens at time C. Hence, if move i is assigned to hoist 1, it must be finished within a cycle and would not go across the cycle.

In order to facilitate the formulation of constraints (5.40) and (5.42) using CPLEX, we add a sufficiently small constant δ into them and they can be equivalently written as:

$$t_i + \delta \leq C, \text{ for all } i \in N, \tag{5.45}$$

$$t_i + d_i \geq C + \delta - M(1 - w_i), \text{ for all } i \in N. \tag{5.46}$$

5.4.2 Other improvements on Leung *et al.*'s MIP model

In this subsection, we report two other improvements on Leung *et al.*'s model. We first demonstrate that the binary variable $\mathcal{L}_i$ is unnecessary to be defined in Leung *et al.*'s model. To be more specific, constraint (5.6) ensures that if move i is the last move executed by hoist 1, then upon completion of move i, hoist 1 has sufficient time

to travel back to the input station (i.e. tank 0) to start move 0 of the next cycle. In fact, as the hoist travelling times satisfy the triangular inequality, constraint (5.6) can be replaced with the following constraint:

$$t_i+(d_i+e_{i+1,\ 0})\, z_i^1 \le C, \ \text{for all} \ i \in N^0, \tag{5.47}$$

The above relation says that $t_i+d_i+e_{i+1,\,0} \le C$ holds for all moves executed by hoist 1. Similar relation can also be found in Chen *et al*. (1998) (see Inequality (8)) for the single-hoist scheduling problem. Thus, it is unnecessary to define the binary variable $\mathcal{L}_i$ in Leung *et al*.'s model. Consequently, constraints (5.2)–(5.5), (5.25) and (5.28) modified from constraint (5.6) can be removed from the model.

We now show that some collision-avoidance constraints given in Leung *et al*.'s MIP model are unnecessary. Suppose that moves *i* and *j* are performed by hoists *k* and *h*, respectively. Without loss of generality, we assume that $i > j$ for any pair of moves (*i*, *j*). That is, given any pair of moves (*i*, *j*), we designate the larger number of move as *i* and the smaller number of move as *j*. For example, if the collision avoidance constraint between move 2 and move 4 is to be considered, we set $i=4$ and $j=2$ and consider the possible collision between them.

As the part processing sequence is same as the tank arrangement sequence, it is understandable that the collision may happen between any two hoists *k* and *h* using a common segment of the track, i.e., $k<h$, $i > j$. That is to say, no collision would happen in the situation of $k>h$, $i > j+1$. It should be noted that constraints (5.37)–(5.39) ensure that no collision would happen between two hoists sharing the same tank (i.e., $k > h$, $i=j+1$), where parts are loaded/unloaded by one hoist and unloaded/loaded by another one.

Based on above analysis, we only need to consider the case $k<h$, $i > j$ in the formulation of the hoist collision avoidance constraints. In this case, hoists *k* and *h* would pass through a common segment of the track. In order to guarantee that no collision would happen between them during the execution of moves *i* and *j*, they cannot be executed at the same time. That is, either move *j* must start after move *i* has finished or move *i* must start after move *j* has finished in order to avoid the collision. Let us first suppose that move *j* starts after move *i* has finished. In this case, move *i* finishes at time t_i+d_i, hoist *k* will pass through tank *j* at time $t_i+d_i+e_{i+1,\,j}$. Knowing that move *j* executed by hoist *h* starts at time t_j, to avoid the possible collision, hoist *k* must

pass through tank j before time t_j. Thus, we have:

$$t_i+d_i+e_{i+1,\,j}\le t_j, \text{ for all } k\le h,\ i>j,\ i,\ j\in N,\ k,\ h\in \mathcal{K} \text{ and } t_i<t_j, \quad (5.48)$$

Similarly, if move i starts after move j has finished, we have:

$$t_j+d_j+e_{j+1,i}\le t_i, \text{ for all } k\le h,\ i>j,\ i,\ j\in N,\ k,\ h\in \mathcal{K} \text{ and } t_j<t_i, \quad (5.49)$$

Besides, consider the possible collision between moves i and j in two consecutive cycles, we must have:

$$t_j+d_j+e_{j+1,i}\le C+t_i, \text{ for all } k\le h,\ i>j,\ i,\ j\in N,\ k,\ h\in \mathcal{K}\ , \quad (5.50)$$

$$t_i+d_i+e_{i+1,\,j}\le C+t_j, \text{ for all } k\le h,\ i>j,\ i,\ j\in N,\ k,\ h\in \mathcal{K}\ , \quad (5.51)$$

Based on above analysis, for any two moves i and j performed by hoists k and h, respectively, (5.48)–(5.51) are their corresponding collision-avoidance constraints. Note that by adding previously defined binary variables into (5.48)–(5.51), they can be transformed into constraints (5.16), (5.17), (5.20), (5.21). We thus can find that constraints (5.16), (5.17), (5.20), (5.21) are sufficient, and constraints (5.18), (5.19), (5.22) and (5.23) are unnecessary and can be removed from the model.

