國 立 交 通 大 學

工 業 工 程 與 管 理 學 系

博士論文

粒子群演算法於多目標排程問題之研究

A Study of Particle Swarm Optimization for Multi-objective Production Scheduling Problems

研究生 ：林信宏

指導教授 ：沙永傑 教授

洪瑞雲 教授

中 華 民 國 九 十 九 年 六 月

粒子群演算法於多目標排程問題之研究

A Study on Particle Swarm Optimization for Multi-objective Production Scheduling Problems

研 究 生： 林信宏　　Student： Hsing-Hung Lin

指導教授： 沙永傑　　Advisor： Dr. D. Y. Sha

洪瑞雲　　Dr. R. Y. Horng

國 立 交 通 大 學
工 業 工 程 與 管 理 學 系
博 士 論 文

A Dissertation

Submitted to Department of Industrial Engineering and Management

College of Management

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Industrial Engineering and Managemen

June 2010

Hsinchu, Taiwan, Republic of China

中 華 民 國 九 十 九 年 六 月

粒子群演算法於多目標排程問題之研究

研究生：林信宏　　　　　　　　　　　　　指導教授：　沙永傑博士
洪瑞雲博士

國立交通大學工業工程與管理系

摘要

以往學術上排程問題的研究主流是尋找單一目標的最佳解(如: 最小完工時間)，然而，實務上生產製造系統的排程需求是達成多目標最佳化。由於運算時間與成本的考量，過去的許多研究已經發展出許多演算法則以搜尋最佳解或近似最佳解。

在本篇論文中，我們分別提出適合求解流程型排程問題(Flow Shop Scheduling Problem, FSSP)、零工型排程問題(Job Shop Scheduling Problem, JSSP)與開放型排程問題(Open Shop Scheduling Problem, OSSP)的粒子群最佳化演算法(Particle Swarm Optimization, PSO)。本研究所提出的演算法針對三種典型排程問題，以同時達到最小完工時間(Makespan)、總流程時間(Total flow time)與機器閒置時間(Machine idle time)作為目標。

粒子群演算法是一種群體搜尋最佳化演算法，於 1995 年被提出。原始的 PSO 是應用於求解連續最佳化問題。因為排程問題為一離散最佳化問題，我們必須修改粒子位置、粒子移動以及粒子速度的表達方式，讓 PSO 更適於求解排程問題。

對於 FSSP 與 JSSP，本研究比較 PSO 與文獻中的基因演算法(Genetic Algorithm, GA)搜尋三大目標的結果，顯示本文提出的 PSO 優於基因演算法。本研究另行發展求解多目標 OSSP 的基因演算法並與 PSO 進行 Benchmark 問題的比較，計算結果顯示，修改後的 PSO 所搜尋到的解，在品質與效率上優於基因演算法。

關鍵字：粒子群最佳化、流程型排程、零工型排程、開放型排程、啟發式演算法

A Study on Particle Swarm Optimization for Multi-objective Production Scheduling Problems

Student：Hsing-Hung Lin Advisor： Dr. D. Y. Sha
Dr. R. Y. Horng

Department of Industrial Engineering and Management
National Chiao Tung University

ABSTRACT

The academic approach of discovering the single optimal solution (ex. makespan) of scheduling for production system is the mainstream although the empirical requirement of production system is to achieve multi-objective optimization. Many algorithms have been developed to search for optimal or near-optimal solutions due to the computational cost of determining exact solutions.

This study provides a Particle Swarm Optimization (PSO) to elaborate multi-objective flow shop scheduling problem (FSSP), job shop scheduling problem (JSSP) and open shop scheduling problem (OSSP). The proposed evolutionary algorithm searches the optimal solution for objectives by considering the makespan, total flow time, and machine idle time simultaneously.

Particle Swarm Optimization (PSO) is a population-based optimization algorithm, which was developed in 1995. The original PSO is used to solve continuous optimization problems. Due to the discrete solution spaces of scheduling optimization problems, the authors modified the particle position representation, particle movement, and particle velocity in this study. The modified PSO could be applied for solving various benchmark problems; moreover, the results demonstrated that the modified PSO outperformed traditional evolutionary heuristics – Genetic Algorithm in searching quality and efficiency.

Keywords: Particle swarm optimization, Multi-objective, Flow shop scheduling, Job shop scheduling problem, Open shop scheduling

致　謝

時光荏苒，一轉眼已經過了六年，終於我也可以寫下論文的致謝辭。在博士求學過程間，首先要感謝的是指導教授沙永傑老師，除了在研究期間給我精神上的支持外，也在出國參加研討會發表論文時，給予相當多的實質幫助。其次，必須感謝的是共同指導教授洪瑞雲老師，協助處理校內相關的行政規定與作業，使我得以順利取得學位。另外，感謝論文口試委員，系上老師彭德保教授、中華大學謝玲芬教授、台灣科技大學王孔政教授、以及雲林科技大學駱景堯教授，在論文計劃書及最後口試期間提供許多寶貴的建議，由於他們的意見，使本篇論文更加豐富與完善。

感謝論文研究期間一直提供幫助的誠佑學長，還要感謝共同修課，一起打球的同學們：新恩、家寧、國良、崑智、俊雄、彰孚、雅甄、惠誠、志偉、清章。

最後要感謝家人支持，父母親六年間不辭辛勞的協助照顧襁褓中的兒女，也感謝儀鵑一路以來的陪伴，工作同時還要花費心思照料家庭，更付出許多心力叮嚀剛開始學習成長的女兒，有了家人的協助，使我可以在無後顧之憂下，全力完成學業。

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

1.1 Research Motivations

Scheduling is an optimization process by which limited resources are allocated over time among parallel and sequential activities. Such situations develop routinely in factories, publishing houses, shipping universities, hospitals, airports, etc. Solving such a problem amounts to making discrete choice such that an optimal solution is found among a finite or a countable infinite number of alternatives. Such problems are called combinational optimization problems. Typically, the task is complex, limiting the practical utility of combinatorial, mathematical programming and other analytical methods in solving scheduling problems effectively.

To find exact solutions of scheduling problems a branch-and-bound or dynamic programming algorithm is often used. However, many shop scheduling problems are NP-hard, which means that the problem cannot be exactly solved in a reasonable computation time. Using problem-specific information sometimes reduces search space, even though the problem is still difficult to solve exactly. Therefore, heuristic algorithms and dispatching rules are developed to obtain the approximate optimal solution. Meta-heuristic is one of the most popular and the most efficient method to obtain the approximate optimal solution. Among the meta-heuristics, particle swarm optimization (PSO) is new and extensively implemented in recent years. However, the original intent of PSO is to solve continuous optimization problems, and PSO methods that work well for combinatorial optimization are still scarce.

1.2 Research Objectives

The objective of this work is to development PSOs for two shop scheduling problems: the flow shop scheduling problem (FSSP) and the job shop scheduling problem (JSSP). In the work of FSSP, the problem is to find a schedule to minimize the makespan ($C_{\max}$), mean flow time and machine idle time. In the work of JSSP, we attempt to search a schedule to minimize the makespan ($C_{\max}$), machine idle time and total tardiness.

Since the original intent of PSO is to solve continuous optimization problems, we have to modify the original PSO when we implement PSO to a combinatorial optimization problem. PSO can be separated several parts to discuss: position representation, particle velocity, and particle movement. We will develop various PSO designs in this work. On the other hand, the PSO developed in this work can be an example of PSO design for other discrete optimization problems.

1.3 Research Process

The research of this dissertation begins with the determination of research topic. The literature consists with flow shop scheduling, job shop scheduling, open shop scheduling, particle swarm optimization and genetic algorithms. The programs of particle swarm optimization and genetic algorithm are coded with programming language C according to the types of scheduling problem. Then, the experiments are compared to evaluate the performance of each algorithm to different problem types. Finally, the conclusion is remarked. The flow chart of this dissertation is as figure 1.1.

