Waseda University Doctoral Dissertation

Study on Quantum-Inspired Optimization Approaches for Flow Shop Scheduling Problems

Xin WEI

Graduate School of Information, Production and Systems
Waseda University

September 2012

Acknowledgements

This dissertation arose in part out of years of research that has been done since I came to Waseda University, IPS. By that time, I have worked with a great number of people whose contribution in assorted ways to the research and the making of the thesis deserved notable mention. It is a pleasure to convey my gratitude to them all in my humble acknowledgment.

In the first place, I would like to record my gratitude to Prof. Shigeru Fujimura for his supervision, advice, and guidance from the early stage of this research as well as giving me extraordinary experiences throughout the work. Above all and the most needed, he provided me unflinching encouragement and support in various ways. When I wrote the first journal paper, he helped me to check the words of manuscript one by one, carefully instructed the way how to express the ideas by using graphs and proper descriptions. And, I remember the many times that he invited me at the exclusive restaurants ordering my preferred drinks and paid the fees. I never forget that his recommendations help me to apply scholarship and hunt jobs. I never forget that he provided the funds for me to attend international conferences. In addition, in these years, he not only provided me the valuable suggestions, profound knowledge and rich experience, but also promoted me with his kindness, humor and patience. Overall, I am indebted to him more than he knows.

Besides, I also would like to express my sincere thanks to Prof. Osamu Yoshie and Prof. Tomohiro Murata, the committee members, for their careful reviews, constructive advices and helpful suggestions in my work.

I would like to express my gratitude to all my lab mates. Dr. Wei Weng and Wenqiang Zhang have always been patient to instruct me all along the ways and given me valuable suggestions for my research. Their professional advices can always encourage me never give up and continue to do the best. My thanks will also go to Mr. Kiyun Woo for his help in research and life. He is always willing to discuss my research and try to give suggestions. I should also extend my thanks to the other lab members: Mr. Yiqing Liu, Mr. Yi Sun, Mr. Qicong Shan, Mr. Wei Wei, Mr. Chenhua Ding, Mr. Dongzhe Zhang, Ms. Sicong Tan, Mr. Yiyong Jin, Ms. Pengfei Jiao, Ms. Ping Gu and etc. We have been together experiencing many things and this will definitely be one of the most precious memories. Special thanks should also be given to the former Japanese lab mates: Mr. Yamada, Mr. Takahashi and Mr. Morimoto. They are always willing to speak to me in Japanese and teach me a lot of things for a better life in Japan.

I also convey acknowledgement to my family members. My parents deserve particular mention for their inseparable support and prayers. My Father, Xunmin Wei in the first place is the person who put the fundament my learning character, showing me the joy of intellectual pursuit ever since I was a child. My Mother, Hongying Zhang, is the one who sincerely raised me with her caring and gently love. Erxin Wei, my brother, thanks for being supportive and caring parents.

Words fail me to express my appreciation to my wife Jia Su whose dedication, love and persistent confidence in me. I owe her for being unselfishly let her intelligence, passions, and ambitions collide with mine. Therefore, I would also thank Su's family for letting me take her hand in marriage, and accepting me as a member of the family, warmly.

Finally, I would like to thank everybody who was pivotal to the successful realization of this dissertation, as well as expressing my apology that I could not mention personally one by one.

Contents

List of Figures

List of Tables

LIST OF TABLES

Chapter 1

Introduction

1.1 Research Background and Motivation

Scheduling is a decision making practice that plays a decisive role in most manufacturing and services industries, which has significant impact on increased productivity, customer satisfaction, cost reduction and overall competitive advantage. However, with the development of modern manufacturing industries, the customers are demanding high-variety products, which have contributed to an increase in product complexity. Therefore, the effective scheduling system becomes more essential to face the competitive markets. An effective scheduling leads to increase in capacity utilization efficiency, thereby shortening the time required to complete jobs and consequently increasing the profitability of an organization in present competitive environment. The Flow Shop Scheduling Problem (FSSP) is generally considered to be one of the most critical issues in manufacturing industries of steel, semiconductor, textile, furniture and etc.

In flow shop production environment, the permutation FSSP and reentrant FSSP are two representative scheduling problems, which are most investigated in this dissertation. The jobs of permutation FSSP are to be processed through a series of machines for optimizing number of required criteria. The permutation FSSP is proved to be NP-hard and even small size problems are difficult to solve. The reentrant

FSSP is a more complex problem than permutation FSSP. It usually appears in the semiconductor industries, in which the routes of jobs on the machines are identical as in permutation FSSP, but the jobs must be processed by several machines in multiple times. Minimizing makespan (the time interval required to complete all the jobs) is one of the most essential criteria for the manufacturing industries to reduce cost and improve productivity. The first target of this dissertation is to minimize makespan for permutation FSSP. Furthermore, the highly competitive industries, such as food, beverage and semiconductor industry, also need on-time delivery to response the stress of competition on the markets. These industries have to offer a great variety of different and individual products while customers are expecting ordered goods to be delivered on time. Hence, there is a requirement of multi-objective scheduling system through which all the objectives can be achieved simultaneously. To solve the multi-objective permutation FSSP and reentrant FSSP is the second target in this dissertation.

Due to the great complexity of permutation FSSP and reentrant FSSP, many researchers have dealt with such problems using some heuristic approaches, such as genetic algorithm, simulated annealing, taboo search and etc. However, based on the traditional methods it is hard to deliver an effective solution in a limited computation time, especially for large size problems. Since last decades, numerously prospective researches have demonstrated that quantum computing is much efficient and fast for solving various complex problems on the quantum computer. Recently, the merging evolutionary computation and quantum computing have taken a great progression. Inspired by the concept of quantum computing, a new approach called Multi-update Mode Quantum Evolutionary Algorithm (MMQEA) has been proposed. MMQEA is innovated by q-bit individual and q-gate. The q-bit individual has a probabilistic representation of the solutions inspired by the superposition of q-bit. The q-gate with multi-update mode is used to renovate the quantum states of q-bit individual to evolve towards better solutions. The numerous experiments have verified that MMQEA have good global search ability, fast convergence behavior to address permutation FSSP with makespan criterion.

Based on the achievement of MMQEA, an effective multi-objective optimization algorithm called Parallel Quantum Evolutionary Algorithm (PQEA) has been developed to solve the multi-objective permutation FSSP. PQEA, a universal computation technique, is applicable for solving a variety of multi-objective optimization problems. The experimental results on benchmark problems has demonstrated that PQEA advances the most of famous algorithms both in low computational cost and dominated solution attainment. Although PQEA has already archived better solutions on the multi-objective permutation FSSP, there is still some space to promote more. To further improve solution quality and accelerate convergence of PQEA, Alternated Neighborhood Search (ANS) procedure is proposed. The wide search space is exploited by the interaction of neighborhoods in ANS which combines with PQEA obviously improved the convergence speed without cost of any external computation time.

In order to verify the practical applicability of developed approach, PQEA has been tested in a multi-objective reentrant FSSP of a real-world semiconductor factory. Due to the complexity of reentrant FSSP, the Quantum-to-Job (Q-to-J) method that transforms quantum states to real reentrant-job's identifiers is proposed. The results show that solutions obtained by PQEA have significantly improved both makespan and on-time delivery criteria, comparing with the previous used one in this factory.

1.2 Scope of Present Work

In present work, quantum-inspired optimization approaches have been proposed for single objective and multi objective flow shop scheduling problems. The MMQEA and PQEA are developed for single objective and multi-objective flow shop scheduling problems with the performance measures including maximum tardiness, makespan, penalty cost for tardy jobs. This work will also benefit the development of optimization techniques that can be applied to other combinatorial optimization problems. Further, present works demonstrate the suitability and applicability of MMQEA and PQEA on single and multi objective knapsack problems. The ANS is neighborhood

based local search strategy, which can also been applied for scheduling the jobs in a job shop or open shop manufacturing system with other performance measures such as total flow time, total weighted tardiness, number of tardy jobs and etc. Proposed approaches are practical for scheduling any number of jobs on m-machines in production shop environment for meeting the said multi-objectives for achieving organizational and individual goals.

1.3 Outline of This Dissertation

This dissertation falls into six chapters. The background, problem, motivations are described in chapter 1. Chapter 2 introduces the MMQEA for permutation FSSP with makespan criterion. Chapter 3 presents the proposed PQEA framework to solve the multi-objective permutation FSSP in which the makespan and maximum tardiness are optimized, simultaneously. To further improve the performance of PQEA, Chapter 4 describes neighborhood based local search ANS which consists of basic procedure and two neighborhood structures MOINS and MOEXC. Chapter 5 states the proposed Q-to-J and combines with PQEA to solve a scheduling problem in a real-world semiconductor factory, providing the comparison of system performance before and after using proposed approach. Finally, conclusions and some suggestions for further work in this area are presented in chapter 6.

Chapter 2

MMQEA for Permutation FSSP with Makespan Criterion

2.1 Introduction

In artificial intelligence, an evolutionary algorithm (EA) is a subset of evolutionary computation and a genetic population based meta-heuristic optimization algorithm. EA uses some mechanisms inspired by biological evolution: reproduction, mutation, recombination, and selection. Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness function determines the environment within which the solutions "live". EA differs with traditional optimization techniques in that it involves a search from a "population" of solutions, not from a single point, and applies the principle of survival of the fittest to produce successively better approximations to a solution. The "recombination" and "mutation" are used in EA to generate new approximate solutions that are biased towards regions of the space for which good solutions have already been seen. Evolution of the population takes place after the repeated application of the above operators. Up to now, several different types of evolutionary search methods were developed independently, such as evolutionary programming, genetic algorithms and etc.

2. MMQEA FOR PERMUTATION FSSP WITH MAKESPAN CRITERION

Quantum computer was proposed in the early 1980s [1], [2], which is a device for computation that makes direct use of quantum mechanical phenomena, such as superposition and entanglement, to perform operations on data. The basic principle behind quantum computation is that quantum properties can be used to represent data and perform operations on data [3]. The most famous quantum algorithms such as Shors quantum factoring algorithm [4],[5] and Grovers database search algorithm [4], [6] demonstrate that quantum computer is more powerful than classical computers on specialized problem. In recently, research on merging evolutionary computing and quantum computing has been progressed actively, such as paper [7], [8], [9] and etc. Han and Kim proposed quantum evolutionary algorithm (QEA) that integrated some concept and principles of quantum computer such as a quantum bit (q-bit) and superposition state to EA. The result of QEA shows that it performs well, even with a small population, without premature convergence compared with the conventional GA. Due to it's high efficacy, QEA has been applied to solve many kinds of NP-hard problems, such as the researches [10], [11] and etc. QEA uses a q-bit as a probabilistic representation, defined as the smallest unit of information. The q-bit individual consists of a string of q-bit, which has the advantage that it can represent a linear superposition of states (binary solutions) in search space probabilistically. The quantum gate is designed as variational operator to update the state of q-bit in individual to search the fitter solution. However, in QEA, each q-bit individual evolves independently, in which the state of q-bit is updated on only situation, that causes the convergence of algorithm in early and effects on the search capability directly, especially in a large population.

Therefore, a novel approach of multi-update mode and corresponding individual structure are proposed. Inspiration from the characteristic quantum entanglement, an individual of MMQEA comprises two interactional q-bit strings. Each q-bit string of the individual provides its evolutionary information to other one and also can receive information from the interactional one in every generation, which maintains the population more diversified and avoid premature. Multi-update mode is proposed as a variational operator to drive the individuals toward better solutions and eventually toward a single state. By these inherent mechanism, the proposed algorithm

can treat the more balance between exploration and exploitation compared with previous quantum evolutionary algorithm. To demonstrate the applicability and efficiency of MMQEA, the permutation FSSP and knapsack problems are applied to test the performance, and comparing with GA and previous QEA.

This chapter is organized as follows. Section 2 firstly introduces some basics of quantum computing and presents the MMQEA details in later. Section 3 applies MMQEA to knapsack problems and permutation FSSP, summarizes the experimental results and analyzes the characteristics of MMQEA. Concluding remarks follow in Section 4.

2.2 Main Framework of MMQEA

2.2.1 Basics of Quantum Computing

Before describing MMQEA, the basics of quantum computing are presented briefly in the following. A q-bit is a smallest unit of quantum information. It has some similarities to a classical bit, but is overall very different. The difference is that whereas a bit must be either 0 or 1, a q-bit can be 0, 1, or a superposition of both. So a q-bit has a characteristic that it can represent three states simultaneously. The state of a q-bit can be represented as Eq. 2.1.

$$|\psi\rangle = \gamma|0\rangle + \eta|1\rangle \tag{2.1}$$

Here, γ and η are probability amplitudes which in general are complex numbers. When we measure this q-bit in the standard basis, the probabilities of outcome $|0\rangle$ and $|1\rangle$ are $|\gamma|^2$, $|\eta|^2$, respectively. As the absolute squares of the amplitudes are probabilities, γ and η must follow the below constraint.

$$|\gamma|^2 + |\eta|^2 = 1 \tag{2.2}$$

In quantum computing, quantum gates, which are represented by matrices, can

be visualized as rotations of the quantum state on the Bloch sphere [12]. It is a reversible gate and can be represented as a unitary operator U acting on the q-bit basis states satisfying $U' * U = U * U'$, where U' is the hermitian adjoint of U. There are several quantum gates, Rotation gate and Hadamard gate and etc, for various quantum algorithms to update the states of quantum system [13].

In MMQEA, the q-bit is also treated as basic unit information. MMQEA is designed with a novel q-bit individual and the multi-update mode as a variational operator, which are presented in following section.

2.2.2 Proposed Quantum Individual Structure

The q-bit string (object) with m q-bits is defined as:

$$q = \left[\begin{array}{c|c|c|c} \gamma_1 & \gamma_2 & \cdots & \gamma_m \\ \eta_1 & \eta_2 & \cdots & \eta_m \end{array} \right] \tag{2.3}$$

where, $|\gamma_i|^2 + |\eta_i|^2 = 1$, $i = 1, 2, ..., m$. If there is, for instance, a three q-bit string with three pairs of amplitudes such as:

$$\left[\begin{array}{c|c|c} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{2} \end{array} \right] \tag{2.4}$$

Then, states of the q-bit string can be represented as:

$$\begin{aligned} & -1/4|000\rangle - \sqrt{3}/4|001\rangle + 1/4|010\rangle \\ & + \quad \sqrt{3}/4|011\rangle - 1/4|100\rangle - \sqrt{3}/4|101\rangle \\ & + \quad 1/4|110\rangle + \sqrt{3}/4|111\rangle \end{aligned} \tag{2.5}$$

The above result means that the probabilities of the states $|000\rangle$, $|001\rangle$, $|010\rangle$, $|011\rangle$, $|100\rangle$, $|101\rangle$, $|110\rangle$ and $|111\rangle$, are 1/16, 3/16, 1/16, 3/16, 1/16, 3/16, 1/16 and 3/16, respectively, and also indicates that this q-bit string contains the information of eight states.

In quantum computing, if there is a system with two or more objects, quantum entanglement is a property of the system in which the quantum states of the these objects are linked together so that the quantum states of one object can no longer be adequately described without full mention of its counterpart [14]. It means that these objects in system interact with each other. The state of one object is not entirely independent of other states, and its state is dependent on another state in some way. The most EAs individuals are lack of evolving themselves to adjust the balance of exploitation and exploration, on which rely on biological operator to maintain the population diversity. The individual in MMQEA inspired by quantum entanglement, which composed of a pair of q-bit strings, the each q-bit string is not evolved lonely, while the other q-bit string provides its evolutionary history information to it to birth next generational q-bit string. So the MMQEA has the ability to keep population diversity by individual itself. The individual of MMQEA is called Dq-bits (double q-bit strings). The individual structure is defined as:

Definition 1: A Dq-bits individual is composed of a pair of q-bit strings are given as

$$d_j(\mathrm{t}) = \begin{bmatrix} q_j^{\alpha}(t) \\ q_j^{\beta}(t) \end{bmatrix} \tag{2.6}$$

Where t is generation times, j is the index of individual in population. The length of q-bit string is m. α and β are q-bit update modes, will be introduced in next subsection. $q_j^{\alpha}(t)$ and $q_j^{\beta}(t)$ are q-bit strings, the states of which are updated by α-update and β-update mode respectively. They are shown in Eq. 2.7.

$$\begin{aligned} q_j^{\alpha}(t) &= \left[\begin{array}{c|c|c|c} \gamma_{\alpha j1}^t & \gamma_{\alpha j2}^t & \cdots & \gamma_{\alpha jm}^t \\ \eta_{\alpha j1}^t & \eta_{\alpha j2}^t & \cdots & \eta_{\alpha jm}^t \end{array}\right] \\ q_j^{\beta}(t) &= \left[\begin{array}{c|c|c|c} \gamma_{\beta j1}^t & \gamma_{\beta j2}^t & \cdots & \gamma_{\beta jm}^t \\ \eta_{\beta j1}^t & \eta_{\beta j2}^t & \cdots & \eta_{\beta jm}^t \end{array}\right] \end{aligned} \tag{2.7}$$

Where, γ and η are coefficient of $|0\rangle$ and $|1\rangle$ as represented in last section. MMQEA is a probabilistic algorithm similar with the other EAs, however, maintains

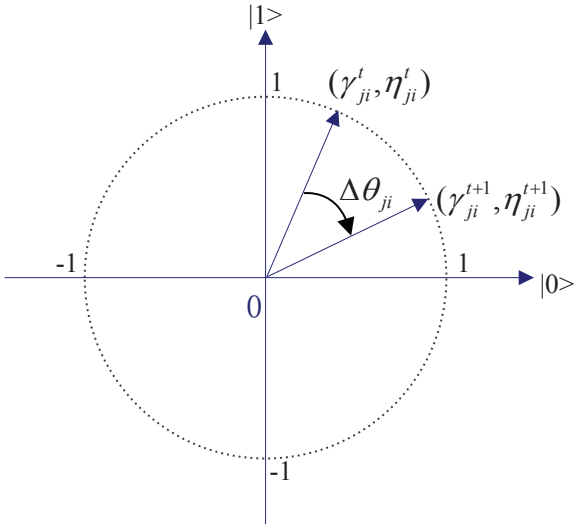


Figure 2.1: Polar plot of rotational gate

a population of Dq-bits individuals, as $MMQ(t) = \{d_1(t), d_2(t), \ldots, d_n(t)\}$, n is size of population.

2.2.3 Quantum Gate Instruction

In quantum computing, q-gate usually is a reversible unitary matrix [15]. Han and Kim designed a q-gate for QEA [16], as Eq. 2.8. The q-gate $U(\Delta\theta_{ji})$ is employed to update the state of q-bit of individual.

$$\mathbf{U}(\Delta\theta_{ji}) = \begin{bmatrix} \cos(\Delta\theta_{ji}), & -\sin(\Delta\theta_{ji}) \\ \sin(\Delta\theta_{ji}), & \cos(\Delta\theta_{ji}) \end{bmatrix} \tag{2.8}$$

The $\Delta\theta_{ji}$ is a rotation angle of each q-bit toward either $|0\rangle$ or $|1\rangle$ state. A q-bit through the q-gate updating as Eq. 2.9 can get a new q-bit and always the next generation q-bit states satisfied $\left|\gamma_{ji}^{t+1}\right|^2 + \left|\eta_{ji}^{t+1}\right|^2 = 1$.

$$\begin{bmatrix} \gamma_{ji}^{t+1} \\ \eta_{ji}^{t+1} \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta_{ji}), & -\sin(\Delta\theta_{ji}) \\ \sin(\Delta\theta_{ji}), & \cos(\Delta\theta_{ji}) \end{bmatrix} * \begin{bmatrix} \gamma_{ji}^{t} \\ \eta_{ji}^{t} \end{bmatrix} \tag{2.9}$$

Algorithm 1: Main Procedure of MMQEA

Input: T- Maximal Generation times, m- Length of q-bit String, n- Population Size.
Output: b- Final Founded Best Solution.

```
begin
    t ← 0;
1   Initialize MMQ(t);
2   Make P_c(t) by observing the states of q_j^α(t) and q_j^β(t) in MMQ(t);
3   Evaluate P_c(t);
4   Store the best solutions among P_c(t) into B(t);
    while t < T do
        t ← t + 1;
5       Make P_c(t) by observing the states of q_j^α(t − 1) and q_j^β(t − 1);
6       Evaluate P_c(t);
7       Store the best solutions among B(t − 1) and P_c(t) into B(t);
8       Store the best solution b among B(t);
9       Update q_j^α(t) and q_j^β(t) using α-update mode and β-update mode,
        respectively;
10      if immigration condition satisfied then
            Cover all the items of B(t) with b;
        end
    end
end
```

The polar plot of rotational q-gate for updating the state of a q-bit is depicted in Fig. 2.1. We should design the value of angle parameters $\Delta\theta_{ji}$ for the different kinds of problems to control the convergence of algorithm. In this chapter, the rotated direction on clockwise is considered as positive rotation.

2.2.4 Proposed Procedure of MMQEA

The structure of MMQEA is described in the Algorithm 1.

(1) MMQ(t) is initialized. All of q-bit states of individuals are given as:

$$\begin{bmatrix} \gamma_{\alpha ji}^t \\ \eta_{\alpha ji}^t \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \quad \begin{bmatrix} \gamma_{\beta ji}^t \\ \eta_{\beta ji}^t \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \tag{2.10}$$

It means that all the individuals that consist of q-bit with the same states, represent the linear superposition of all possible states with same probability in the first step. Where, i is the position of q-bit in the q-bit string, from $[1, m]$, j is the individual index in population, from $[1, n]$.

(2) The $P_c(t)$ is population of solutions as Eq. 2.11. The binary solutions in $P_c(t)$ are transformed from the q-bit strings by observing procedure. The observing procedure is described as following. Let δ be a random number generated from the uniform distribution $[0, 1]$. If $\eta^t_{\alpha ji}$ satisfies $\left|\eta^t_{\alpha ji}\right|^2 > \delta$, then the bit $x^\alpha_{ji}(t)$ of the binary string is set to 1, otherwise set to 0. Thus, binary string $x^\alpha_j(t) = \{x^\alpha_{j1}(t), x^\alpha_{j2}(t), ..., x^\alpha_{jm}(t)\}$ is formed. The same way is applied to produce $x^\beta_j(t)$. The binary solutions $x^\alpha_j(t)$ and $x^\beta_j(t)$ are observed from $q^\alpha_j(t)$, $q^\beta_j(t)$, respectively.

$$P_c(t) = \begin{pmatrix} x^\alpha_1(t), & x^\alpha_2(t), & ..., & x^\alpha_n(t) \\ x^\beta_1(t), & x^\beta_2(t), & ..., & x^\beta_n(t) \end{pmatrix} \tag{2.11}$$

(3) All the binary solutions $x^\alpha_j(t)$ and $x^\beta_j(t)$ are evaluated to give their fitness values.

(4) The initial best solutions are selected among the binary solutions in $P_c(t)$, and stored into $B(t)$, where $B(t) = \{b_1(t), b_2(t), \ldots, b_n(t)\}$, $b_j(t) = \max\{x^\alpha_j(t), x^\beta_j(t)\}$[1]. $x^\alpha_j(t)$ and $x^\beta_j(t)$ are observed by two q-bit strings of the jth individual, until here $t = 0$.

(5, 6) In the repetition loop, binary solutions in $P_c(t)$ are formed by observing $x^\alpha_j(t-1)$ and $x^\beta_j(t-1)$ as the step 2, each binary solution is evaluated for the fitness value.

(7) In this step, $B(t)$ is updated. $B(t)$ saves n current best solutions that are used to update the states of q-bit strings in each individual. If the solution $x^\alpha_j(t)$ or $x^\beta_j(t)$ is better than $b_j(t-1)$, then $b_j(t) \leftarrow x^\alpha_j(t)$ or $x^\beta_j(t)$, otherwise $b_j(t) \leftarrow b_j(t-1)$.

(8) The solution b, which is selected the best one from $B(t)$, is recorded as the current global best solution.

[1]If the optimization problem is for minimization, $b_j(t) = \min\{x^\alpha_j(t), x^\beta_j(t)\}$.

Table 2.1: Rotational direction θ_i

	x_{ji}	$b_{ji}(t)$	$\gamma^t_{\alpha ji} * \eta^t_{\alpha ji} \geq 0$	$\gamma^t_{\alpha ji} * \eta^t_{\alpha ji} < 0$
α update mode	0	0	0	0
	0	1	$+\Delta\theta_i$	$-\Delta\theta_i$
	1	0	$-\Delta\theta_i$	$+\Delta\theta_i$
	1	1	0	0
β update mode	0	0	0	0
	0	1	$-\Delta\theta_i$	$+\Delta\theta_i$
	1	0	$+\Delta\theta_i$	$-\Delta\theta_i$
	1	1	0	0

(9) Update procedure is a core technique of MMQEA. For describing the update approach, we should firstly denote that "F" is objective function, which is used to calculate a binary solution to a value, the binary solution $x_j = x^\alpha_j(t)$ or $x^\beta_j(t)$ and $b_j(t)$ is the jth item in $B(t)$. The q-gate represented in last section used to update all the q-bits of individual, where $\Delta\theta_{ji}$ is a rotational angle of each q-bit toward either $|0\rangle$ or $|1\rangle$ state in polar coordinates. The rotational direction of $\Delta\theta_{ji}$ is obtained as a function of $b_{ji}(t)$ (the ith bit of the $b_j(t)$) and x_{ji} (the ith bit of the binary solution x_j). We should contrast the binary numbers that are in the same positions of $b_j(t)$ and x_j to deicide rotational direction of $\Delta\theta_{ji}$ for the q-bit in same position of q-bit string. The rotational directions are set by some intuitive reasons. There are four kinds of situation to contrast two binary numbers to decide rotational direction, which are represented as Table 2.1. The absolute value of $\Delta\theta_{ji}$ should be designed in compliance with the application problem, usually from 0 to 0.02π [11]. In MMQEA, an individual is consisted of double q-bit strings, while they are updated by different strategies which are called α-update mode and β-update mode defined as bellow:

Definition 2: α-update mode: Binary solution x_j used to update $q^\alpha_j(t)$, only if $F(x_j) \geq F(b_j(t))$ [1], where, rotated direction of $\Delta\theta_{ji}$ as Table 2.1.

The means of "+" and "−" in the above tables is that rotated angle on clockwise or anticlockwise to change the probability of the 1 (or 0) state to that of 0 (or 1)

[1]If the optimization problem is for minimization, condition in Definition 2 and Definition 3 change to $F(x_j) < f(b)$ and $F(x_j) \geq f(b)$, respectively.

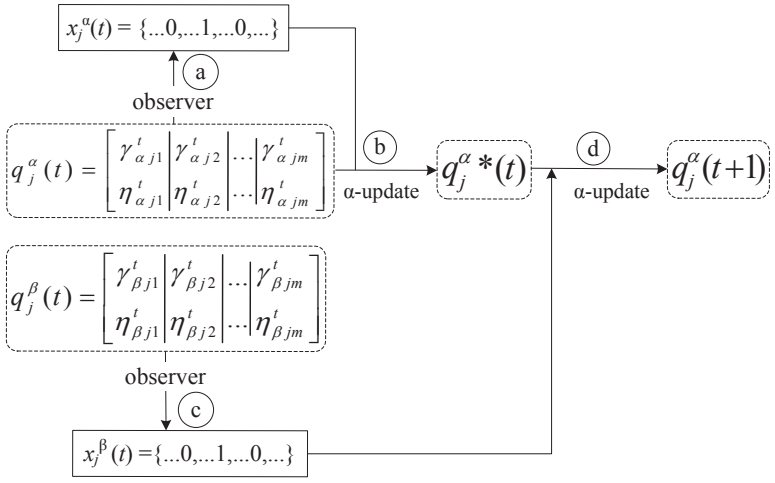


Figure 2.2: One part of multi update mode

state. The state of q-bit also decides the rotated direction, if the q-bit located in the first or third quadrant rotated direction is opposite with in second or fourth quadrant.

Definition 3: β**-update mode**: Binary solution x_j used to update $q_j^{\beta}(t)$, only if $F(x_j) < F(b_j(t))$, where, rotated direction of $\Delta\theta_{ji}$ as Table 2.1.

The q-bit strings $q_j^{\alpha}(t)$ and $q_j^{\beta}(t)$ have the evolved history information of themselves by α and β update modes respectively, after some generational evolution. If the $q_j^{\alpha}(t)$ and $q_j^{\beta}(t)$ are always evolved by above mentioned α and β independently, obviously, they are both easily trap on local optimum. So this chapter proposes a multi-update structure that makes $q_j^{\alpha}(t)$ and $q_j^{\beta}(t)$ are not evolved alone, in every generation while each q-bit string is updated, it always considers the other ones history evolutionary information. As at the generation t, firstly, both $q_j^{\alpha}(t)$ and $q_j^{\beta}(t)$ are observed to get binary strings $x_j^{\alpha}(t)$ and $x_j^{\beta}(t)$ (as `a` and `c` procedure in Fig. 2.2), respectively. The $F(x_j^{\alpha}(t))$, $F(x_j^{\beta}(t))$ and $F(b_j(t))$ values of the solutions $x_j^{\alpha}(t)$, $x_j^{\beta}(t)$ and $b_j(t)$ decide the update mode on the individual. If the condition "$F(x_j^{\alpha}(t)) \geq F(b_j(t))\&\&F(x_j^{\beta}(t)) \geq F(b_j(t))$" is satisfied, the update mode that is presented in Fig. 2.2 is employed. Since $x_j^{\alpha}(t)$ and $x_j^{\beta}(t)$ are both fitter that current best solution $b_j(t)$, the state of q-bit string $q_j^{\alpha}(t)$ is updated twice by $x_j^{\alpha}(t)$ and $x_j^{\beta}(t)$

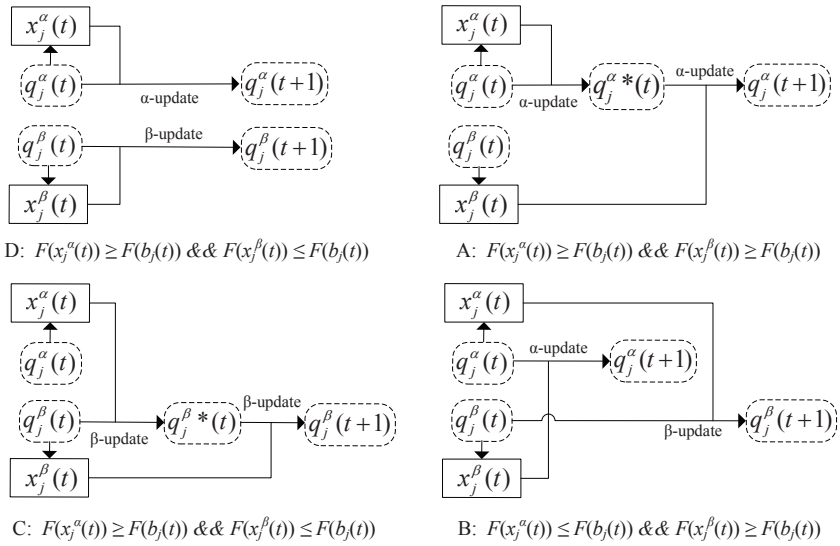


Figure 2.3: Status of multi update mode

(`b` and `d` procedures in Fig. 2.2), whereas the state of $q_j^{\beta}(t)$ is not updated to keep the foregone state.