In what follows, we give an illustration to further demonstrate the above observation. Let us consider the collision avoidance constraints between move 3 and move 4 in Figure 5.1 with K=2. We have from Figure 5.1 that y_{34}=0, y_{43}=1, i.e., move 3 starts after move 4 has finished. We also have z_3^1=0, z_3^2=1, z_4^1=1 and z_4^2=0, i.e., move 3 and move 4 are executed by hoist 2 and hoist 1, respectively. We now see for this hoist assignment, what relation between the start times of move 3 and move 4 should satisfy to avoid the possible collision between them. As required by Leung *et al*. (2004), we first let i=3 and j=4 and substitute the values of y_{34}=0, y_{43}=1, z_3^1=0, z_3^2=1, z_4^1=1 and z_4^2=0 into the collision avoidance constraints (5.18), (5.19), (5.22) and (5.23). We obtain the following inequalities:

$$t_4+d_4+e_{5,3}\le t_3 \quad (5.52)$$

$$t_4+d_4+e_{5,3}\le C+t_3 \quad (5.53)$$

$$t_3+d_3+e_{4,4}\le C+t_4 \quad (5.54)$$

As required by Leung *et al*. (2004), we now let i=4 and j=3. By substituting the above values into the collision avoidance constraints (5.16), (5.17), (5.20) and (5.21), we obtain exactly the same inequalities as (5.52)–(5.54). Hence, constraints (5.18), (5.19), (5.22), (5.23) can be removed from the model with the consideration of constraints (5.16), (5.17), (5.20), (5.21).

The model becomes more compact due to the two improvements presented in this subsection.

5.4.3 The improved MIP model

With the extension presented above, the improved MIP model allowing loaded moves to go across the cycle can be formulated as follows:

Minimize C

subject to

Hoist assigning and cycle-time definitional constraints: (5.1), (5.7), (5.8), (5.47).

Time window constraints: (5.29)–(5.39).

Hoist capacity constraints: (5.14)–(5.15).

Collision avoidance constraints: (5.16), (5.17), (5.20), (5.21).

Move cycle-crossing constraints: (5.41), (5.43), (5.45), (5.46).

Binary variable definitional constraints: (5.24), (5.26), (5.27), (5.44).

Note that we do not consider the safe distance between the hoists in the above improved model in order to facilitate the comparison with Leung *et al*.'s model. However, the model can be easily modified to take the safe distance into account. Let β be the minimum interval between two adjacent hoists on the track to avoid collision. For simplicity, β is measured in time and is equal to the width of the hoist divided by its travelling speed. For instance, if the safe distance is considered, constraint (5.16) can be rewritten as follows:

$$t_i + d_i + e_{i+1,j} + (\sum_{h=k}^{K} hz_j^h - kz_i^k)\beta - t_j \le M(3 - y_{ij} - z_i^k - \sum_{h=k}^{K} z_j^h),$$

for all $i, j \in N, j<i, k \in \mathcal{K}$ (5.55)

In the above inequality, if $z_i^k = 1$ and $\sum_{h=k}^{K} z_j^h = 1$ for some $h \geq k$, then we have $(\sum_{h=k}^{K} hz_j^h - kz_i^k)\beta = (h-k)\beta$, which is the minimum safe distance required between hoists *k* and *h* to avoid collision. Similar modifications can also be done to constraints (5.17), (5.20), (5.21), (5.37)–(5.39).

5.5 Computational results

In this section, we evaluate our improved model using both benchmark and randomly generated instances. Both Leung *et al*.'s model and our improved model were coded using C++. The models were then solved using the MIP solver of CPLEX (Version 12.4). All computational experiments were conducted on a HP PC with a Pentium IV Processor 3.0GHZ and on a windows XP environment.

5.5.1 Computational results on benchmark instances

We compare our improved model with Leung *et al*.'s model using five benchmark instances in the literature: BO1, BO2, Phillips and Unger (P&U), Ligne1 and Ligne2. Their data can be found in Leung *et al*. (2004), Phillips and Unger (1976) and Manier (1994). For these benchmark instances, the part processing sequence is assumed to be the same as the tank arrangement sequence.

Table 5.2 is used to test the effectiveness of the two improvements presented in subsection 5.4.2 of Section 5.4. Note that the partially improved model is derived by removing the two improvements presented in subsection 5.4.2 of Section 5.4 from our improved model. The optimal solutions obtained with the partially improved model and our improved model must be the same. In Table 5.2, "B&B" indicates the size of branch-and-bound tree measured in the number of nodes, while "CPU" denotes the computation time measured in CPU seconds. We can see from Table 5.2 that the computation times spent by our improved model are generally smaller than those spent by the partially improved model. However, the B&B sizes seem to show a mixed trend among these instances.

Table 5.3 is used to demonstrate if a smaller cycle time can be found by our improved model compared with Leung *et al*.'s model. In Table 5.3, the numbers on the left and right sides of the slash (/) are the optimal cycle times obtained with Leung

et al.'s model and our improved model, respectively. The number marked with * means that at least one hoist move in the optimal solution goes across the cycle. We can see that both Leung *et al.*'s model and our improved model obtained the same optimal solutions for most instances except problem P&U with K=3. For this problem, the optimal cycle time obtained with Leung *et al.*'s model is 205 while a better solution with the cycle time 198 was found by our improved model. For other solutions marked with *, although at least one hoist move in the optimal solution obtained with our improved model goes across the cycle, the optimal cycle times obtained with the two models remain the same.