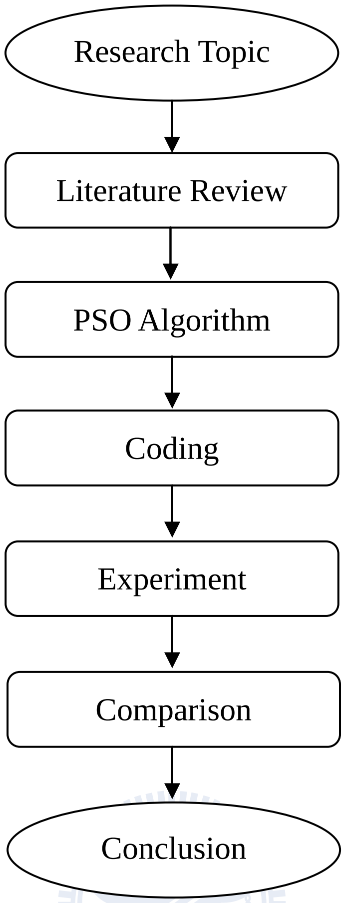

Figure 1. 1 The flow chart of this dissertation

1.4 Organization

The organization of the remaining chapters for this research is as follows. Chapter 2 reviews the literatures of the background of shop scheduling problems and PSO. Chapter 3 describes the factors of PSO design and PSO for FSSP. PSO for JSP is modified and illustrated in Chapter 4. We also proposed a novel PSO for OSSP in Chapter 5. In chapter 6 we draw our conclusion and indicate the direction for further research.

CHAPTER 2 LITERATURE REVIEW

2.1 Particle Swarm Optimization

Particle swarm optimization (PSO) is an evolutionary technique for unconstrained continuous optimization problems proposed by Kennedy and Eberhart (1995). The PSO concept is based on observations of the social behavior of animals such as birds in flocks, fish in schools, and swarm theory. The advantages of the PSO method: simple structure, immediate applicability to practical problems, ease of implementation, quick solution, and robustness. Particle swarm optimization proposed recently for unconstrained continuous optimization problems is one of the latest evolutionary techniques. PSO has been successfully applied to different field of applications due to the easy implementation and computational efficiency. Nevertheless, the applications of the PSO on the combination optimization problem are still scarce.

The major idea of PSO is based on observations of the social behaviors of animals such as bird flocking, fish schooling, and swarm theory. The population is initialized by random solutions. The population consists with individuals (i.e. particles). Each particle is assigned with a randomized velocity according to its own and populations' movement experience. The relationship between swarm and particles in PSO is similar to the relationship between population and chromosomes in GA.

In PSO, the problem solution space is formulated as a search space. Each position of the particles in the search space is a correlated solution of the problem. Particles cooperate to find out the best position (solution) in the search space (solution space).

Suppose that the searching space is D-dimensional and ρ particles comprise the swarm. Each particle locates at the position say $X_i=\{x_{1i}, x_{2i}, \ldots, x_{Di}\}$ with the velocity $V_i=\{v_{1i}, v_{2i}, \ldots, v_{Di}\}$, where i=1, 2, …,ρ. Based on the PSO algorithm, each particle move toward its own best position (*pbest*) denoted as $Pbest_i=\{pbest_{1i}, pbest_{2i},\ldots, pbest_{ni}\}$ and the best position of the whole swarm (*gbest*) denoted as $Gbest=\{gbest_1, gbest_2, \ldots, gbest_n\}$with each iteration. Each particle changes its position according to its velocity which is randomly generated toward *pbest* and *gbest* positions. For each particle r and dimension s, the new velocity v_{sr} and position x_{sr} of particles can be calculated by the following equations:

$$v_{kj} \leftarrow w \times v_{kj} + c_1 \times rand_1 \times (pbest_{kj} - x_{kj}) + c_2 \times rand_2 \times (gbest_j - x_{kj}) \qquad (2.1)$$

$$x_{kj} \leftarrow x_{kj} + v_{kj} \qquad (2.2)$$

In Eqs. (2.1) and (2.2), τ means the iteration number. The inertia weight w is employed to control exploration and exploitation. A large w keeps particles with high velocity and prevents particles from trapping in local optima. A small w maintains low velocity of particles and urges particles to exploit the same search area. The constant c_1 and c_2 are acceleration coefficients to determine whether particles prefer to move closer to *pbest* position or *gbest* position. The $rand_1$ and $rand_2$ are two independent random numbers uniformly distributed between 0 and 1. The termination criterion of the PSO algorithm includes the maximal number of generations, designated value of *pbest* and no further improved *pbest*. The standard process of PSO is outlined as follows:

(1)Initialize a population of particles with random positions and velocities on d dimensions in the search space.

(2)Update the velocity of each particle, according to Eq. (2.1).

(3)Update the position of each particle, according to Eq. (2.2).

(4)Map the position of each particle into solution space and evaluate its fitness value according to the desired optimization fitness function. Meanwhile, update *pbest* and *gbest* position if necessary.

(5)Loop to Step2 until a criterion is met, usually a sufficient good fitness or a maximum number of iterations.

The original PSO is designed to suit continuous solution space. For better applying to combinational optimization problems, we have to modify PSO position representation, particle velocity, and particle movement.

2.2 Genetic Algorithm

The concept of genetic algorithms (GA) was introduced by Holland (1975) as a general search technique which mimics biological evolution, with the survival of the fittest individuals and a structured, yet randomized, information exchange like in population genetics. GAs have been applied with a growing success to combinational problems (Reeves, 1996). GAs works on a set (population) of solutions. Each solution is encoded as a string of symbols called chromosome, and is associated with a measure of adaptation, the fitness, often related to the objective function. Starting from an initial population, new solutions are generated by selecting some *parents* randomly, but with a probability growing with fitness, and by applying genetic operators such as *crossover* (an exchange of substrings of the parent chromosomes) and *mutation* (a random perturbation of a chromosome). Some existing solutions are then selected at random and replaced by some of the offspring, to keep a constant population size. The process is repeated until a satisfactory solution is found.

For solving optimization problems, genetic algorithms have been investigated and shown to be effective at exploring a large and complex space in an adaptive way guided by the equivalent biological evolution mechanism (Huang and Adeli, 1994). Many conventional optimization methods start from one point in the search area and then move sequentially to achieve the optimal solution, thereby operating rather locally and highly prone to falling inside a coincidental local optimum.

GAs are known for their robustness: they can be applied to a wide range of problems without special knowledge about the problem structure. The price to pay is that they cannot compete with meta-heuristics which explore problem-specific neighborhoods. However, more and more paper have showed that GAs can outperform meta-heuristics on some problems, when they are enriched by some problem-specific knowledge, or when they are hybridized with other improvement techniques such as local search.

2.3 Flow Shop Scheduling Problem

Production scheduling in real environments has become a significant challenge in enterprises maintaining their competitive positions in rapidly changing markets. Flow shop scheduling problems have attracted much attention in academic circles in the last five decades since Johnson’s initial research. Most of these studies have focused on finding the exact optimal solution. A brief overview of the evolution of flow shop scheduling problems and possible approaches to their solution over the last fifty years has been provided by Gupta and Stafford (2006). That survey indicated that most research on flow shop scheduling has focused on single-objective problems, such as minimizing completion time, total flow time, or total tardiness. Numerous heuristic techniques have been developed for obtaining the approximate optimal solution to NP-hard scheduling problems. A complete survey of flow shop scheduling problems with makespan criterion and contributions, including exact methods, constructive heuristics, improved heuristics, and evolutionary approaches from 1954 to 2004, was offered by Hejazi and Saghafian (2005). Ruiz and Maroto (2004) also presented a review and comparative evaluation of heuristics and meta-heuristics for permutation flowshop problems with the makespan criterion. The NEH algorithm (Nawaz, Enscore and Ham, 1983) has been shown to be the best constructive heuristic for Taillard's benchmarks (Taillard, 1993) while the iterated local search (Stützle, 1998) method and the genetic algorithm (GA) (Reeves, 1995) are better than other meta-heuristic algorithms.

Most studies of flow shop scheduling have focused on a single objective that could be optimized independently. However, empirical scheduling decisions might not only involve the consideration of more than one objective, but also require minimizing the conflict between two or more objectives. In addition, finding the exact solution to scheduling problems is computationally expensive because such problems are NP-hard. Solving a scheduling problem with multiple objectives is even more complicated than solving a single-objective problem. Approaches including meta-heuristics and memetics have been developed to reduce the complexity and improve the efficiency of solutions.

Hybrid heuristics combining the features of different methods in a complementary fashion have been a hot issue in the fields of computer science and operational research (Liu et al., 2007). Ponnambalam et al. (2004) considered a weighted sum of multiple objectives, including minimizing the makespan, mean flow time, and machine idle time as a performance measurement, and proposed a multi-objective algorithm using a traveling salesman algorithm and the GA for the flow shop scheduling problem. Rajendran and Ziegler (2004) approached the problem of scheduling in permutation flow shop using two ant colony optimization (ACO) approaches, first to minimize the makespan, and then to minimize the sum of the total flow time. Yagmahan and Yenisey (2008) was the first to apply ACO meta-heuristics to flow shop scheduling with the multiple objectives of makespan, total flow time, and total machine idle time.