There are four situations to compare the value of $F(x_j^{\alpha}(t))$, $F(x_j^{\beta}(t))$ and $F(b_j(t))$. The multi-update structure is as Fig. 2.3. The update methods for updating individual depend on magnitude $F(x_j^{\alpha}(t))$, $F(x_j^{\beta}(t))$ and $F(b_j(t))$, which present at the part A, B, C and D of Fig. 2.3. Part A is same as Fig. 2.2. Part B: if $b_j(t)$ is worse than $x_j^{\beta}(t)$ but fitter than $x_j^{\alpha}(t)$, $x_j^{\beta}(t)$ is copied to x_j ($x_j^{\beta}(t) \rightarrow x_j$), Then, the states of $q_j^{\alpha}(t)$ are updated by α-update on x_j and $b_j(t)$, at one time $x_j^{\alpha}(t) \rightarrow x_j$, β-update is processed on $q_j^{\beta}(t)$ with x_j and $b_j(t)$, to get $q_j^{\alpha}(t+1)$ and $q_j^{\beta}(t+1)$, respectively. If $b_j(t)$ is better than $x_j^{\alpha}(t)$ and $x_j^{\beta}(t)$, firstly $x_j^{\beta}(t)$ and $b_j(t)$ are used to update the state of $q_j^{\beta}(t)$ on β-update to get a new q-bit string $q_j^{\beta^*}(t)$, and then $x_j^{\alpha}(t) \rightarrow x_j$, β-update is continuously processed to update $q_j^{\beta^*}(t)$ with x_j and $b_j(t)$ to get $q_j^{\beta}(t+1)$, as Part C. Part D: if $b_j(t)$ is worse than $x_j^{\alpha}(t)$ but fitter than $x_j^{\beta}(t)$, α-update and β-update are processed on $x_j^{\alpha}(t)$, $q_j^{\alpha}(t)$ and $x_j^{\beta}(t)$, $q_j^{\beta}(t)$ respectively, to get $q_j^{\alpha}(t+1)$ and $q_j^{\beta}(t+1)$.

(10)The immigration condition is designed as a parameter of generation times. The immigration mechanism is used to lead MMQEA having a fast convergence for

finding the optimal solution. Here, "t_m" and "t" are immigration-term and current generation time, respectively. If the remainder of "t/t_m" is zero, best solution b is transferred to cover for all items of $B(t)$. All the individuals have a same global best solution b to guide the evolvement of q-bit strings in individuals.

2.3 Comparisons and Experimental Results Analysis

2.3.1 MMQEA Application for Knapsack Problems

2.3.1.1 Knapsack Problem and Test Data

The knapsack problem [17], a representative combinatorial optimization problem, is used to investigate the performance of MMQEA and to demonstrate the applicability of the proposed algorithm. It can be described as selecting from among various items, which are most profitable, and the given knapsack has limited capacity. The knapsack problem is formalized as:

Given a set of m items and a knapsack, select a subset of the items so as to maximize the profits $f(x)$:

$$f(x) = \sum_{i=1}^{m} p_i * x_i \tag{2.12}$$

Subject to

$$\sum_{i=1}^{m} w_i * x_i \leq C \tag{2.13}$$

Where, $x = (x_1, x_2, \ldots, x_m)$, x_i is 0 or 1, p_i is the profit of item i, w_i is the weight of item i, and C is the capacity of knapsack. If $x_i = 1$, the ith item is selected for the knapsack. In this chapter, strongly correlated sets of data are considered as w_i = uniformly random $[1, v]$; $p_i = w_i + r$; $v = 10$ and $r = 5$.

The following average knapsack capacity was used:

$$C = \frac{1}{2}\sum_{i=1}^{m} w_i \tag{2.14}$$

The data are unsorted. Three knapsack problems with 100, 250, and 500 items were considered.

2.3.1.2 Implementation on Knapsack Problem

MMQEA for the knapsack problem consists of a basic structure of MMQEA and a repair procedure to satisfy the capacity constraint. An MMQEA individual with a pair of q-bit strings that have m q-bits represents two linear superposition solutions at one time to the problem. The length of q-bit string is same as the number of items. Firstly, the all q-bits of individuals are initialized with same states. It means that all the items are selected with the same probability. Since the evolved state of q-bit is a similar probabilistic evolution, the observing procedure, which makes the superposition solution to binary solution, can be considered as a summarization of evolved history information of q-bit strings. The observing procedure that is presented in step 2 of MMQEA in section 2.2.4 transforms the quantum states to a binary solution. Then binary solutions should be evaluated on step 3 and 6 of MMQEA. The repair procedure presented as Algorithm 2 is used before evaluating because of some solutions that observed from q-bit string may be infeasible. Only if the solutions exceed the constraints of problems, the repair procedure is used to. QEA also uses the same repair procedure.

In the knapsack problem, the magnitude of $|\Delta\theta_{ji}|$ has an effect on the speed of convergence. In our experiments, for comparing with the QEA, MMQEA and QEA for knapsack problem selected the same value, as $|\Delta\theta_{ji}| = 0.015\pi$. In our experiments, the generation time "T" of MMQEA and QEA are both set as 1000. The immigration-term "t_m" is designed with 200. Until the termination condition is satisfied, MMQEA is still running in the while loop.

Algorithm 2: Infeasible solution repair procedure

Input: $x_j(t)$- Infeasible binary solution.
$x_j(t) = x_{j1}(t), x_{j2}(t), ..., x_{jm}(t)$.
Output: $x_j^*(t)$- Feasible binary solution.

```
begin
    Knapsack-overfilled ← true;
    while Knapsack-overfilled do
        Select an ith item from knapsack;
        if x_ji(t) == 1 then
            x_ji(t) ← 0;
            if Σ_{i=1}^{m} w_i * x_ji(t) ≤ C then
                Knapsack-overfilled ← false;
            end
        end
    end
    while Not Knapsack-overfilled do
        Select an kth item from knapsack;
        if x_jk(t) == 1 then
            x_jk(t) ← 0;
            if Σ_{i=1}^{m} w_i * x_ji(t) > C then
                Knapsack-overfilled ← true;
            end
        end
    end
    x_jk(t) ← 0;
    x*_j(t) ← x_j(t);
end
```

2.3.2 MMQEA Performance Comparisons

In this Section, The experiment results of MMQEA for knapsack problems are used to compare with QEA and GA. The population size of MMQEA1, MMQEA5, and MMQEA10 are set as 1, 5, and 10, respectively. As a performance measure of the algorithms, it collects the best solutions found within 1000 generations over 30 runs, recorded the elapsed time per run, which is summarized in Table 2.2 and Table 2.3, where the data of QEAs and GAs columns are organized from Han's paper[7].

The population size of GA10 and GA50 are 10 and 50, where $(0.3, 0.15)$ is selected

Table 2.2: Performance comparisons of MMQEA and QEA

Items	Value	QEAs		MMQEAs		
		QEA2	QEA10	MMQEA1	MMQEA5	MMQEA10
100	b	597.7	603.3	622.4	634.3	640.1
	m	591.8	592.3	614.9	621.1	629.8
	w	582.5	582.4	608.5	616.9	620
	t	0.021	0.04	0.047	0.201	0.489
	σ	4.84	3.308	2.01	1.89	1.48
250	b	1480.2	1499.7	1544.9	1558.4	1571.9
	m	1464.5	1476.2	1512.7	1520.4	1542.7
	w	1445.1	1460.8	1505	1510.2	1519
	t	0.055	0.101	0.109	0.519	0.911
	σ	9.554	6.321	5.909	4.98	3.901
500	b	2899.7	2917.1	2999.8	3023.6	3030.1
	m	2876.4	2907.9	2962.9	2996.7	2998.3
	w	2836.2	2874.1	2908.8	2982.9	2984.4
	t	0.117	0.211	0.22	1.232	2.51
	σ	12.832	9.411	10.01	4.079	3.001

as ordered pair of the mutation and crossover probabilities that gave the maximum profits. In the QEA, one q-bit string is regarded as an individual. The population size of QEA1, QEA2 and QEA10 are 1, 2, and 10 and have 1, 2 and 10 q-bit strings, respectively. In Table 2.2, where b, m and w are the final best values, mean values and the worst values, respectively. Unit of t is second per run; σ represents the mean squared error of results. As Table 2.2 and Table 2.3 shows, MMQEAs yielded much better results compared to GAs and QEAs with 100 items which is a relatively simple one compared with the other cases. In the cases 250 and 500 items, MMQEAs found the better solutions within a short time compared to GAs. The results also gained that MMQEAs perform well even with a small population. Compared the MMQEA1 with QEA2 and MMQEA5 with QEA5 that have same number of q-bit strings, MMQEA1 and MMQEA5 get the better solutions due to Multi-update modes are applied.

Fig. 2.6 shows the progress of the best values, worst values and the mean values of the population evolvement found by QEAs and MMQEAs over 30 runs for 100, 250,

Table 2.3: Performance comparisons of MMQEA and GA

Items	Value	GAs		MMQEAs		
		GA10	GA50	MMQEA1	MMQEA5	MMQEA10
100	b	597.6	602.2	622.4	634.3	640.1
	m	587.8	593.6	614.9	621.1	629.8
	w	577.6	582.6	608.5	616.9	620
	t	0.154	0.786	0.047	0.201	0.489
	σ	5.227	4.958	2.01	1.89	1.48
250	b	1455	1472.5	1544.9	1558.4	1571.9
	m	1436.7	1452.4	1512.7	1520.4	1542.7
	w	1415.2	1430.1	1505	1510.2	1519
	t	0.357	1.804	0.109	0.519	0.911
	σ	11.377	10.324	5.909	4.98	3.901
500	b	2828.1	2856.1	2999.8	3023.6	3030.1
	m	2807.2	2831	2962.9	2996.7	2998.3
	w	1781	2810.1	2908.8	2982.9	2984.4
	t	0.706	3.559	0.22	1.232	2.51
	σ	14.142	11.264	10.01	4.079	3.001

and 500 items. Each picture exhibits the two algorithms in same number of q-bits strings. Three different scale problems are tested. MMQEA1 perform significantly better than QEA2 on the best values of every generation for the three different scale problems. With the increase of individuals in MMQEA and QEA, MMQEA5 also is better than QEA10 on the progress of the best values, worst values. The lines of average values in each figure are calculated by 30 times run. The best values of each run in every generation are collected. The average values of MMQEAs are also larger than QEAs in 1000 generations.

Fig. 2.4 and Fig. 2.5 show the best running of MMQEAs and QEAs for three scaled problems, in which the mean error square values of solutions in every generation time are presented. We can see from above Fig. 2.4 and Fig. 2.5, whatever problems are, the value of mean error square in MMQEAs is greater than QEAs at every generation time. So MMQEAs can maintain the solutions of population more diversity. The mean error square line of QEA2 varies smoothly in Fig. 2.4, however the outputs of MMQEA1 in three different scaled problems fluctuate markedly

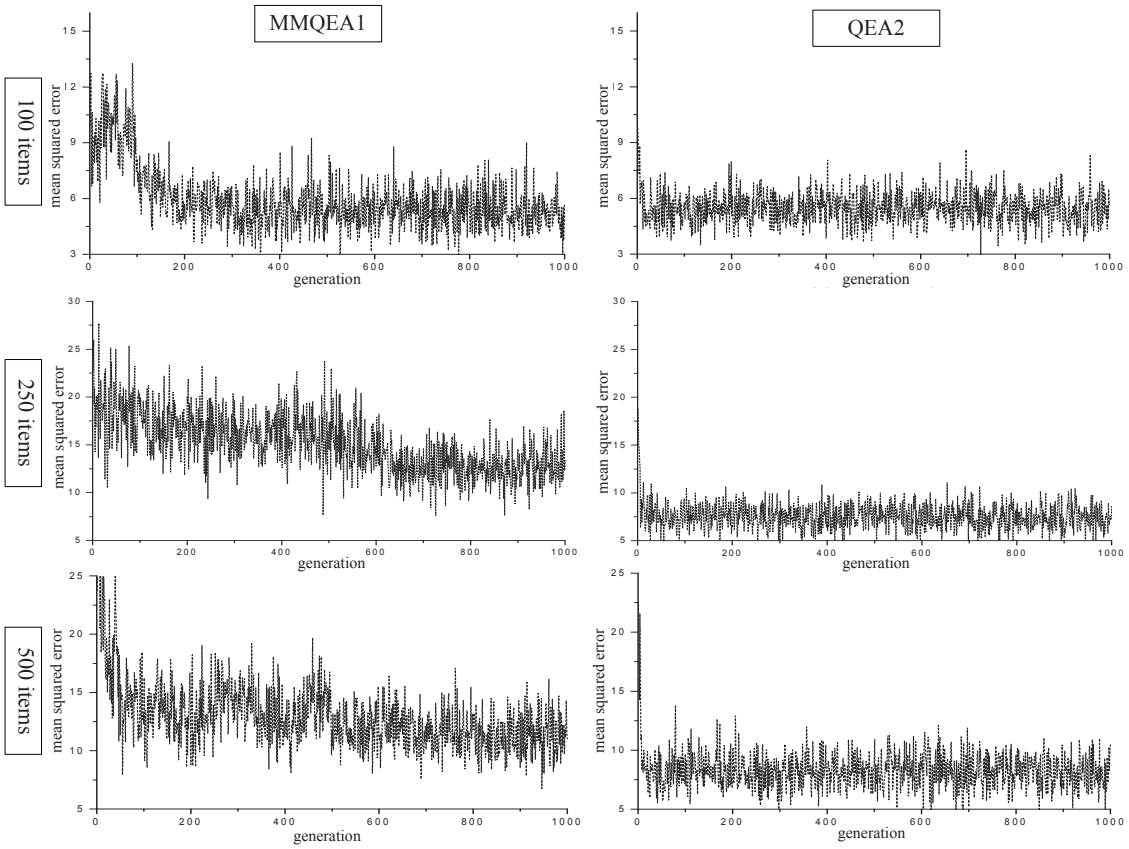


Figure 2.4: Mean squared error in every generation of MMQEA1 and QEA2

even with only one individual. That illustrated our MMQEA could preserve the population of solutions with a variation of diversity. With the increase of number of individuals, as showing in Fig. 2.5, MMQEA5 always keep the population of solutions diversity to guide the evolution avoiding in early convergence.

2.3.3 MMQEA for Permutation FSSP

2.3.3.1 Permutation FSSP Statement

The permutation FSSP can be formulated as follows. Each of n jobs from the set $J = \{J_1, J_2, \ldots, J_n\}$ has to be processed on m machines $M_1, M_2, \ldots, M_m$. Without loss of generality, it is assumed that the jobs are processed in the order of indices

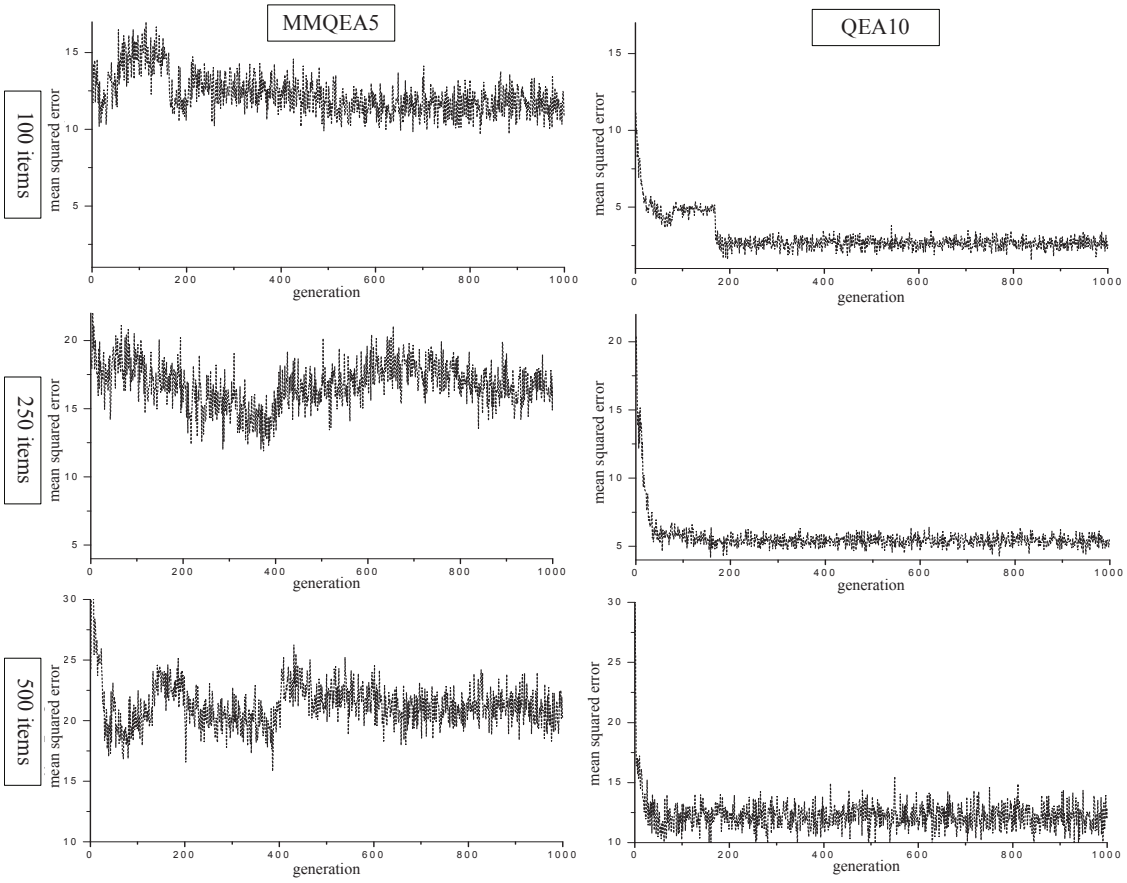
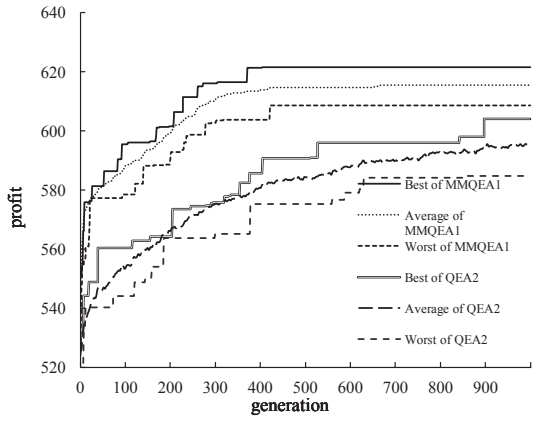


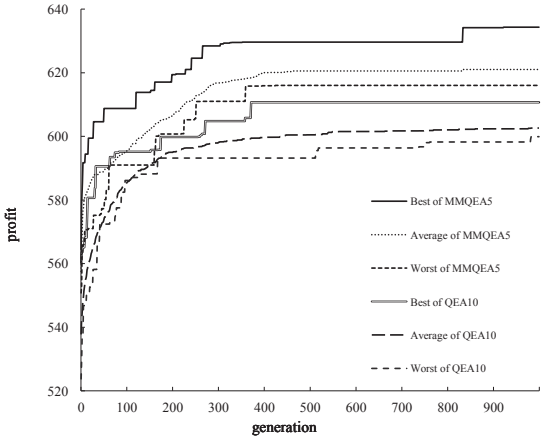
Figure 2.5: Mean squared error in every generation of MMQEA5 and QEA10

of machines. O_{ji} is defined as the ith operation of job J_j processed on machine M_i, and working time of O_{ji} is p_{ji}. At any time, each machine can process at most one job and each job can be processed on at most one machine. The sequence in which the jobs are to be processed is the same for each machine.

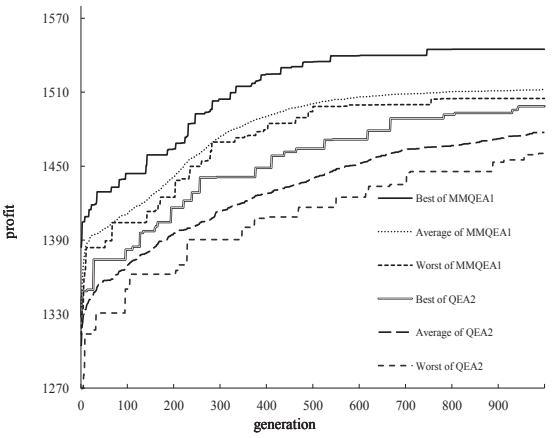
In literature, this criterion is usually referenced by $F/permu/C_{max}$ [18], considered as objective function to reduce the makespan, which is the overall processing time. Let $\pi = (\,\pi(1), \pi(2), \ldots, \pi(n)\,)$ be a permutation, where $\pi(a)$ is the ath job of permutation π, n is number of jobs. Ω denotes the set of all such permutations, and $\pi \in \Omega$. $C_{max}(\pi)$ is makespan of permutation π. Suppose that $C^i_{\pi(j)}$ is the completion time of job $\pi(j)$ on machine M_i. The mathematical formulation can be



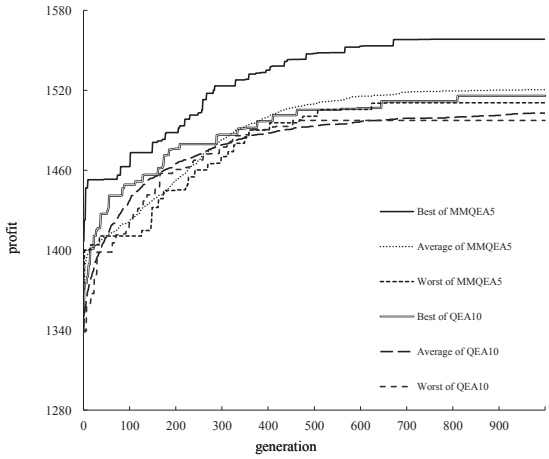
(a) MMQEA1 and QEA2 for 100 items



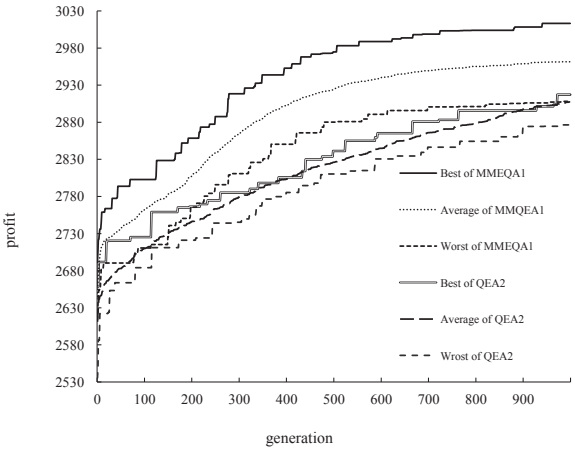
(b) MMQEA5 and QEA10 for 100 items



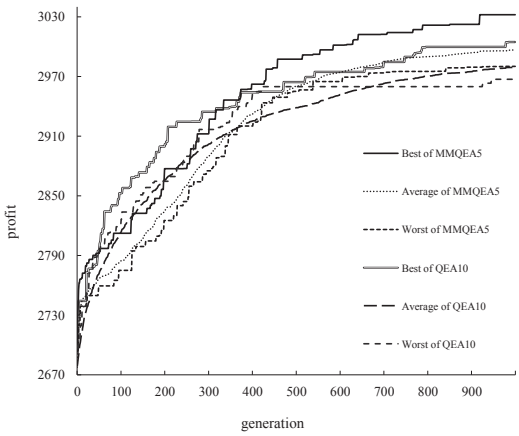
(c) MMQEA1 and QEA2 for 250 items



(d) MMQEA5 and QEA10 for 250 items



(e) MMQEA1 and QEA2 for 500 items



(f) MMQEA5 and QEA10 for 500 items

Figure 2.6: Experimental results of QEAs and MMQEAs for knapsack problems

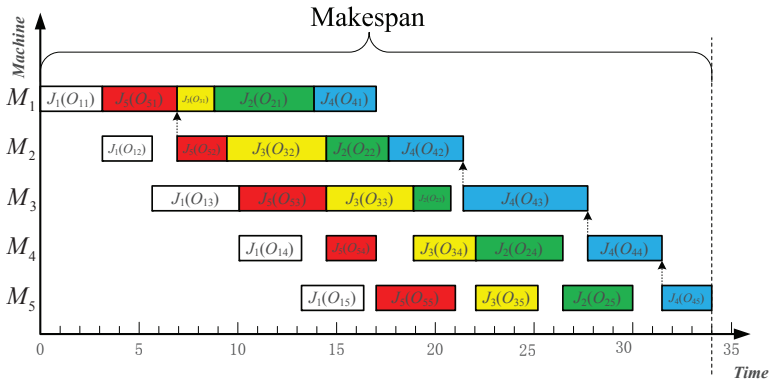


Figure 2.7: Gantt chart of permutation FSSP with makespan criterion

described as below.

$$\begin{aligned} C^1_{\pi(1)} &= p_{\pi(1),1} \\ C^i_{\pi(1)} &= C^{i-1}_{\pi(1)} + p_{\pi(1),i} \\ C^1_{\pi(j)} &= C^1_{\pi(j-1)} + p_{\pi(j),1} \\ C^i_{\pi(j)} &= Max\{C^i_{\pi(j-1)}, C^{i-1}_{\pi(j)}\} + p_{\pi(j),i} \end{aligned} \tag{2.15}$$

where $j = 2, 3, \ldots n$, $i = 2, 3, \ldots m$. Under these specifications, the makespan $C_{max}(\pi)$ is given by $C^m_{\pi(n)}$, which is the completion time of last job $\pi(n)$ on machine M_m. The permutation FSSP with makespan criterion is to find a permutation π^* from Ω, such that:

$$\begin{aligned} \pi^* &= \arg\min_{\pi\in\Omega} C_{\max}(\pi) \\ C_{\max}(\pi) &= C^m_{\pi(n)} \end{aligned} \tag{2.16}$$

Fig. 2.7 shows a gantt chart of the permutation π sequentially process on 5 machines, where π=($\pi(1), \pi(2), \pi(3), \pi(4), \pi(5)$)= ($J_1, J_5, J_3, J_2, J_4$). The makespan $C_{max}(\pi)$ is obtained by the last finished operation O_{45} of job J_4 on machine M_5.

2.3.3.2 Implementation of MMQEA for Permutation FSSP

Although q-bit string can represent a linear superposition of solutions, it cannot be used directly for solving FSSP. Because the flowshop code we want is a permutation

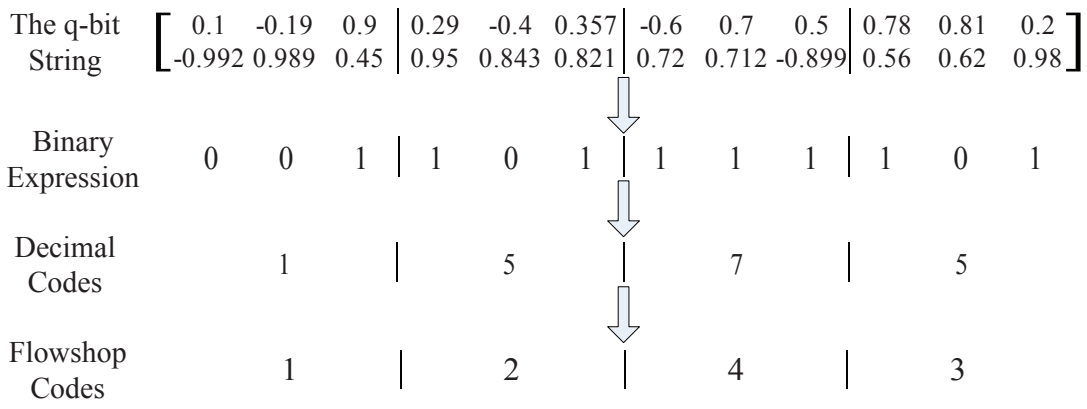


Figure 2.8: Transformation of quantum states to permutation FSSP's job code

of all jobs in a decimal system from 1 to M, but q-bit representation is a pair of complex numbers γ and η. So a converting mechanism should be put forward specially aiming at FSSP for evaluation, which is described as following.

A q-bit individual $d_j(t)$ of length $\lfloor \log_2{}^n + 1 \rfloor * n$ represents a linear superposition of solutions to the problem. In the step of initialization, all the q-bit states initialized as $1/\sqrt{2}$. The observing procedure as represented in section 2.4 are employed to transform quantum states to binary string. Thus, a binary string $x(t)$ with the length of $\lfloor \log_2{}^n + 1 \rfloor * n$ is formed. Every $\lfloor \log_2{}^n + 1 \rfloor * n$ bits of binary string need convert into a decimal code, and then we get a decimal string $D(t)$ of length J. Order a permutation of $D(t)$ from small to big. If values of two numbers are different, let smaller number denotes the job with smaller index; otherwise, let the first one denote the job with smaller index. This permutation is the final flow shop code. The decode process could be carried out as following: firstly finished the arrangement of all operations upon machines for the first job, and then turn to the next job and until last job. Fig. 2.8 shows an example with 4 jobs converts from quantum codes to job codes.

Based on the above encoding method, MMQEA does not produce infeasible solutions. So it does not require any additional repair procedure for MMQEA solved permutation FSSP. The other parts of MMQEA are same as Algorithm 1. The value of $|\Delta\theta_i|$ is also set as 0.015π.

Table 2.4: Experimental results of each algorithm for permutation FSSP

Problem	Value	GAs		QEAs		MMQEAs	
		GA10	GA50	QEA2	QEA10	MMQEA1	MMQEA5
TA-20×5	***b***	1313	1300	1297	1297	1297	1297
	m	1366.4	1344.8	1334	1315.3	1310.2	1308.3
	w	1416	1370	1377	1339	1339	1339
	t	0.343	1.982	0.23	1.38	0.22	1.39
	σ	35.09	16.15	17	12.4	11.39	10.26
TA-20×10	***b***	1791	1711	1680	1686	1670	1656
	m	1903	1806.5	1741.3	1711.2	1711.7	1703.7
	w	1995	1860	1803	1734	1758	1734
	t	0.423	2.32	0.283	1.69	0.259	1.7
	σ	47.62	31.7	30.24	14.3	19.71	21.3
TA-20×20	***b***	2453	2453	2403	2394	2400	2388
	m	2538.9	2538.9	2440.5	2435.5	2437.5	2435.1
	w	2588	2588	2487	2469	2473	2456
	t	0.53	2.69	0.39	2.11	0.4	2.03
	σ	50.6	33.03	21.56	16.48	16.4	16.79
TA-50×5	***b***	2889	2845	2753	2741	2752	2746
	m	2974.7	2907.1	2788.3	2782.1	2787.8	2793.6
	w	3070	2993	2833	2844	2827	2842
	t	0.792	3.78	0.61	3.12	0.61	3.12
	σ	53.3	33.9	18.37	22.62	22.44	27.68
TA-50×10	***b***	3503	3413	3316	3296	3279	3246
	m	3645	3562.1	3381.3	3366.7	3348.1	3358
	w	3727	3646	3477	3446	3415	3421
	t	0.841	3.95	0.69	3.22	0.62	3.19
	σ	54.08	52.58	36.1	39.03	36.42	44.2
TA-50×20	***b***	4541	4528	4249	4280	4230	4189
	m	4654.4	4601.4	4349.5	4362.4	4328.6	4310.1
	w	4798	4684	4441	4502	4412	4400
	t	0.982	4.02	0.73	3.41	0.71	3.33
	σ	59.15	34.4	44.7	55.6	44.72	44.18

2.3.4 Experiments and Analysis on Permutation FSSP

The experiment data are from the benchmark on flowshop problem which can be found in Taillard's home page [1]. The 6 different scale problems are tested on the experiments. In Taillard's problems, there are many cases for a scale problem. The first case of these problems are used.