Table 5.2 Comparison of computation times for benchmark instances

Instances	Partially improved model		Our improved model	
	B&B	CPU	B&B	CPU
BO1(K=2)	1928	1.03	708	0.44
BO1(K=3)	952	1.38	612	0.55
BO1(K=4)	283	0.81	1544	1.27
BO2(K=2)	1421	0.89	572	0.44
BO2(K=3)	1925	2.25	60	0.38
BO2(K=4)	151	0.78	1556	1.99
P&U(K=2)	43759	21.44	27086	9.94
P&U(K=3)	60081	45.88	29279	14.84
P&U(K=4)	2147	5.92	4776	4.77
Ligne1(K=2)	2419	2.47	3107	1.70
Ligne1(K=3)	3049	3.03	1513	1.02
Ligne1(K=4)	1939	2.38	2487	2.44
Ligne2(K=2)	2488	1.89	1501	1.08
Ligne2(K=3)	1200	2.53	1666	1.44
Ligne2(K=4)	1387	2.97	2040	2.13

Table 5.3 Comparison of optimal cycle times for benchmark instances

Instances	K=2	K=3	K=4
BO1	255.2/255.2*	255.2/255.2	255.2/255.2*
BO2	255.2/255.2	255.2/255.2*	255.2/255.2
P&U	251/251*	205/198*	170/170
Ligne1	317.5/317.5	317.5/317.5	317.5/317.5*
Ligne2	675/675	675/675*	675/675*

We note that the optimal cycle times remain unchanged when the number of hoist increases to 3 and 4 for problems BO1, BO2, Ligne1 and Ligne2. We explain

the above observation as follows. In a multi-hoist system, the cycle time C is bounded from below by:

$$C \geq \max_{i \in N} (L_i + \varepsilon_i + \rho_i) . \tag{5.56}$$

That is to say, the cycle time C is greater than or equal to the sum of minimum processing time and the unloading and loading times in any tank. For problems BO1, BO2, Ligne1 and Ligne2, the optimal cycle time for K=2 reaches the lower bound given by (5.56). As a result, the optimal cycle time remains unchanged when the number of hoist increases. In other words, for these cases ($K \geq 2$), the critical resource becomes processing tanks and not transportation hoist.

5.5.2 Computational results on randomly generated instances

Randomly generated instances were also used to further evaluate the performance of our improved model. All the random instances were generated as described below. We set $K \in \{2, 3, 4\}$, and $n \in \{8, 10, 12, 14\}$. Let $U(a, b)$ be a uniform distribution between parameters a and b. The lower bound on processing time was generated as $L_i = U(50, 200)$. The upper bound on processing time was generated using the following three scenarios with different widths of time windows: $U_i = L_i$, $U_i = L_i + U(0, 50)$ and $U_i = L_i + U(0, 100)$. The travelling time between adjacent tanks was generated as follows: $e_{i, i+1} = U(2, 6)$. The travelling time between tank i and tank j can be computed with the formula $e_{i, j} = e_{j, i} = \sum_{k=i}^{j-1} e_{k,k+1}$, $i<j$, $i, j \in N^0 \cup \{n+1\}$. The loaded move time is computed by $d_i = 25 + e_{i, i+1}$, $i \in N^0$, where $\varepsilon_i + \rho_i = 25$, $i \in N$. For each given values of n and K, 20 random instances were generated.

Tables 5.4, 5.5 and 5.6 are used to test the effectiveness of the two improvements presented in subsection 5.4.2 of Section 4 under three scenarios $U_i = L_i$, $U_i = L_i + U(0, 50)$ and $U_i = L_i + U(0, 100)$, respectively. For each given values of n and K, the data for columns “B&B” and “CPU” in these tables represent the average size of branch-and-bound trees and average computation time (in CPU seconds) among 20 test instances, respectively. We can see from these tables that the B&B sizes explored by our improved model are generally smaller than those explored by the partially improved model. However, the computation times spent by our improved model are always shorter than those spent by the partially improved model.

Table 5.4 Comparison of computation times for random instances $U_i = L_i$

Random	Partially improved model		Our improved model		Ratio of
Instances	B&B	CPU	B&B	CPU	CPUs
n=8, K=2	1375	0.49	1075	0.31	1.58
n=8, K=3	1192	0.69	980	0.44	1.57
n=8, K=4	1337	0.99	984	0.48	2.06
n=10, K=2	3994	1.88	3382	1.26	1.49
n=10, K=3	5410	4.52	4783	2.44	1.85
n=10, K=4	3671	3.89	3121	1.96	1.99
n=12, K=2	6983	4.89	6514	3.11	1.57
n=12, K=3	12449	11.30	8504	4.72	2.39
n=12, K=4	5554	8.69	4947	3.95	2.20
n=14, K=2	11138	9.27	8753	5.05	1.84
n=14, K=3	51413	43.58	20324	11.15	3.91
n=14, K=4	263390	288.25	18562	11.38	25.33

Table 5.5 Comparison of computation times for random instances $U_i = L_i + U(0, 50)$

Random	Partially improved model		Our improved model		Ratio of
Instances	B&B	CPU	B&B	CPU	CPUs
n=8, K=2	1368	0.53	857	0.31	1.71
n=8, K=3	1592	0.89	1612	0.64	1.39
n=8, K=4	1209	0.94	1051	0.56	1.69
n=10, K=2	6028	2.91	5129	1.77	1.64
n=10, K=3	7252	5.92	6103	2.83	2.09
n=10, K=4	4283	4.39	4165	2.34	1.88
n=12, K=2	18644	9.40	15309	5.14	1.83
n=12, K=3	39609	24.19	27505	10.20	2.37
n=12, K=4	6844	9.21	13697	6.56	1.40
n=14, K=2	39998	23.63	34652	13.37	1.77
n=14, K=3	203217	150.39	112123	43.39	3.47
n=14, K=4	674087	696.77	128213	50.15	13.89

We explain the above observations as follows. In fact, our improved model is more compact than the partially improved model in terms of the number of variables and constraints. With our improved model, a smaller linear program is solved at each node, which requires shorter computation time at each node. Hence, our improved model is always more efficient (in terms of the computation time) than the partially improved model although the B&B size of the former is not always smaller than that of the latter. This means that the two improvements presented in subsection B of Section 5.4 are effective. Furthermore, we can also notice that the ratios of CPU times

spent by the partially improved model and our improved model increase generally with the values of n and K. Therefore, it seems that the larger the instance size, generally the more saving in computation time achieved by our improved model.