The literature on multi-objective flow shop scheduling problems can divided into two groups: a priori approaches with assigned weights of each objective, and a posteriori approaches involving a set of non-dominated solutions (Pasupathy et al., 2006). There is also a multi-objective GA (MOGA) called PGA-ALS, designed to search non-dominated sequences with the objectives of minimizing makespan and total flow time. The multi-objective solutions are called non-dominated solutions (or Pareto-optimal solutions in the case of Pareto-optimality). Eren and Güner (2007) tackled a multi-criteria two-machine flow shop scheduling problem with minimization of the weighted sum of total completion time, total tardiness, and makespan.

To minimize the objective of maximum completion time (i.e., the makespan), Liu et al. (2007) invented an effective PSO-based memetic algorithm for the permutation flow shop scheduling problem. Jarboui et al. (2008) developed a PSO algorithm for solving the permutation flow shop scheduling problem; this was an improved procedure based on simulated annealing. PSO was recommended by Tasgetiren et al. (2007) to solve the permutation flow shop scheduling problem with the objectives of minimizing makespan and the total flow time of jobs. Rahimi-Vahed and Mirghorbani (2007) tackled a bi-criteria permutation flow shop scheduling problem where the weighted mean completion time and the weighted mean tardiness were minimized simultaneously. They exploited a new concept called the ideal point and a new approach to specifying the superior particle's position vector in the swarm that is designed and used for finding the locally Pareto-optimal frontier of the problem. Due to the discrete nature of the flow shop scheduling problem, Lian et al. (2008) addressed permutation flow shop scheduling with a minimized makespan using a novel PSO.

2.4 Job Shop Scheduling Problem

Job shop scheduling problem (JSSP) has been studied for more than 50 years in both academic and practical fields. Jain and Meeran (1999) gave a concise overview of JSSP over the last decades and highlighted the main techniques. JSSP is the toughest class in the combinational optimization. Garey et al. (1976) demonstrated that JSSP is NP-hard (NP stands for non-deterministic polynomial), hence we cannot find the exact solution of it in reasonable computation time. The single objective JSSP has attracted researching concentration widely. Most studies of single objective JSSP are discovering a schedule to minimize the time required to complete all jobs, namely makespan (C_{max}). In order to conquer the limitation the exact enumeration techniques, many approximate methods have been developed in the last decades. These approximate approaches includes simulated annealing (Lourenco, 1995), tabu search (Sun et al., 1995; Nowicki and Smutnicki 1996; Pezzella and Merelli 2000) and genetic algorithm (Bean, 1995; Kobayashi et al., 1995; Wang and Zheng, 2001; Goncalves et al., 2005). However, in real world, the multi-objectives requirements of production system should be achieved at the same time. This makes the academic concentration of objectives in JSSP has been extended from single to multiple. Related works of JSSP with multiple objectives in recent years is summarized as below.

Ponnambalam et al. (2001) has offered a multi-objective genetic algorithm to derive the optimal machine-wise priority dispatching rules to resolve the job shop problems with the objective functions considered minimization of makespan, minimization of total tardiness, and minimization of total idle time of machines. Verified by the benchmark problem in the literatures, the proposed MOGA is capable of providing optimal or near-optimal solutions. A Pareto front provides a set of best solutions to determine the trade-offs between the various objects. Good parameter settings and appropriate representations can enhance the behavior of an evolution algorithm. Esquivel et al. (2002) conducted a study of the influence of distinct parameter combinations as well as different chromosome representations. Initial result shows that: (i)Larger numbers of generations favor the building of a Pareto front because the search process (if rather slow) does not stagnate. (ii)Multi-recombination helps to speed the search and to find a larger set size when seeking the Pareto optimal set. (iii)Operation based representation is the best of the three representations selected for contrast under both methods of recombination. A meta-heuristic procedure based on the simulated annealing algorithm called Pareto archived simulated annealing (PASA) is proposed by Suresh and Mohanasndaram (2006) to discover non-dominated solution sets for the job shop scheduling problem with the objectives of minimizing the makespan and the mean flow time of jobs. The superior performance of the PASA can be attributed to its acceptance mechanism used to accept the candidate solution. Candido et al. (1998) addressed job shop scheduling problems with numbers of more realistic constraints such as job with several subassembly levels, alternative processing plans for parts and alternative resources of operations, requirement of multiple resources to process an operation, etc. The robust procedure worked well in all problem instances, showing to be a promising tool to solve more realistic job shop scheduling problems. Lei and Wu (2006) firstly designed a crowding-measure-based multi-objective evolutionary algorithm (CMOEA) which makes use of the crowding measure to adjust the external population and assign different fitness for individuals. The comparison between CMOEA and SPEA demonstrates that CMOEA performs well in job shop scheduling with two objectives including minimization of makespan and total tardiness.

Coello et al. (2004) provided an approach in which Pareto dominance is incorporated into particle swarm optimization in order to allow the heuristic to handle problems with several object functions. The algorithm used secondary repository of particles to guide particle flight. The proposed approach is validated using several test functions and metrics taken from the standard literature on evolutionary multi-objective optimization. The results show that the approach is highly competitive. Liang et al. (2005) have invented a novel PSO-based algorithm for job- shop scheduling problems. The algorithm effectively exploits the capability of distributed and parallel computing systems, with simulation result showing the possibility of high quality solutions for typical benchmark problems. Lei (2008) presented a particle swarm optimization for multi-objective job shop scheduling problem to simultaneously minimize makespan and total tardiness of jobs. By constructing the corresponding relation between real vector and the chromosome obtained by using priority rule-based representation method, job shop scheduling is converted into a continuous optimization problem. The global best position selection is combined with the crowding measure-based archive maintenance to design a Pareto archive particle swarm optimization. The proposed algorithm is capable of producing a number of high-quality Pareto optimal scheduling plans.

Incorporating different approaches to take the strength of them, some hybrid algorithms have been proposed lately and lead to another research branch. Wang and Zheng (2001) reasonably combined GA with SA to invent a hybrid framework, in which GA was introduced to present a parallel search architecture, and SA was introduced to increase escaping probability from local optimal at high temperatures. Computer simulation results based on some b showed that the hybrid strategy was very effective and robust, and could almost find optima for all benchmark instances. Based on the hybridization of PSO and SA, Xia and Wu (2005) developed an easily implemented approach for the multi-objective flexible job shop scheduling problem. The results obtained from the computational study have shown that the proposed algorithm is a viable and effective approach for the multi-objective FJSP, especially for problems on a large scale. Ripon (2007) extends the idea called Jumping Genes Genetic Algorithm (JGGA) to propose a hybrid approach which can search for the near-optimal and non-dominated solutions with better convergence by optimizing criteria simultaneously.

2.5 Open Shop Scheduling Problem

Shop scheduling problems, including flow-, job-, and open-shop problems, have attracted the interest of many researchers. Shop scheduling has become a significant factor used by shops to maintain their competitive position in a rapidly changing marketplace. Most previous research into the open-shop scheduling problem has concentrated on finding a single optimal solution (e.g., makespan). However, in the real world, the multiple-objective requirements of shop scheduling must be achieved simultaneously. Thus, the academic study of open-shop scheduling has been extended from a single objective to multiple objectives.

Because the open-shop scheduling problem is non-deterministic polynomial-time hard (NP hard) for more than two machines (m > 2) (Gonzalez and Sahni ,1976), we cannot solve it exactly using a reasonable amount of computation time. Most published research has concentrated on developing heuristic algorithms to search for the optimal makespan of open-shop scheduling problems. A neighborhood search algorithm based on the simulated annealing technique was proposed by Liaw (1999) to addresses the problem of scheduling a non-preemptive open shop with the objective of minimizing the makespan. An efficient local search algorithm based on the tabu search technique was also proposed by Liaw (1999) to minimize the makespan.