In this chapter, MMQEA compares with QEA and GA without any external local search procedure. The same encoding method is employed on MMQEA and QEA for FSSP, in which other parts are similar with the basic structures of MMQEA and Han's QEA, respectively. The permutations of job numbers are considered as a coding strategy for GA algorithm, of which mutation is implemented by exchanging the two jobs on chromosomes. The two point crossover is used. In GA, the probability of crossover and mutation are both set to 0.3, which obtained the better results compared with other parameters. The obtained final makespan of three algorithms are summarized in Table 2.4. These algorithms are tested on 6 different scaled problems in where such as TA-20×5 represents the problem with 20 jobs and 5 machines. Each case for every algorithm runs 30 times at the 500 generations. Table 2.4 gathers the mean value and best, worst value. The population size of GA10, GA50, QEA2, QEA10, MMQEA1 and MMQEA5 are 10, 50, 2, 10, 1, 5, respectively. We can see clearly from this table that MMQEAs is evidently better than GAs even with little individuals. MMQEA5 outperforms QEA10 with 3%, 6% and 7% improvement on 20×10 (20 jobs 10 machines), 50×10 and 50×20 problems. On the other hand, Although MMQEAs and QEAs consumed the similar computational times on all test problems, comparing with GAs, MMQEA costed less computational times to find near-optimal solutions.

2.4 Summary

This chapter presents a novel evolutionary algorithm, called MMQEA, inspired by the concept of quantum computing. An individual is defined as a pair of interactional

[1] http://mistic.heig-vd.ch/taillard/ problemes.dir/ ordonnancement.dir/ordonnancement.html

q-bit strings for the probabilistic representation. Multi-update mode is designed as a variation operator on the q-bit individuals, inspired by quantum entanglement. The proposed MMQEA is characterized in the q-bit representation for the population diversity and the evolutionary history information of double q-bit strings also is considered to guide the evolutionary direction of individuals to avoid premature convergence.

For driving the individuals toward better solutions, the multi-update modes is employed, in which the evolutionary information of q-bit string in individual will be considered to evolve the other q-bit string that also provides its evolutionary information. The most advantages of MMQEA are little parameters to be set and the individual has the ability that evolves itself to keep a balance between exploration and exploitation. Firstly, knapsack problems were used for testing the performance of MMQEA. The experiment showed MMQEA could maintain population of solutions more diversity and gain more quality result. For further testing the ability of MMQEA's application, the permutation FSSP was applied for MMQEA. The results also were comparable.

Chapter 3

PQEA for Multi-Objective Permutation FSSP

3.1 Introduction

In the world around us, it is rare for any problem to concern only a single value or objective. Generally, multiple objectives or parameters have to be met or optimized. Multi-objective Optimization (MOO) is the process of simultaneously optimizing two or more conflicting objectives subject to certain constraints. MOO problems can be found in various fields: product and process design, aircraft design, the oil and gas industry, or wherever optimal decisions need to be taken in the presence of trade-offs between two or more conflicting objectives. Maximizing profit and minimizing the cost of a product; maximizing performance and minimizing fuel consumption of a vehicle; and minimizing weight while maximizing the strength of a particular component are examples of MOO problems. For these kinds of problems, there is no single optimal solution, but a set of alternative solutions. If these solutions are optimal, they are called pareto-optimal solutions[19], [20]. It means that no other solutions in the search space are superior to them when all objectives are considered. The multi-objective permutation FSSP is a typical MOO problem, while minimizing makespan and job tardiness are considered as two criteria. Because makespan is a

measure of system utilization while maximum tardiness can be regarded as a measure of performance in meeting customer's due dates. A. Nagar et al. [21] has summarily surveyed that the most widely used criteria are related to minimizing makespan and job tardiness in permutation FSSP, and the trade-off between the criteria has been demonstrated.

Up to now, a number of different evolutionary algorithms (EA) have been suggested to solve MOO problems. For reviewing of previous researches, Zhang's MOEA/D [22], Deb's NSGA-II[23] and Ishibushi's MOGLS [24] have been paid great attention to. These algorithms demonstrated the necessary additional operators for converting a simple EA to a (Multi-Object Evolutionary Algorithm) MOEA. All above algorithms can be summarized in two common features that are based on non-dominated sorting and decomposing MOO problem into a number of scalar subproblems. The first feature is employed in the NSGA-II, where the individuals are ranked in different levels, in other words, each of them is given a fitness value. Then, the individuals in diverse levels were selected as population members for the next generation. The Pareto Front which is decomposed into a number of scalar objective optimization sub-problems is the feature of MOEA/D and MOGLS. MOEA/D uniformly generates a set of weighted vectors and transfers MOO problem to many sub-problems that are simultaneously optimized in a generation. Several methods for building scalar functions can be found in the literatures [25], [26], [27]. The most popular ones among them include the weighted sum approach and Tchebycheff approach [26]. The weighted sum approach is widely used for solving the discrete MOO problems in the paper[28]. The Tchebycheff approach works well when the optimal solutions are a concave or convex pareto fronts[29].

Based on achievement of last chapter, a multi-objective quantum evolutionary algorithm based on parallel model, which is called PQEA (parallel quantum evolutionary algorithm) is proposed in this chapter. In the initialization step, PQEA evenly decomposes MOO problem into many scalar optimization sub-problems by a set of weighted vectors. One sub-problem is associated with a weighted vector. All sub-problems are classified into several groups based on the similarities of the weighted vectors. In the main evolution part, the groups of sub-problems are parallel

evolved by a population of quantum individuals. The probabilistic representation is the most feature of q-bit individual, in which the evolutionary information for optimizing a problem can be memorized during the evolution progression. Because of the similarities of neighboring sub-problems, a q-bit individual is applied to orderly address the sub-problems in one group. Thus, the evolutionary information in the q-bit individual for optimizing a sub-problem can be shared with its neighborhoods. By parallel evolving the groups of neighboring sub-problems, we attempt to obtain the pareto solutions that exhibit more extensive distribution and dominated performance. To demonstrate the effectiveness of PQEA, it is applied to solve multi-objective permutation FSSP to compare with the current state-of-the-art methods, such as MOEA/D, NSGA-II and MOGLS. The performance of PQEA on Multi-Objective Knapsack Problems (MOKPs) is also investigated.

This chapter is organized as follows. Section 2 introduces some key concepts of quantum evolutionary algorithm and gives an overview of the MOO problems. The weighted sum approach for MOO problem is also reviewed. The proposed PQEA is described in Section 3 in detail. The comparison of PQEA and other famous MOO evolutionary algorithms on multi-objective permutation FSSP and MOKPs is the subject of Section 4, which is divided into two parts: methodology of the comparison and experimental results. The last section concludes this research and offers the future perspectives.

3.2 Basic Components of PQEA

Quantum-inspired evolutionary algorithm (QEA) applies quantum computing principles to enhance classical evolutionary algorithms. QEA is firstly proposed by Han and Kim in paper[7] where some major principles of quantum computing are used, such as the quantum bit, the linear superposition of states and the quantum rotation gate.

Like other EAs, QEA is also characterized by the representation of the individual, the evaluation function, and the population dynamics. Some definitions and operators in QEA are given as follows:

(1) *Population Construction*: The QEA maintains a population of q-bit individuals, likes $Q(t)$, and $Q(t) = \{q_t(1), q_t(2), \ldots, q_t(v)\}$ at generation t, where v is the size of population, and $q_t(i)$ is a q-bit individual defined as

$$q_t(i) = \left[\begin{array}{c|c|c|c} \gamma_{i1}^t & \gamma_{i2}^t & \cdots & \gamma_{im}^t \\ \eta_{i1}^t & \eta_{i2}^t & \cdots & \eta_{im}^t \end{array} \right] \tag{3.1}$$

where, m is the number of q-bits in a q-bit individual, and $i = 1, 2, \ldots, v$.

The $|\psi_{ij}^t\rangle$, which is quantum state of jth q-bit in $q_t(i)$, can be represented as:

$$|\psi_{ij}^t\rangle = \gamma_{ij}^t|0\rangle + \eta_{ij}^t|1\rangle \tag{3.2}$$

(2) *Observing Procedure*: The observing procedure is used to transform a q-bit individual to a binary solution. Suppose $x_t(i)$ is the binary solution, and $x_t(i) = \{x_t^1(i), x_t^2(i), \ldots, x_t^m(i)\}$, here, $x_t^j(i)$ is 0 or 1, $j = 1, 2, \ldots, m$. The $x_t(i)$ is generated from the q-bit individual $q_t(i)$, in which the $x_t^j(i)$ is obtained by observing the state of $|\psi_{ij}^t\rangle$. The observing procedure can be described as:

Let τ_j be a random number generated from the uniform distribution $(0, 1]$. If τ_j satisfies $\left|\eta_{ij}^t\right|^2 > \tau_j$, then $x_{ij}^t \leftarrow 1$, otherwise $x_{ij}^t \leftarrow 0$.

Thus, the binary solution $x_t(i)$ can be obtained from $q_t(i)$ by iterating above observing procedure m times. After that, a fitness value of $x_t(i)$ could be evaluated based on the problem specification.

(3) *Updating q-bit Individual States*: In classical EA, variation operators like crossover or mutation operations are used to explore the search space. The quantum analog for these operators is called a q-gate. In QEA, the q-gate, which is represented by matrices, can be visualized as rotations of the quantum state on the Bloch Sphere [6]. The q-gate $U(\triangle\theta)$ is designed for QEA, as below equation:

$$\mathbf{U}(\Delta\theta) = \left[\begin{array}{ll} \cos(\Delta\theta), & -\sin(\Delta\theta) \\ \sin(\Delta\theta), & \cos(\Delta\theta) \end{array} \right] \tag{3.3}$$

where, $\Delta\theta$ is a rotation angle of each q-bit toward either $|0\rangle$ or $|1\rangle$ state. In this study, the $U(\triangle\theta)$ is used to modify the quantum state of q-bit. The jth q-bit at

generation time t in individual $q_t(i)$ is updated as follows:

$$\begin{bmatrix}\gamma_{ij}^{t+1}\\ \eta_{ij}^{t+1}\end{bmatrix} = \begin{bmatrix}\cos(\Delta\theta), & -\sin(\Delta\theta)\\ \sin(\Delta\theta), & \cos(\Delta\theta)\end{bmatrix} * \begin{bmatrix}\gamma_{ij}^{t}\\ \eta_{ij}^{t}\end{bmatrix} \tag{3.4}$$

The q-bit state of $|\psi_{ij}^{t+1}\rangle$ also hold $\left|\gamma_{ij}^{t+1}\right|^2+\left|\eta_{ij}^{t+1}\right|^2=1$. The $q_{t+1}(i)$ can be obtained by sequentially updating the state of q-bit in $q_t(i)$ based on Eq. 3.4. We should design the value of angle parameters $\Delta\theta$ for different kinds of problems to control the convergence of algorithm [30]. In this chapter, the rotated direction on clockwise is considered as positive rotation, and $\Delta\theta$ is limited in the range $[0, \pi/2]$.

The main procedure of QEA is to iteratively execute the observing and update procedure until the terminal condition is met. The more details of QEA can be referred in the researches [31].

3.3 Main Framework of PQEA

3.3.1 MOO Problem Statements

A general MOO problem can be described as a vector function maps a set of decision variables to another set of objectives, as follow:

Formally:

$$\begin{aligned} &Min/Max\ \ Y = f(X)\\ &\qquad\qquad = \{f_1(X), f_2(X), \ldots, f_J(X)\}\\ &Where, \left\{\begin{array}{l} X = (x_1, x_2, \ldots, x_m)\\ Y = (y_1, y_2, \ldots, y_J)\\ X \in \Omega \end{array}\right\} \end{aligned} \tag{3.5}$$

where, X is decision variable vector, x_i is decision variable ($i = 1, 2, \ldots, m$). Y is objective value vector and f_j is one of objective function (j is from 1 to J). The solution space is Ω. The set of solutions of a multi-objective optimization problem consists of all decision vectors for which the corresponding objective vectors cannot be improved in any dimension without degradation in another. These vectors are

known as Pareto Optimal. Mathematically, the concept of pareto optimality is as follows: Assume, without loss of generality, a maximization problem and consider two decision vectors $a, b \in X$. then, a is said to dominate b. That is also written as $a \succ b$. If and only if $f_i(a) \geq f_i(b)$, $i \in \{1, 2, \ldots, J\}$ and $f_j(a) > f_j(b)$ for at least one index $j \in \{1, 2, \ldots, J\}$. All decision vectors which are not dominated by any other decision vector of a given set are called non-dominated solutions[32].

3.3.2 Decomposition Approach Used in PQEA

There are several approaches for converting the problem of approximation of the pareto optimality into a number of scalar optimization problems. The weighted sum approach is one of most popular methods to convert a multi-objective problem into a number of scalar optimization problems. It is widely employed on discrete MOO problems, such as MOGLS for flow shop scheduling problem [33], MOEA/D for knapsack problem[34], and so on. The explanations of this approach present as follow:

Let $\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_J)^T$ be a weighted vector, $\lambda_j \geq 0$ for all $j = 1, 2, \ldots, J$ and $\sum_{j=1}^{J} \lambda_j = 1$. MOO problem can be changed to a single objective sub-problem as:

$$\min / \max \quad g(x) = \sum_{i=1}^{J} \lambda_i * f_i(x) \tag{3.6}$$

To obtain the pareto optimal solutions of a MOO problem, many different weighted vectors Λ are used in the above scalar optimization problem.

Parallel quantum evolutionary algorithm (PQEA) using the weighted sum approach decomposes the MOO problem by a set of even spread weighted vectors. Firstly, the set of weighted vectors will be classified into some groups according to the euclidean distance between each two vectors. After that, the evolution of a q-bit individual is used to address the sub-problems of one group. The population size of q-bit individuals is same as the number of groups.

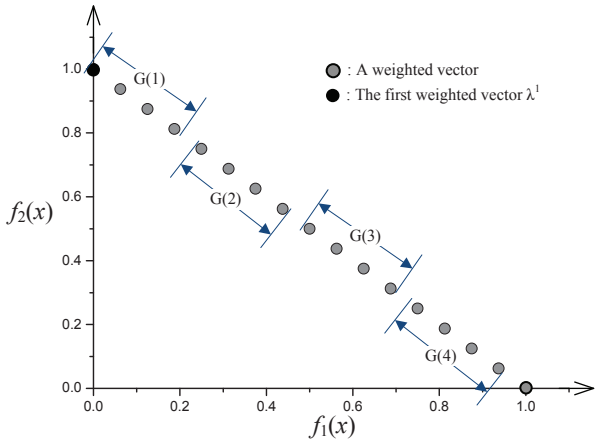


Figure 3.1: Weighted vectors classification on two objective problems

3.3.3 Weighted vector classification for PQEA

Suppose $S(\Lambda)$=$\lambda^1, \lambda^2, \ldots, \lambda^N$, is a set of even spread weighted vectors, and N is number of vectors. Here, λ^i= $\{\lambda^i_1 ..., \lambda^i_J\}$, and $\sum_{j=1}^{J} \lambda^i_j = 1$, i=1, 2, ..., N. The set of weighted vectors are equally divided into v groups, such as $G(1)$, $G(2)$,..., $G(v)$. The $G(u)$ likes as $G(u) = \{\lambda^1(u), \lambda^2(u), ..., \lambda^T(u)\}$ which maintains T closest weighted vectors of $\lambda^1(u)$, where $u = 1, \ldots, v$.

The procedure of weighted vectors classification is described as follow:

- **Step 1**. The $\lambda^1(1)$, which is the first weighted vector of $G(1)$, is set as: $\lambda^1(1) = \lambda^1$, and λ^1 is removed from $S(\Lambda)$. Where, λ^1 specialized as λ^1=$\{\lambda^1_1, ..., \lambda^1_{J-1}, \lambda^1_J\}$ =$\{0.0, ..., 0.0, 1.0\}$.

- **Step 2**. Select T-1 closest Euclidean distances of $\lambda^1(1)$ from $S(\Lambda)$ to create the group 1, such as $G(1)$=$\{\lambda^1(1), \lambda^2(1), \ldots., \lambda^T(1)\}$. The selected weighted vectors will be removed from $S(\Lambda)$.

- **Step 3**. The $G(2)$ is formed by selecting the T closest Euclidean distances of $\lambda^T(1)$ from $S(\Lambda)$. The same way is used to form $G(3)$, $G(4)$,..., $G(v)$, where $v = \lceil \frac{N}{T} \rceil$.

Fig.3.1 presents an example of weighted vector classification on bi-objective, where N, v and T are 17, 4 and 4, respectively. There are 4 q-bit individuals is used to address the the sub-problems in 4 (v=4) groups with parallel evolution.

Algorithm 3: General Framework of PQEA

Input : $MaxGeneration$, N, T, C
Output: $PNDS$
Initialization:
Generate a set of even spread weight vectors, $\lambda^1, \lambda^2, ..., \lambda^N$;
Classify the weight vectors into $G(1), G(2), ...G(v)$ by weight vectors classification, *where* $v = \lceil \frac{N}{T} \rceil$;
Initialize $Q(t)$, $t \leftarrow 0$, $\delta \leftarrow 1$;
for $i \leftarrow 1$ **to** v **do**
    Select $g\left(x \left| \lambda^{\delta}(i)\right.\right)$ as objective function for $q_t(i)$;
    Generate a binary solution $x_t(i)$ by observing the states of $q_t(i)$;
    Repair\Improve $x_t(i)$ to $p_t(i)$ by the problem specialized approach;
    Evaluate $p_t(i)$ to fitness value by $g\left(x \left| \lambda^{\delta}(i)\right.\right)$;
    `UpdatePNDS(`$PNDS$, $p_t(i)$`)`;
    $b^{\delta}(i) \leftarrow p_t(i)$;
end
Main parts:
$t \leftarrow 1$;
while $t < MaxGeneration$ **do**
    for $i \leftarrow 1$ **to** v **do**
        if `MOD`$(t, C) = 0$ **then**
            Select the $g\left(x \left| \lambda^{\delta}(i)\right.\right)$ as objective function for $q_t(i)$;
            Find the best solution x for $g\left(x \left| \lambda^{\delta}(i)\right.\right)$ from PNDS, and $b^{\delta}(i) \leftarrow x$;
        end
        Generate a binary solution $x_t(i)$ by observing the states of $q_t(i)$;
        Repair\Improve $x_t(i)$ to $p_t(i)$ by the problem specialized approach;
        Evaluate $p_t(i)$ to fitness value by $g\left(x \left| \lambda^{\delta}(i)\right.\right)$;
        if $p_t(i)$ *is better than* $b^{\delta}(i)$ **then**
            $b^{\delta}(i) \leftarrow p_t(i)$;
        end
        `UpdatePNDS(`$PNDS$, $p_t(i)$`)`;
        `UpdateQuanStates(`$q_t(i)$, $b^{\delta}(i)$, $p_t(i)$`)`;
    end
    $t++$;
end

3.3.4 General Framework of PQEA

At each generation t, PQEA with sum weighted approach maintains:

- $Q(t)$=$\{q_t(1), q_t(2), ..., q_t(v)\}$ is a population of q-bit individuals at generation t. The $q_t(i)$ is evolved for the sub-problems in $G(i)$, i=$1, 2, ..., v$. The population size of $Q(t)$ is same as the number of weighted vector groups v.

- $B(1), B(2), \ldots, B(v)$ are set of solutions. For $i = 1, 2, ..., v$, $B(i)$ likes as: $B(i) = \left\{b^1(i), b^2(i), \ldots, b^T(i)\right\}$. The $B(i)$ holds the best solutions for the sub-problems that are formulated by the weighted vectors in $G(i)$, such as, $b^j(i)$ holds the best solutions for $g(x\,|\lambda^j(i))$.

- Population of non-dominated solutions ($PNDS$), which is used to maintain the non-dominated solutions found during the evolution.

The q-bit individual $q_t(i)$ in $Q(t)$ can be represented as below:

$$q_t(i) = \left[\begin{array}{c|c|c|c} \gamma_{i1}^t & \gamma_{i2}^t & \cdots & \gamma_{im}^t \\ \eta_{i1}^t & \eta_{i2}^t & \cdots & \eta_{im}^t \end{array}\right] \tag{3.7}$$

where m is the number of q-bits, i.e., the string length of the q-bit individual.

The pseudo code of PQEA is presented in Algorithm 3, more details explanations are given in the following.

(1) Some parameters are given before PQEA is started:

- $MaxGeneration$: a stop criterion for PQEA.

- N: number of total weighted vectors in PQEA.

- T: number of weight vectors in one group.

- C: a constant integer used to control the evolution times on a sub-problem.

(2) Output the solutions while stop condition is met.

(3,4) A set of uniform spread of N weighted vectors: $S(\Lambda) = \{\lambda^1, \lambda^2, \ldots, \lambda^N\}$ is generated. And, $S(\Lambda)$ is classified into some groups by the weighted vector classification.

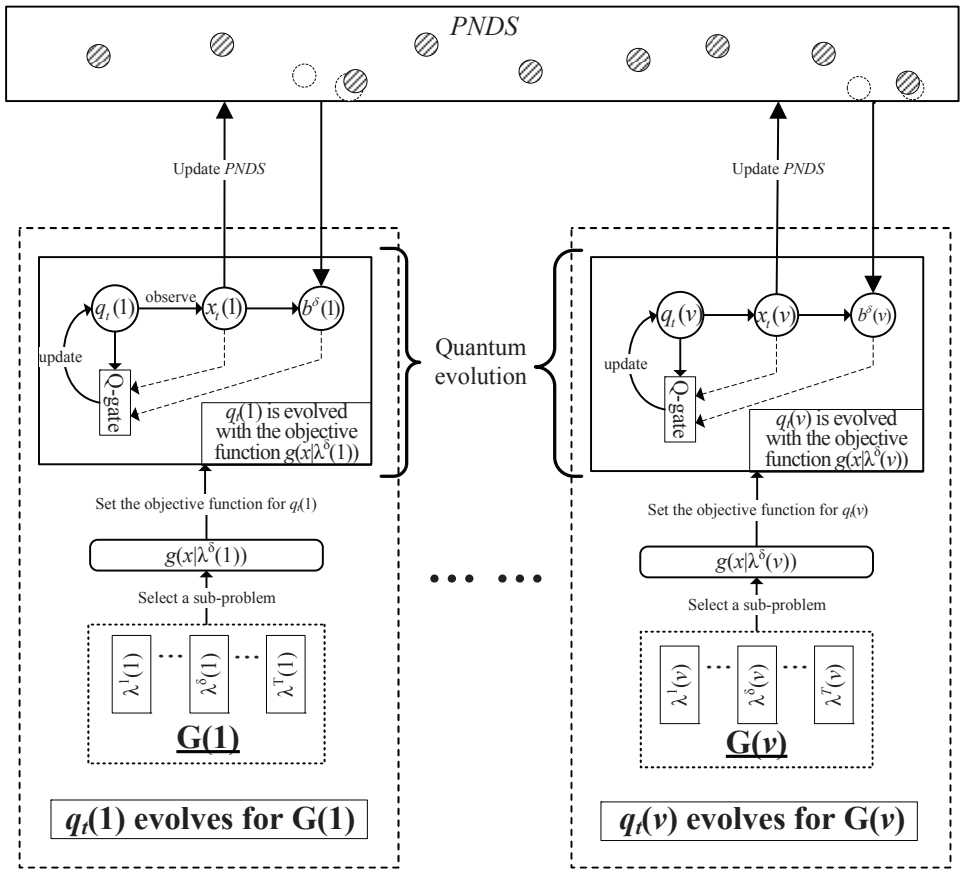


Figure 3.2: General framework of proposed PQEA

(5) All of q-bit states of individuals are initialized as:

$$\begin{bmatrix} \gamma_{ij}^t \\ \eta_{ij}^t \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} \tag{3.8}$$

Here, i is the individual index in population and i=1, 2, ..., v, j is the position of q-bit in, j=1, 2, ..., m. It means that all the q-bit individuals that consist of q-bit with the same states, which represents the linear superposition of all possible states with same probability in the initialization step.

(6) In the loop of initialization step, each q-bit individual is set with a corresponding fitness function that is formulated by the first weighted vector in every group.

(7∼9) The binary solution $x_0(i)$, which is transformed from the q-bit individual $q_0(i)$ by observing procedure, is repaired or improved to $p_0(i)$ by some problem specialized approaches. After that, the $p_0(i)$ is evaluated to give a level of its fitness by $g(x\,|\lambda^1(i))$.

(10,11) `UpdatePNDS`(PNDS, $p_0(i)$): all the solutions that are dominated by $p_0(i)$ are removed from PNDS, and $p_0(i)$ is added into PNDS if no solutions in PNDS dominate $p_0(i)$. Then, $b^0(i)$ is initialized by $p_0(i)$, and i=1, 2, ..., v.

(12) In the main parts, all individuals $q_t(1)$, $q_t(2)$,..., $q_t(v)$ are parallel evolved to deal with the sub-problems in every group, which proceeds in the inner loop.

(13∼15) Each q-bit individual is independently processed in dealing with a sub-problem within C generation times. After the C generation evolution, the neighboring sub-problem in the same group is continuously called to evolve by the same q-bit individual. Here, the δ controls which sub-problem is operated in successively. The δ, which relates with C, T and current generation time t, can be represented as below.

$$\delta = 1 + MOD(\lceil t/C \rceil, T) \tag{3.9}$$

It's clear that the δ is limited in $[1, T]$. So, the "T" sub-problems in a group will be successively evolved by a q-bit individual, until the stop condition is met.

(16∼20) These procedures are similar to (7∼11).

(21) `UpdateQuanStates`($q_t(i)$, $b^\delta(i)$, $p_t(i)$): This step makes the q-bit individuals converge to the fitter states. We should contrast the binary numbers that are in the same positions of $b^\delta(i)$ and $p_t(i)$ to deicide the value of $\Delta\theta$ for the q-bit in same position of q-bit individual. The $\Delta\theta$ are set by some intuitive reasoning for different problems, and more details of this step can be found in papers[35][11].

3.4 PQEA for Multi-Objective Permutation FSSP

In this section, first of all, PQEA is implemented to compare with the famous multi-objective algorithms MOEA/D, MOGLS and NSGA-II on multi-objective

permutation FSSP. The computational experiments are conducted on the benchmark problems of multi-objective permutation FSSP to verify the four algorithms performance. To further confirm the applicability of proposed algorithm, the Multi-Objective Knapsack Problems (MOKPs) are used to test the efficiency of PQEA.

As the description in Chapter 2, the permutation FSSP consists of scheduling given jobs with same order at all machines. The job can be processed on at most one machine; meanwhile one machine can process at most one job. The most common objective for this problem is makespan. However, multi-objective approach for scheduling to reduce maximum tardiness is important. Hence, in this chapter, we consider the permutation FSSP with multi-objectives of makespan and maximum tardiness. The objective is to find a set of non-dominated solutions, which minimize makespan and maximum tardiness simultaneously, for decision maker.

3.4.1 Criteria Formulations for Permutation FSSP

Among due date based criteria, the minimization of the maximum tardiness is the most common one, which is often represented as $F/permu/T_{max}$ [18]. Let $\pi = (\,\pi(1),\, \pi(2),\, \ldots,\, \pi(n)\,)$ be a permutation, where $\pi(a)$ is the ath job of permutation π, n is number of jobs. Ω denotes the set of all such permutations and $\pi \in \Omega$. The permutation FSSP with this criterion is to find a permutation π^* from Ω, which is defined as follows:

$$\pi^* = \arg\min_{\pi \in \Omega} T_{\max}(\pi) \tag{3.10}$$

$$\begin{aligned} T_{max}(\pi) &= \underset{j \in [1,n]}{Max} \left\{ T_{\pi(j)} \right\} \\ T_{\pi(j)} &= Max \left\{ C^m_{\pi(j)} - D_{\pi(j)}, 0 \right\} \end{aligned} \tag{3.11}$$

where $T_{\max}(\pi)$ is maximum tardiness of π, $T_{\pi(j)}$ and $D_{\pi(j)}$ present the tardiness and due date of job $\pi(j)$, respectively. When the value of $T_{\pi(j)}$ is negative, tardiness of job $\pi(j)$ is given as 0, otherwise, the difference between the completion time and due date of job $\pi(j)$ is set as tardiness value.

Suppose that $C^i_{\pi(j)}$ is the completion time of job $\pi(j)$ on machine M_i. As mentioned in chapter 2, the makespan $C_{max}(\pi)$ is given by $C^m_{\pi(n)}$ which is the completion

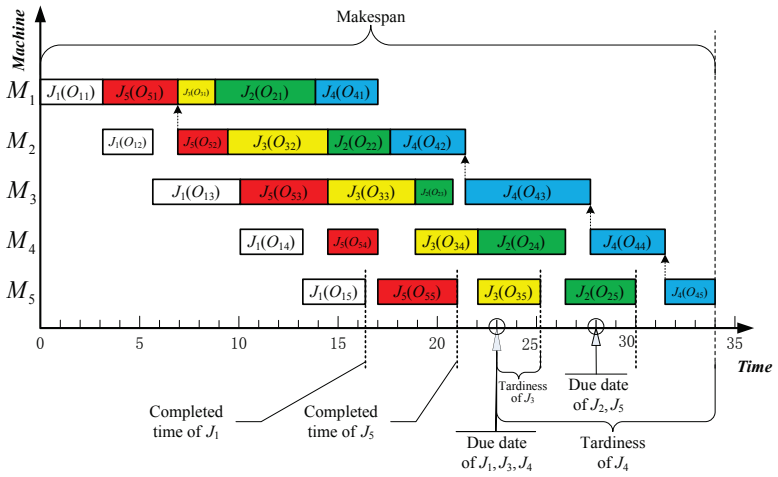


Figure 3.3: Gantt chart of 5×5 permutation FSSP

time of last job $\pi(n)$ on machine M_m. The permutation FSSP with makespan criterion is to find a permutation π^* from Ω, such that:

$$\begin{aligned} \pi^* &= \arg\min_{\pi\in\Omega} C_{\max}(\pi) \\ C_{\max}(\pi) &= C^m_{\pi(n)} \end{aligned} \tag{3.12}$$

A gantt chart of the permutation π sequentially process on 5 machines presents in Fig. 4.1, where π=$(\pi(1), \pi(2), \pi(3), \pi(4), \pi(5))$= $(J_1, J_5, J_3, J_2, J_4)$. The makespan $C_{max}(\pi)$ is obtained by the last finished operation O_{45} of job J_4 on machine M_5. The due date of every job is fixed given in the figure. For this example, the completion time of jobs J_1, J_5 are early than their due date, whereas the jobs J_2, J_3 and J_4 have tardiness to catch the due dates. Maximum tardiness of the π is generated by J_4. We want to find some solutions that can simultaneously minimize makespan and maximum tardiness.

Based on the above mentioned, the bi-criteria optimization problem for permutation FSSP can be formalized as

$$\underset{\pi\in\Omega}{Minimize}\,\{C_{\max}(\pi), T_{\max}(\pi)\} \tag{3.13}$$

The final solutions for bi-criteria optimization problem consist of all permutations π ($\pi \in \Omega$) for which the corresponding objective values cannot be improved in any dimension without degradation in another. These permutations are known as Pareto Optimal Solutions [36]. Mathematically, the concept of pareto optimality is as follows: assume that $\pi,\ \pi^* \in \Omega$, π dominates π^*, if and only if the condition Eq. 3.14 or Eq. 3.15 is satisfied.

$$C_{\max}(\pi) \leq C_{\max}(\pi^*) \bigwedge T_{\max}(\pi) < T_{\max}(\pi^*) \tag{3.14}$$

$$C_{\max}(\pi) < C_{\max}(\pi^*) \bigwedge T_{\max}(\pi) \leq T_{\max}(\pi^*) \tag{3.15}$$

The permutations π and π^* which are not dominated by each other of a given set are called non-dominated solutions.