Table 5.6 Comparison of computation times for random instances $U_i = L_i + U(0, 100)$

Random Instances	Partially improved model		Our improved model		Ratio of CPUs
	B&B	CPU	B&B	CPU	
n=8, K=2	1514	0.58	1326	0.39	1.49
n=8, K=3	1773	0.93	1371	0.56	1.66
n=8, K=4	1203	0.95	1107	0.63	1.51
n=10, K=2	7833	3.73	5537	1.93	1.93
n=10, K=3	6206	5.13	4689	2.31	2.22
n=10, K=4	3334	3.80	2977	2.00	1.90
n=12, K=2	27397	12.52	21992	6.76	1.85
n=12, K=3	22239	16.30	15334	6.59	2.47
n=12, K=4	10798	10.58	16092	6.87	1.54
n=14, K=2	140203	82.14	79586	27.94	2.94
n=14, K=3	239951	177.27	154389	59.25	2.99
n=14, K=4	616542	722.49	261087	98.80	7.31

Table 5.7 Average number of improved instances with shorter cycles for random instances

Random Instances	$U_i=L_i$	$U_i=L_i+U(0,50)$	$U_i=L_i+U(0,100)$
n=8, K=2	4	2	0
n=8, K=3	12	1	1
n=8, K=4	10	1	0
n=10, K=2	2	4	2
n=10, K=3	14	3	1
n=10, K=4	13	2	0
n=12, K=2	9	3	2
n=12, K=3	10	6	2
n=12, K=4	15	4	1
n=14, K=2	3	1	2
n=14, K=3	14	6	3
n=14, K=4	12	2	2

Table 5.7 indicates that how many instances for which the optimal cycle time obtained with our improved model is smaller than that by Leung *et al*.'s model among 20 test instances. We can see from Table 5.7 that the number of improved instances seems to decrease generally with the width of the time windows. That is, the smaller

the width of the time windows, generally the larger the number of improved instances achieved by our improved model. We explain the above observation as follows. When the width of the time window is large, it provides a greater possibility of gaining a better solution with Leung *et al.*'s model by exploring the flexibility resulting from the time windows. Thus, it provides a smaller possibility of achieving a better solution with our improved model compared with the one obtained by Leung *et al.*'s model.

5.6 Summary

In this chapter, we gave a counterexample to demonstrate that the optimal solution obtained with the existing MIP approach for the multi-hoist cyclic scheduling problem with unidirectional part flow is not necessarily a global optimal solution. To find a global optimal solution, we proposed an improved MIP approach, in which loaded moves are allowed to go across the cycle. Computational results demonstrated that the smaller the width of the processing time windows, generally the greater possibility of achieving a better optimal solution by allowing the loaded moves to go across the cycle. The results also showed that our improved MIP approach is more efficient than Leung *et al.*'s MIP approach.

Chapter 6 Conclusions and Future Research

6.1 Conclusions

Hoist scheduling problem with processing time windows (HSP for short) is often encountered in surface treatment industry, which plays a key role in changing surface properties of metals and other electronics. A typical example from surface treatment industry is the automated electroplating plant, in which computer-controlled hoists are widely used to transport part from one processing stage to another. This research focused on the hoist scheduling issues arising from automated electroplating lines. More precisely, three typical hoist scheduling problems with processing time windows have been examined in this thesis: the basic cyclic HSP, the cyclic HSP with bi-objective and the cyclic HSP with multiple hoists. These scheduling problems are all NP-complete.

The main contributions of this thesis are summarized as follows. Firstly, we have proposed a hybrid QEA (HQEA) to find the best hoist move schedule with minimal cycle time for the basic HSP. As usual, each chromosome is encoded by Q-bits in the proposed HQEA. For a better population diversification, a new decoding scheme consisting of three different procedures was proposed for transforming Q-bits chromosome into hoist move sequences. It has several advantages over the commonly used ones, such as better ability to exploit the diversity of Q-bits chromosome and shorter length of chromosome. As infeasible hoist move sequences are inevitable, a simple and effective repairing procedure was designed to deal with this issue. Besides, quantum-rotation gate and adaptive genetic operators were applied to evolve the population towards best solution. The experimental results indicate that the proposed algorithm can provide high-quality solutions within a reasonable time. Our contribution was valorized through one communication (Lei *et al.*, 2013) and one submitted paper in the international journal *Applied Soft Computing* (Lei *et al.*, 2014).

Secondly, we formulated a mathematical model and proposed an efficient bi-objective QEA with local search (LS) procedure for a cyclic HSP with minimizing the cycle time and the production cost simultaneously. More precisely, a bi-objective mathematical model was formulated using the MPI approach (Levner *et al.*, 1997) providing that the actual processing times are known (In fact they are decision variables). After that, an efficient QEA with LS procedure was proposed for enumerating the actual processing times and finding a set of Pareto-optimal solutions

for the studied problem. Particularly, for providing a better diversity of population, each chromosome is converted into two different individuals by a double-decoding scheme. For finding the non-dominated individuals, Pareto-dominance procedure was suggested for individual evaluation. A specific chaotic quantum-rotation gate was designed for updating Q-bits individuals. To increase the diversity, mutation operator was also implanted. Moreover, an efficient LS procedure was periodically applied to improve all the non-dominated solutions stored in external archive.

A real zinc electroplating problem was used to investigate the performance of the proposed algorithm. We have run the bi-objective QEA algorithm with different parameter settings. For testing its performance, we also run the algorithm without LS procedure. Computational results show that the proposed algorithm is efficient in solving the studied problem, and the LS procedure is very helpful for improving the solution quality. Our results were presented at the international conference *IEEE ICIII* 2014 (Lei *et al.*, 2014).