Liaw (2000) developed and applied a hybrid genetic algorithm (HGA) to the open-shop scheduling problem. The hybrid algorithm incorporated a local improvement procedure based on the tabu search (TS) into the basic genetic algorithm (GA). Blum (2005) proposed the Beam-ACO technique to tackle open-shop scheduling; this technique consisted of a hybridized solution construction mechanism for ant colony optimization (ACO) with a beam search. Several competitive GAs have also been presented to detect global optimal values disseminated among many quasi-optimal schedules of the open-shop problem (Prins, 2000). A heuristic technique for the open-shop scheduling problem using the genetic algorithm to minimize the makespan was developed by Senthilkumar and Shahbudeen (2006), and Tang and Bai (2010) proposed a heuristic algorithm, known as the shortest processing time block (SPTB), to solve the open-shop problem by minimizing the sum of the completion time.

Liang (2005) considered the problem of scheduling preemptive open shops to minimize the total tardiness. He developed an efficient constructive heuristic to solve large problems. To solve medium-sized problems, he proposed a branch-and-bound algorithm that incorporated a lower bound scheme based on the solution of an assignment problem as well as various dominance rules.

Blazewicz et al. (2004) applied a non-classical performance measure, the late work criterion, to scheduling problems. They estimated the quality of the obtained solution with regards to the duration of the late parts of the tasks, but did not take into account the quality of these delays.

One of the latest evolutionary techniques, particle swarm optimization (PSO), was recently proposed by Kennedy and Eberhart (1995) for unconstrained continuous optimization problems. The idea behind PSO is based on observations of the social behavior of animals such as flocks of birds or schools of fish, combined with swarm theory. PSO has been successfully applied to different fields due to its easy implementation and computational efficiency. Nevertheless, applications of PSO to combinations of optimization problems are still scarce.

2.6 Multiple Objective Programming

There are several ways to classify the different approaches to multiobjective optimization. Adulbhan and Tabucanon (1989) classified the techniques into three main approaches based on the way the initial multiobjective problem is transformed into a mathematically manageable format. These approaches are, respectively, (a) conversion of secondary objectives into constraints, (b) development of a single combined objective function, and (c) treatment of all objectives as constraints. Hwang, Masud, Paidy and Yoon (1982), on the other hand, propose grouping of techniques according to the stage at which the analyst needs information from the decision-maker. The classification is divided into four approaches: (a) no articulation of decision maker’s preference data, (b) a priori articulation of preference data, (c) progressive articulation of preference data, and (d) a posteriori articulation of preference data.

A recently proposed method for treating the analytical phase of the MCDM process is called multiple criteria optimization or, in short, multiobjective optimization (Seo and Sakawa, 1988). According to this viewpoint, multiple criteria optimization contains two key concepts: (a) Pareto optimality and (b) the preferred decision (or preferred solution). In general, the decisions with Pareto optimality are not uniquely determined, unlike, for instance, what goal programming produces. In multiobjective optimization problems, the usually exist many solutions that are optimal in the Pareo sense, a concept put forth by economists. Owing to such plurality of optimal decisions, the most desirable decision may be selected after one has generated the Pareto optimal or nondominated solutions. The final solution thus selected as the most desirable, or at least the best-compromised solution, is called preferred solution.

Many approaches have been developed in the domain of multi-objective meta-heuristic optimization. Hsu, Dupas, Jolly & Goncalves (2002) focus our presentation on evolutionary approaches that can be classified into three types: (a) The transformation towards a mono-objective problem consists of combining the different objective into a weighted sum. (b) The non-Pareto approach utilizes operators for processing the different objectives in a separated way. (c) The Pareto approach is directly based on the Pareto optimization concept. It aims at satisfy two goals: coverage to the Pareto front and obtain diversified solutions scattered all over the Pareto front.

In real world, empirical scheduling decisions should not only involve the deliberation of more than one objective at a time, but also need to prevent the conflict of two or more objectives. The solution set of multi-objective optimization problem with conflicting objective function consisted with the solutions that no other solution is better than all other objective functions is called Pareto optimal. A multi-objective minimization problem with m decision variables and n objectives is given below to describe the concept of Pareto optimality.

$$\begin{aligned}&\textit{Minimize}\quad F(x) = (f_1(x), f_2(x), \ldots, f_n(x))\\&\textit{where},\ x \in \Re^m,\ F(x) \in \Re^n\end{aligned}$$

A solution *p* is said to dominate solution q if and only if:

$$\begin{aligned}&f_k(p) \le f_k(q) \quad \forall k \in \{1, 2, \ldots, n\}\\&f_k(p) < f_k(q) \quad \exists k \in \{1, 2, \ldots, n\}\end{aligned}$$

The non-dominated solution is defined as solutions which dominate the others but do not dominate themselves. Solution *p* is said a Pareto-optimal solution if there exist no other solution *q* in the feasible space which could dominate *p*. The set including all Pareto-optimal solutions is termed the Pareto-optimal Set, or the efficient set. The graph plotted using collected Pareto-optimal solutions in feasible space is designated as Pareto front.

The external Pareto optimal set is employed to deposit a limited size of non-dominated solutions (Knowles et al., 2000; Zitzler et al. 2001). Maximum size of archive set is specified in advance. This method is applied to forbid missing fragment of non-dominated front during the searching process. The Pareto-optimal front is getting formed as archive updated iteratively. While the archive set is empty enough and a new non-dominated solution is detected, the new solution will enter the archive set. As the new solution enters the archive set, any solution in the archive set dominated by this solution will be withdrawn from the archive set. In case the maximum archive size reaches its preset value, the archive set have to decide which solution could be replaced.

In this study, we propose a novel Pareto archive set updating process in order to preclude from losing non-dominated solutions when the Pareto archive set is full. When a new non-dominated solution is discovered, the archive set would be updated when one of the following situation occurs: (a) number of solutions in the archive set is less than the maximum value; (b) number of the solutions in the archive set is equal to (greater than) the maximum value, then one of the solutions in the archive set that is most dissimilar to the new solution will be replaced by the new solution. We measure the dissimilarity by Euclidean distance. A longer distance implies a higher dissimilarity is. The non-dominated solution in the Pareto archive set with the longest distance to the new found solution will be replaced.

CHAPTER 3 PSO for Multi-objective FSSP

In this chapter, we will discuss the probably success factors to develop a PSO design for a discrete optimization problem. We will compare PSO with another population-based meta-heuristic—genetic algorithm (GA). The principles of a GA design may be also suitable to a PSO design.

There are two different representations of particle position associated with a schedule. Zhang et al. (2005) demonstrated that permutation-based position representation outperforms priority-based representation. While we have chosen to implement permutation-based position representation, we must also adjust the particle velocity and particle movement.

There are four types of feasible schedules in JSP, including inadmissible, semi-active, active and non-delay. The optimal schedule is guaranteed to be an active schedule. We can decode a particle position into an active schedule employing Giffler and Thompson's (1960) heuristic. There are two different representation of particle position associated with a schedule. The results of Zhang (2005) demonstrated that permutation-based position representation outperforms priority-based representation. While choosing permutation-based position presentation to implement, we also have to adjust the particle velocity and particle movement. In addition, the maintenance of Pareto optima and diversification procedure are proposed finally for better performance.

3.1 Problem Formulation

The problem of scheduling in flow shops has been the subject of much investigation.

The primary elements of flow shop scheduling include a set of *m* machines and a collection of *n* jobs to be scheduled on the set of machines. Each job follows the same process of machines and passes through each machine only once. Each job can be processed on one and only one machine at a time, whereas each machine can process only one job at a time. The processing time of each job on each machine is fixed and known in advance. We formulate the multi-objective flow shop scheduling problem using the following notation:

n total number of jobs to be scheduled

m total number of machines in the process

$t(i, j)$ processing time for job i on machine j (i=1,2,…n), (j=1,2,…m)

L_i the lateness of job i

$\{\pi_1, \pi_2, \ldots, \pi_n\}$ permutation of jobs

The objectives considered in this paper are formulated as follows:

Completion time (makespan) $C(\pi, j)$

$$C(\pi_1,1) = t(\pi_1,1) \tag{3.1}$$

$$C(\pi_i,1) = C(\pi_{i-1},1) + t(\pi_i,1) \quad i = 2,...,n \tag{3.2}$$

$$C(\pi_1, j) = C(\pi_1, j-1) + t(\pi, j) \quad j = 2,...,m \tag{3.3}$$

$$C(\pi_i, j) = \max\{C(\pi_{i-1}, j), C(\pi_i, j-1)\} + t(\pi_i, j) \quad i = 2,...,n; \quad j = 2,...,m \tag{3.4}$$

Makespan, $f_{C\max} = C(\pi_n, m)$ (3.5)

Mean flow time, $f_{MFT} = [\sum_{i=1}^{n} C(\pi_i, m)] / n$ (3.6)

Machine idle time,

$$f_{MIT} = \{C(\pi_1, j-1) + \sum_{i=2}^{n} \{\max\{C(\pi_i, j-1) - C(\pi_{i-1}, j), 0\}\} \mid j = 2...m\} \quad (3.7)$$

3.2 Particle Position Representation

In the study of flow shop scheduling, we randomly generated a group of particles (solutions) represented by a permutation sequence that is an ordered list of operations. The following example is a permutation sequence for a six-job permutation flow shop scheduling problem, where j_n is the operation of job n.