3.4.2 Implement of PQEA for Multi-Objective Permutation FSSP

To solve the multi-objective permutation FSSP, PQEA needs transformation procedure and normalization objective to vary some steps in its general framework. Each q-bit individual in PQEA contains $\lfloor \log_2^n + 1 \rfloor * n$ q-bits to represent a linear superposition of solutions to this problem, where n is number of jobs to be processed. As the description in Chapter 2, using the transformation procedure converts the quantum states to real permutation FSSP codes. The transformation procedure is implemented in the steps (7)$\sim$(8) and (17)$\sim$(18) of PQEA.

Due to the disparately scaled objectives of multi-objective permutation FSSP, we incorporate a simple objective normalization technique into PQEA. For example, the makespan and maximum tardiness of 20×20 (20 jobs and 20 machines) benchmark problem are belong to the range of $[3315, 3696]$ and $[97, 1075]$, respectively, which are all normalized into the space of $[0, 1]$ to evaluate the fitness values. The

benchmark problems can be found at here[1]. A lot of effort has been made on the issue of objective normalization in the communities of both mathematical programming and evolutionary computation. Normalize objective values of members x using the equations below:

$$\overline{C_{\max}(x)} = \frac{C_{\max}(x) - C_{\min}}{C_{\max} - C_{\min}} \tag{3.16}$$

$$\overline{T_{\max}(x)} = \frac{T_{\max}(x) - T_{\min}}{T_{\max} - T_{\min}} \tag{3.17}$$

Where, $\overline{C_{max}(x)}$ and $\overline{T_{max}(x)}$ are the normalized objective values for makespan and maximum tardiness. In these equations, C_{max} and C_{min} are maximum and minimum makespan values in reference solutions. The maximum and minimum tardiness values in reference solutions are represented by T_{max} and T_{min}, respectively. In 20×20 case, learning from reference solutions C_{max}, C_{min}, T_{max} and T_{min} are equal to 3696, 3315, 97 and 1075. In such way, the range of each objective value becomes $[0, 1]$. Thus, the fitness evaluation function $g(x|\lambda^{\delta}(i))$ in PQEA for multi-objective permutation FSSP can be replace by below equation:

$$g(x|\lambda^{\delta}(i)) = \lambda_1^{\delta}(i) * \overline{C_{\max}(x)} + \lambda_2^{\delta}(i) * \overline{T_{\max}(x)} \tag{3.18}$$

3.4.3 Related Approaches for Multi-Objective Permutation FSSP

In the literatures, for solving multi-objective permutation FSSP with the criteria of makespan and maximum tardiness, some good results are reported by Arroyo[37] and Chang[38], who employ MOGLS algorithm and NSGA-II, respectively. PQEA compares with MOGLS and NSGA-II through computational experiments. The main procedure and genetic operations of MOGLS and NSGA-II for multi-objective permutation FSSP are simply described as follows.

[1] `http://www.ie.osakafu-u.ac.jp/~hisaoi/ci_lab_e/research/pdf_file/multiobjective/EMO_ReSubmission_Ishibuchi.html`

The MOGLS and NSGA-II are genetic algorithm based multi-objective optimization techniques. A list of job number is represented as a chromosome in the population of MOGLS and NSGA-II. The genetic operations in MOGLS and NSGA-II are implemented by two point crossover and insertion mutation. Because good results appear in the researches [39] and [40], in which these genetic operations are used.

- **MOGLS:** it is same as PQEA, also decomposes this problem into a number of scalar optimization sub-problems using sum weighted approach. An iteration of the MOGLS algorithm starts with a population P with L solutions, and a set of random weights is selected to determine the objective function which should be normalized by using the Eq. 3.18. The operators of selection, recombination and mutation are applied to the elements of P until reaching a population P' with L elite solutions. Then L elite solutions are randomly selected from the current set of non-dominated solutions and included in the set P'. A restricted local search is applied to each solution in P'. More specifically, a random neighbor $y\prime$ of $y \in P'$ is generated and if its fitness is better than that of y, it replaces y, otherwise, the local search initiated from y terminates. The number of neighborhoods examined from each $y \in P'$ is limited, and a new population P is formed to start a new iteration.

- **NSGA-II:** it uses non-dominated sorting for fitness assignments of individual in population. All individuals are not dominated by any other individuals, are assigned as front rank 1. All individuals only dominated by individuals in front rank 1 are assigned into front rank 2, and so on. Selection is made using tournament between two individuals. The individual with the lowest front number is selected if the two individuals are from different fronts. The individual with the highest crowding distance is selected if they are from the same front. i.e., a higher fitness is assigned to individuals located on a sparsely populated part of the front. There are S parents and in every iteration S new individuals (offspring) are generated. Both parents and offspring compete with each other for inclusion in the next iteration.

3.4.4 PQEA Performance Measure

For the naturalness of multi-criteria problems, many performance indices should be used for comparing the performance of different algorithms [41]. The distance measure (D-measure) and coverage measure (C-measure), which are most popular metric methods, are simply described as following.

- *D-measure*: This performance measure is based on calculating the average distance of obtained non-dominated solutions with actual parato optimal results. Let G be the actual parato optimal results and A is obtained non-dominated solutions. The average distance from A to G is defined as:

$$D(A, G) = \frac{\sum_{v \in G} d(v, A)}{|G|} \tag{3.19}$$

  where $d(v, A)$ is the minimum Euclidean distance between v and the front in A. The *D-measure* should measure both the diversity and convergence at one time. The low value of *D-measure* illustrates that obtained A is very close to the actual pareto optimal solutions without missing any part of the pareto optimum. The low value of D-measure illustrates that obtained A is very close to the reference solutions, without missing any part of the reference solutions. In our experiments, the reference solutions of the tested problems are generated by great parameters with huge number of solution evaluations.

- *C-measure*: It is also called set coverage. Let A and B be two approximations to the pareto solutions of a MOO problem, $C(A, B)$ is defined as the percentage of the solutions in B that are dominated by at least one solution in A.

$$C(A, B) = \frac{|\{u \in B | \exists v \in A : v \succ u\}|}{|B|} \tag{3.20}$$

  Here, v dominates u that is represented by $v \succ u$. $C(A, B) = 1$ means that all fronts in B are dominated by or same with some fronts in A. $C(A, B) = 0$

means that there are no front in B is covered by any front of A. In general, $C(A, B) = 1$ is not necessarily equal to $1 - C(B, A)$.

The objective space of each test problem is normalized so that the minimum and maximum values of each objective among the reference solutions are 0 and 100, respectively. For example, the rectangle $[3315, 3696] \times [97, 1075]$ specified by the reference solutions of 20×20 problem was normalized into the square $[0, 100] \times [0, 100]$. Using the normalized objective space, the D-measure is calculated to compare the performance of each algorithm on multi-objective permutation FSSP.

3.4.5 Simulation Results and Analysis

To demonstrate the efficiency of proposed method, four different scaled multi-objective permutation FSSP benchmark problems are applied for comparing, 20×20 (20 jobs and 20 machines), 40×20, 60×20 and 80×20 problems, which is provided by the research[28].

The good results are reported by Arroy [37] and Chang[38] who adopt the parameters setting as follow.

- Population Size: $popSize$=80;
- Mutation Probability: P_M=0.6;
- Crossover Probability: P_C=0.8;

For fair comparisons, MOGLS and NSGA-II use the same parameters as the advisement of two papers. In MOGLS, a elite population size L needs to be set. The size of L is given by 50.

PQEA for multi-objective FSSP used the below parameters designs. The value of rotated angel is same as the one that used in Chapter 2. Based on these parameters, the number of weighted vector groups becomes 15 (N/T). Consequently, there are 15 q-bit individuals exit in population.

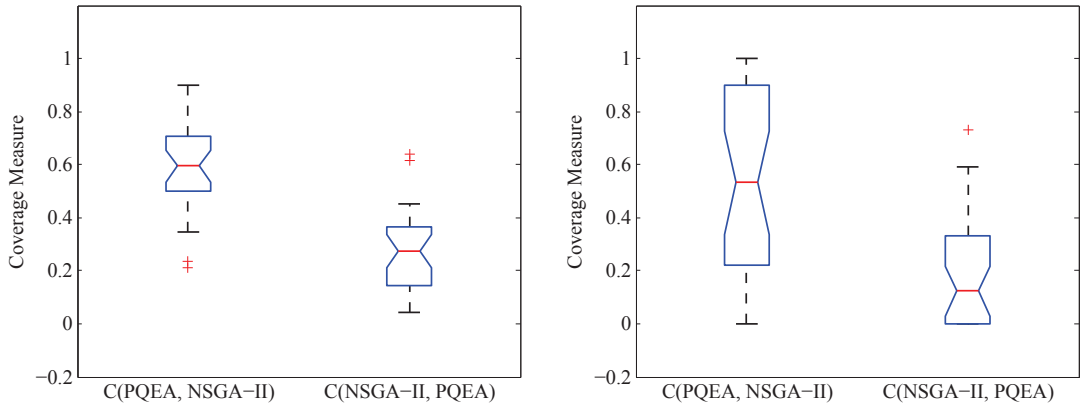
- Number of initialized weight vectors: N=150;
- Number of neighbor sub-problem in one group: T=10;
- Rotation angle value for updating quantum state: $\Delta\theta = 0.01\pi$;

Table 3.1: Comparison of PQEA using D-measure under same computational times

Test Problem	Algorithms	MOGLS	NSGA-II	PQEA	Improved
20×20	Ave	18.4	18.9	16.7	11.6%
	Min	12.8	10.5	9.7	7.6%
	Max	25.6	27.2	23.0	15.4%
	SD	3.6	3.5	4.0	—
40×20	Ave	82.1	76.9	69.5	9.6%
	Min	60.9	50.7	45.4	10.5%
	Max	95.6	99.4	97.8	1.6%
	SD	10.6	12.8	7.2	—
60×20	Ave	62.0	59.1	48.3	18.3%
	Min	54.7	50.2	35.4	29.5%
	Max	67.8	72.3	64.8	10.4%
	SD	4.5	6.1	3.4	—
80×20	Ave	549.0	505.5	453.2	10.3%
	Min	505.5	380.3	326.0	14.3%
	Max	591.9	621.5	538.4	13.4%
	SD	50.3	55.0	24.1	—

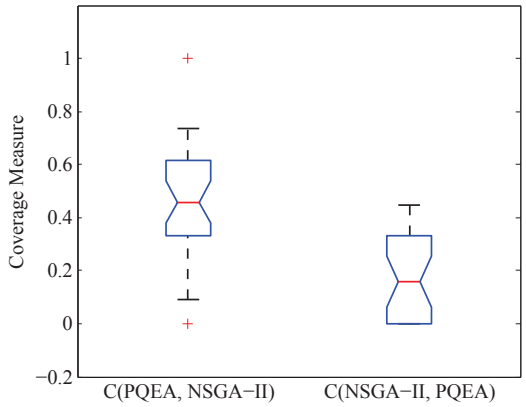
All the compared algorithms PQEA, MOGLS and NSGA-II have been independently run for 30 times for each test instance on identical computers (Centrino (R) 2.0 GHZ, 2.00 GB). The three algorithms are implemented by Java language with Eclipse SDK 3.62 version programmed by same data structures to create a fair comparison environment. The terminated condition for the three algorithms is using 5 seconds.

To have a fair comparison, three algorithms are independently executed 30 times. Table 3.1 collects the average, maximum, minimum and standard deviation of D-measure values among the 30 times execution. Comparing with NSGA-II, based on the Eq. 3.21 the average dispersion performance for 20×20, 40×20, 60×20 and 80×20 problems have been further improved by 11.6%, 9.6%, 18.3% and 10.3%, respectively. Here, D_m(PQEA) and D_m(NSGA-II) mean the average D-measure

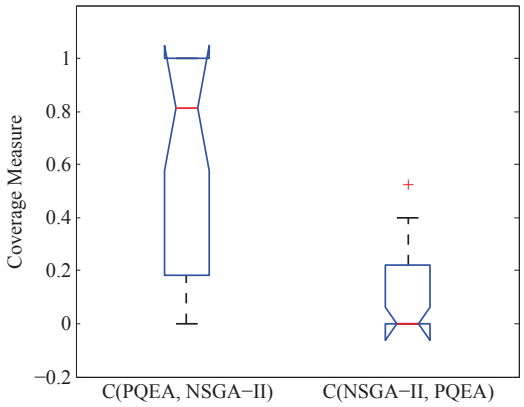


(a) 20 × 20

(b) 40 × 20



(c) 60 × 20



(d) 80 × 20

Figure 3.4: Boxplot of C-measure values on PQEA and NSGA-II for multi-objective permutation FSSP

values generated by PQEA and D_m(NSGA-II).

$$Imp\ \% = \frac{|D_m(\text{NSGA-II}) - D_m(\text{PQEA})|}{D_m(\text{NSGA-II})} \times 100 \tag{3.21}$$

The lower D-measure results illustrates that PQEA obtains near-pareto optimal solutions with even dispersion and without solutions loss of entire pareto front.

Because NSGA-II is better than MOGLS on the D-measure, to further verify the effectiveness of PQEA, NSGA-II and PQEA are compared based on the C-measure. Fig.3.4 presents the boxplot that are summarizing 30 results of C-measure values for

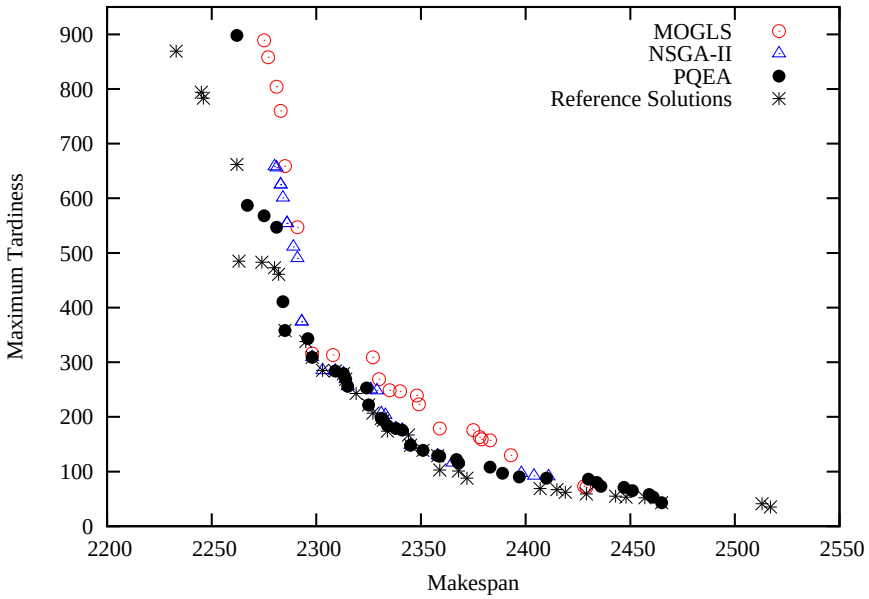


Figure 3.5: Non-dominated solutions attained by each method for 20×20 case

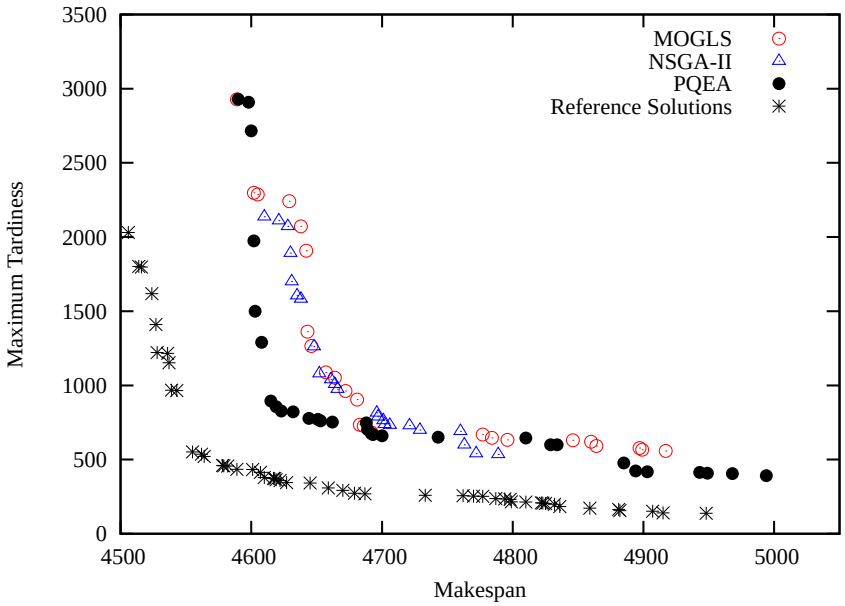


Figure 3.6: Non-dominated solutions attained by each method for 60×20 case

PQEA and NSGA-II. The larger value of coverage rate is generated by C(PQEA, NSGA-II), which indicates that most solutions of PQEA are dominating solutions of NSGA-II, reversely, only little solutions of PQEA are dominated by NSGA-II.

Fig. 3.5 and Fig. 3.6 give the non-dominated solution that generated by PQEA, MOGLS and NSGA-II under 30 runs. We can observe that PQEA performs well. In 20×20 and 60×20 problem, PQEA shows dominant on the both criteria. The dispersion of solutions of PQEA is also smooth than MOGLS and NSGA-II.

3.5 Experimental Comparisons of PQEA on MOKPs

In this section, PQEA compares with MOEA/D, MOGLS and NSGA-II on multi-objective knapsack problems (MOKPs). The MOKPs is widely used to test the performance of the multi-objective algorithms, such as the researches [22] [29]. The experiment described in this chapter is a continuation of the experiments performed by Zhang[22], Zitzler[42] and Jaszkiewicz[43]. We use a similar way of evaluating the approximations to the pareto optimal solutions.

3.5.1 Multi-objective Knapsack Problems

The MOKPs are briefly given as follows. Given a set of I items and a set of J knapsacks, where p_{ji}, w_{ji} are the profit and weight values of item i in knapsack j. The c_j is the capacity of knapsack j, we want to find a vector $x = \{x_1, x_2, \ldots, x_I\} \in \{0,1\}^I$ that maximizes

$$f_j(x) = \sum_{i=1}^{I} x_i * p_{ji}, j = 1, 2, \ldots, J \tag{3.22}$$

subject to

$$\sum_{i=1}^{I} w_{ji} * x_i \leq c_j, j = 1, 2, \ldots, J \tag{3.23}$$

The "$x_i = 1$" means that item i is selected and put in all knapsacks. The MOKPs, which is the classic combinational problem, has been demonstrated to be a NP-hard problem. Since Jaszkiewicz[43] has proposed a set of test instances as benchmark problems for MOO algorithms, many MOO algorithms has used it for testing, such as papers[22], [26], [44] [45], [46], and etc. We will also use these instances for comparison. The instances have two, three, and four objectives and 250, 500, and 750 items. The instances were generated randomly with uncorrelated profits and weights. The capacities of knapsacks are equal to half the total weight of items. As a result, about 50% of items are expected to be in the pareto optimal solutions.

3.5.2 Implementations of Comparing Algorithms for MOKPs

To implement MOO algorithm on MOKPs, it needs a repair method to recover the infeasible solutions. The greedy repair procedure presents in Algorithm 4 that proposed by Jaszkiewicz [47]. Let $x = (x_1, x_2, ..., x_I)$ be an infeasible solution for MOKPs. Due to p_{ji} and w_{ji} are nonnegative, one can remove some items from it (i.e., change the values of some x_i from 1 to 0) to make it feasible. In the approach of Fig.4, items are removed one by one from x until x^* becomes feasible. An item with the heavy weights (i.e., $\sum_{j\in\omega} w_{ji}$) in the overfilled knapsacks and little contribution to the profits of all knapsacks is more likely to be removed.

Note that the greedy repair procedure is the only heuristic that takes into account the value of the current scalarizing function. We did not use iterative improvement. The main reason for that was preservation of compatibility with the experiment of MOEA\D and MOGLS.

For the fair comparisons, the four algorithms use the same repair approach.

3.5.3 Parameters Setting

Same with the PQEA, MOEA/D also decomposes the MOO into N sub-problems by a set of even spread weighted vectors $\lambda^1, ..., \lambda^N$. In order to guarantee a fair comparison, the N for PQEA are set same as MOEA/D of Zhang[22] on all the tested instances, where N is 150, 200 and 250 for 250*2, 500*2 and 750*2, respectively.

Algorithm 4: Repair Procedure On MOKP

Input: A solution $x = (x_1, x_2, \ldots, x_I)^T$
Output: A feasible solution $x^* = (x_1^*, x_2^*, \ldots, x_I^*)^T$
begin
  while *x is not feasible* **do**
    Create the set of ψ, where
      $\psi = \{i \,|x_i = 1, and\ 1 \leq i \leq I\}$;
    Set ω is built as:
      $\omega = \{j \,|1 \leq j \leq J\ , and\ \ \sum_{i=1}^{I} w_{ji}x_i > c_j\}$;
    Select $k \in \psi$ such that:

      $k = \arg\min_{j \in \psi} \frac{g(x|\lambda) - g\left(x^{j-}|\lambda\right)}{\sum_{i \in \omega} w_{ij}}$
      where, $x^{j-} = (x_1^{j-}, x_2^{j-}, \ldots, x_I^{j-})^T$, $x_j^{j-} = 0$
      and for all $i \neq j$, $x_i^{j-} = x_i$;
    $x_k \leftarrow 0$;
  end
  $x^* \leftarrow x$.
end

(The 250*2 means the problem have 250 items and 2 knapsacks). The N is 351 for all the instances of three objectives. The neighbor size of weighted vector in a group "T" is set as 10 for all the instances. The rotation angle $\triangle\theta$ is 0.01π which is advised by last chapter for knapsack problems. The investigation of setting different T and C values is taken in next section.

The genetic operators in MOEA/D, MOGLS and NSGA-II used the one-point crossover operator and the standard mutation operator. The setting of the parameters of MOEA/D for MOKPs is identical with the paper[22], in which the mutation and crossover probabilities are selected as 0.01 and 1.00, respectively. The MOGLS and NSGA-II used same probabilities for genetic operators. In order to guarantee fair comparisons, the MOGLS adopts same parameters with Jaszkiewicz[43].

The both algorithms terminating at 100,000 individuals have been evaluated.

In our experiments, the 250, 500 and 750 items for 2, 3 objectives are considered. Zitzler [41] provided us the actual pareto solutions of 250, and 500 items for bi-criteria problems. But there is not actual pareto optimal solutions of other test

Table 3.2: Comparison of each algorithm using D-measure on two objective problem. (The smaller one is better, "SD" means standard deviation.)

Algorithms		MOEA/D	MOGLS	NSGA-II	PQEA
250*2	Ave	26.83	32.33	39.01	20.92
	Min	21.54	20.14	28.21	15.76
	Max	28.07	40.86	43.31	26.60
	SD	2.94	4.15	3.28	2.50
500*2	Ave	64.61	82.40	97.85	34.69
	Min	55.49	63.10	70.93	28.10
	Max	73.47	92.13	103.31	50.12
	SD	4.76	9.40	5.52	4.56
750*2	Ave	176.99	274.21	300.21	70.41
	Min	156.60	206.30	260.08	53.81
	Max	214.34	321.12	340.21	84.20
	SD	13.73	22.72	9.80	6.69

instances. Jaszkiewicz [43] has produced a very good upper approximation pareto optimal solutions to the other instances by using the linear programming methods, in which a number of uniformly distributed weighted vectors are applied to convert the MOKPs to many single objective problems. These single objective problems are solved one by one. In our experiments, G is set as such an approximation to compare the performance of D-measure for each algorithm.

3.5.4 Experiment Results and Discussions

E-ach algorithm has been independently executed in 30 times. Table 3.2 and 3.3 present the average, minimum, maximum and standard deviation of D-measure values in PQEA, MOEA/D, MOGLS and NSGA-II for each instance. The comparisons of average C-measure values between PQEA and other algorithms on all instances are given in Table 3.4 and Fig. 3.7.

Fig.3.9 plots the distributions of non-dominated solutions with the lowest D-measure in 30 runs of PQEA, MOEA/D and MOGLS on the 750*2 Instances. We

Table 3.3: Comparison of each algorithm using D-measure on three objectives problem. (The smaller one is better, "SD" means standard deviation.)

Algorithms		MOEA/D	MOGLS	NSGA-II	PQEA
250*3	Ave	92.14	141.71	169.21	98.81
	Min	82.97	99.12	120.12	81.97
	Max	105.45	190.67	223.10	105.36
	SD	13.02	31.23	15.31	13.03
500*3	Ave	257.39	419.23	439.10	251.5
	Min	223.68	278.91	310.63	213.72
	Max	306.27	500.10	512.32	297.30
	SD	24.42	39.02	22.19	20.69
750*3	Ave	460.98	768.41	942.30	412.57
	Min	417.93	610.82	720.32	338.87
	Max	534.18	810.23	1001.29	511.62
	SD	23.08	51.79	28.65	19.97

Table 3.4: Average C-measure values on the 6 tested instances. (The MOEA/D, MOGLS, NSGA-II, and PQEA are represented by 'E','G','N' and 'P', respectively and the bigger one is better.)

Instance	$C(P,E)$	$C(E,P)$	$C(P,G)$	$C(G,P)$	$C(P,N)$	$C(N,P)$
250*2	45.91	38.74	57.21	36.82	71.32	19.32
500*2	88.82	5.48	92.11	3.82	79.72	12.91
750*2	97.78	0.89	94.21	5.91	89.31	12.10
250*3	39.31	25.63	45.78	32.91	60.86	23.93
500*3	60.92	7.43	54.31	12.12	70.67	19.82
750*3	78.55	1.54	69.32	2.51	82.30	22.31

can learn from Table 3.2 and 3.3 that MOEA/D performs better than MOGLS and NSGA-II. For further teasing the PQEA and MOEA/D, Fig.3.10 shows the evolution of the average D-measure value in 30 runs with the number of the evaluation times (the times of calling of the repair method) in PQEA and MOEA/D for each test instance.

Based on the experiment results, we can make the following remarks:

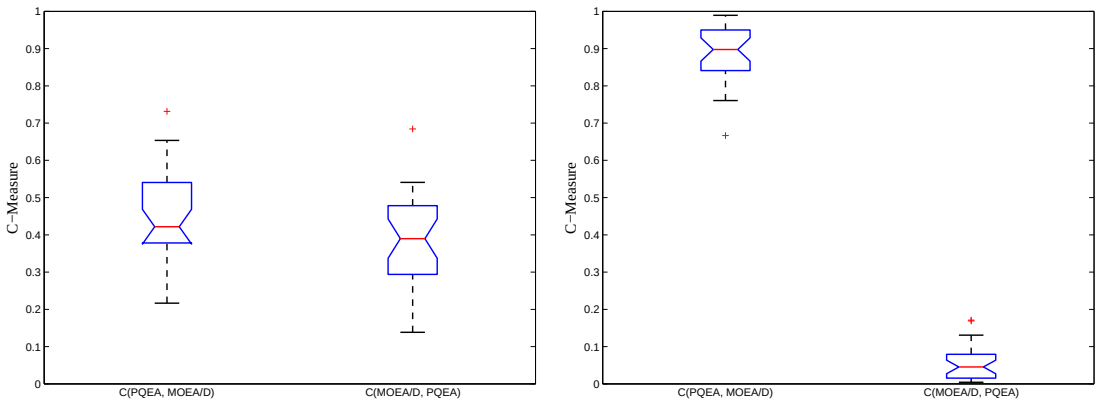
- Table 3.2, 3.3 and 3.4 reveal that the final obtained non-dominated solutions by PQEA is better than that obtained by MOEA/D, MOGLS and NSGA-II, in terms of both D-measure and C-measure, for the most test instances except instance 250*3 in which PQEA is slightly worse than MOEA/D in D-measure. Taking instance 750*2 as an example, on average, 97.78% of the final non-dominated solutions generated by MOEA/D are dominated by those generated by PQEA, and only 2.42% vice versa. The 92.11% and 89.31% of the final non-dominated solutions of MOGLS and NSGA-II are dominated by PQEA.

- Table 3.2, 3.3 also show that the standard deviation of D-measure values in PQEA, MOEA/D, MOGLS and NSGA-II. The smaller standard deviation values has implied that PQEA and MOEA/D have more stable performance than MOEA/D and NSGA-II.

- The difference between the approximations generated by PQEA, MOEA/D, MOGLS and NSGA-II on instances 750*2 can be visually detected from Fig.3.9. The approximations of PQEA are closer to reference solutions than MOEA/D's and MOGLS's, while the approximations of NSGA-II are only centralized in middle part.

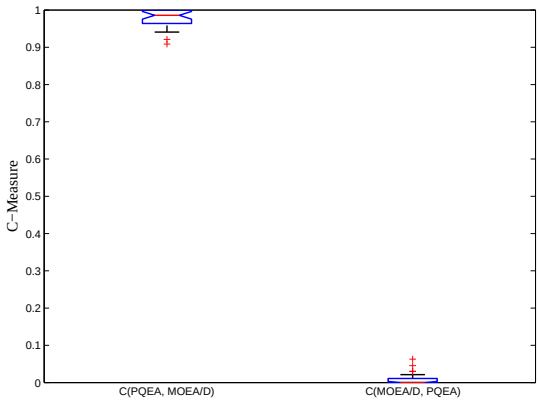
- Fig.3.10 clearly indicates that for the test instances of two objectives, PQEA costs fewer times of evaluation (number of solutions are tried) than MOEA/D for minimizing the D-measure value, which suggests that PQEA is more efficient and effective than MOEA/D on the two objectives. The two algorithms perform similar on the three objectives of MOKPs.

Based on the above analysis, we can summarize that the PQEA obviously outperforms MOGLS, NSGA-II and MOEA/D on the bi-objective problems, and PQEA produces similar results with MOEA/D for the more objectives on these MOKPs test instances.