At last, we have proposed an improved MIP model for the cyclic HSP with unidirectional multiple hoists to minimize the cycle time. Our improved MIP model was formulated with two improvements on Leung *et al.*'s MIP model (Leung *et al.*, 2004). The first improvement is the reformulation of the time window constraints by allowing the loaded hoist moves to start at the one cycle and end at the next one if necessary, which is a relaxation of the existing assumption that all loaded hoist moves start and end within the same cycle used in most related works, such as Leung *et al.* (2004), Chtourou *et al.* (2013) and Jiang and Liu (2014). The second one is to remove some unnecessary hoist collision-avoidance constraints from Leung *et al.*'s MIP model. Based on the above works, an improved and relatively more compact MIP model was formulated for the studied problem.

Computational results verify that our improved MIP approach can always find the global optimal solution for the studied problem, while the existing ones may identify a non-optimal solution to be an optimal one. Our results were published in the international journal *IEEE Transactions on Automation Science and Engineering* (Che *et al.*, 2014).

6.2 Limitations and future research

As described above, we have proposed efficient scheduling approaches for the

considered HSPs in this thesis. However, there are a lot of limitations in this search, so it still has enough room to conduct further research. In what follows, we discuss the limitations of this thesis and some potential directions for future research.

In chapter 3, the studied basic cyclic HSP only deals with a single part type. However, to improve the productivity and meet the diverse demands, multi-type parts are often produced within a same cycle in practice. Besides, duplicated tanks are often used to overcome the bottleneck processing stages in practices. Note that for HSP with multi-type parts and duplicated tanks, part input sequence must be optimized along with the sequencing of hoist moves. So how to extend the proposed HQEA for solving multi-type parts HSP with duplicated tanks is worth investigating in future. A key issue for the algorithm extension is to develop an efficient encoding and decoding scheme for sequencing of parts and hoist moves.

In chapter 4, optimizing HSP with two different objectives (i.e. cycle time and production cost) was investigated. To reduce the problem complexity, the second objective (i.e. the production cost) was supposed to be a linear function of the actual processing times. But from the practical point of view, a non-linear objective function may be more suitable for simulating the process of resource consumption. Thus, future interesting research direction is to introduce the non-linear objective function into the formulated bi-objective model. Moreover, it is also interesting to extend the proposed model and algorithm for solving the HSP with more than two objectives.

In chapter 5, all tanks are arranged in a row according to their index numbers, and each part is supposed to be processed through tank 1 to tank n. In other words, the part is moved in only one direction, i.e. from left to right. However, the part processing sequence may be different from the tanks layout in many real-world applications. Consequently, the hoist may move the part from left to right and from right to left. Therefore, how to extend the developed MIP model to the multi-hoist system with bidirectional part flow is worth investigating in future. Moreover, it is also worthwhile to develop efficient QEAs for multi-hoist scheduling problem with multiple objectives based on this research.

Bibliography

Alcaide, D., Chu, C., Kats, V., Levner, E., Sierksma, G., 2007. Cyclic multiple-robot scheduling with time-window constraints using a critical path approach. *European Journal of Operational Research*, 177(1): 147–162.

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.

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.

Baptiste, P., Legeard, B., Manier, M.-A., Varnier, C., 1993. Optimization with constraint logic programming: the hoist scheduling problem solved with various solvers. *Application of Artificial Intelligence in Engineering*, Toulouse, France, Elsevier Science Publishers B. V., Amsterdam, 2(June-July 1993): 599–614.

Bloch, C., Manier, M.-A., Baptiste, P., Varnier, C., 2008. Hoist scheduling problem. Chapter 8 in book: Production Scheduling, Control Systems, Robotics and Manufacturing Series. New York, NY, USA: Wiley, 2008, 193–231.

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.

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.

Che, A., Chu, C., 2007. Cyclic hoist scheduling in large real-life electroplating lines. *OR Spectrum*, 29(3): 445–470.

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.

Che, A., Zhou, Z., Chu, C., Chen, H., 2011. Multi-degree cyclic hoist scheduling with time window constraints. *International Journal of Production Research*, 49(19): 5679–5693.

Chen, H.X., Chu, C., Proth, J.M., 1998. Cyclic scheduling of a hoist with time window constraints. *IEEE Transaction on Robotics and Automation*, 14(1): 144–152.

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.

Crama, Y., Kats, V., Van de Klundert, J., Levner, E., 2000. Cyclic scheduling in robotic flowshops. *Annals of Operations Research*, 96(1–4): 97–124.

Deb, K., 2001. Multi-Objective Optimization Using Evolutionary Algorithms. Chichester, U.K.: Wiley.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2): 182–197.

Dettmer, R., 1993. Chaos and engineering, *IEE Review*, 39 (5): 199–203.

Deutsch, D., 1985. Quantum theory, the church-turing principle and the universal quantum computer. *Proceeding of the Royal Society of London*, 97–117.

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 Science and Techniques of Automatic control & Computer Engineering*, 11, special issue, 612–623.

El Amraoui, A., Manier, M.-A., El Moudni, A., Benrejeb, M., 2013a. A linear optimization approach to the heterogeneous r-cyclic hoist scheduling problem. *Computers & Industrial Engineering*, 65(3): 360–369.

El Amraoui, A., Manier, M.-A., El Moudni, A., Benrejeb, M., 2013b. A genetic algorithm approach for a single hoist scheduling problem with time windows constraints. *Engineering Applications of Artificial Intelligence*, 26(7): 1761–1771.