Index:	1	2	3	4	5	6
Permutation:	j_4	j_3	j_1	j_6	j_2	j_5

An operation earlier in the list has a higher priority of being placed into the schedule. We used a list with a length of n for an n-job problem in our algorithm to represent the position of particle k, i.e.

$X^k = [x_1^k \; x_2^k \; ... \; x_n^k]$,

x_i^k is the priority of j_i in particle k.

Then, we convert the permutation list to a priority list. x_i^k is a value randomly initialized to some value between (p – 0.5) and (p + 0.5). This means $x_i^k \leftarrow p + rand - 0.5$, where p is the location (index) of j_i in the permutation list, and *rand* is a random number between 0 and 1. Consequently, the operation with smaller x_i^k has a higher priority for scheduling. The permutation list mentioned above can be converted to X^k = [2.7 5.2 1.8 0.6 6.3 3.9].

We describe the conversion between integers and float-point numbers as follows. The permutation list is represented in integer, while the priority list is presented in floating-point number. At first, we generate integers randomly for permutation list. The permutation list could convert to priority list via the equation $x_i^k = p_i + rand() - 0.5$, where *rand()* is the random number between 0 and 1.

Index	1	2	3	4	5	6
Permutation	j_4	j_3	j_1	j_6	j_2	j_5

Permutation list	4	3	1	6	2	5
p_i	1	2	3	4	5	6

$$x_i \leftarrow p_i + rand - 0.5$$

Permutation	j_4	j_3	j_1	j_6	j_2	j_5
Priority list	0.6	1.8	2.7	3.9	5.2	6.3

Figure 3.1 The conversion between integers and float-point numbers

The priority list contains real number is used in our PSO. The priority list stored in the array is as follows.

Index	*1*	*2*	*3*	*4*	*5*	*6*
Priority list	*2.7*	*5.2*	*1.8*	*0.6*	*6.3*	*3.9*

Figure 3.2 The priority list stored in the array

As the particle move, the value of priority list may change. We assume that the priority list change to be followed.

Index	*1*	*2*	*3*	*4*	*5*	*6*
Priority list	*2.7*	*5.2*	*1.8*	*0.6*	*2.6*	*3.9*

Figure 3.3 The priority list changed as particle movement

Finally, we sort the priority list and we can get a new permutation list. The new list can be used to calculate fitness function.

Index	1	2	3	4	5	6
Priority list	2.7	5.2	1.8	0.6	2.6	3.9

Sorting

Permutation list	4	3	5	1	6	2

Figure 3.4 A new permutation list

3.3 Particle Velocity

The original PSO velocity concept is that each particle moves according to the velocity determined by the distance between the previous position of the particle and the *gbest* (*pbest*) solution. The two major purposes of the particle velocity are to move the particle toward the *gbest* and *pbest* solutions, and to maintain the inertia to prevent particles from becoming trapped in local optima.

In the proposed PSO of flow shop scheduling, we concentrated on preventing particles from becoming trapped in local optima rather than moving them toward the *gbest* (*pbest*) solution. If the priority value increases or decreases with the present velocity in this iteration, we maintain the priority value increasing or decreasing at the beginning of the next iteration with probability w, which is the PSO inertial weight. The larger the value of w is, the greater the number of iterations over which the priority value keeps increasing or decreasing, and the greater the difficulty the particle has returning to the current position. For an n-job problem, the velocity of particle k can be represented as

$$V^k = [\, v_1^k \; v_2^k \; \ldots v_n^k \,], v_i^k \in \{-1, 0, 1\}$$

where v_i^k is the velocity of j_i of particle k.

The initial particle velocities are generated randomly. Instead of considering the distance from x_i^k to $pbest_i^k$ ($gbest_i$), our PSO considers whether the value of x_i^k

is larger or smaller than $pbest_i^k$ ($gbest_i$) If x_i^k has decreased in the present iteration, this means that $pbest_i^k$ ($gbest_i$) is smaller than x_i^k, and x_i^k is set moving toward $pbest_i^k$ ($gbest_i$) by letting $v_i^k \leftarrow -1$. Therefore, in the next iteration, x_i^k is kept decreasing by one (i.e., $x_i^k \leftarrow x_i^k - 1$) with probability w. Conversely, if x_i^k has increased in this iteration, this means that $pbest_i^k$ ($gbest_i$) is larger than x_i^k, and x_i^k is set moving toward $pbest_i^k$ ($gbest_i$) by letting $v_i^k \leftarrow 1$. Therefore, in the next iteration, x_i^k is kept increasing by one (i.e. $x_i^k \leftarrow x_i^k + 1$) with probability w.

The inertial weight w influences the velocity of particles in PSO. We randomly update velocities at the beginning of iterations. For each particle k and operation j_{i}, if v_i^k is not equal to 0, v_i^k is set to 0 with probability $(1-w)$. This ensures that x_i^k stops increasing or decreasing continuously in this iteration with probability $(1-w)$.

3.4 Particle Movement

The particle movement of flow shop scheduling is based on the insertion operator proposed by Sha and Hsu (2008). The insertion operator is introduced to the priority list to reduce computational complexity. We illustrate the effect of the insertion operator using the permutation list example described above. If we wish to insert j_4 into the third location of the permutation list, we must move j_6 to the sixth location, move j_1 to the fifth location, move j_2 to the fourth location, and then insert j_4 in the third location. The insertion operation comprising these actions costs $O(n/2)$ on average. However, the insertion operator used in this study need only set $x_i^k \leftarrow 3 + rand - 0.5$ when we want to insert j_5 in the third location of the permutation.

This requires only one step for each insertion. If the random number *rand* equals 0.1, for example, after j_4 is inserted into the third location, then X^k becomes X^k = [2.7 5.2 1.8 0.6 2.6 3.9].

If we wish to insert j_i into the *p*th location in the permutation list, we could set $x_i^k \leftarrow p + rand - 0.5$. The location of operation j_i in the permutation sequence of the *k*th *pbest* and *gbest* solutions are $pbest_i^k$ and $gbest_i$, respectively. As particle *k* moves, if v_i^k equals 0 for all j_i, then x_i^k is set to $pbest_i^k + rand - 0.5$ with probability c_1 and set to $gbest_i + rand - 0.5$ with probability c_2, where *rand* is a random number between 0 and 1, c_1 and c_2 are constants between 0 and 1, and $c_1 + c_2 \le 1$. We explain this concept by assuming specific values for V^k, X^k, $pbest^k$, *gbest*, c_1, and c_2.

$$
\begin{aligned}
&V^k = [-1\ 0\ 0\ 1\ 0\ 0],\\
&X^k = [2.7\ 5.2\ 1.8\ 0.6\ 6.3\ 3.9],\\
&pbest^k = [5\ 1\ 4\ 6\ 3\ 2],\\
&gbest = [6\ 3\ 4\ 5\ 1\ 2], \mathrm{c}_1 = 0.8, c_2 = 0.1.
\end{aligned}
$$

For j_1, since $v_1^k \neq 0$ and $x_1^k \leftarrow x_1^k + v_1^k$, then $x_1^k = 1.7$.

For j_2, since $v_2^k = 0$, the generated random number $rand_1 = 0.6$. Since $rand_1 \le c_1$, then the generated random number $rand_2 = 0.3$. Since $pbest_2^k \le x_2^k$, set $v_2^k \leftarrow -1$ and $x_2^k \leftarrow pbest_2^k + rand_2 - 0.5$, i.e., $x_2^k = 0.8$.

For j_3, since $v_3^k = 0$, the generated random number $rand_1 = 0.93$. Since $rand_1 > c_1 + c_2$, x_3^k and v_3^k do not need to be changed.