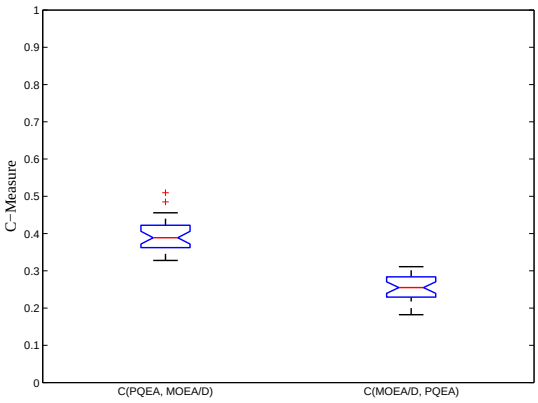


(a) 250*2

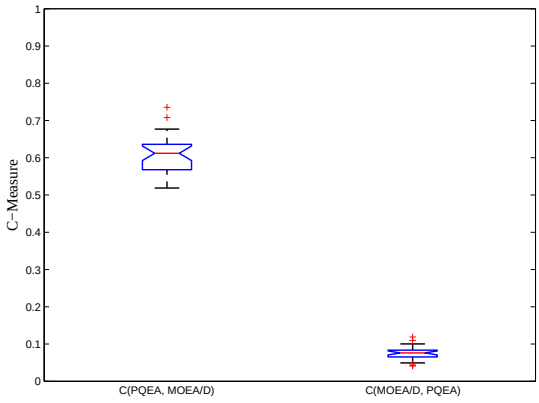
(b) 500*2



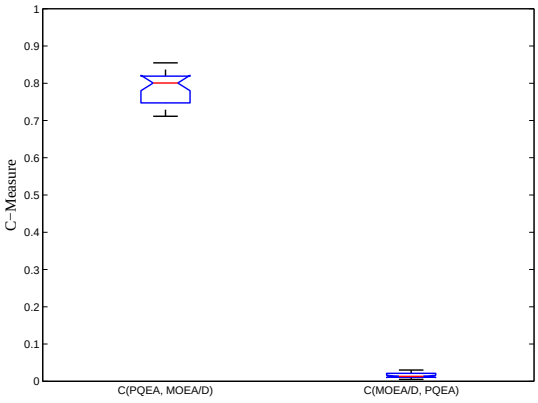
(c) 750*2



(d) 250*3



(e) 500*3



(f) 750*3

Figure 3.7: Boxplot of C-measure compared values for PQEA and MOEA/D

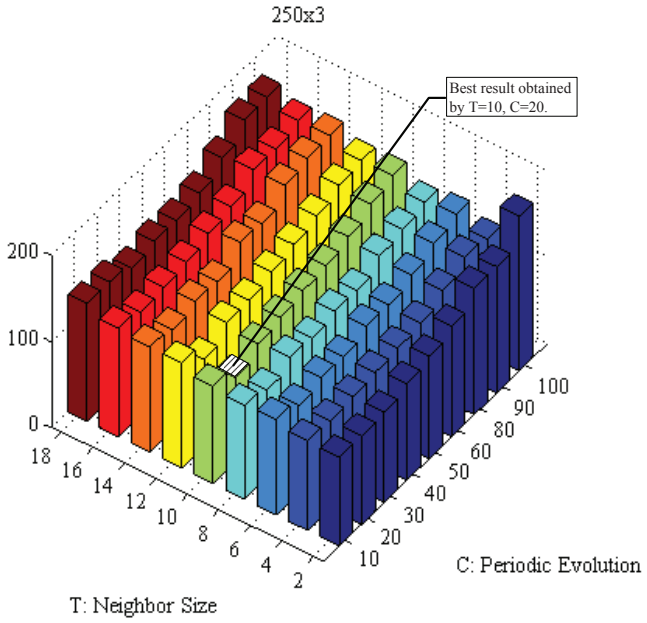


Figure 3.8: Parameters examination for PQEA

3.5.5 Investigation of Parameters Setting in PQEA

T and C are two important parameters in PQEA. In order to examine the effect of parameters T and C on the search ability of PQEA, we performed computational experiments using various specifications of T and C. More specifically, we examined 90 combinations of 9 values of T (i.e., T=2, 4, 6, 8, 10, 12, 14, 16, 18) and 10 values of C (i.e., 10, 20, 30, 40, 50, 60, 70, 80, 90, 100). Using each combination of T and C, our PQEA was applied to each test problem 30 times under the same stopping condition (i.e., evaluation of 100,000 solutions). The average value of the D-measure obtained from each combination of T and C is shown in Fig.3.8 for the 250×3 test problem where shorter bars mean better solutions. In this figure, we can observe that good results were obtained from combinations of T and C in where $C = 20$, and $T = 8, 10, 12$.

3.6 Summary

This chapter proposes a parallel quantum evolutionary algorithm (PQEA) for multi-objective optimization on discrete problems. PQEA uses a set of even weighted vectors to decompose the MOOP into a number of scalar optimization sub-problems. Each sub-problem is associated with a weighted vector. All sub-problems are classified into several groups based on the their similarities. The groups of sub-problems are parallel evolved by a population of q-bit individuals. Because the neighboring sub-problems in one group should have close optimal solutions, a q-bit individual is applied to orderly address the sub-problems in one group. All groups of sub-problems are optimized in parallel. On the other hand, the probabilistic representation is the most features of q-bit individual. The evolutionary information for optimizing the sub-problem in the q-bit individual can be shared to optimize its neighborhoods. This work utilizes the feature of q-bit individual and the similarities of solutions of sub-problems in one group to find pareto optimal solutions for MOOP.

To solve the multi-objective permutation FSSP, PQEA are implemented with a transformation procedure which is describing in last chapter. The benchmark problem is employed to compare the performance of PQEA and state of art algorithms MOGLS and NSGA-II. Through numerous experimental tests, the results show that PQEA outperforms than MOGLS and NSGA-II with near 12.4% improved dispersion performance on the 4 different scaled problems. The coverage rate comparisons among each algorithm are evaluated, PQEA is also competitive.

To further confirm the applicability of PQEA, other kind of combinatorial problem multi-objective knapsack problems are applied to compare with MOEA/D, due to the high efficacy of MOEA/D on this problem. The experimental results have shown that PQEA is obviously better than MOEA/D in two objective problems. The PQEA is able to generate much better non-dominated solutions than those produced by MOGLS and NSGA-II in the same number of function evaluations on more objective problems, in which PQEA and MOEA/D obtain similar performance.

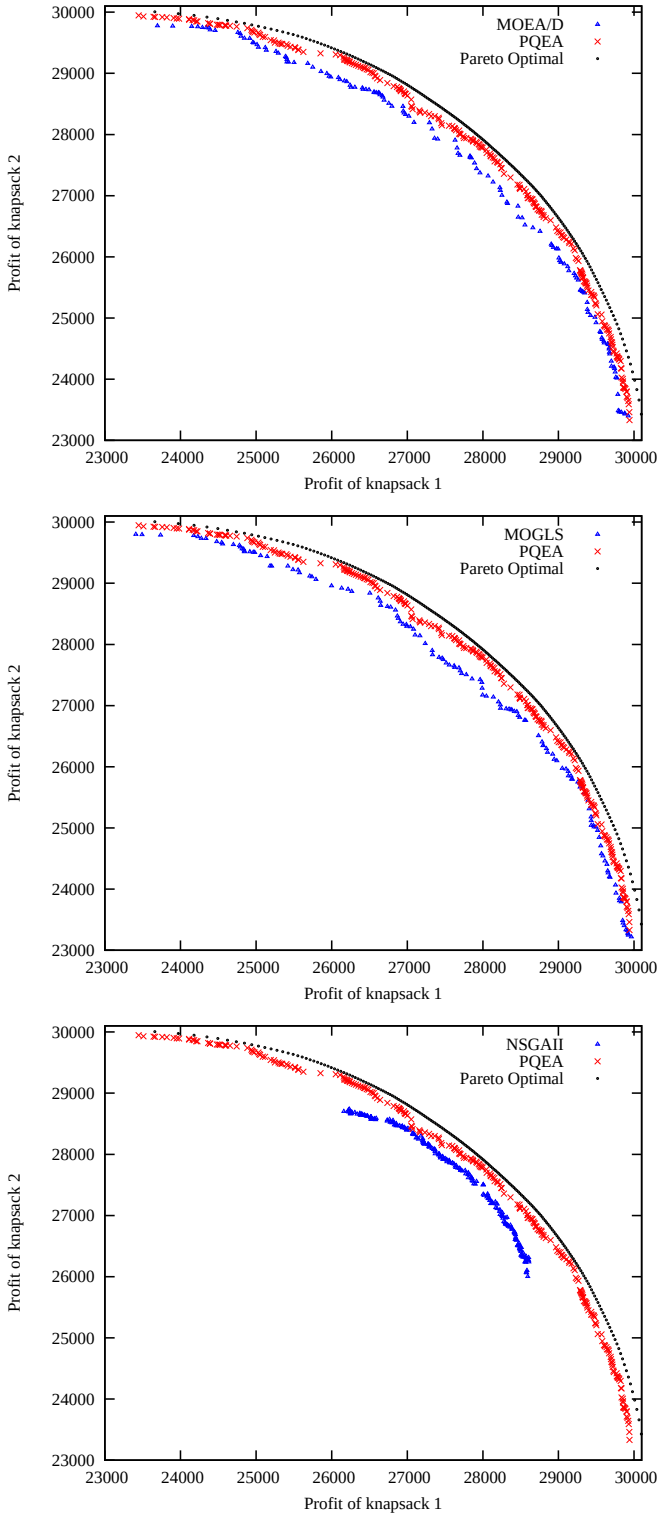
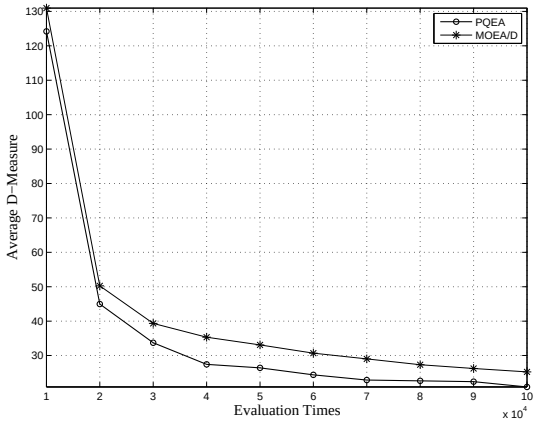
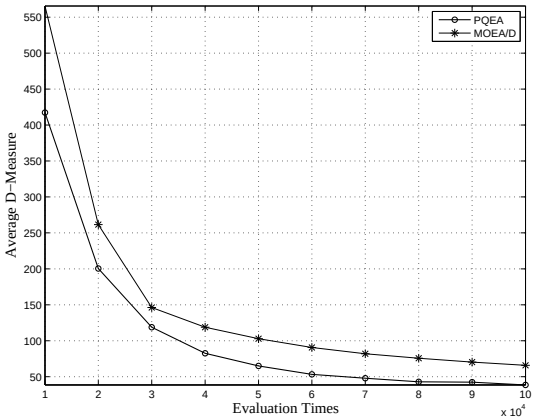


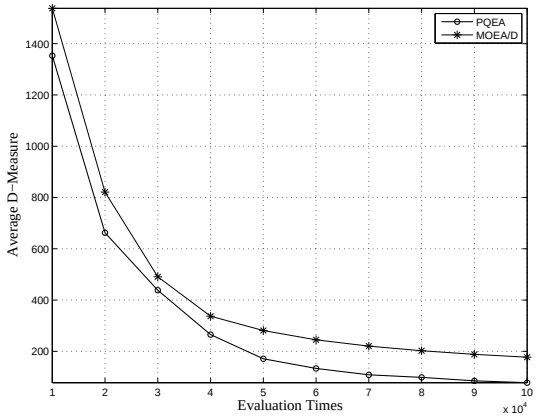
Figure 3.9: Plots of non-dominated solutions with the lowest D-measure in 30 Runs of PQEA, MOEA/D and MOGLS on the 750*2 instances



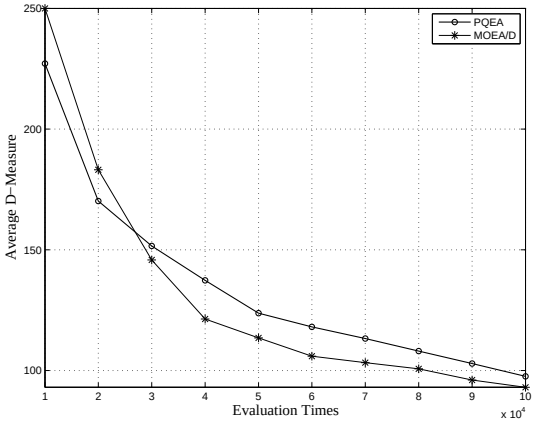
(a) 250 * 2



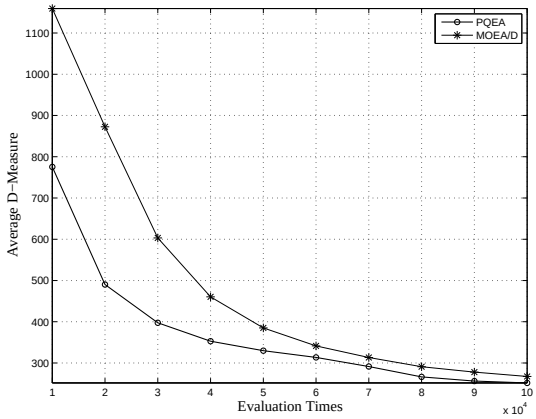
(b) 500 * 2



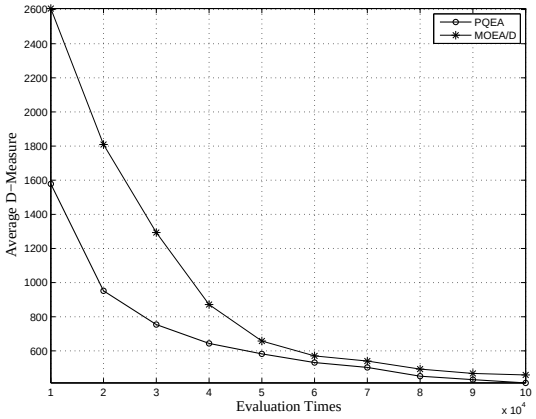
(c) 750 * 2



(d) 250 * 3



(e) 500 * 3



(f) 750 * 3

Figure 3.10: Comparison of average D-measure values for PQEA and MOEA/D during evolution progression

Chapter 4

Hybridization of ANS for Multi-Objective Permutation FSSP

4.1 Introduction

The hybridization of a proper local search with evolutionary algorithm can improve the convergence and stable performance of an algorithm to find a near-optimal solution [48]. For permutation FSSP, there are many researchers developed hybridization methods to address permutation FSSP on the criterion of minimizing makespan, such as Ekşioğlu et al. [49], Tasgetiren et al. [50], and Ben Daya et al. [51]. Because of many benefits can be yielded from makespan minimization, such as increasing productivity, saving electricity and labor cost etc [52]. On the other hand, Grabowski et al. [53] [54] and Koulamas [55] proposed the hybrid approaches to solve the permutation FSSP with the single criterion of minimizing maximum tardiness. These approaches gained better results on permutation FSSP with the criterion of minimizing makespan or maximum tardiness. However, it cannot be used in multi-objective permutation FSSP, because one criterion is considered and

other criterion is ignored.

The last chapter has discussed the performance of PQEA, MOGLS and NSGA-II on multi-objective permutation FSSP. These approaches are pure evolution algorithm without integrating any properties of permutation FSSP. To further improve solution quality, the multi-objective local search should be designed to combine with these approaches. However, in the literatures, there is seldom found the such local search procedure except Ishibuchi et al. [56]. In his approach, the individuals of population are selected to perform local search based on a probability parameter. The local search procedure is implemented by generating some neighborhood solutions using a neighborhood structure. A set of arbitrarily weighted parameters is applied to control the search direction of local search procedure. They also discuss the balance between genetic search and local search for multi-criteria permutation FSSP, and asserted that local search can be much more efficiently executed than genetic search while more effective neighborhood structures are constructed.

Although this approach has obtained lots of good solutions, the common neighborhood structure is a cause of low efficiency on optimizing bi-criteria simultaneously. And, using local search procedure for an individual that bases on randomly weighted parameters would waste much computational resource on searching an uncertain and unpredicted direction. To improve the search ability to reach more extensive and distributed solutions, in the beginning, this chapter proposes two neighborhood structures that are called as MOINS and MOEXC. Any movement based on MOINS and MOEXC attempts to reduce both criteria: makespan and maximum tardiness. Moreover, Alternated Neighborhood Search (ANS) is developed. The search direction of ANS on an individual is naturally decided by interaction of MOINS and MOEXC instead of randomly weighted parameters. To verify the efficiency of ANS, four different scaled problems are used to test. The formal local search procedure is implemented with various kinds of neighborhood structures compares to ANS.

This chapter is organized as follows. Section 2 briefly introduces how to identify the critical jobs and active blocks in a permutation FSSP. The proposed neighborhood structures MOINS and MOEXC are described in Section 3 in details. Mean-

while, the core of ANS is taken in later. The hybridization of ANS with multi-objective evolutionary algorithm on bi-criteria permutation FSSP and comparison are the subject of Section 4, which is divided into two parts: methodology of comparison and experimental results. The last section concludes this research and offers the future perspectives.

4.2 Critical Jobs and Active Block Identification

Although the critical jobs and active blocks are very significant properties for permutation FSSP, they are only applied on minimizing makespan until now. Grabowski [57] [54] and Ruiz [58] have used the properties to develop some neighborhood structures on the single criterion permutation FSSP (makespan minimization). However, in this chapter, the new neighborhood structures that simultaneously minimize both criteria are proposed by utilizing the properties. The definitions of critical jobs and active blocks are reviewed as following.

The permutation FSSP can be formulated as follows. Each of n jobs from the set $J = \{J_1, J_2, \ldots, J_n\}$ has to be processed on m machines $M_1, M_2, \ldots, M_m$. Without loss of generality, it is assumed that the jobs are processed in the order of indices of machines. O_{ji} is defined as the ith operation of job J_j processed on machine M_i, and working time of O_{ji} is p_{ji}. A sequencing of jobs can be represented by a permutation $\pi = (\pi(1), \ldots, \pi(n))$, where $\pi(a)$ is the ath element of permutation π. As we described in Chapter 2, the permutation FSSP with this criterion is to find a such permutation π^* from Ω:

$$\pi^* = \arg\min_{\pi \in \Omega} C_{\max}(\pi) \tag{4.1}$$

For any permutation π, the makespan $C_{max}(\pi)$ value can be found by the following equation:

$$C_{max}(\pi) = \underset{1 \le t_1 \le \ldots \le t_{m-1} \le n}{Max} \sum_{j=1}^{t_1} p_{\pi(j)1} + \sum_{j=t_1}^{t_2} p_{\pi(j)2} + \ldots + \sum_{j=t_{m-1}}^{n} p_{\pi(j)m} \tag{4.2}$$

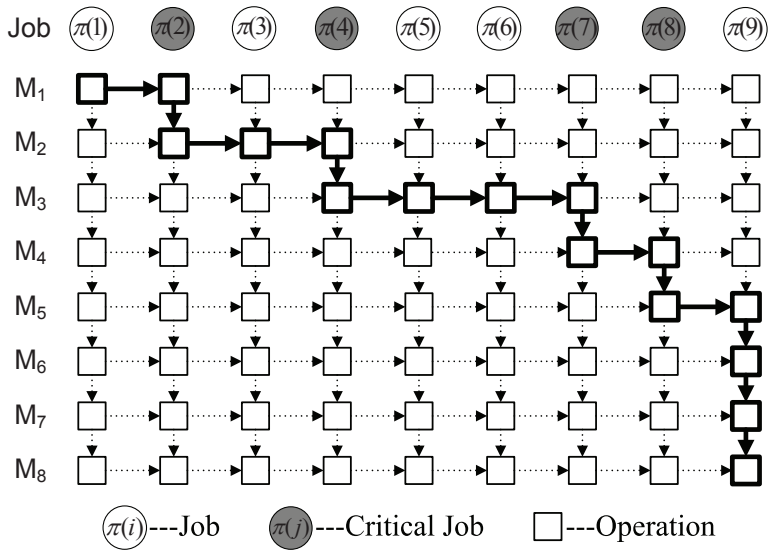


Figure 4.1: Grid graph of a permutation FSSP instance with 8 machines and 9 jobs

Here, the sequence of integers $t = (t_1, t_2, ..., t_{m-1})$ defines a path $P(\pi, t)$ from beginning operation $O_{\pi(1),1}$ to last operation $O_{\pi(n),m}$, where the t_i ($t_i \in t$) presents the element index in permutation π.

Let $L(P(\pi, t))$ be the length of path $P(\pi, t)$. It can be written as:

$$L(P(\pi,t)) = \sum_{j=1}^{t_1} p_{\pi(j)1} + \sum_{j=t_1}^{t_2} p_{\pi(j)2} + ... + \sum_{j=t_{m-1}}^{n} p_{\pi(j)m} \tag{4.3}$$

Then, for any permutation π we can get that:

$$C_{max}(\pi) \geq L(P(\pi,t)) \tag{4.4}$$

$P(\pi, t)$ is any path in π. The longest path $P(\pi, l)$, $l = (l_1, l_2, ..., l_{m-1})$, gives the makespan value of permutation π such as:

$$C_{max}(\pi) = L(P(\pi,l)) \tag{4.5}$$

The path $P(\pi, l)$ is also called as critical path [51]. The jobs of set A are defined

as critical jobs of π, where

$$A = \{\pi(l_j) \,|\pi(l_j) \neq \pi(l_{j'}) \;\; l_j, l_{j'} \in l\,\} \tag{4.6}$$

A sequence of jobs

$$B_k = \{\pi(l_{k-1}),\, \pi(l_{k-1}+1),\, ...,\, \pi(l_k)\} \tag{4.7}$$

is called the kth active block in π, and $k \in \{1, 2, ..., |A|\}$. The $\pi(l_{k-1})$ and $\pi(l_k)$ are adjacent critical jobs of permutation π. If $k = 1$, l_0 is defined as l_0=1.

For the instance of Fig. 4.1, this permutation $\pi = (\pi(1), \pi(2), \pi(3), \pi(4), \pi(5), \pi(6), \pi(7), \pi(8), \pi(9))$, we can find the longest path $P(\pi, l)$, $l = (2, 4, 7, 8, 9, 9, 9)$, and the set of critical jobs is $A = \{\pi(2), \pi(4), \pi(7), \pi(8), \pi(9)\}$. Based on above definitions, there are 5 active blocks B_1, B_2, B_3, B_4 and B_5 for this instance, where, B_1=$\{\pi(1), \pi(2)\}$, B_2=$\{\pi(2), \pi(3), \pi(4)\}$, B_3=$\{\pi(4), \pi(5), \pi(6), \pi(7)\}$, B_4=$\{\pi(7), \pi(8)\}$, B_5=$\{\pi(8),\, \pi(9)\}$.

4.3 Alternated Neighborhood Search

Alternated Neighborhood Search (ANS) contains two main parts: "multi-objective neighborhood structures" and "ANS procedure". The multi-objective neighborhood structures are proposed based on two theorems in permutation FSSP. The interaction of neighborhood structures that guides search direction presents in ANS procedure.

We can find many types of neighborhood structures that are using a movement based on interchanging jobs on machines in the literatures[51][59][60]. The neighborhood structures of "Two Point Insertion" and "Two Point Exchange" are the most commonly applied and described as follows.

Firstly, let $(u,\, v)$ be a pair of position numbers in a permutation π.

- Two Point Exchange (EXC): exchange the positions of the jobs that currently located in u and v.

- Two Point Insertion (INS): step 1, remove the job $\pi(u)$ from the schedule π, and the jobs between u and v are arranged forward one unit; step 2, set job $\pi(u)$ to position v.

It is noted that u is always smaller than v, because the double numbers are randomly generated, the small one is given to u, and bigger one is set as v.

4.3.1 Theorems about Two Criteria

Based on the two theorems, new neighborhood structures for bi-criterion permutation FSSP are presented in this subsection. The theorem 1 has been stated by Graboski [57]. It was applied to minimizing makespan using INS or EXC by the papers [61] [57] [59]. Theorem 2 only focuses on minimizing maximum tardiness. The description and proof procedure of theorem 2 are originally illustrated by us. Here, $\pi(u)$ and $\pi(v)$ represent the uth and vth jobs in π.

Theorem 1 (Grabowski[57]): Suppose the permutation π'' is generated from π by a movement. The movement is to exchange the job $\pi(u)$ and $\pi(v)$ or insert $\pi(u)$ to v on π. If $\pi(u)$ and $\pi(v)$ are in the same active block, then:

$$C_{max}(\pi'') \geq C_{max}(\pi) \tag{4.8}$$

Theorem 2: For any permutation π, $\pi(t)$ is maximum tardiness job in π. The permutation π'' is obtained by the movement of $\pi(t)$ to v. If $v > t$, then:

$$T_{max}(\pi'') \geq T_{max}(\pi) \tag{4.9}$$

Proof:

Because of $v > t$, we can assume that x is an integer from $\{0, 1, ..., v - t\}$ and

$$t + x = v \tag{4.10}$$

The permutation π likes as:

$$\pi = (\pi(1), \pi(2), \ldots, \pi(t-1), \pi(t), \ldots, \pi(v), \ldots, \pi(n)).$$

After the movement of $\pi(t)$ to v, we get that:

$$\pi'' = (\pi(1), \pi(2), ..., \pi(t-1), ..., \pi(v), ..., \pi(t), ..., \pi(n)).$$
$$\pi''=(\pi''(1), \pi''(2), ..., \pi''(t-1), \pi''(t), ..., \pi''(v), ..., \pi''(n)).$$

The completed time of jobs $\pi(t)$ and $\pi''(v)$ are $C^m_{\pi(t)}$, $C^m_{\pi''(v)}$ on the last machine M_m.

Besides, It can be learned from π and π'' that sub-permutations $(\pi(1), \pi(2), ..., \pi(t-1))$ and $(\pi''(1)\ , \pi''(2), ..., \pi''(t-1))$ are same, moreover $\pi(t)$ and $\pi''(v)$ are identical jobs.

So, we have

$$\begin{aligned} &C^m_{\pi(t-1)} = C^m_{\pi''(t-1)} \\ &D_{\pi''(v)} = D_{\pi(t)} \\ &P_{\pi''(v),m} = P_{\pi(t),m} \end{aligned} \tag{4.11}$$

Based on the Eq. 2.15, the following formulas hold,

$$C^m_{\pi(t)} \geq C^m_{\pi(t-1)} + P_{\pi(t),m} \tag{4.12}$$

$$\begin{aligned} C^m_{\pi''(v)} \quad = \quad & C^m_{\pi''(t+x)} \geq C^m_{\pi''(t-1)} + P_{\pi''(t),m} \\ & + P_{\pi''(t+1),m} + ... + P_{\pi''(t+x),m} \end{aligned} \tag{4.13}$$

Using Eq. 4.13 to minus Eq. 4.12, below equation can be obtained.

$$C^m_{\pi''(v)} - C^m_{\pi(t)} \geq P_{\pi''(t+1),m} + ... + P_{\pi''(t+x),m} \tag{4.14}$$

and,

$$P_{\pi''(t+1),m} + ... + P_{\pi''(t+x),m} \geq 0 \tag{4.15}$$

therefore,

$$C^m_{\pi(v)} - C^m_{\pi(t)} \geq 0 \tag{4.16}$$

According to the Eq. 3.11, we can get that:

$$\begin{aligned} &T_{\max}(\pi) = T_{\pi(t)} = C^m_{\pi(t)} - D_{\pi(t)} \\ &T_{\max}(\pi'') \geq T_{\pi''(v)} = C^m_{\pi''(v)} - D_{\pi''(v)} \end{aligned} \tag{4.17}$$

Thus, the following equation is generated.

$$T_{\max}(\pi'') - T_{\max}(\pi) \geq C^m_{\pi''(v)} - C^m_{\pi(t)} \tag{4.18}$$

By the Eq. 4.16 and Eq. 4.18, we can conclude that:

$$T_{\max}(\pi'') - T_{\max}(\pi) \geq 0 \tag{4.19}$$

This completes the proof.

Theorem 1 encourages the movement on $\pi(u)$ and $\pi(v)$ that belong to different active blocks to decrease the makespan $C_{max}(\pi)$. Theorem 2 states a way to reduce search space for finding a solution that minimizes maximum tardiness.

4.3.2 Multi-objective Neighborhood Structures

The targets of above theorems are for minimizing makespan and maximum tardiness, respectively. Integrating the two theorems, any movement based on the multi-objective neighborhood structures simultaneously improves two objectives. The multi-objective neighborhood structures: multi-objective insertion (MOINS) and multi-objective exchange (MOEXC) are proposed as below.

4.3.2.1 Definitions of MOINS and MOEXC

Firstly, some notations are given. Let permutation π is π=$(\pi(1), \pi(2), ..., \pi(t), ..., \pi(n))$. The job $\pi(t)$ has maximum tardiness. There are k active blocks in π, such as $\{B_1, B_2, ..., B_{h-1}, B_h, B_{h+1}, ..., B_k\}$. According to Eq. 4.7, suppose the job $\pi(t)$ in the hth active block B_h ($\pi(t) \in B_h$).

Definition 1. MOINS: Let π be a current permutation. A neighbor of π using "two point insertion" scheme is generated by inserting job $\pi(u)$ to position v. The values of u and v are listed as Table 4.1.

The work flow of MOINS presents as:

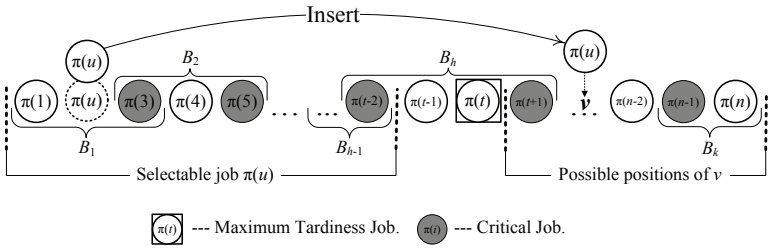


Figure 4.2: MOINS neighborhood structure

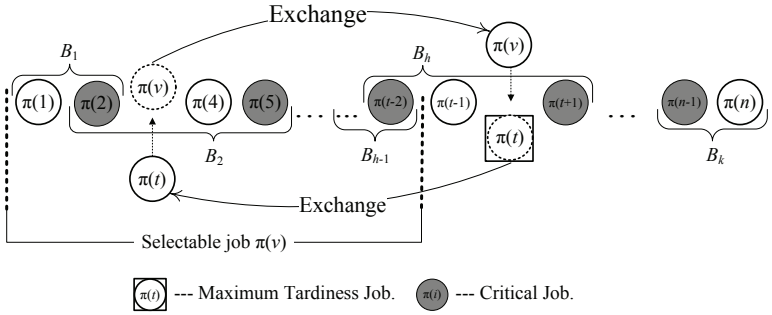


Figure 4.3: MOEXC neighborhood structure

Step 1. Randomly generate an integer u, which is limited to $[1, t)$. Here, $[1, t)$ means u belongs to the range that is from 1 to t, including 1 but t is exclusive.

Step 2. Identify which category the job $\pi(u)$ belongs to. As Table 4.1, the category 1 is $\pi(u) \in B_i$, $i = 1, 2, ..., h-1$. If $u \in \{x|(\pi(x) \in B_h) \cap (x < t)\}$, it is defined as category 2.

Step 3. Determine the parameter v based on categorization of $\pi(u)$. If the $\pi(u)$ is in category 1, v is randomly generated from $[t+1, n]$. Otherwise, v is given by $\pi(v)$ which is randomly selected from B_j, $j = h+1, h+2, ..., k$.

Step 4. Move the job $\pi(u)$ to v by "two point insertion" scheme.

In MOINS, job $\pi(u)$ which is in front of $\pi(t)$ will be inserted to v, and v is behind t. Fig. 4.2 shows the new permutation π'' is obtained by MOINS using the first category. The job $\pi(u)$, $(u = 2, \pi(2) \in B_1)$ is moved to position v $(v > t+1)$ to get π'', such as: $\pi'' = (\pi(1), \pi(3), \pi(4), ..., \pi(t), ..., \pi(v), \pi(2), \pi(v+1), ..., \pi(n))$.