Fargier, H., Lamothe, J., 2001. Handling soft constraints in hoist scheduling problems: the fuzzy approach. *Engineering Applications of Artificial Intelligence*, 14(3): 387–399.

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.

Fleury, G., Gourgand, M., Lacomme, P., 2001. Metaheuristics for the stochastic hoist scheduling problem (SHSP). *International Journal of Production Research*, 39(15): 3419–3457.

Ge, Y., Yih, Y., 1995. Crane scheduling with time windows in circuit board production lines. *International Journal of Production Research*, 33(5): 1187–1199.

Gu, J., Gu, M., Cao, C., Gu, X., 2010. A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem. *Computers & Operations Research*, 37(5): 927–937.

Gu, J., Gu, X, Gu, M., 2009. A novel parallel quantum genetic algorithm for

stochastic job shop scheduling. *Journal of Mathematical Analysis and Applications,* 35(2009): 63–81.

Han, K.-H., Kim, J.-H., 2002. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE Transactions on Evolutionary Computation*, 6(6): 580–593.

Han, K.-H., Kim, J.-H., 2004. Quantum-inspired evolutionary algorithms with a new termination criterion, H-gate and two-phase scheme. *IEEE Transactions on Evolutionary Computation*, 8(2): 156–169.

Hey, T., 1999. Quantum computing: an introduction. *Computing&Control Engineering Journal*, 10(3): 105–112.

Hindi, K., Fleszar, K., 2004. A constraint propagation heuristic for the single-hoist, multiple-products scheduling problem. *Computers & Industrial Engineering*, 47(1): 91–101.

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.

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.

Kats, V., 1982. An exact optimal cyclic scheduling algorithm for multi-operator service of a production line. *Automation and Remote Control*, 42(4II): 538–543.

Kats, V., and Levner, E., 2011. Cyclic routing algorithms in graphs: Performance analysis and applications to robot scheduling. *Computers & Industrial Engineering*, 61(2): 279–288.

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.

Kats, V., Levner, E., 2011a. Parametric algorithms for 2-cyclic robot scheduling with interval processing times. *Journal of Scheduling*, 14(3): 267–279.

Kats, V., Levner, E., 2011b. 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.

Kim, J.H., Lee, T.E., 2008. Schedulability analysis of time-constrained cluster tools with bounded time variation by an extended Petri Net. *IEEE Transactions on Automation Science and Engineering*, 5(3): 490–503.

Kujawski, K., Świątek, J., 2011. Electroplating production scheduling by cyclogram unfolding in dynamic hoist scheduling problem. *International Journal of Production Research*, 49(17): 5355–5371.

Kuntay, I., Xu, Q., Uygun, K., Huang, Y., 2006. Environmentally Conscious Hoist Scheduling for Electroplating Facilities. *Chemical Engineering Communications*, 193(3): 273–292.

Lamothe, J., Correge, M., Delmas, J., 1995. A dynamic heuristic for the real-time hoist scheduling problem. *Proceedings of 1995 INRIA/IEEE Symposium on Emerging Technologies and Factory Automation (ETFA)*, 2: 161–168.

Lei, L., 1993. Determining the optimal starting times in a cyclic schedule with a given route. *Computers & Operations Research*, 20(8): 807–816.

Lei, L., Liu, Q., 2001. Optimal cyclic scheduling of a robotic processing line with two-product and time-window constraints. *INFOR*, 39(2): 185–199.

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.

Lei, L., Wang, T. J., 1994. Determining optimal cyclic hoist schedules in a single-hoist electroplating line. *IIE Transactions*, 26(2): 25–33.

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.

Lei, W., Che, A., Manier, H., Manier, M-A., 2014. Quantum-inspired evolutionary algorithm for bi-objective scheduling in an automated electroplating line. *7th International Conference on Information Management, Innovation Management and Industrial Engineering* (*ICIII*), 2:157–160, October 25–26, Xi'an, China.

Lei, W., Manier, H., Manier, M-A., 2013. A quantum-inspired evolutionary for the cyclic hoist scheduling problem. *Quatorzième Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision* (*ROADEF'13*), Troyes, France (13–15, février, 2013).

Lei, W., Manier, H., Manier, M-A., Che, A., 2014. A hybrid quantum evolutionary algorithm with improved decoding scheme for a robotic flowshop scheduling problem. *Applied soft computing*, under review.

Leung, J., Zhang, G.Q., 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.

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.

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, I-TECH Education and Publishing, Vienna,

Austria, 1–20.

Levner, E., Kats, V., de Pablo, D.A.L., Cheng, T.C.E., 2010. Complexity of cyclic scheduling problems: A state-of-the-art survey. *Computers & Industrial Engineering*, 59(2): 352–361.

Levner, E., Kats, V., Levit, V., 1997. An improved algorithm for cyclic flowshop scheduling in a robotic cell. *European Journal of Operational Research*, 97(3): 500–508.

Li, B., Wang, L., 2007. A Hybrid quantum-inspired genetic algorithm for multiobjective flow shop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 37(3): 576–591.

Li, P., Li, S., 2008. Quantum-inspired evolutionary algorithm for continuous spaces optimization based on Bloch coordinates of qubits. *Neurocomputing*, 72(1–3): 581–591.

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.

Lim, M.-J., 1997. A genetic algorithm for a single hoist scheduling in the printed-circuit-board electroplating line. *Computers & Industrial Engineering*, 33(3–4): 789–792.

Liu, J.Y., Jiang, Y., Zhou, Z.L., 2002. Cyclic scheduling of a single hoist in extended electroplating lines: a comprehensive integer programming solution. *IIE Transactions*, 34(10): 905–914.