For j_4, since $v_4^k = 1$, then $x_4^k \leftarrow x_4^k + v_4^k$, i.e., $x_4^k = 1.6$.

For j_5, since $v_5^k = 0$, the generated random number $rand_1 = 0.85$. Since $c_1 < rand_1 \le c_1 + c_2$, the generated random number $rand_2 = 0.7$. Since $gbest_5 \le x_5^k$, set $v_5^k \leftarrow -1$. Then $x_5^k \leftarrow gbest_5 + rand_2 - 0.5$, i.e., $x_5^k = 1.2$.

For j_6, since $v_6^k = 0$, the generated random number $rand_1 = 0.95$. Since $rand_1 > c_1 + c_2$, x_6^k and v_6^k do not need to be changed.

Therefore, after particle k moves, the V^k and X^k are

$$V^k = [-1 \quad -1 \quad 0 \quad 1 \quad -1 \quad 0]$$
$$X^k = [1.6 \quad 0.8 \quad 1.8 \quad 1.7 \quad 1.2 \quad 3.9]$$

In addition, we use a mutation operator in our PSO algorithm. After moving a particle to a new position, we randomly choose an operation and then mutate its priority value x_i^k in accordance with v_i^k. If $x_i^k \leq (n/2)$, we randomly set x_i^k to a value between (n/2) and n, and set $v_i^k \leftarrow 1$. If $x_i^k > (n/2)$, we randomly set x_i^k to a value between 0 and (n/2), and set $v_i^k \leftarrow -1$.

3.5 Pareto optimal set maintenance

Real empirical scheduling decisions often involve not only the consideration of more than one objective at a time, but also must prevent the conflict of two or more objectives. The solution set of the multi-objective optimization problem with conflicting objective functions consistent with the solutions so that no other solution is better than all other objective functions is called *Pareto optimal*. A multi-objective minimization problem with m decision variables and n objectives is given below to describe the concept of Pareto optimality.

$$\begin{aligned}&\textit{Minimize} \quad F(x) = (f_1(x), f_2(x), \ldots, f_n(x)) \\ &\textit{where}, \ x \in \Re^m, \ F(x) \in \Re^n\end{aligned}$$

A solution p is said to dominate solution q if and only if

$$\begin{aligned}&f_k(p) \leq f_k(q) \quad \forall k \in \{1, 2, \ldots, n\} \\ &f_k(p) < f_k(q) \quad \exists k \in \{1, 2, \ldots, n\}\end{aligned}$$

Non-dominated solutions are defined as solutions that dominate the others but do not dominate themselves. Solution p is said to be a Pareto-optimal solution if there exists no other solution q in the feasible space that could dominate p. The set including all Pareto-optimal solutions is referred to as the Pareto-optimal or *efficient* set. A graph plotted using collected Pareto-optimal solutions in feasible space is referred to as the *Pareto front*.

The external Pareto optimal set is used to produce a limited size of non-dominated solutions (Knowles and Corne (1999); Zitzler et al. (2001)). The maximum size of the archive set is specified in advance. This method is used to avoid missing fragments of the non-dominated front during the search process. The Pareto-optimal front is formed as the archive is updated iteratively. When the archive set is sufficiently empty and a new non-dominated solution is detected, the new solution enters the archive set. As the new solution enters the archive set, any solution already there that is dominated by this solution will be removed. When the maximum archive size reaches its preset value, the archive set must decide which solution should be replaced. In this study, we propose a novel Pareto archive set update process to preclude losing non-dominated solutions when the Pareto archive set is full. When a new non-dominated solution is discovered, the archive set is updated when one of the following situations occurs: either the number of solutions in the archive set is less than the maximum value, or if the number of solutions in the archive set is equal to or greater than the maximum value, then the one solution in the archive set that is most dissimilar to the new solution is replaced by the new solution. We measure the dissimilarity by the Euclidean distance. A longer distance implies a higher dissimilarity. The non-dominated solution in the Pareto archive set with the longest distance to the newly found solution is replaced. For example, the distance (d_{ij}) between X^1 and X^2 is calculated as

$X^1 = [2.7\ \ 5.2\ \ 1.8\ \ 0.6\ \ 6.3\ \ 3.9]$
$X^2 = [1.6\ \ 0.8\ \ 1.8\ \ 1.7\ \ 1.2\ \ 3.9]$
$d_{ij} =$
$\sqrt{(2.7-1.6)^2 + (5.2-0.8)^2 + (0.6-1.7)^2 + (6.3-1.2)^2}$
$= 6.91$

The Pareto archive set is updated at the end of each iteration in the proposed PSO.

3.6 Computational Results

The proposed PSO algorithm was verified by benchmark problems obtained from the OR-Library that were contributed by Carlier (1978), Heller (1960), and Reeves (1995). The test program was coded in Visual C++ and run 20 times on each problem using an Intel Pentium 4 3.0-GHz processor with 1 GB of RAM running Windows XP. We used four swarm sizes N (10, 20, 60, and 80) to test the algorithm during a pilot experiment. A value of $N = 80$ was best, so it was used in all subsequent tests. The algorithm parameters were set as follows: c_1 and c_2 were tested over the range 0.1–0.7 in increments of 0.2, and the inertial weight w was reduced from w_{max} to w_{min} during the iterations. Parameter w_{max} was set to 0.5, 0.7, and 0.9 corresponding to w_{min} values of 0.1, 0.3, and 0.5. Settings of $c_1 = 0.7$, $c_2 = 0.1$, $w_{max} = 0.7$, and $w_{min} = 0.3$ worked best.

The presented PSO algorithm is compared with two heuristic algorithms: CDS and NEH. We briefly describe these two methods here. CDS heuristic named by the three authors was proposed by Campbell, et al. (1970). The CDS procedure is a heuristic generalization of Johnson’s algorithm. The process generates a set of m-1 artificial two-machine problem, each of which is then solved by Johnson’s rule. In this study, we modified original CDS and compared the makespan, mean flow time and machine idle time of all m-1 generated problems. The non-dominated solution was picked to compare with the solutions obtained from our PSO algorithm. The other comparison is based on the solutions constructed from NEH algorithm that was presented by Nawaz M. et al. (1983). The NEH enumerates *n(n+1)/2* permutations to find near-optimal solutions. Similar to CDS, we modified the original NEH and compared the three objectives of all *n(n+1)/2* sequences. We compared the non-dominated solution from those sequences with the solutions from our PSO.

The makespan, mean flow time, and machine idle time from sequence given by the PSO, CDS and NEH are denoted MS_{PSO}, MFT_{PSO}, and MIT_{PSO}; MS_{CDS}, MFT_{CDS}, and MIT_{CDS}; and MS_{NEH}, MFT_{NEH}, and MIT_{NEH} respectively. The relative error in makespan, mean flow time, and machine idle time for schedule S_{PSO} are as follows.

$$[MS_{PSO} - MIN(MS_{PSO}, MS_{CDS}, MS_{NEH})]/MIN(MS_{PSO}, MS_{CDS}, MS_{NEH}) \tag{3.8}$$

$$[MFT_{PSO} - MIN(MFT_{PSO}, MFT_{CDS}, MFT_{NEH})]/MIN(MFT_{PSO}, MFT_{CDS}, MFT_{NEH}) \tag{3.9}$$

$$[MIT_{PSO} - MIN(MIT_{PSO}, MIT_{CDS}, MIT_{NEH})]/MIN(MIT_{PSO}, MIT_{CDS}, MIT_{NEH}) \tag{3.10}$$

Furthermore, the relative error in makespan, mean flow time, and machine idle time for schedule S_{CDS} could be derived using the following equations.

$$[MS_{CDS} - MIN(MS_{PSO}, MS_{CDS}, MS_{NEH})]/MIN(MS_{PSO}, MS_{CDS}, MS_{NEH}) \tag{3.11}$$

$$[MFT_{CDS} - MIN(MFT_{PSO}, MFT_{CDS}, MFT_{NEH})]/MIN(MFT_{PSO}, MFT_{CDS}, MFT_{NEH}) \tag{3.12}$$

$$[MIT_{CDS} - MIN(MIT_{PSO}, MIT_{CDS}, MIT_{NEH})]/MIN(MIT_{PSO}, MIT_{CDS}, MIT_{NEH}) \tag{3.13}$$

At last, the relative error in makespan, mean flow time, and machine idle time for schedule S_{NEH} could be derived using the following equations:

$$[MS_{NEH} - MIN(MS_{PSO}, MS_{CDS}, MS_{NEH})]/MIN(MS_{PSO}, MS_{CDS}, MS_{NEH}) \tag{3.14}$$

$$[MFT_{NEH} - MIN(MFT_{PSO}, MFT_{CDS}, MFT_{NEH})] / MIN(MFT_{PSO}, MFT_{CDS}, MFT_{NEH}) \quad (3.15)$$

$$[MIT_{NEH} - MIN(MIT_{PSO}, MIT_{CDS}, MIT_{NEH})] / MIN(MIT_{PSO}, MIT_{CDS}, MIT_{NEH}) \quad (3.16)$$

Finally, the following functions are used to measure the aggregated objectives performance of the three heuristics.