Definition 2. MOEXC: Let π be a current permutation. The job $\pi(t)$ has

Table 4.1: Movement of job positions in neighborhood structure MOINS

	Job $\pi(u)$	Possible positions of v
Category 1	$\pi(u) \in B_i$, $i = 1, 2, ..., h-1$	$v \in [t+1,\ n]$
Category 2	$u \in \{x \mid (\pi(x) \in B_h) \cap (x < t)\}$	$\pi(v) \in B_j$, $j = h+1, h+2, ..., k$

maximum tardiness in π. A neighbor of π using "two point exchange" scheme is generated by interchanging the positions of job $\pi(t)$ and $\pi(v)$, where $\pi(v) \in B_i$, $i = 1, 2, ..., h-1$.

The processes of MOEXC like as:

Step 1. Randomly selected a job $\pi(v)$ from B_i, where $i = 1, 2, ..., h-1$. Specially, if $h = 1$, B_i is defined to B_1.

Step 2. Exchange the jobs $\pi(t)$ and $\pi(v)$ by "two point exchange" scheme.

The maximum tardiness job $\pi(t)$ exchanges with the jobs that belong to different active blocks, and these active blocks are before $\pi(t)$. The movement that is based on MOEXC can be visualized on Fig. 4.3. The new permutation π'' is generated as: $\pi''=(\pi(1), \pi(2), \pi(t), \pi(4),\ ..., \pi(t-1), \pi(v), \pi(t+1), ..., \pi(n)$.

4.3.2.2 Investigation of MOINS and MOEXC

According to Theorem 1, in order to reduce the makespan $C_{max}(\pi)$, the jobs that are applied for MOINS or MOEXC are selected from different active blocks. On the other hand, both MOINS or MOEXC forwardly move the maximum tardiness job $\pi(t)$ from the previous position, to reduce $T_{max}(\pi)$.

Suppose π_{ins} and π_{exc} are generated by π using MOINS and MOEXC, respectively.

π_{ins}=$(\pi(1), ..., \pi(u-1), \pi(u+1), ..., \pi(t), \pi(t+1), ..., \pi(v), \pi(u), \pi(v+1), ..., \pi(n))$.

Because the job $\pi(u)$ is inserted to position v in MOINS, the maximum tardiness job $\pi(t)$ moves from t to $t-1$. As the description of theorem 2, the $T_{max}(\pi_{ins})$ is smaller than $T_{max}(\pi)$ in most cases.

π_{exc}=$(\pi(1), ..., \pi(v-1), \pi(t), \pi(v+1), ..., \pi(t-1), \pi(v), \pi(t+1), ..., \pi(n))$.

In π_{exc}, the job $\pi(t)$ exchanges with $\pi(v)$. The maximum tardiness job $\pi(t)$ is moved to position v, where $v < t$. It also complies with theorem 2 for reducing maximum tardiness.

As above mentioned, MOEXC and MOINS are designed by integrating the two theorems. Any movement based on MOEXC or MOINS simultaneously considers the effect of two criteria.

4.3.3 Alternated Neighborhood Search Procedure

The basic idea of ANS is systematically changing of neighborhood structures within a local search. The interaction between neighborhood structures naturally explore the search direction into dominant area. Let $N_k, (k = 1, 2)$ denote a set of pre-selected neighborhood structures, permutation δ is selected for applying Alternated Neighborhood Search procedure. N_1 and N_2 use neighborhood structures of MOINS, MOEXC respectively. The pseudo-codes of ANS procedure present on Algorithm 5. The parameter k decides when and which neighborhood structure is called. While k is equal to 1, neighborhood structure N_1 always takes to search until it cannot deduce makespan any more. Otherwise, k is set to 2, neighborhood structure N_2 is invoked. However, in the process of N_2, if a better permutation δ^{N_2} with low maximum tardiness is found, k is renewed by 1 so that N_1 is applied for continuing search based on permutation δ ($\delta^{N_2} \rightarrow \delta$). If the procedure cannot find a permutation with lower maximum tardiness by N_2, then $k \leftarrow 3$. The procedure of ANS will stop.

The interaction of double neighborhood structures can be conclusively visualized as Fig. 4.4. The permutation δ is applied for local search. The new permutations that are sequentially generated by ANS represent in alphabetical order (for example, $\delta_a^{N_1} \rightarrow \delta_b^{N_1} ... \rightarrow \delta_h^{N_2}$). Solid lines represent a permutation with lower makespan or

Algorithm 5: Alternated Neighborhood Search

Input: δ - The schedule is selected to apply local search.
m - Makespan of current schedule δ.
t - Maximum tardiness of current schedule δ
Output: S(δ) - A set of neighborhood schedules of δ
begin
  Set $k \rightarrow 1$, and $k_{max} \rightarrow 2$;
  while $k < k_{max}$ **do**
    if $k == 1$ **then**
      Schedule δ^{INS} is obtained by MOINS neighborhood structure based on δ;
      if $f_M(\delta^{INS}) < m$ **then**
      $m \leftarrow f_M(\delta^{INS})$;
      $\delta \leftarrow \delta^{INS}$;
      else $k = k + 1$;
      $S(\delta) \leftarrow \{S(\delta) \bigcup \delta^{INS}\}$;
    end
    if $k == 2$ **then**
      Schedule δ^{EXC} is obtained by MOEXC neighborhood structure based on δ;
      if $f_T(\delta^{EXC}) < t$ **then**
      $t \leftarrow f_T(\delta^{EXC})$;
      $\delta \leftarrow \delta^{EXC}$;
      $k \leftarrow 1$;
      else $k = k + 1$;
      $S(\delta) \leftarrow \{S(\delta) \bigcup \delta^{EXC}\}$;
    end
  end
end

maximum tardiness (such as $\delta_a^{N_1}$, $\delta_b^{N_1}$ and etc.) is obtained. If the obtained permutation (such as $\delta_c^{N_1}$, $\delta_e^{N_1}$ and etc.) has big objective value, it is drawn as broken lines and curve lines indicate the former permutation gets back. In the beginning, N_1 is used. When the makespan cannot be reduced any more by N_1, the objective is turned into finding lower maximum tardiness using N_2. But if the better permutation with lower maximum tardiness is generated, the movement based on N_1 is reemployed to reduce makespan. In this instance, N_1, N_2 are called 5 times and 3

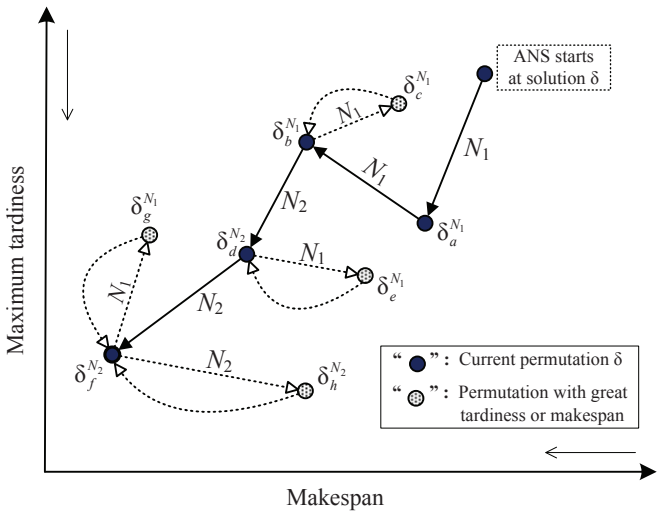


Figure 4.4: Search direction of ANS

times, respectively. Finally, local search procedure will generate a set of permutations $s(\delta)$, in which any permutation that is generated by whichever neighborhood structure is contained. Regardless of their objective values, 8 permutations of the instance in Fig. 4.4 are saved in $s(\delta)$. The search direction of this example is from δ to $\delta_f^{N_2}$, such as $\delta \rightarrow \delta_a^{N_1} \rightarrow \delta_b^{N_1} \rightarrow \delta_d^{N_2} \rightarrow \delta_f^{N_2}$.

4.3.4 Combination of ANS with Multi-Objective Approaches

4.3.4.1 Hybridization of ANS with NSGA-II

The genetic operations in NSGA-II are implemented by INS mutation and two point crossovers for multi-objective permutation FSSP. Because better solutions are generated from the combination of two point crossover and the INS mutation at the last chapter. NSGA-II uses non-dominated sorting for fitness assignments which are presented in Fig. 4.5. Individuals that are most dominant are assigned as front rank 1. The individuals only dominated by individuals in front rank 1 are assigned into front rank 2, and so on. Selection is made using tournament between two individuals.

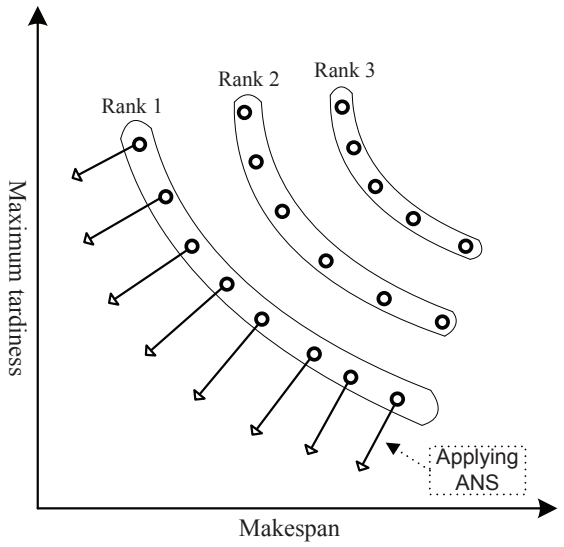


Figure 4.5: Non-dominated sorting method in NSGA-II

The combination of ANS with NSGA-II is called as NSGA-II-ANS in which the INS are used as mutation method. At every generation, only the individuals that are assigned in rank 1 employ ANS procedure. An individual which applies ANS procedure will generate an additional set of neighborhood individuals. To maintain the extensity of pareto fronts, all neighborhood individuals of the set are added into current population. Such as the example of Fig. 4.5, eight individuals in rank 1 are applied local search procedure, the obtained neighborhood individuals and former individuals in current population are put together to form a larger population. Non-dominated sorting method of NSGA-II classifies all individuals of the larger population into several ranks. The elites of individuals in larger population are filtered by tournament selection to construct the population of next generation.

4.3.4.2 Hybridization of ANS with PQEA

The combination of ANS with PQEA calls as PQEA-ANS. As the description in last chapter, the PNDS stores the non-dominated solution during evolution progression. At every generation, PNDS is updated when a new binary solution is produced by q-bit individuals. Because of the solutions in PNDS with the binary expression, it is

necessary to construct a new set 'EP' to maintain the job-based solutions. Because ANS is a job-based local search procedure. At every generation of PQEA, all binary solutions in PNDS are converted to job-based solutions to apply ANS procedure. The solutions that generated by ANS are saved in 'EP'. Until meeting the stop condition, 'EP' will output the non-dominated solutions found by PQEA-ANS.

4.4 Experiments

4.4.1 Test Problems and Performance Measures

To demonstrate the efficiency of proposed method, four different scaled multi-criteria permutation FSSP problems are applied for comparing, 20×20 (20 jobs and 20 machines), 40×20, 60×20 and 80×20. The same test data used in last chapter are applied to compare the performance of each algorithm. The D-measure and C-measure also are relied on verifying the efficiency of proposed ANS.

4.4.2 Comparisons of Neighborhood Structures and ANS

The motivation of proposing MOINS and MOEXC is explained in before section. More specifically, we have pointed out that any movement base on MOINS or MOEXC attempts to improve the two objectives, simultaneously. For examining the validity of MOINS and MOEXC, the INS, EXC, MOINS and MOEXC are implemented as mutation methods for NSGA-II to compare with our NSGA-II-ANS.

For fair comparison, the same parameters are set as following.

- Population Size: $popSize$=60;
- Mutation Probability: P_M=0.6;
- Crossover Probability: P_C=0.8;
- Terminated Condition: Eva_{max}=100,000;

Table 4.2: Comparison of neighborhood structures using D-measure by 100,000 individual evaluations (The smaller one is better)

Test Problems	Ave&SD Values	NSGA-II				NSGA-II-ANS
		INS	EXC	MOINS	MOEXC	
20×20	Ave	5.7	6.0	4.4	5.8	3.6
	SD	4.6	4.5	1.6	2.3	1.3
40×20	Ave	18.6	18.2	13.4	16.3	8.8
	SD	3.8	4.6	2.4	3.0	2.1
60×20	Ave	17.6	16.8	11.3	15.3	7.1
	SD	3.5	2.8	2.7	2.9	2.1
80×20	Ave	110.2	115.3	69.3	62.3	37.9
	SD	35.6	32.0	15.2	11.1	7.6

The Eva_{max} is the maximum number of evaluations time. The algorithms terminate at Eva_{max} solutions tried.

The performances of five methods are compared using the D-measure. The average values (Ave) of D-measure over 30 times are shown together with the standard deviation (SD) in Table 4.2. We can see this table that MOINS and MOEXC obtain better D-measure values than INS and EXC on most of problems. In 20×20, 40×20 and 60×20 problems, MOINS gets lower D-measure values than MOEXC, but MOEXC outperforms than MOINS on more complicated problem 80×20. The ANS has improved the convergence of NSGA-II by learning the results of NSGA-II-ANS, which not only products lower D-measure values but also has little SD values on all problems. The small D-measure values indicate that the non-dominated solutions of NSGA-II-ANS are closer to reference solutions than other methods. In other words, the solutions of NSGA-II-ANS are superior on both criteria.

In order to further verify the validity of proposed methods, all the comparing algorithms will terminate at same computation time. In Table 4.3, we show the average values of D-measure obtained by each algorithm when the same computation time was used as the stopping condition. The execution of each method was iterated in 15 seconds. From this table, we can get that better results were obtained from our NSGA-II-ANS than other ones. Although NSGA-II using MOEXC generates a

Table 4.3: Comparison of neighborhood structures using D-measure under same computation time. (The small one is better)

Test Problems	Ave&SD Values	NSGA-II				NSGA-II-ANS
		INS	EXC	MOINS	MOEXC	
20×20	Ave	5.8	6.7	4.1	6.0	3.9
	SD	3.5	3.8	1.8	2.3	1.4
40×20	Ave	18.2	18.0	12.4	16.5	9.7
	SD	5.0	5.8	1.8	2.3	1.64
60×20	Ave	18.4	18.9	9.5	15.3	8.1
	SD	3.1	3.8	2.7	2.9	2.2
80×20	Ave	130.3	121.8	76.4	67.1	63.9
	SD	37.0	35.8	11.2	12.4	11.4

comparable result on 20×20 problem, MOEXC greatly outperforms than INS and EXC on large scale problems. Also, the results of MOINS obviously better than INS and EXC on all problems.

4.4.3 Hybridization Approaches Comparisons

The performance of NSGA-II can be improved by hybridization with local search. To increase the convergence speed of NSGA-II to the pareto front, Ishibuchi et al. [56] have proposed a local search method combined with NSGA-II. Here, it is called as "F_NSGA-II" and simply described as follows.

F_NSGA-II: The former local search approach is implemented as following. At every generation of F_NSGA-II, first of all, it randomly generates a weight $[w_1, w_2]$ for two objectives, and $w_1 + w_2 = 1$, to birth a single objective $f(\delta)$.

$$f(\delta) = w_1 * T_{\max}(\delta) + w_2 * C_{\max}(\delta) \tag{4.20}$$

Next, based on objective $f(\delta)$ tournament selection is applied to choose a permutation δ from current population. Permutation δ with a local search probability P_{LS} is improved by a local search procedure. The local search procedure is implemented

Table 4.4: The evaluation of F_NSGA-II using different local search operations by D-measure. (The smaller one is better)

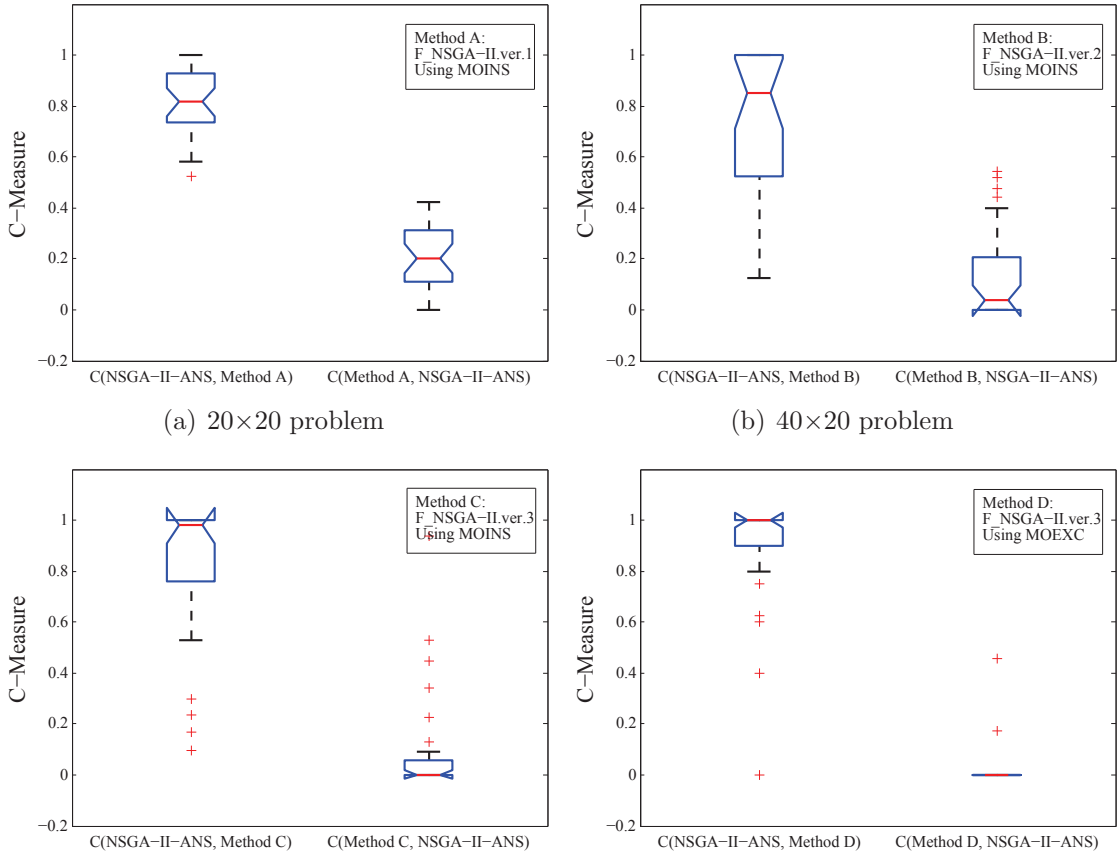
Test Problems	Ave&SD Values	F_NSGA-II.ver.1				
		INS	INSAB	EXCAB	MOINS	MOEXC
20×20	Ave	5.7	5.0	5.1	4.2	5.5
	SD	2.3	2.1	2.1	1.3	2.3
40×20	Ave	18.3	20.8	20.2	14.0	17.0
	SD	6.4	6.4	4.5	3.8	3.8
60×20	Ave	16.7	17.0	17.8	13.1	15.8
	SD	4.3	2.7	2.8	2.8	3.1
80×20	Ave	100.7	112.2	114.3	84.5	96.0
	SD	27.7	34.6	31.0	23.6	25.3

Table 4.5: Examination of F_NSGA-II using different local search operations by D-measure. (The smaller one is better)

Test Problems	Ave&SD Values	F_NSGA-II.ver.2		F_NSGA-II.ver.3		NSGA-II-ANS
		MOINS	MOEXC	MOINS	MOEXC	
20×20	Ave	4.7	4.5	4.7	5.9	3.6
	SD	1.2	1.6	1.3	2.4	1.3
40×20	Ave	12.8	12.9	13.3	16.7	8.8
	SD	2.7	3.1	2.9	3.3	2.1
60×20	Ave	9.6	9.8	9.5	11.3	7.1
	SD	2.5	2.5	2.3	3.7	2.1
80×20	Ave	67.4	65.4	63.5	62.0	37.9
	SD	10.5	9.8	9.0	10.9	6.6

by generating k neighbors of δ using a local search operation (such as INS or EXC). The objective $f(\delta)$ decides whether δ is replaced by neighbors of δ. All the permutations in current population are iterated using local search procedure. There are double parameters P_{LS} and k should be set in F_NSGA-II.

There are three versions of F_NSGA-II in our experiments. The F_NSGA-II.ver.1, F_NSGA-II.ver.2 and F_NSGA-II.ver.3 have used the mutation method INS, MOINS and MOEXC, respectively. The parameters of *popSize*, P_M, P_C and Eva_{max}



(a) 20×20 problem (b) 40×20 problem

(c) 60×20 problem (d) 80×20 problem

Figure 4.6: Comparison of C-measure values in 30 executed times using Boxplot. (A is better than B, only if $C(A, B)$ is bigger than $C(B, A)$)

for F_NSGA-II are set as 60, 0.6, 0.8, and 100,000, respectively. Specifically, it is needed to give the parameters of k and P_{LS} for F_NSGA-II, which are advised as $k = 80$ and $P_{LS} = 0.02$ by many experiments. All the algorithms are severally executed 30 times for comparing the D-measure. As Table 4.5 shown, firstly, the five neighborhood structures are implemented as local search operations for F_NSGA-II.ver.1. The five neighborhood structures are INS, INSAB (jobs insertion based on adjacent active block), EXCAB (jobs exchange based on active block), MOINS and MOEXC. The INSAB and EXCAB are proposed by B. Ekşioğlu [49] and Grabowski [57], respectively, for the single objective of minimizing makespan. The INSAB is

inserting a job into adjacent active blocks, more details can be found in those papers. The EXCAB exchanges the jobs that belong to different active blocks. We can find that F_NSGA-II.ver.1 performs very well when the MOINS and MOEXC are applied as local search operations.

Learning from Table 4.2, while MOINS and MOEXC act as the mutation in pure NSGA-II, the good results are obtained. So, MOINS and MOEXC are selected as mutation methods for F_NSGA-II.ver.2 and F_NSGA-II.ver.3, respectively. Table 4.5 also gives out the results of F_NSGA-II.ver.2 and F_NSGA-II.ver.3 in which the MOEXC and MOINS are also applied as local search operations. We can see that although F_NSGA-II.ver.2 and F_NSGA-II.ver.3 have improved the D-measure results, our NSGA-II-ANS are still obviously better than F_NSGA-II.ver.2 and F_NSGA-II.ver.3.

Using the C-measure, the NSGA-II-ANS compare to the best performances of F_NSGA-II in four problems, such as F_NSGA-II.ver.1 (MOINS) for 20×20 problem, F_NSGA-II.ver.2 (MOINS) for 40×20 problem, F_NSGA-II.ver.2 (MOINS) for 60×20 problem and F_NSGA-II.ver.3 (MOEXC) for 80×20 problem. The comparisons of C-measure are shown in Fig. 4.6. Each boxplot collects the set coverage between NSGA-II-ANS and F_NSGA-II. A boxplot provides a simple graphical summary of a set of data. It shows a measure of central location (the median), two measures of dispersion (the range and inter-quartile range), the skewness (from the orientation of the median relative to the quartiles) and potential outliers (marked individually). We can see from this figure that coverage rate of NSGA-II-ANS is bigger than F_NSGA-II. It indicts that most solutions of NSGA-II-ANS are dominating F_NSGA-II, whereas only little solutions are covered by F_NSGA-II. Especially, in the larger scale 80 × 20 problem, averagely, above 80 percent non-dominated solutions of F_NSGA-II is covered by NSGA-II-ANS.

To see Table 4.2 and 4.3, some similar D-measure results are gotten by pure NSGA-II, while MOINS and MOEXC are used as mutation methods. To investigate the distribution of final non-dominated solutions, Fig. 4.7 and Fig. 4.8 state the non-dominated solutions that are generated by NSGA-II, F_NSGA-II and NSGA-II-ANS under 30 runs. It can be observed from this figure that NSGA-II-ANS performs well

Table 4.6: Improvement of ANS for PQEA and NSGA-II on D-measure. (Computation time is 15 seconds)

Test Problem	D-Measure	NSGA-II	NSGA-II-ANS	PQEA	PQEA-ANS	*Imp* %
20×20	Ave	5.8	3.9	5.1	4.2	23.5%
	SD	3.5	1.4	4.2	2.3	-
40×20	Ave	18.2	9.7	14.3	11.8	32.2%
	SD	5.0	1.64	2.9	2.3	-
60×20	Ave	18.4	8.1	13.4	10.1	39.6%
	SD	3.1	2.2	3.4	2.9	-
80×20	Ave	130.3	63.9	119.3	68.9	46.4%
	SD	35.8	11.4	38.7	7.9	-

in most instances. In 40×20 problems, The NSGA-II-ANS shows dominant on both criteria. With problem scale increased, more extensive non-dominated solutions are produced by NSGA-II-ANS. The solutions of NSGA-II-ANS not only are widely and smoothly distributed but also have great advance on both objectives on the problem of 80×20.

An attainment surface [62] is a kind of trade-off surface obtained by a single run of MOEA algorithm. It uses a dashed/dotted line to enclose the dominated area of non-dominated solutions. The 50% attainment surface shows the estimated attainment surface obtained by at least 50% of multiple runs, which represents the average performance of algorithms in many runs. As shown in Fig. 4.9 and Fig. 4.10, the performance of three methods is very close on the 20×20 problem, however with the increasing of complexity of problems, the NSGA-II-ANS is obviously better than NSGA-II and F_NSGA-II. The dominant area of NSGA-II-ANS are more wide than others, which also indicates that the ANS has improved the convergence of NSGA-II. The experimental results also certify the guidance of search direction in NSGA-II-ANS is better than randomly weighted strategy of F_NSGA-II.

4.4.4 Improvement of ANS for PQEA

In last section, numerous experiments have demonstrated the efficiency of hybridization of ANS with NSGA-II to compare with the previous research of Ishibuchi et al. [56]. In this section, the performance of hybridization of ANS with PQEA is investigated. The PQEA-ANS, NSGA-II-ANS, PQEA and NSGA-II are independently executed 30 times. All algorithms are terminated at 15 seconds. The NSGA-II adopts the INS mutation and two point crossover, because the better results are founded by these genetic operations. PQEA and NSGA-II use the same parameters as in last chapter. Table 4.6 collects average D-measure values and standard deviation (SD) obtained by each algorithm in 30 execution times.

We can learn from this table that although PQEA shows its better performance than NSGA-II, after merging ANS procedure, the NSGA-II-ANS is slightly better than PQEA-ANS. Comparing with the achievement of PQEA in last chapter, based on the Eq. 4.21 the dispersion performance for 20×20, 40×20, 60×20 and 80×20 problems have been further improved by 23.5%, 32.2%, 39.6% and 46.4%, respectively. Here, D_m(PQEA) and D_m(NSGA-II-ANS) mean the average D-measure values generated by PQEA and D_m(NSGA-II-ANS).

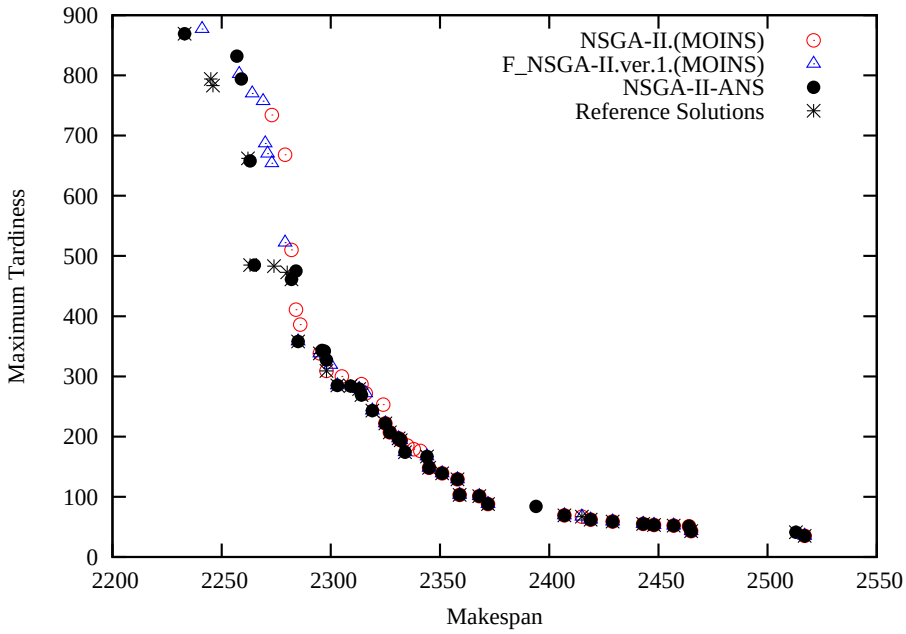
$$Imp\ \% = \frac{|D_m(\text{NSGA-II-ANS}) - D_m(\text{PQEA})|}{D_m(\text{PQEA})} \times 100 \tag{4.21}$$

4.5 Summary

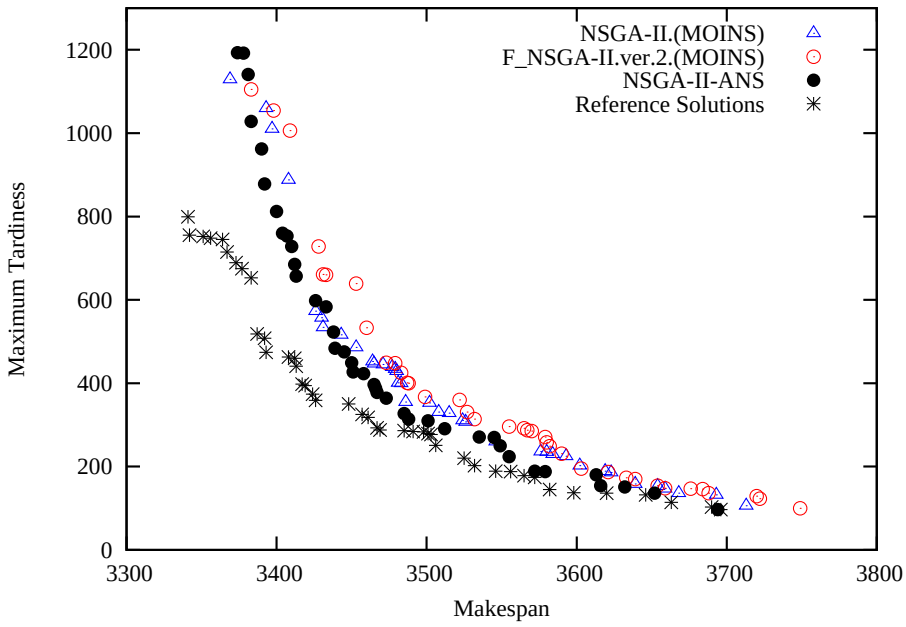
This chapter deals with the permutation flow shop scheduling problem on the bi-criteria: makespan and maximum tardiness. Alternated Neighborhood Search (ANS) procedure which contains double neighborhood structures is proposed. Any movement on a schedule based on proposed neighborhood structures simultaneously takes effect on double objectives. These neighborhood structures are not found in any literature research before. The search direction of ANS is naturally guided by interaction of proposed neighborhood structures instead of random weighted strategy. ANS integrates with famous multi-objective evolutionary algorithm, NSGA-II

(Non-dominated Sorting Genetic Algorithm-II) has improved the distribution of solutions and convergence. The numerous comparisons show efficiency of ANS is better than most of algorithms even with same number of individual evaluations, same computational time and parameter setting.

Although results of proposed approach are very attractive on most measures, as the future study, the permutation FSSP with three or more criteria motivate us to modify ANS and develop more advanced neighborhood structures.