Livshits, E.M., Mikhailetsky, Z.N., Chervyakov, E.V., 1974. A scheduling problem in an automated flow time with an automated operator. *Computational Mathematics and Computerized Systems*, 5, 151–155, in Russian.

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.

Manier, M.-A, Bloch, C., 2003. A classification for hoist scheduling problems. *International Journal of Flexible Manufacturing Systems*, 15(1): 37–55.

Manier, M.-A., 1994. Contribution à l'ordonnancement cyclique du système de manutention d'une ligne de galvanoplastie. PhD thesis in Automatics and Computer Science, Université de Franche-Comté, France.

Manier, M.-A., Lamrous, S., 2008. An evolutionary approach for the design and scheduling of electroplating facilities. *Journal of Mathematical Modelling and Algorithm*, 7(2): 197–215.

Manier, M.-A., Varnier, C., Baptiste, P., 2000. Constraint-based model for the cyclic multi-hoists scheduling problem. *Production Planning & Control*, 11(3):

244–257.

Miettinen, K., 1999. Nonlinear Multiobjective Optimization. Boston. MA: Kluwer Academic Publishers.

Narayanan, A., Moore, M., 1996. Quantum-inspired genetic algorithms. *Proceedings of IEEE International Conference on Evolutionary Computation*, 61–66.

Neto, J.X.V., De Andrade Bernert, D.L., Dos santos coelho, L., 2011. Improved quantum-inspired evolutionary algorithm with diversity information applied to economic dispatch problem with prohibited operating zones. *Energy Conversion and Management*, 52(2011): 8–14.

Ng, W. C., 1995. Determining the optimal number of duplicated process tanks in a single-hoist circuit board production line. *Computers & Industrial Engineering*, 28(4): 681–688.

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.

Ng, W. C., Leung, J., 1997. Determining the optimal move times for a given cyclic schedule of a material handling hoist. *Computers & Industrial Engineering*, 32(3): 595–606.

Ni, B., 2010. Galvanizing technology. Beijing: China Machine Press.

Niu, Q., Zhou, T., Ma, S., 2009. A quantum-inspired immune algorithm for hybrid flow shop with makespan criterion. *Journal of Universal Computer Science*, 15(4): 765–785.

Paul, H., Bierwirth, C., Kopfer, H., 2007. A heuristic scheduling procedure for multi-item hoist production lines. *International Journal of Production Economics*, 105(1): 54–69.

Phillips, L.W., Unger, P.S., 1976. Mathematical programming solution of a hoist scheduling program. *AIIE Transactions*, 8(2): 219–225.

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.

Schleinger, M., Paunovic, M., 2010. Modern electroplating, fifth edition, John Wiley & Sons, Inc., Hoboken, New Jersey.

Shapiro, G. W., Nuttle, H. L. W., 1988. Hoist scheduling for a PCB electroplating facility. *IIE Transactions*, 20(2): 157–167.

Spacek, P., Manier, M.-A., El Moudni, A., 1999. Control of an electroplating line in the max and min algebras. *International Journal of Systems Science*, 30(7): 759–778.

Subaï, C., Baptiste, P., Niel, E., 2006. Scheduling Issues for Environmentally

Responsible Manufacturing: The Case of Hoist Scheduling in an Electroplating Line. *International Journal of Production Economics*, 99(1–2): 74–87.

Talbi, H., Draa, A., Batouche, M., 2004. A new quantum-inspired genetic algorithm for solving the travelling salesman problem. *2004 IEEE International Conference on Industrial Technology*, 3: 1192–1197.

Tian, N., Che, A., Feng, J., 2013. Real-time hoist scheduling for multistage material handling process under uncertainties. *AIChE Journal*, 59(4): 1046–1048.

Varnier, C., Bachelu, A., Baptiste, P., 1997. Resolution of the cyclic multi-hoists scheduling problem with overlapping partitions. *Information System and Operations Research* (*INFOR*), 35(4): 309–324.

Wang, Y., Che, A., 2013. Robotic cells scheduling based on hybrid quantum evolutionary algorithm. *Computer Integrated Manufacturing Systems*, 19(9): 2193–2201, in Chinese.

Wu, N., Zhou, M., 2012a. Modeling, Analysis and control of dual-arm cluster tools with residency time constraint and activity time variation based on Petri Nets. *IEEE Transactions on Automation Science and Engineering*, 9(2): 446–454.

Wu, N., Zhou, M., 2012b. Schedulability analysis and optimal scheduling of dual-arm cluster tools with residency time constraint and activity time variation. *IEEE Transactions on Automation Science and Engineering*, 9(1): 203–209.

Xu, Q., Huang, Y. L., 2004. Graph-assisted cyclic hoist scheduling for environmentally benign electroplating. *Industrial and Engineering Chemistry Research*, 43(26): 8307–8316.

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(10): 128–140.

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.

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.

Yih, Y., 1994. An algorithm for hoist scheduling problems. *International Journal of Production Research*, 32(3): 501–516.

Yih, Y., Liang, T.-P., Moskowitz, H., 1993. Robot scheduling in a circuit board production line: a hybrid OR/ANN approach. *IIE Transactions*, 25(2): 26–33.

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.

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.

Zhang, R., Gao, H., 2007. Improved quantum evolutionary algorithm for combinatorial optimization problem. *2007 International Conference on Machine Learning and Cybernetics* (*ICLMC*), 6: 3501–3505.

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 and Chemical Engineering*, 50(5): 28–38.

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.

Zhao, Z., Peng, X., Peng, Y., Yu, E., 2006. An effective constraint handling method in quantum-inspired evolutionary algorithm for knapsack problems. *WSEAS Transactions on Computers*, 5(6): 1194–1199.