$$\frac{[(MS_{PSO} - MIN_{MS}) + (MFT_{PSO} - MIN_{MFT}) + (MIT_{PSO} - MIN_{MIT})]}{MIN_{MS} + MIN_{MFT} + MIN_{MIT}} \quad (3.17)$$

$$\frac{(MS_{CDS} - MIN_{MS}) + (MFT_{CDS} - MIN_{MFT}) + (MIT_{CDS} - MIN_{MIT})}{MIN_{MS} + MIN_{MFT} + MIN_{MIT}} \quad (3.18)$$

$$\frac{(MS_{NEH} - MIN_{MS}) + (MFT_{NEH} - MIN_{MFT}) + (MIT_{NEH} - MIN_{MIT})}{MIN_{MS} + MIN_{MFT} + MIN_{MIT}} \quad (3.19)$$

where

$$MIN_{MS} = MIN(MS_{PSO}, MS_{CDS}, MS_{NEH})$$
$$MIN_{MFT} = MIN(MFT_{PSO}, MFT_{CDS}, MFT_{NEH})$$
$$MIN_{MIT} = MIN(MIT_{PSO}, MIT_{CDS}, MIT_{NEH})$$

In order to examine the performance including efficiency and quality of the proposed PSO algorithm, we have applied our PSO to totally 161 benchmark problems. For problem Rec01 to Rec41, the average relative error of C_{max} and MFT are given in Table 3.1. Table 3.2 shows average relative error of MIT and aggregate performance. From Table 3.3 to Table 3.6, we demonstrated the average relative error of C_{max}, MFT and MIT with the problem Tai20×5 to Tai500×20. The aggregate performance of problem Tai20×5 to Tai500×20 are given in Table 3.7.

At last, we observed that the PSO perform better than other two heuristics while only one objective is considered. Table 3.8 shows the superior number and percentage of problems among the three different algorithms. As we consider the three objectives at the same time, we can prove the performance of proposed PSO by Table 3. 9.

Table 3. 1 The average relative error in C_{max} and MFT of problem RecXX

Problem	Makespan			MFT		
	CDS	NEH	PSO	CDS	NEH	PSO
Rec01_20×5	0.0798	**0.0000**	0.0850	0.4089	**0.0000**	0.3119
Rec03_20×5	0.1867	**0.0000**	0.1278	0.4447	**0.0000**	0.3274
Rec05_20×5	0.1068	**0.0008**	0.0315	0.3974	**0.0000**	0.2931
Rec07_20×10	0.0437	**0.0000**	0.0970	0.1330	**0.0000**	0.0382
Rec09_20×10	0.1712	**0.0000**	0.1106	0.1370	**0.0004**	0.0495
Rec11_20×10	0.1430	**0.0000**	0.0666	0.2467	0.0621	**0.0002**
Rec13_20×15	0.2233	**0.0000**	0.1146	0.0487	0.1487	**0.0015**
Rec15_20×15	0.0796	**0.0000**	0.0935	0.0801	0.1639	**0.0000**
Rec17_20×15	0.1990	**0.0000**	0.1190	0.0721	0.1549	**0.0006**
Rec19_30×10	0.1059	**0.0000**	0.1090	0.2520	**0.0000**	0.1955
Rec21_30×10	0.2029	**0.0000**	0.1531	0.3009	**0.0000**	0.2171
Rec23_30×10	0.1542	**0.0000**	0.1170	0.2376	**0.0000**	0.2060
Rec25_30×15	0.1640	**0.0000**	0.0934	0.1249	**0.0004**	0.0231
Rec27_30×15	0.1365	**0.0000**	0.0983	0.0988	**0.0000**	0.0359
Rec29_30×15	0.2419	**0.0000**	0.1546	0.1576	**0.0000**	0.0466
Rec31_50×10	0.4748	0.2951	**0.0000**	0.7323	0.1225	**0.0000**
Rec33_50×10	0.4603	0.3596	**0.0000**	0.6403	0.2224	**0.0000**
Rec35_50×10	0.5053	0.3202	**0.0000**	0.6703	0.1520	**0.0000**
Rec37_75×20	0.9534	0.6410	**0.0000**	1.3879	0.7679	**0.0000**
Rec39_75×20	0.9371	0.6362	**0.0000**	1.5575	0.8042	**0.0000**
Rec41_75×20	0.9938	0.7155	**0.0000**	1.6152	0.8441	**0.0000**
Average	0.3125	0.1413	**0.0748**	0.4640	0.1640	**0.0832**

Table 3. 2 The average relative error in MIT and Aggregate of problem Rec

Problem	MIT			Aggregate		
	CDS	NEH	PSO	CDS	NEH	PSO
Rec01_20×5	4.1112	2.4282	**0.0000**	0.3410	**0.0797**	0.1676
Rec03_20×5	1.4296	1.0185	**0.0014**	0.3665	**0.0707**	0.1887
Rec05_20×5	3.5144	2.1824	**0.0000**	0.3585	**0.0904**	0.1281
Rec07_20×10	0.7301	0.4095	**0.0019**	0.3020	0.1384	**0.0480**
Rec09_20×10	0.5385	0.2741	**0.0127**	0.2872	0.0832	**0.0594**
Rec11_20×10	2.0732	**0.0082**	0.1115	0.7760	**0.0203**	0.0616
Rec13_20×15	0.2396	0.2623	**0.0097**	0.1948	0.1580	**0.0405**
Rec15_20×15	0.4336	0.4493	**0.0000**	0.2550	0.2521	**0.0287**
Rec17_20×15	0.4120	0.2383	**0.0131**	0.2650	0.1366	**0.0451**
Rec19_30×10	0.5509	**0.0166**	0.1057	0.2844	**0.0049**	0.1323
Rec21_30×10	0.9548	**0.0343**	0.0979	0.4315	**0.0082**	0.1577
Rec23_30×10	0.1880	0.0990	**0.0067**	0.1864	**0.0362**	0.0974
Rec25_30×15	0.6217	0.2278	**0.0060**	0.3703	0.1056	**0.0367**
Rec27_30×15	0.3343	0.4371	**0.0000**	0.2297	0.2262	**0.0342**
Rec29_30×15	0.5644	0.0862	**0.0101**	0.3860	0.0421	0.0592
Rec31_50×10	1.4631	0.4220	**0.0000**	0.8552	0.2808	**0.0000**
Rec33_50×10	0.6859	0.3633	**0.0040**	0.5684	0.3117	**0.0013**
Rec35_50×10	0.7108	0.2891	**0.0000**	0.6201	0.2607	**0.0000**
Rec37_75×20	1.1915	0.8135	**0.0000**	1.1581	0.7601	**0.0000**
Rec39_75×20	1.8247	0.5223	**0.0000**	1.5418	0.5946	**0.0000**
Rec41_75×20	1.7428	1.1721	**0.0000**	1.5445	1.0152	**0.0000**
Average	1.1579	0.5597	**0.0181**	0.5392	0.2227	**0.0613**