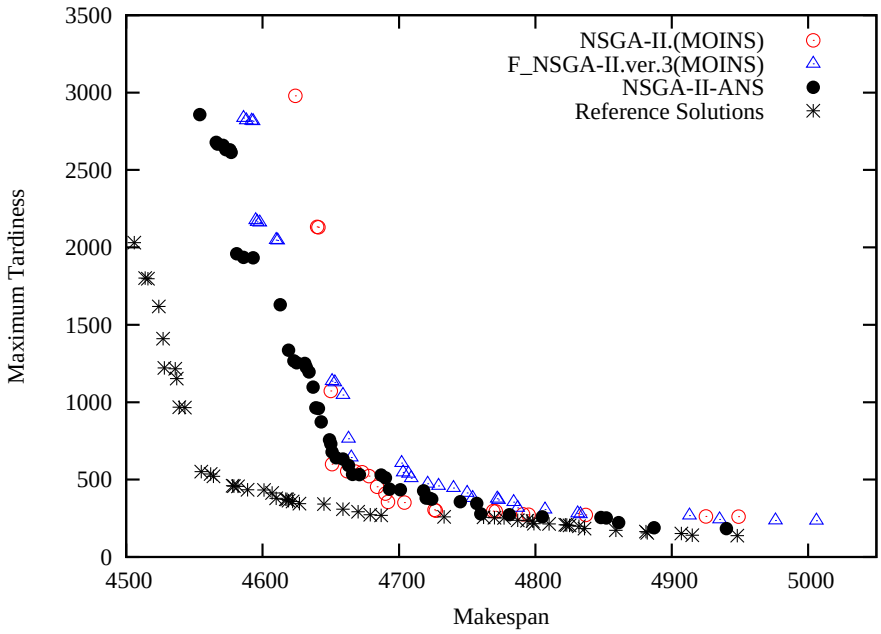


(a) 20×20 problem

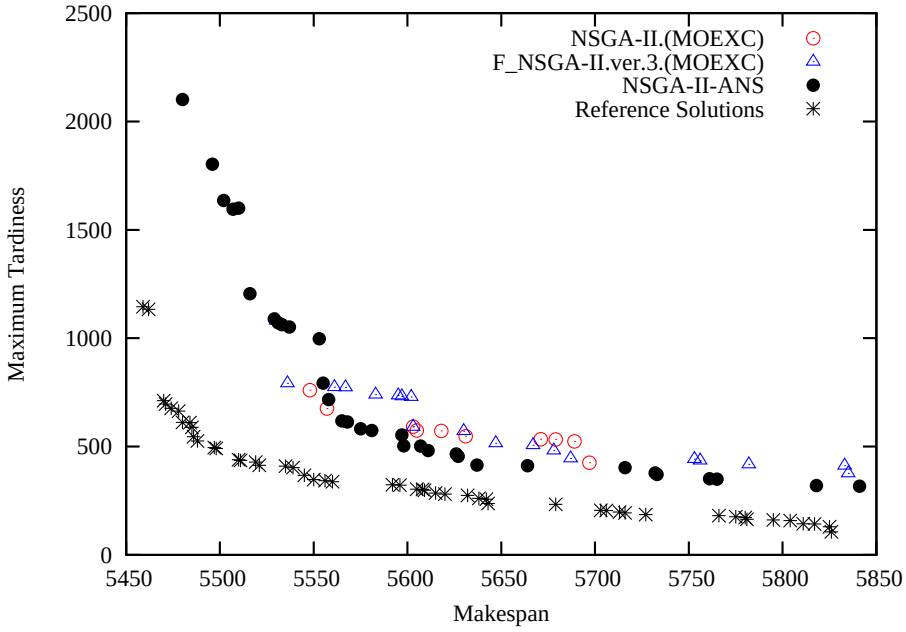


(b) 40×20 problem

Figure 4.7: Attainment of non-dominated solutions for 20×20 and 40×40 cases under 100,000 evaluation times

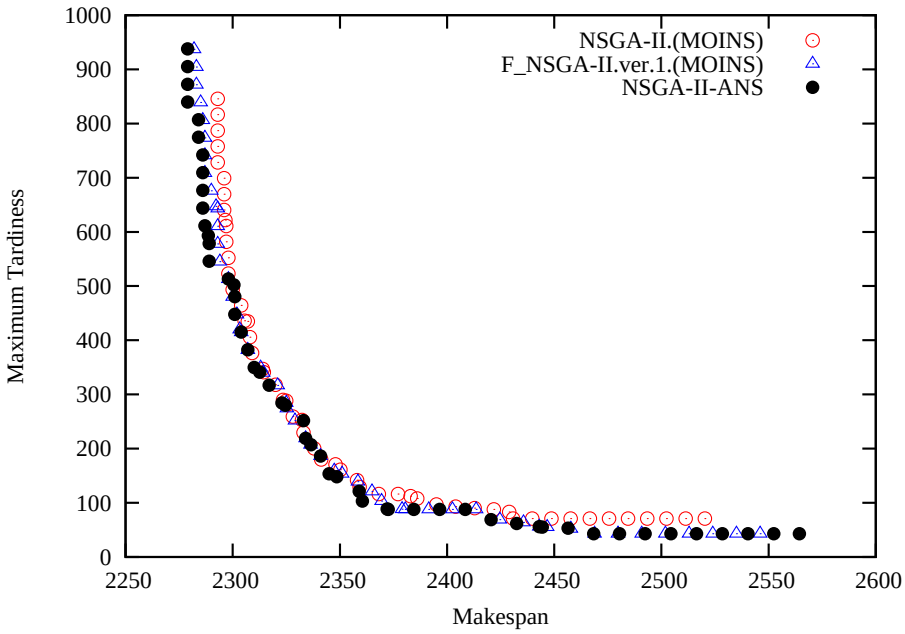


(a) 60×20 problem

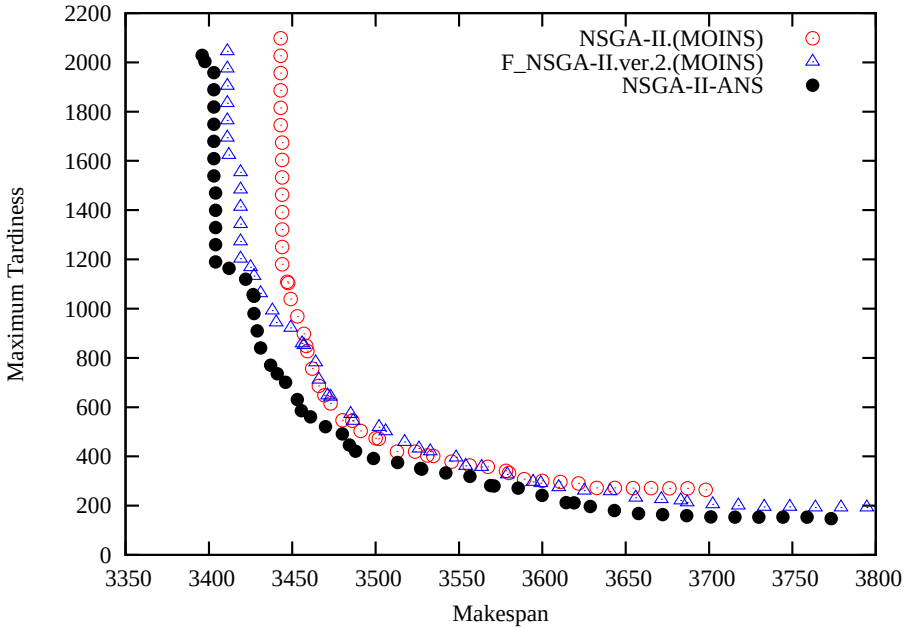


(b) 80×20 problem

Figure 4.8: Attainment of non-dominated solutions for 60×60 and 80×80 cases under 100,000 evaluation times

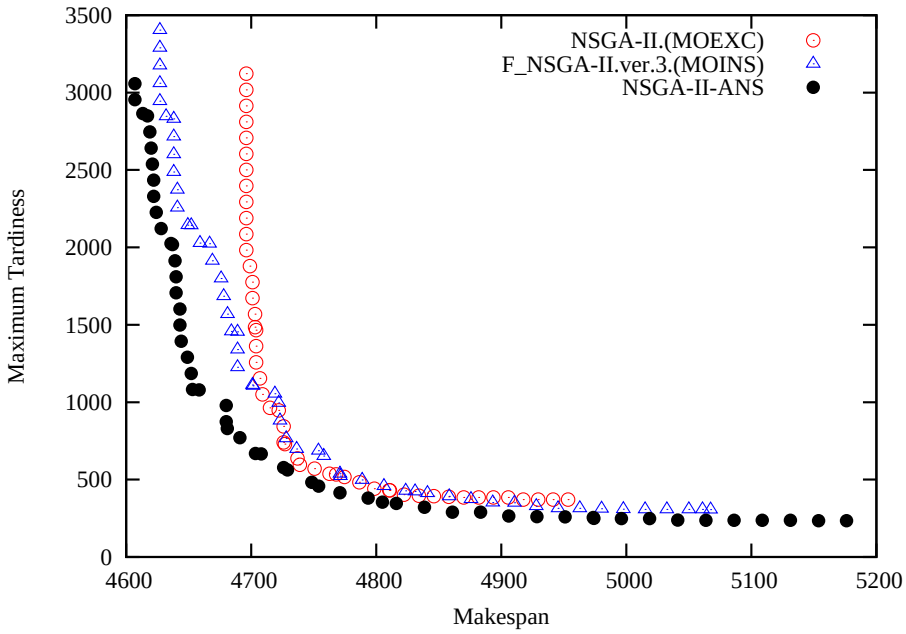


(a) 20×20 problem

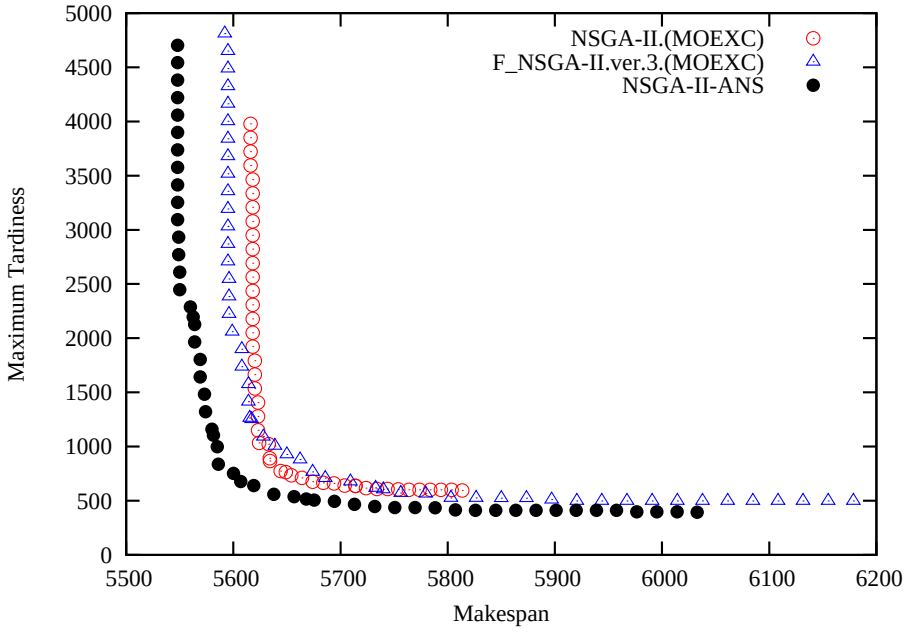


(b) 40×20 problem

Figure 4.9: 50% attainment surfaces of each algorithm on 20×20 and 40×40 cases under 100,000 evaluation times



(a) 60×20 problem



(b) 80×20 problem

Figure 4.10: 50% attainment surfaces of each algorithm on 60×60 and 80×80 cases under 100,000 evaluation times

Chapter 5

PQEA with Q-to-J Method for Multi-Objective Reentrant FSSP

5.1 Introduction

The reentrant FSSP is a more complex problem than permutation FSSP. It usually appears in the semiconductor manufacturing, in which the routes of jobs on the machines are identical as in permutation FSSP, but the jobs must be processed by several machines in multiple times. For example, in semiconductor manufacturing, each wafer re-visits the same machines for multiple processing steps. The wafer traverses flow lines several times to produce different layer on each circuit.

The evaluation of proposed approaches for actual production cases is studied in this chapter. Simulations upon data are conducted from a real-world semiconductor factory, in which the procedure final testing of semiconductor products is main concerned. The semiconductor final testing is the last procedure of semiconductor production, which is responsible for the function detection, final assembly and package work before the semiconductor products are shipped out to customers. This Semiconductor Final Testing Problem (SFTSP) case includes almost all the flow-shop factors as reentry characteristic, serial and batch processing stages, job-clusters and parallel machines. Since the critical equipment needs to be utilized efficiently

at a specific testing stage, the scheduling arrangement for the jobs is then playing an important role in order to reduce both the makespan and penalty cost in total final testing progress.

In order to verify the practical applicability of developed approach in before chapter, PQEA has been tested in this multi-objective reentrant FSSP of a real-world case of SFTSP. Due to the complexity of reentrant FSSP, the Quantum-to-Job (Q-to-J) method that transforms quantum states to real reentrant-jobs identifiers is proposed. The famous multi-objective algorithm NSGA-II is also implemented to solving this problem comparing with PQEA. The investigation of the efficacy of PQEA by contrasting between obtained schedules and factory previous one is taken out.

This chapter is organized as follows. Section 2 firstly introduces the process procedure of SFTSP in this factory and presents the formulations of two objectives in later. Section 3 applies PQEA and NSGA-II to this actual production case. Section 4 summarizes the simulation results and analyzes the performance of PQEA.

5.2 Reentrant FSSP Statement

In semiconductor industry, reentrant operations occur because the wafers may undergo some processes several times during production, in which multiple layers are required for the final product. We call this manufacturing shop as reentrant FSSP, which is defined as that a set of n jobs need to be processed on m machines following the same route and every job must be processed on certain machines more than once. In this section, firstly, the investigation of production model for a SFTSP in real-world semiconductor factory is presented. And, the formulation of objective function is described later.

5.2.1 Instruction of Real-world Reentrant FSSP

The manufacturing processes of a real-world factory for a semiconductor final testing is studied in here. The production model of this factory covers all manufacturing

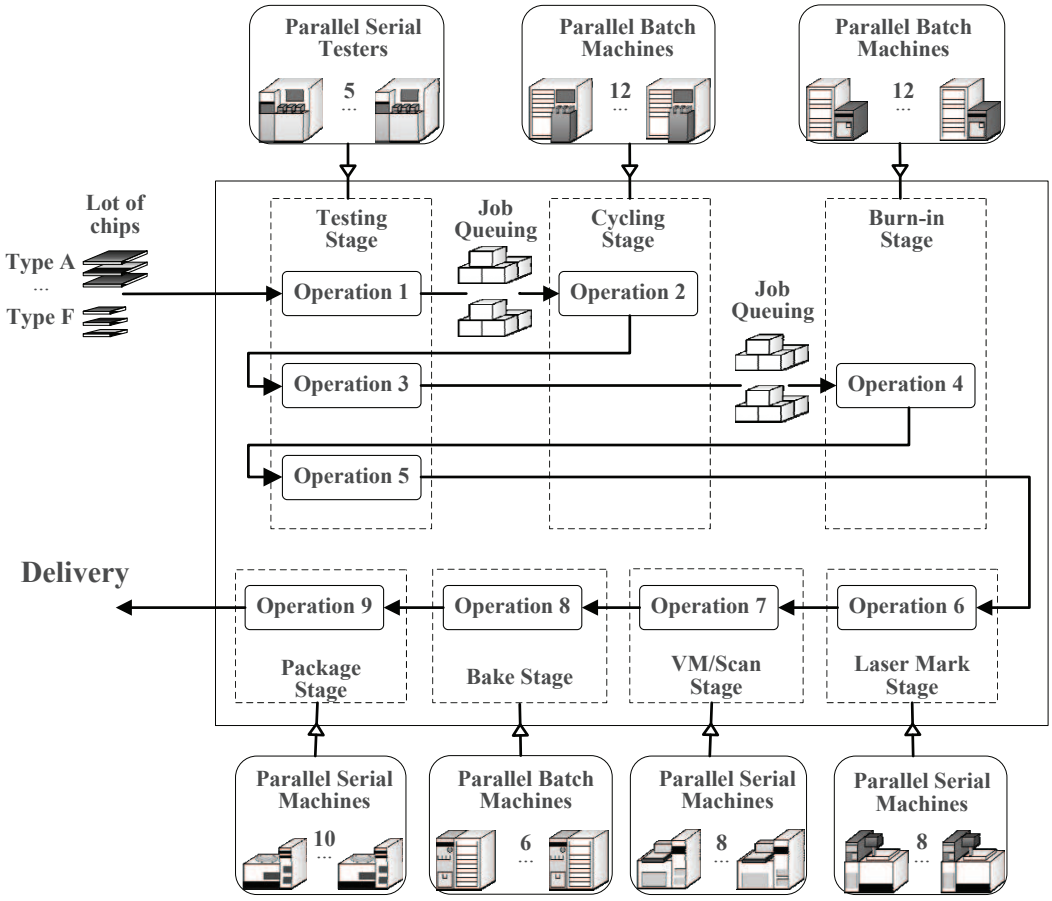


Figure 5.1: Workflow of actual process in reentrant FSSP

process stages in the semiconductor factory, which includes the stages processing jobs serially and the stages processing jobs in a batch manner. The case is taken from a semiconductor final testing factory located in Taiwan. This factory receives orders from other many semiconductor factories required to test their product's quality. For the case investigated, jobs of various product types that came from different semiconductor factories are to be processed on the flow line with reentry at critical resource stage (final testing) many times.

Due to the large increasing demand in semiconductor products, most semiconductor final testing factories allocate the machine capacity to the semiconductor product. Therefore, we concentrate on the scheduling problem of semiconductor products in this factory. For the Semiconductor Final Testing Scheduling Problem

Table 5.1: Processing types and number of machines at each stage

Work Center	Test	Cycling	Burn	Laser	VM/scan	Bake	Package
Process stage	1, 3, 5	2	4	6	7	8	9
Processing type	Serial	Batch	Batch	Serial	Serial	Batch	Serial
Machine Numbers	5	12	12	8	8	6	10

(SFTSP), the manufacturing process includes the following nine stages, (1) FT-1, (2) Cycling, (3) FT-2, (4) Burn-In, (5) FT-3 (6) Laser Mark, (7) VM/Scan, (8) Bake/Package, and (9) Shipping, as displayed in Fig. 5.1. We note that jobs in the following three stages, FT-1, FT-2, and FT-3, share the same group of resources (testers). The FT-1, FT-2, and FT-3 are final testing 1, 2 and 3, respectively.

At FT-1 stage, the tester combined with handler is used to test the basic functions and conductivity of chips, which are tested individually through a specific load board with socket bases by some parallel serious machines. For the FT-2 stage, chips are tested within certain times. FT-2 determines the failure rate and detects potential reliability problems of products. At Burn-In stage, ten thousands of chips are arrayed in the oven, where the Burn-In operation is performed at the same time on the batch machines. For the FT-3 stage, basic testing such as the conductivity testing is executed again. After completing all testing operations, some information is marked on the top surface of the chips, such as the device code, manufacturing date. For the UM/Scan stage, the appearance of each chip is examined manually or by computer scanner. At the last step, before shipping to customer, chips are packaged with plastic tubes.

5.2.2 Case Study of Real-world Reentrant FSSP

For the case investigated, at each stage, a set of identical machines of two types are arranged in parallel. One type of machine is dedicated to the serial stage, which can receive only one lot of chips each time. The other type of machine is dedicated to the batch stage, which can receive lots of chips up to 10 lots at the same time. Table 5.1 shows the process stage, processing type, and the number of parallel machines

Table 5.2: Processing time of operations on each process stage

Operation	Product type					
	A	B	C	D	E	F
1	30	30	30	30	35	30
2	360	360	360	360	720	720
3	30	30	30	30	35	30
4	1440	1440	1440	1440	1440	1440
5	30	30	30	30	35	30
6	30	30	30	30	35	30
7	30	30	30	30	35	30
8	0	600	0	360	0	600
9	30	30	30	30	35	30

at each work center. The reentries occur at stage 1, 3, 5, which are processed by 5 identical machines.

In this case, there are six different product types of jobs are required to go through all these nine operations in sequence, and the processing time may vary, depending on the product type of the jobs. Further, only jobs with the same product type can be processed together in a batch machine. Table 5.2 presents the processing time of six different product types of jobs at 9 stages. The processing time of each operation uses the "Minute"as the time unit. Usually, this factory receives two orders for the six different production types of chips at a week. Each order includes a variety of product type containing the required quantity of lots and the due date. Table 5.3 describes the two orders considered in our case, which includes 53 lots in the first order and 67 lots in the second order. Due to the reentry characteristic in

Table 5.3: product types, required quantity of lots and the due date for the two orders in the case

Order	1						2					
Product type	A	B	C	D	E	F	A	B	C	D	E	F
Number of lots	10	12	15	9	3	4	30	4	25	3	1	4
Due date (in days)	3.5	1.5	2	2.5	1	0.5	4.5	4	3.5	4.5	3.5	3

FT-1, all the jobs have to visit this stage multiple times. And learned from above tables, a batch process takes quite a bit longer time than a serial process. Therefore, a thoughtless arrangement of the scheduling may result in a large waste of waiting time and cost. The target in this case is to find the schedule solution that minimizes the makespan and reduce the penalty cost for the tardy jobs.

5.2.3 Lot-dispatching Approach for Reentrant FSSP

The Burn-In machines are critical resource in this case, because of long processing time at a batch stage, high electricity cost and limitation of machine capacity. At this case, each Burn-In machine can parallel process the same type of chips with a maximum batch size. To adequately utilize the Burn-In machines, a full-load policy is needed to collect lots of the same product type chips until meeting the maximum batch size, and then to choose a possible machine to perform the operation for the entire batch simultaneously.

In this case, the maximum batch size of Burn-In machine is 10. Under the full-load policy, the number of required lots chip in each type is divided by 10. The 10 lots chips are considered as a job. And also the remainder is held as a job. For the example of Table 5.3, the 40 lots chips of 'A' product type are divided into 4 jobs, of which each job has 10 lots chips. Consequently, there are 18 jobs $J_1, J_2, \ldots, J_{18}$ are generated by the lot-dispatching approach for this real-world production case.

5.2.4 Objective Functions of Reentrant FSSP

There are 18 jobs to be processed at 7 stages in order of nine stages. The jobs that visit stage 1 for the first time is called as first-pass jobs; those visit stage 1 for the second time second-pass jobs and those for the third time third-pass jobs. Therefore, we can consider that there are total 54 jobs containing 18 first-pass jobs, 18 second-pass jobs and 18 third-pass jobs in the schedule. A job is regarded as completed only when its third-pass job is finished at stage 9. Fig. 5.2 shows an example of schedule in which each job visits stage 1 three times and its index number occurs exactly three times.

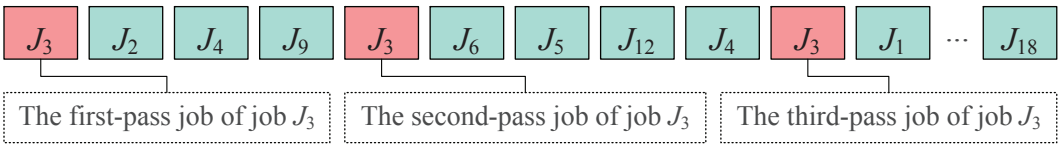


Figure 5.2: Reentrant job sequence

The notations and formulations used in the objective functions of minimizing makespan and penalty cost are described. The index for jobs is denoted by i ($i = 1, 2, \ldots, 18$) and the index for operations is expressed as j ($j = 1, 2, \ldots, 9$). At a batch stage, we suggest arranging a job to be processed on a machine which has least present workload (waiting time). Let F_j ($j = 2, 4, 8$) denote the minimum machine workload at batch stage 2, 4, 8. The workload of the chosen machine is then added by a processing time of this job. Other notations are as follows:

- $J_i^{(1)}$, $J_i^{(2)}$, $J_i^{(3)}$: first-pass job, second-pass job and third-pass job of job J_i.
- $I_i^{(1)}$, $I_i^{(2)}$, $I_i^{(3)}$: the index for jobs $J_i^{(1)}$, jobs $J_i^{(2)}$ and jobs $J_i^{(3)}$ in a schedule.
- π: the schedule includes all jobs $J_i^{(1)}$, jobs $J_i^{(2)}$ and jobs $J_i^{(3)}$.
- π^*: sub-schedule extracted from π which includes only jobs $J_i^{(3)}$.
- $p_j(J_i^{(1)})$, $p_j(J_i^{(2)})$, $p_j(J_i^{(3)})$: processing time of job $J_i^{(1)}$, $J_i^{(2)}$ and $J_i^{(3)}$, respectively, at stage j.
- ϱ: sub-schedule of $J_i^{(3)}$ in increasing order of completion time at stage 8.
- $C_s(\pi(I_i-1))$, $C_s^*(\pi(I_i-1))$, $C_s(\varrho(I_i-1))$: completion time of the predecessor of job J_i, in sequence π, π^* and ϱ, respectively, on the stage s.
- $C_{max}(\pi)$: makespan of schedule π.
- $P_c(\pi)$: penalty cost of all late lots of schedule π.

Our target is to find a schedule π in the solution space Ω, which can be represented as Eq. 5.1. Where, $C_{\max}(\pi) = \max_{i\in[1,18]}\{C_9(J_i^{(3)})\}$ and $P_c(\pi) = \sum \mu\,(C_l - D_l)$, in which penalty cost coefficient is denoted by μ. C_l and D_l are the completion time and due date of the lot l.

$$\underset{\pi\in\Omega}{Minimize}\{C_{\max}(\pi), P_c(\pi)\} \tag{5.1}$$

The calculation of makespan $C_{\max}(\pi)$ is displayed as below equations. The Eq. 5.2, 5.4, 5.6, 5.7, 5.8, 5.10 present that the completion time of the job at a serial

stage is depending on the bigger one of its finishing time on the previous operation and the completion time of its preceding job on this operation. While at a batch stage such as Eq. 5.3, Eq. 5.5, Eq. 5.9, the completion time is decided by the bigger one of the present least machine workload and the job's completion time on the previous operation.

$$C_1(J_i^{(1)}) = C_1(\pi(I_i^{(1)} - 1)) + p_1(J_i^{(1)}) \tag{5.2}$$

$$C_2(J_i^{(1)}) = \max\{F_2, C_1(J_i^{(1)})\} + p_2(J_i^{(1)}) \tag{5.3}$$

$$C_3(J_i^{(2)}) = \max\{C_1(\pi(I_i^{(2)} - 1)), C_2(J_i^{(1)})\} + p_3(J_i^{(2)}) \tag{5.4}$$

$$C_4(J_i^{(2)}) = \max\{F_4, C_3(J_i^{(2)})\} + p_4(J_i^{(2)}) \tag{5.5}$$

$$C_5(J_i^{(3)}) = \max\{C_1(\pi(I_i^{(3)} - 1)), C_4(J_i^{(2)})\} + p_5(J_i^{(3)}) \tag{5.6}$$

$$C_6(J_i^{(3)}) = \max\{C_6(\pi^*(I_i^{(3)} - 1)), C_5(J_i^{(3)})\} + p_6(J_i^{(3)}) \tag{5.7}$$

$$C_7(J_i^{(3)}) = \max\{C_7(\pi^*(I_i^{(3)} - 1)), C_6(J_i^{(3)})\} + p_7(J_i^{(3)}) \tag{5.8}$$

$$C_8(J_i^{(3)}) = \max\{F_8, C_7(J_i^{(3)})\} + p_8(J_i^{(3)}) \tag{5.9}$$

$$C_9(J_i^{(3)}) = \max\{C_9(\sigma(I_i^{(3)} - 1)), C_8(J_i^{(3)})\} + p_9(J_i^{(3)}) = C_{max}(\pi) \tag{5.10}$$

The other objective function for penalty cost $P_c(\pi)$ likes as $P_c(\pi) = \sum \mu\,(C_l - D_l)$, where the coefficient μ is considered as 0.2, C_l and D_l are the completion time and due date of the lot l. The μ is set as experience value that are learned from history trade information.

5.3 PQEA with Q-to-J Method for Reentrant FSSP

Due to the probabilistic representation of q-bit individual in PQEA, it is needed a Quantum-to-Job (Q-to-J) method that translates the quantum states to real job

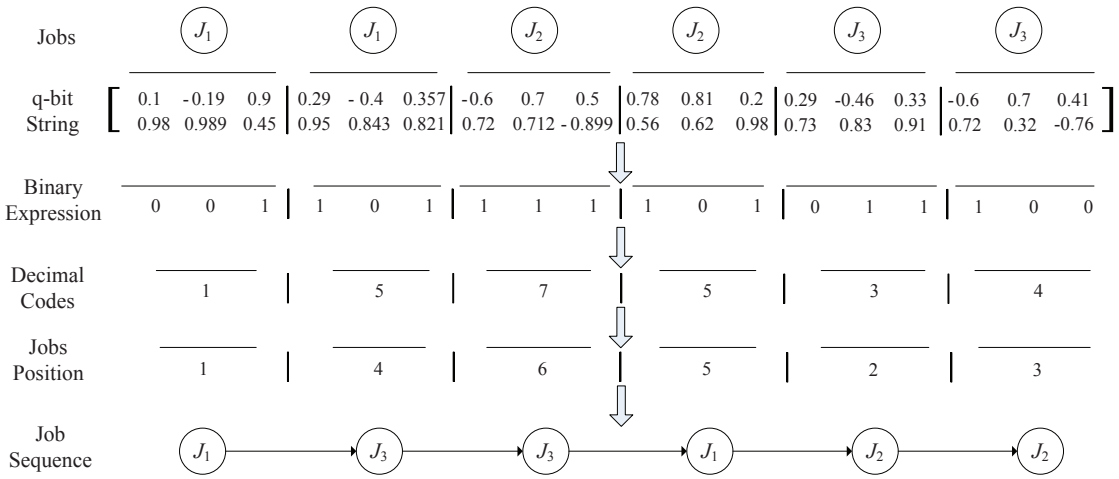


Figure 5.3: Transformation of quantum States to reentrant job's identifiers

identities. The PQEA combines Q-to-J method to solve this multi-objective reentrant FSSP, which is firstly introduced in this section. Additionally, to illustrate the effectiveness of PQEA, the NSGA-II is also implemented to address this problem comparing with PQEA.

5.3.1 Q-to-J Method Instruction

Although q-bit can represent a linear superposition of solutions, it cannot be used directly for solving reentrant FSSP. Because the reentrant job code we want is a permutation of jobs with reentry, but q-bit representation is a pair of complex numbers $[\gamma, \eta]$. So a converting mechanism (Q-to-J method) should be put forward specially aiming at reentrant FSSP for evaluation, which is an innovation in this chapter. It is described as following.

A q-bit individual with length of $\lfloor \log_2^n + 1 \rfloor * n * m$ probabilistically represents the positions of job $J_1, J_1, J_1, \ldots, J_n, J_n, J_n$ in the solution. Here, n and m are number of jobs and reentry times (in this case, $n = 18$, $m = 3$). Every $\lfloor \log_2^n + 1 \rfloor$ q-bits are used to determine a job or reentrant job's position which is from 1 to $n * m$. There are 3 steps in procedure of Q-to-J method.

- **Step 1**. Apply the observing procedure to convert q-bit individual into binary

string. Every $\lfloor \log_2^n + 1 \rfloor$ bits of binary string is transferred into a decimal number, and then a decimal string with length n is obtained.