Zheng, T., Yamashiro, M., 2010. Solving flow shop scheduling problems by quantum differential evolutionary algorithm. *International Journal of Advanced Manufacturing Technology*, 49(5–8): 643–662.

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.

Zhou, Z., Li, L., 2003. Single hoist cyclic scheduling with multiple tanks: a material handling solution. *Computers & Operations Research*, 30(6): 811–819.

Zhou, Z., Li, L., 2009. A Solution for Cyclic Scheduling of multi-hoists without Overlapping. *Annals of Operations Research*, 168(1): 5–21.

Zhou, Z., Liu, J., 2008. A heuristic algorithm for the two-hoist cyclic scheduling problem with overlapping hoist ranges. *IIE Transactions*, 40(8): 782–794.

Cyclic Hoist Scheduling Problems in Classical and Sustainable Contexts

ABSTRACT

Automated surface treatment facilities, which employ computer-controlled hoists for part transportation, have been extensively established in various kinds of industrial companies, because of its numerous advantages over manual system, such as higher productivity, better product quality, and reduced labor intensity. This research investigates three typical hoist scheduling problems with processing time windows in surface treatment facilities, which are (I) cyclic single-hoist scheduling problem to minimize the cycle time; (II) cyclic single-hoist scheduling problem to minimize the cycle time and processing resource consumption (and consequently production cost); and (III) cyclic multi-hoist scheduling problem to minimize the cycle time.

Due to the NP-completeness of the studied problems and numerous advantages of quantum-inspired evolutionary algorithm (QEA), we first propose a hybrid QEA with improved decoding mechanism and repairing procedure to find the best cycle time for the first problem. After that, to enhance with both the economic and environmental performance, which constitute two of the three pillars of the sustainable strategy nowadays deployed in many industries, we formulate a bi-objective mathematical model for the second problem by using the method of prohibited interval (MPI). Then we propose a bi-objective QEA with local search procedure to simultaneously minimize the cycle time and the production cost, and we find a set of Pareto-optimal solutions for this problem. As for the third problem, we find that most existing approaches, such as mixed integer programming (MIP) approach, may identify a non-optimal solution to be an optimal one due to an assumption related to the loaded hoist moves which is made in many existing researches. Consequently, we propose an improved MIP approach for this problem by relaxing the above-mentioned assumption. Our approach can guarantee the optimality of its obtained solutions.

For each problem, experimental study on industrial instances and random instances has been conducted. Computational results demonstrate that the proposed scheduling algorithms are effective and justify the choices we made.

Keywords: cyclic hoist scheduling problem; processing time windows; bi-objective optimization; quantum-inspired evolutionary algorithm; mixed integer programming approach

RÉSUMÉ

Les ateliers de traitement de surface automatisés, qui utilisent des robots de manutention commandés par ordinateur pour le transport de la pièce, ont été largement mis en place dans différents types d'entreprises industrielles, en raison de ses nombreux avantages par rapport à un mode de production manuel, tels que: une plus grande productivité, une meilleure qualité des produits, et l'impact sur les rythmes de travail. Notre recherche porte sur trois types de problèmes d'ordonnancement associés à ces systèmes, appelés hoist scheduling problems, caractérisés par des contraintes de fenêtres de temps de traitement: (I) un problème à une seule ressource de transport où l'objectif est de minimiser le temps de cycle; (II) un problème bi-objectif avec une seule ressource de transport où il faut minimiser le temps de cycle et la consommation de ressources de traitement (et par conséquent le coût de production); et (III) un problème d'ordonnancement cyclique mono-objectif mais multi-robots.

En raison de la NP-complétude des problèmes étudiés et de nombreux avantages de les outils de type quantum-inspired evolutionary algorithm (QEA), nous proposons d'abord un QEA hybride comprenant un mécanisme de décodage amélioré et une procédure réparation dédiée pour trouver le meilleur temps de cycle pour le premier problème. Après cela, afin d'améliorer à la fois la performance économique et environnementale qui constituent deux des trois piliers de la stratégie de développement durable de nos jours déployée dans de nombreuses industries, nous formulons un modèle mathématique bi-objectif pour le deuxième problem en utilisant la méthode de l'intervalle interdit. Ensuite, nous proposons un QEA bi-objectif couplé avec une procédure de recherche locale pour minimiser simultanément le temps de cycle et les coûts de production, en générant un ensemble de solutions Pareto-optimales pour ce problème. Quant au troisième problème, nous constatons que la plupart des approaches utilisées dans les recherches actuelles, telles que la programmation entière mixte (MIP), peuvent conduire à l'obtention d'une solution non optimale en raison de la prise en compte courante d'une hypothèse limitant l'exploration de l'espace de recherche et relative aux mouvements en charge des robots. Par conséquent, nous proposons une approche de MIP améliorée qui peut garantir l'optimalité des solutions obtenues pour ce problème, en relaxant l'hypothèse mentionnée ci-dessus.

Pour chaque problème, une étude expérimentale a été menée sur des cas industriels ainsi que sur des instances générées aléatoirement. Les résultats obtenus montrent que l'efficacité des algorithmes d'ordonnancement proposés, ce qui justifie les choix que nous avons faits.

Mots-clés: ordonnancement cyclique des ateliers de traitement de surface, fenêtres de temps de traitement; optimisation bi-objectif; algorithme évolutionnaire quantique; approche de programmation mixte en nombres entiers.

SPIM

■ École doctorale SPIM - Université de Technologie Belfort-Montbéliard
F - 90010 Belfort Cedex ■ tél. +33 (0)3 84 58 31 39
■ ed-spim@univ-fcomte.fr ■ www.ed-spim.univ-fcomte.fr

utbm université de technologie Belfort-Montbéliard