- **Step 2**. Order this decimal string from small to large. If values of two numbers are different, let smaller number has priority to decide the corresponding job's position in ahead of sequence; otherwise, the first number firstly determines the corresponding job's position in ahead.

- **Step 3**. Generate reentrant flow shop codes by arranging jobs $J_1, J_1, J_1, \ldots, J_n, J_n, J_n$ to their appointed positions.

For example, consider a three-job and twice-reentries problem. Suppose a binary string is [001 | 101 | 111 | 101 | 011 | 100], which is observed from a q-bit representation, then the decimal string is [1 5 7 5 3 4]. From small to large, the positions of jobs J_1, J_1, J_2, J_2, J_3, and J_3 are [1 4 6 5 2 3]. Thus, the reentrant flow shop codes can be got as $J_1, J_3, J_3, J_1, J_2, J_2$. Example can be seen from Fig. 5.3.

5.3.2 PQEA with Q-to-J Method

To solve multi-objective reentrant FSSP, the main procedure of PQEA is similar with the description of PQEA for multi-objective permutation FSSP in chapter 3. The Q-to-J Method is used to evaluate a binary solutions when it is generated by observing the quantum state of q-bit individual.

The parameter of PQEA for multi-objective reentrant FSSP are set as following:

- Number of initialized weight vectors : N=100;
- Number of neighbor sub-problem in one group: T=10;
- Rotation angle value for updating quantum state: $\Delta\theta = 0.01\pi$;

Based on the Q-to-J method, It does not require any other repair procedure to deal with the infeasible solutions.

5.3.3 Implement of NSGA-II for Actual Reentrant FSSP Case

The NSGA-II algorithm and its detailed implementation on reentrant FSSP can be found in paper [63]. An individual is represented by sequence of reentrant jobs. This algorithm has been demonstrated as one of the most efficient algorithms for multi-objective optimization on a number of scheduling problems. A brief description of NSGA-II for reentrant FSSP is as follows: NSGA-II uses non-dominated sorting for fitness assignments. Individuals that are not dominated by any other individuals are assigned in front rank 1. The others only dominated by individuals in front rank 1 are assigned into front rank 2, and so on. Selection is made using tournament between two individuals. The individual with the lowest front number is selected if the two individuals are from different fronts. The individual with the highest crowding distance is selected if they are from the same front. i.e., a higher fitness is assigned to individuals located on a sparsely populated part of the front. There are p parents and in every iteration p new individuals (offspring) are generated. Both parents and offspring compete with each other for inclusion in the next iteration. list of job numbers is represented as an individual for encoding the problem. The mutation and crossover is implemented by one point insertion and two point crossover. The two point crossover is an ordinary one that calls for two points to be selected on the parent organism strings. Segment genes between the two points are swapped between the parent organisms, rendering two child organisms. Several crossover and mutation operations are measured in genetic algorithms for reentrant FSSP problems. The better results are reported by the combination of one point insertion and two point crossover, and parameters are set as following.

- Population Size: $popSize$=90;
- Mutation Probability: P_M=0.3;
- Crossover Probability: P_C=0.8;

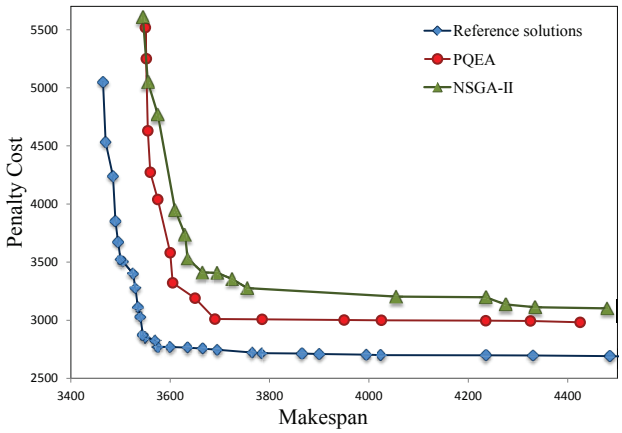
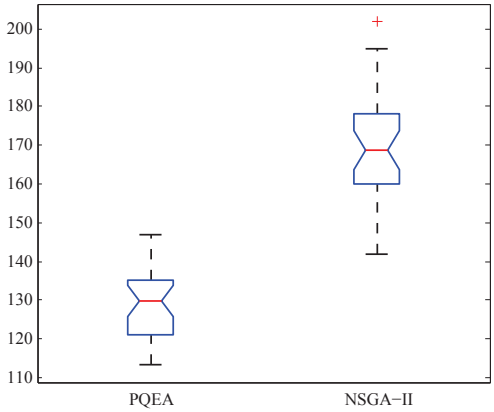


Figure 5.4: Obtained non-dominated solutions of PQEA and NSGA-II for multi-objective reentrant FSSP

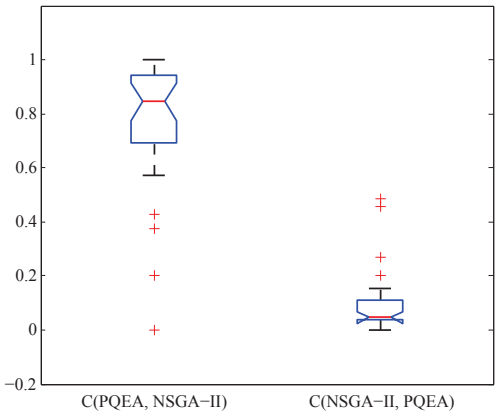
5.4 Experimental Results and Discussion

5.4.1 Comparison of PQEA and NSGA-II

Both algorithms are independently executed 30 times. The C-measure and D-measure are used to test the performance of PQEA and NSGA-II on this actual production case. D-measure needs a set of Pareto-optimal solution or a near Pareto-optimal solution for evaluating the obtained solutions by PQEA and NSGA-II. The average distance of reference solution (near Pareto-optimal solution) set to obtained solutions is the criterion of D-measure to compare the performance of each algorithm. The reference solution set of this problem is found by using the PQEA and NSGA-II. Each algorithm was applied to this actual problem with much longer computation time and larger memory storage than the other computational experiments in this chapter. More specifically, we used the many combination of parameter specification in all two algorithms for finding the reference solution, in which the terminal condition is cost computation time more than 3 days. We chose only non-dominated solutions as reference solutions from 20 solution sets obtained by ten runs of the three algorithms for this problem. The obtained reference are presented in Fig. 5.4, where, we can observe the existence of a clear tradeoff between the two objectives.



(a) D-measure of PQEA and NSGA-II



(b) C-measure of PQEA and NSGA-II

Figure 5.5: Comparison of PQEA and NSGA-II on multi-objective Reentrant FSSP

Three algorithms are executed 30 times severally. Fig. 5.4 shows the non-dominated solutions that are generated by each algorithm, in which the most obtained non-dominated solutions in 30 executed times. We can roughly gain from this figure that the performance of NSGA-II seems having good behavior on this problem. The non-dominated solutions of PQEA have both smaller makespan and penalty cost.

Fig. 5.5 presents the boxplots that are summarizing 30 results of D-measure and C-measure for the two algorithms. Such as Fig. 5.5(a), boxplot of PQEA states the results of D-measure by 30 times runs of PQEA algorithm, of which the median value is about 130. We can learn from this figure that PQEA generates the low value of

Table 5.4: Dominated solutions generated by PQEA on multi-objective reentrant FSSP

	PQEA								*Imp* %
Makespan	3765	3670	3650	3585	3600	3560	3525	3545	8.1%
Penalty Cost	3008	3010	3190	3322	3581	4039	4273	4590	24.4%

Table 5.5: Dominated solutions generated by NSGA-II on multi-objective reentrant FSSP

	NSGA-II							*Imp* %
Makespan	3755	3725	3695	3665	3635	3630	3610	4.8%
Penalty Cost	3276	3354	3408	3411	3532	3736	3946	27.2%

D-measure comparing with NSGA-II. It illustrates that the obtained non-dominated solutions of PQEA are close to the reference solutions, with seldom missing any part of the reference solutions. Besides, the shape of boxplots of PQEA appears shorter than NSGA-II. It means that the standard deviation variances of D-measure in this problem of 30 runs are smaller than NSGA-II. PQEA has a stabile performance. Furthermore, the coverage rate of PQEA is bigger than NSGA-II, which is presented at Fig. 5.5(b). It is indicted that most solutions of PQEA dominates solutions of NSGA-II, whereas only little solutions are dominated by NSGA-II.

5.4.2 Improvement of PQEA and NSGA-II in Real-World Case

The previous scheduling system using in this factory applies the Earliest Due Date (EDD) policy to generate a schedule. The jobs which has little time to deliver to customs will be processed in a high priority. By the EDD policy, only one solution is obtained, because the penalty cost for tardiness job is only considered. However, finally, PQEA and NSGA-II obtains a lot of non-dominated solutions which are optimal on two objectives: minimize makespan and penalty cost. To compare with

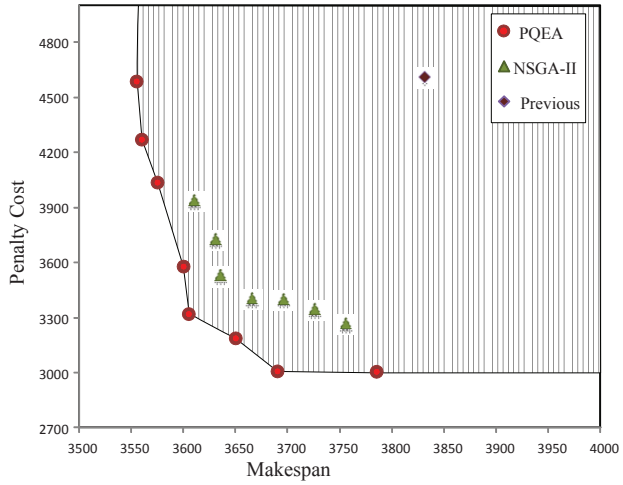


Figure 5.6: Dominated solutions obtained by PQEA and NSGA-II comparing with factory previously used one

PQEA, NSGA-II and EDD, the solutions that are better than previous on both objective generated by PQEA and NSGA-II are collected in Table 5.4 and Table 5.5. These comparing solutions are also shown in Fig. 5.6.

$$Imp\ _{(C_{max})}\% = \left(\frac{C_{\max}(\pi^*) - \frac{1}{n}\sum_{i=1}^{n} C_{\max}(\pi_i)}{C_{\max}(\pi^*)} \right) \times 100 \tag{5.11}$$

$$Imp\ _{(P_{cost})}\% = \left(\frac{P_{cost}(\pi^*) - \frac{1}{n}\sum_{i=1}^{n} P_{cost}(\pi_i)}{P_{cost}(\pi^*)} \right) \times 100 \tag{5.12}$$

Based on the previous scheduling system, solution π^* with $C_{max}(\pi^*) = 3860$ of makespan and $P_{cost}(\pi^*) = 4620$ penalty cots is caused. To compare with obtained non-dominated solutions of PQEA and NSGA-II, based on Eq. 5.11 and Eq. 5.12, the makespan is averagely improved about 8.1%, 4.8% and penalty cost is reduced near 24.4% and 27.2%, respectively. Here, π_i $(i = 1, 2, ..., n)$ is non-dominated solution obtained by PQEA or NSGA-II.

5.5 Summary

In this chapter, we present an actual SFTSP case includes almost all the flow-shop factors as reentry characteristic, serial and batch processing stages, job-clusters and parallel machines. The target of the problem is to find optimal solutions with both the makespan and penalty cost optimized. To apply PQEA on the reentrant FSSP, a transformation method Q-to-J that translates the quantum states to real job identities is proposed. The computational simulation results reveal that PQEA with Q-to-J has a good performance for finding better solutions in this problem. By using the proposed scheduling approach, there are 8 schedules that obtained by PQEA are dominating the previous one. The makespan and penalty cost for tardiness are obviously be reduced. Moreover, NSGA-II is also implemented to solve this problem comparing with our PQEA. The experimental results shows that PQEA with Q-to-J method is significantly better than NSGA-II both in dispersion of non-dominated solutions and high convergence rate.

Chapter 6

Conclusion

6.1 Conclusion and Summary

This thesis has deal with permutation FSSP and reentrant FSSP with multiple objectives including minimization of makespan, maximum tardiness and maximum penalty cost of tardy jobs simultaneously. The two problems are NP-hard and even with small size problems are difficult to solve. To obtain some good solutions, the quantum-inspired optimization approaches are proposed by merging the concept and principles of quantum computing to evolutionary algorithm. The comparative analysis and experiment have demonstrated its effectiveness for solving the two problems.

The makespan is an important criterion to measure the utilization of production recourses and strongly related with production cost. In the literature, there are many heuristic methods have been proposed to address the permutation FSSP with makespan criterion. However, based on the traditional methods it is hard to deliver an effective solution in a limited computation time, especially for large size problems. In order to effectively deliver a solution, an innovated evolution algorithm, called MMQEA, is proposed based on the concept of quantum computing. MMQEA is characterized by the representation of individual, evaluation function and population. However, instead of numeric or symbolic representation, MMQEA

uses q-bit string to probabilistically represent the job sequences. A q-bit individual of MMQEA contains a pair of q-bit strings. There are two update modes that are proposed to update the quantum state of q-bit individual, in which each q-bit string relies on its corresponding update mode to renovate quantum state. Based on the two update modes, each q-bit string of the q-bit individual provides its evolutionary information to other one and also receives other one's evolutionary information during the evolution, which maintains the population more diverse to exploit global optimal solution. For permutation FSSP with makespan criterion, MMQEA outperforms traditional approaches, which has improved the makespan near 3%, 6% and 7% on 20×10 (20 jobs 10 machines), 50×10 and 50×20 problems. Furthermore, to verify the applicability of MMQEA, knapsack problems are used to compare MMQEA with current famous approaches, the performance of MMQEA still are attractive.

The highly competitive industries, such as food, beverage, semiconductor industry, also needs on-time delivery to response the stress of competition on the markets. These factories desire a multi-objective scheduling system that simultaneously optimizes makespan and maximum tardiness to improve the competitiveness. Utilizing the achievement of MMQEA, PQEA framework is proposed to solve this multi-objective problem. To obtain an even distribution of solutions, firstly, PQEA decomposes uniformly this multi-objective problem into a number of scalar optimization sub-problems by decomposition method. All the sub-problems are classified into several groups according to their similarities. One q-bit individual is used to address the sub-problems of a group. Since q-bit individual is a probabilistic representation, it can share evolutionary information of the neighboring sub-problems in same group. Without loss any solutions of Pareto Front, a population of q-bit individuals are parallel evolved. Comparing with current state-of-the-art algorithms, PQEA outperforms all of them, with 11.6%, 9.6%, 18.3% and 10.3% improvement on dispersion performance for 20×20, 40×20, 60×20 and 80×20 problems.

Although PQEA has generated better solution on multi-objective FSSP, there are still some spaces to be promoted. PQEA combines ANS to further improve the convergence and dispersion performances of PQEA on multi-objective permutation

FSSP. ANS is developed by utilizing the properties of active blocks for permutation FSSP. Firstly, two neighborhood structures Multi-Objective Insertion (MOINS) and Multi-Objective Exchange (MOEXC) are designed in order to improve efficiency of perturbation. Any perturbation based on MOINS and MOEXC take positive effect on all objectives simultaneously, which can obviously improve convergence. ANS controls the search direction by systematically changing the neighborhood of MOINS and MOEXC to exploit more dominated solutions. The other advantage of ANS is no need to set parameters. Under the same computational times as terminal condition, numerous comparisons show that ANS has improved dispersion performance near 23.5%, 32.2%, 39.6%, and 46.4% on 20×10, 40×20, 60×20 and 80×20 problems.

Finally, the application of proposed algorithms in a real-world semiconductor factory is studied. For the real-world case investigated, jobs of various product types are to be processed on the flow line with reentry at critical resource stage (final testing) many times, which is a reentrant FSSP. This factory desires a scheduling system that to reduce makespan and penalty costs for tardiness job simultaneously. To apply PQEA on the reentrant FSSP, a transformation method Q-to-J that translates the quantum states to real job identities is proposed. By using the proposed scheduling approach, there are 8 schedules that obtained by PQEA are dominating the previous one. The makespan and penalty cost for tardiness are averagely reduced by 8% and 27%, respectively.

6.2 Future Work Prospects

From the previous investigation on quantum-inspired optimization approaches to permutation FSSP and reentrant FSSP, although these approaches achieve some attractive solutions, there is still much space to improve. In the present work, the procedure of translation quantum states to real job identifies is a factor to increase the complexity of proposed methods, because one job needs several quantum bits to represent its position in job sequence. In the future, it is necessary to develop a way that uses little q-bit for job's representation. The initialization of quantum state is

other point to improve more. The quantum-inspired optimization approaches use a random strategy to start the searching procedure. In other words, all the q-bit are initialized with same quantum states. Merging the knowledge-based dispatching rules of FSSP to the initialization steps of proposed approaches, is a better way to find more dominated solutions effectively.

On the other hand, recently, the factory automation has been advanced that each machine has self diagnostic sensors as well as a network interface and is monitored online even remotely. The flexibility and the accuracy of controlling machines for reconfiguration and rescheduling are also improved. Therefore, the need for extending the proposed to generate an efficient schedule with consideration of more complicated constraints, more flexible objective functions and more dynamic features.

Bibliography

[1] M. Boyer, G. Brassard, A. Tapp, *et al.*, "Tight bounds on quantum searching," *American Journal of Physics*, vol. 2, pp. 11–31, 1996.

[2] A. Steane, "Quantum computing," *Reports on Progress in Physics*, vol. 61, pp. 117–129, 1998.

[3] C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, "Strengths and weaknesses of quantum computing," *Arxiv preprint quant-ph/9701001*, vol. 6, pp. 11–31, 1997.

[4] L. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. ACM, 1996, pp. 212–219.

[5] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Foundations of Computer Science, 1994 Proceedings, 35th Annual Symposium on*. IEEE, 1994, pp. 124–134.

[6] M. Nielsen, I. Chuang, and L. Grover, "Quantum computation and quantum information," *American Journal of Physics*, vol. 70, pp. 558–598, 2002.

[7] K. Han and J. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 6, pp. 580–593, 2002.

[8] L. Wang, F. Tang, and H. Wu, "Hybrid genetic algorithm based on quantum computing for numerical optimization and parameter estimation," *Applied Mathematics and Computation*, vol. 171, no. 2, pp. 1141–1156, 2005.

[9] M. Platel, S. Schliebs, and N. Kasabov, "Quantum-inspired evolutionary algorithm: a multimodel eda," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 6, pp. 1218–1232, 2009.

[10] J. Jang, K. Han, and J. Kim, "Quantum-inspired evolutionary algorithm-based face verification," in *Genetic and Evolutionary ComputationGECCO 2003*. Springer, 2003, pp. 214–214.

[11] A. da Cruz, M. Vellasco, and M. Pacheco, "Quantum-inspired evolutionary algorithm for numerical optimization," *Hybrid Evolutionary Algorithms*, pp. 19–37, 2007.

[12] A. Narayanan, "Quantum computing for beginners," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 3. IEEE, 1999.

[13] L. Spector, H. Barnum, H. Bernstein, and N. Swamy, "Finding a better-than-classical quantum and/or algorithm using genetic programming," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 3. IEEE, 1999.

[14] K. Bergmann, H. Theuer, and B. Shore, "Coherent population transfer among quantum states of atoms and molecules," *Reviews of Modern Physics*, vol. 70, no. 3, pp. 1003–1031, 1998.

[15] F. Schmidt-Kaler, H. Häffner, M. Riebe, S. Gulde, G. Lancaster, T. Deuschle, C. Becher, C. Roos, J. Eschner, and R. Blatt, "Realization of the cirac–zoller controlled-not quantum gate for loose quantum computing," *Nature*, vol. 422, no. 6930, pp. 408–411, 2003.

[16] K. Han and J. Kim, "Quantum-inspired evolutionary algorithms with a new termination criterion, hε gate, and two-phase scheme," *Evolutionary Computation, IEEE Transactions on*, vol. 8, no. 2, pp. 156–169, 2004.

[17] Y. Kim, J. Kim, and K. Han, "Quantum-inspired multiobjective evolutionary algorithm for multiobjective 0/1 knapsack problems," in *Evolutionary Computation, 2006. IEEE Congress on.* IEEE, 2006, pp. 2601–2606.

[18] M. Garey, D. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, pp. 117–129, 1976.

[19] K. Deb, *Multi-objective optimization using evolutionary algorithms.* Wiley, 2001, vol. 16.

[20] D. Corne, N. Jerram, J. Knowles, M. Oates, *et al.*, "Pesa-ii: Region-based selection in evolutionary multiobjective optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference.* Citeseer, 2001.

[21] A. Nagar, J. Haddock, and S. Heragu, "Multiple and bicriteria scheduling: A literature survey," *European journal of operational research*, vol. 81, no. 1, pp. 88–104, 1995.

[22] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 6, pp. 712–731, 2007.

[23] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, 2002.

[24] H. Ishibuchi and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 28, no. 3, pp. 392–403, 1998.

[25] M. Wiecek, W. Chen, and J. Zhang, "Piecewise quadratic approximation of the non-dominated set for bi-criteria programs," *Journal of Multi-Criteria Decision Analysis*, vol. 10, no. 1, pp. 35–47, 2001.

[26] A. Jaszkiewicz, "Genetic local search for multi-objective combinatorial optimization," *European journal of operational research*, vol. 137, no. 1, pp. 50–71, 2002.

[27] G. Celli, E. Ghiani, S. Mocci, and F. Pilo, "A multiobjective evolutionary algorithm for the sizing and siting of distributed generation," *Power Systems, IEEE Transactions on*, vol. 20, no. 2, pp. 750–757, 2005.

[28] H. Ishibuchi and H. Tanaka, "Multiobjective programming in optimization of the interval objective function," *European Journal of Operational Research*, vol. 48, no. 2, pp. 219–225, 1990.

[29] A. Jaszkiewicz, "Multiple objective genetic local search algorithm," *Lecture notes in economics and mathematical systems*, pp. 231–240, 2001.

[30] K. Han and J. Kim, "On setting the parameters of quantum-inspired evolutionary algorithm for practical application," in *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, vol. 1. IEEE, 2003, pp. 178–194.

[31] X. Wei and S. Fujimura, "Multiupdate mode quantum evolutionary algorithm and its applications to combination and permutation problems," *IEEJ Transactions on Electrical and Electronic Engineering*, pp. 102–110, 2012.

[32] W. Ponweiser and M. Vincze, "The multiple multi objective problem–definition, solution and evaluation," in *Evolutionary Multi-Criterion Optimization*. Springer, 2007, pp. 877–892.

[33] B. Liu, L. Wang, and Y. Jin, "An effective pso-based memetic algorithm for flow shop scheduling," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 37, no. 1, pp. 18–27, 2007.

[34] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 2, pp. 284–302, 2009.

[35] X. Wei and S. Fujimura, "Multiupdate mode quantum evolutionary algorithm and its applications to combination and permutation problems," *IEEJ Transactions on Electrical and Electronic Engineering*, pp. 534–545, 2012.

[36] J. Knowles and D. Corne, "Approximating the nondominated front using the pareto archived evolution strategy," *Evolutionary computation*, vol. 8, no. 2, pp. 149–172, 2000.

[37] J. Arroyo and V. Armentano, "Genetic local search for multi-objective flowshop scheduling problems," *European Journal of Operational Research*, vol. 167, no. 3, pp. 717–738, 2005.

[38] P. Chang, S. Chen, and C. Liu, "Sub-population genetic algorithm with mining gene structures for multiobjective flowshop scheduling problems," *Expert systems with applications*, vol. 33, no. 3, pp. 762–771, 2007.

[39] O. Engin and A. Döyen, "A new approach to solve hybrid flow shop scheduling problems by artificial immune system," *Future Generation Computer Systems*, vol. 20, no. 6, pp. 1083–1095, 2004.

[40] Z. Lian, X. Gu, and B. Jiao, "A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan," *Chaos, Solitons & Fractals*, vol. 35, no. 5, pp. 851–861, 2008.

[41] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *Evolutionary Computation, IEEE Transactions on*, vol. 3, no. 4, pp. 257–271, 1999.

[42] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary computation*, vol. 8, no. 2, pp. 173–195, 2000.

[43] A. Jaszkiewicz, "On the performance of multiple-objective genetic local search on the 0/1 knapsack problem-a comparative experiment," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 4, pp. 402–412, 2002.

[44] J. Knowles and D. Corne, "M-paes: A memetic algorithm for multiobjective optimization," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 1. IEEE, 2000, pp. 325–332.

[45] M. Alves and M. Almeida, "Motga: A multiobjective tchebycheff based genetic algorithm for the multidimensional knapsack problem," *Computers & operations research*, vol. 34, no. 11, pp. 3458–3470, 2007.

[46] T. Erlebach, H. Kellerer, and U. Pferschy, "Approximating multiobjective knapsack problems," *Management Science*, pp. 1603–1612, 2002.

[47] A. Jaszkiewicz, "On the computational efficiency of multiple objective metaheuristics. the knapsack problem case study," *European Journal of Operational Research*, vol. 158, no. 2, pp. 418–433, 2004.

[48] A. Martínez-Estudillo, C. Hervás-Martínez, F. Martínez-Estudillo, and N. García-Pedrajas, "Hybridization of evolutionary algorithms and local search by means of a clustering method," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 36, no. 3, pp. 534–545, 2005.

[49] B. Ekşioğlu, S. Ekşioğlu, and P. Jain, "A tabu search algorithm for the flowshop scheduling problem with changing neighborhoods," *Computers and Industrial Engineering*, vol. 54, no. 1, pp. 1–11, 2008.

[50] M. Tasgetiren, M. Sevkli, Y. Liang, and G. Gencyilmaz, "Particle swarm optimization algorithm for permutation flowshop sequencing problem," *Ant Colony Optimization and Swarm Intelligence*, pp. 366–385, 2004.

[51] M. Ben-Daya and M. Al-Fawzan, "A tabu search approach for the flow shop scheduling problem," *European Journal of Operational Research*, vol. 109, no. 1, pp. 88–95, 1998.

[52] R. Linn and W. Zhang, "Hybrid flow shop scheduling: a survey," *Computers & Industrial Engineering*, vol. 37, no. 1-2, pp. 57–61, 1999.

[53] J. Grabowski, "On two-machine scheduling with release and due dates to minimize maximum lateness," *Opsearch*, vol. 17, pp. 133–154, 1999.

[54] J. Grabowski, E. Skubalska, and C. Smutnicki, "On flow shop scheduling with release and due dates to minimize maximum lateness," *The Journal of the Operational Research Society*, vol. 34, no. 7, pp. 615–620, 1993.

[55] C. Koulamas, "A new constructive heuristic for the flowshop scheduling problem," *European Journal of Operational Research*, vol. 105, no. 1, pp. 66–71, 1998.

[56] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 2, pp. 204–223, 2003.

[57] J. Grabowski and M. Wodecki, "A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion," *Computers & Operations Research*, vol. 31, no. 11, pp. 1891–1909, 2004.

[58] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 2033–2049, 2007.

[59] E. Taillard, "Some efficient heuristic methods for the flow shop sequencing problem," *European Journal of Operational Research*, vol. 47, no. 1, pp. 65–74, 1990.

[60] L. Wang, L. Zhang, and D. Zheng, "An effective hybrid genetic algorithm for flow shop scheduling with limited buffers," *Computers & Operations Research*, vol. 33, no. 10, pp. 2960–2971, 2006.

[61] C. Wang, X. Li, and Q. Wang, "Accelerated tabu search for no-wait flowshop scheduling problem with maximum lateness criterion," *European Journal of Operational Research*, vol. 206, no. 1, pp. 64–72, 2010.

[62] J. Knowles, "A summary-attainment-surface plotting method for visualizing the performance of stochastic multiobjective optimizers," *Evolutionary computation*, vol. 2, no. 3, pp. 14–17, 2005.

[63] F. Dugardin, F. Yalaoui, and L. Amodeo, "New multi-objective method to solve reentrant hybrid flow shop scheduling problem," *European Journal of Operational Research*, vol. 203, no. 1, pp. 22–31, 2010.

Publications

Journals (with review)

[1] Wei Weng, **Xin Wei**, Shigeru Fujimura. "Dynamic routing strategies for JIT production in hybrid flow shops", European Journal of Computers & Op -erations Research. (Accepted)

◯ [2] **Xin Wei**, Shigeru Fujimura. "Multi-update Mode Quantum Evolutionary Algorithm and Its Applications to Combination and Permutation Problems", IEEJ Transactions on Electrical and Electronic Engineering, Vol.7, No.2, pp. 166-173, 2012.

◯ [3] **Xin Wei**, Wenqiang Zhang, Wei Weng, Shigeru Fujimura. "Multi- -objective Local Search Combined with NSGA-II for Bi-criteria Permutation Flow Shop Scheduling Problem", IEEJ Transactions on Electronics, Informa -tion and Systems, Vol.132, No.1, pp. 32-41, 2012.

International Conference (with review)

[1] Wei Weng, **Xin Wei**, Shigeru Fujimura. "Implementation of Pull System along a Production Chain", The 2012 International Conference on Managem -ent and Service Science (MASS 2012), Aug. 2012. (Accepted)

◯ [2] **Xin Wei**, Shigeru Fujimura. "Parallel Quantum Evolutionary Algorithms with Client-Server Model for Multi-Objective Optimization on Discrete Pro- -blems", The 2012 IEEE World Congress on Computational Intelligence (IE-

-EE WCCI 2012), Jun. 2012. (Accepted)

◯ [3] **Xin Wei**, Cheng Xia Sun, Shigeru Fujimura. "Hybrid Bi-Criterion Evoluti--onary Algorithm for Permutation Flow Shop Scheduling Problem", Proceedings of the 21st International Conference on Production Research (ICPR21), Aug. 2011.

[4] Wenqing Zhang, **Xin Wei**, Shigeru Fujimura. "Fast and effective evolution -ary algorithm for multi-objective process planning and scheduling problem", Proceedings of the 21st International Conference on Production Research (ICPR, -21) Aug. 2011.

◯ [5] **Xin Wei**, Shigeru Fujimura. "Multi-Objective Quantum Evolutionary Algorithm for Discrete Multi-Objective Combinational Problem", The 15th International Conference on Technologies and Applications of Artificial Intelligence (TAAI2010), Vol. 32, pp. 39-46, Nov. 2010.

[6] Wei Wei, **Xin Wei**, C.H.Ding, Shigeru Fujimura. "A Hybrid Local Search Approach in Solving the Mirrored Traveling Tournament Problem", The IEEE 17th International Conference of Industrial Engineering and Engineering Manag--ement (IE&EM), Vol. 978, pp. 620-624, Oct. 2010.

[7] C. H. Ding, **Xin Wei**, Shigeru Fujimura. "Time and Cost Trade-off Problem under Uncertainty Incorporating Multi-Objective Genetic Algorithm and Fuzzy Sets Theory", The IEEE 17th International Conference of Industrial Engineering and Engineering Management (IE&EM), Vol. 8, pp. 290-294, Oct. 2010.

◯ [8] **Xin Wei**, Shigeru Fujimura. "Multi-update Mode Quantum Evolutionary Algorithm For A Combinatorial Problem", The 2nd International Conference on Computer and Automation Engineering (ICCAE2010), Vol. 2, pp. 281-285, Feb. 2010.