Table 3. 3 The average relative error in C_{max} , MFT and MIT of problem Tai_20

	Makespan			MFT			MIT		
Problem	CDS	NEH	PSO	CDS	NEH	PSO	CDS	NEH	PSO
Tai_20×5_1	**0.0010**	0.1205	0.0323	0.1135	0.1052	**0.0000**	3.8211	3.2056	**0.0000**
Tai_20×5_2	**0.0013**	0.1180	0.0278	0.0377	0.0310	**0.0000**	8.7129	0.0296	**0.9753**
Tai_20×5_3	**0.0000**	0.1834	0.0626	0.1000	0.0178	**0.0000**	3.1344	0.0205	**0.1308**
Tai_20×5_4	**0.0088**	0.1487	0.0096	0.0757	0.0627	**0.0000**	4.3087	2.2174	**0.0000**
Tai_20×5_5	**0.0003**	0.1761	**0.0504**	0.1418	0.0468	**0.0000**	2.2425	0.4427	**0.0000**
Tai_20×5_6	0.0654	0.1170	**0.0001**	0.0296	0.0570	**0.0000**	0.3327	0.8453	**0.0000**
Tai_20×5_7	**0.0027**	0.0229	0.0350	0.0642	0.0000	**0.0000**	16.158	1.2283	**0.6350**
Tai_20×5_8	**0.0003**	0.0654	0.0517	0.0633	0.0231	**0.0000**	5.8688	3.8646	**0.0000**
Tai_20×5_9	**0.0033**	0.0814	0.0328	0.0312	0.0366	**0.0000**	1.5550	1.2519	**0.0000**
Tai_20×5_10	0.0353	0.0590	**0.0039**	0.0774	0.0794	**0.0000**	5.1159	5.3127	**0.0000**
Average	0.0119	0.1092	**0.0306**	0.0734	0.0460	**0.0000**	5.1250	1.8419	**0.1741**
Tai_20×10_1	0.0733	0.0543	**0.0000**	0.0406	0.0496	**0.0000**	0.1589	0.3794	**0.0155**
Tai_20×10_2	**0.0014**	0.1127	0.0166	**0.0024**	0.0768	0.0093	0.0132	0.6637	**0.1071**
Tai_20×10_3	0.1509	0.0978	**0.0000**	0.0529	0.0460	**0.0003**	0.2847	0.3143	**0.0044**
Tai_20×10_4	0.1079	0.0792	**0.0000**	0.0688	0.0524	**0.0000**	0.5260	0.4282	**0.0000**
Tai_20×10_5	**0.0022**	0.1666	0.0255	0.0524	0.0420	**0.0006**	0.6851	0.4076	**0.0113**
Tai_20×10_6	0.1863	0.1647	**0.0000**	0.1108	0.0480	**0.0000**	0.6342	0.1983	**0.0043**
Tai_20×10_7	0.1230	0.0938	**0.0000**	0.0454	**0.0092**	0.0133	0.3709	**0.0616**	0.1693
Tai_20×10_8	0.0766	0.1262	**0.0000**	0.0768	0.0537	**0.0002**	0.4519	0.3661	**0.0020**
Tai_20×10_9	0.0902	0.1124	**0.0000**	0.1244	0.0292	**0.0000**	1.2666	0.2793	**0.0016**
Tai_20×10_10	0.0687	0.1368	**0.0000**	0.1527	0.0845	**0.0000**	1.3403	0.6326	**0.0003**
Average	0.0880	0.1144	**0.0042**	0.0727	0.0491	**0.0024**	0.5732	0.3731	**0.0316**
Tai_20×20_1	0.0335	0.0639	**0.0009**	0.0605	0.0707	**0.0000**	0.2408	0.2813	**0.0000**
Tai_20×20_2	0.0334	0.0812	**0.0009**	0.0262	0.0262	**0.0015**	0.1384	0.1135	**0.0109**
Tai_20×20_3	0.0406	0.0672	**0.0000**	0.0693	0.0693	**0.0000**	0.2703	0.3117	**0.0000**
Tai_20×20_4	0.0268	0.0978	**0.0005**	0.0783	0.0673	**0.0001**	0.3266	0.2998	**0.0005**
Tai_20×20_5	0.0691	0.0702	**0.0000**	0.0337	0.0109	**0.0069**	0.1362	0.0368	**0.0325**
Tai_20×20_6	0.0234	0.0894	**0.0004**	0.1383	0.0373	**0.0103**	0.6739	0.1864	**0.0490**
Tai_20×20_7	0.0232	0.1210	**0.0008**	0.0541	0.0868	**0.0007**	0.2585	0.3830	**0.0031**
Tai_20×20_8	0.0421	0.0725	**0.0000**	0.0616	0.0655	**0.0002**	0.2999	0.3040	**0.0003**
Tai_20×20_9	**0.0003**	0.0764	0.0275	0.0588	0.0588	**0.0005**	0.2485	0.2851	**0.0019**
Tai_20×20_10	0.1108	0.0526	**0.0000**	0.0688	0.0432	**0.0015**	0.2640	0.1535	**0.0071**
Average	0.0403	0.0792	**0.0031**	0.0650	0.0536	**0.0022**	0.2857	0.2355	**0.0105**

Table 3. 4 The average relative error in C_{max} , MFT and MIT of problem Tai_50

	Makespan			MFT			MIT		
Problem	CDS	NEH	PSO	CDS	NEH	PSO	CDS	NEH	PSO
Tai_50×5_1	**0.0003**	0.1044	0.0288	0.0149	0.0320	**0.0000**	0.3238	0.4924	**0.0000**
Tai_50×5_2	**0.0006**	0.0699	0.0173	0.0242	0.0456	**0.0002**	0.5121	0.6752	**0.0043**
Tai_50×5_3	0.0208	0.1076	**0.0015**	0.0225	0.0065	**0.0001**	0.6046	0.3555	**0.0000**
Tai_50×5_4	0.0174	0.1230	**0.0017**	0.0532	0.0897	**0.0000**	1.4981	0.1268	**0.0519**
Tai_50×5_5	0.0098	0.0903	**0.0042**	0.0441	0.0210	**0.0000**	0.9365	**0.0518**	0.0698
Tai_50×5_6	0.0285	0.0972	**0.0005**	0.0427	**0.0019**	0.0023	0.9723	1.1903	**0.0000**
Tai_50×5_7	0.0094	0.0626	**0.0021**	0.0430	0.0882	**0.0000**	0.6055	1.4488	**0.0000**
Tai_50×5_8	0.0688	0.1347	**0.0000**	0.0785	0.0479	**0.0000**	1.3355	0.7859	**0.0000**
Tai_50×5_9	0.1240	0.0624	**0.0000**	0.1097	**0.0001**	0.0158	2.5634	0.1153	**0.0648**
Tai_50×5_10	0.0013	0.0769	**0.0130**	0.0181	0.0001	**0.0070**	1.0134	1.2503	**0.0000**
Average	0.0281	0.0929	**0.0069**	0.0451	0.0333	**0.0025**	1.0365	0.6492	**0.0191**
Tai_50×10_1	0.0801	0.0923	**0.0000**	0.0602	0.0463	**0.0000**	0.5719	0.5740	**0.0801**
Tai_50×10_2	0.0164	0.0644	**0.0015**	0.0660	0.0264	**0.0004**	0.6506	0.2583	**0.0164**
Tai_50×10_3	0.0313	0.0885	**0.0000**	0.0356	0.0503	**0.0000**	0.3509	0.5253	**0.0313**
Tai_50×10_4	0.0748	0.1291	**0.0000**	0.0952	0.0597	**0.0000**	1.1933	0.7006	**0.0748**
Tai_50×10_5	0.0317	0.1246	**0.0000**	0.0100	0.0438	**0.0015**	0.1101	0.6553	**0.0317**
Tai_50×10_6	**0.0001**	0.0754	0.0242	0.0072	0.0337	**0.0026**	0.0741	0.2831	**0.0001**
Tai_50×10_7	0.0678	0.1042	**0.0000**	0.0474	0.0481	**0.0000**	0.3679	0.4048	**0.0678**
Tai_50×10_8	0.0417	0.0798	**0.0000**	0.0608	0.0117	**0.0011**	0.4762	0.1079	**0.0417**
Tai_50×10_9	0.0428	0.0454	**0.0002**	0.0954	0.0295	**0.0000**	0.8816	0.4151	**0.0428**
Tai_50×10_10	0.0498	0.1492	**0.0000**	0.0350	0.0322	**0.0000**	0.2690	0.1005	**0.0498**
Average	0.0436	0.0953	**0.0026**	0.0513	0.0382	**0.0006**	0.4946	0.4025	**0.0436**
Tai_50×20_1	0.0255	0.0946	**0.0005**	0.0494	0.0494	**0.0000**	0.2822	0.2279	**0.0000**
Tai_50×20_2	0.0305	0.0659	**0.0004**	0.0458	0.0478	**0.0000**	0.2430	0.2715	**0.0006**
Tai_50×20_3	0.0038	0.1261	**0.0062**	0.0222	0.0458	**0.0016**	0.1191	0.2660	**0.0080**
Tai_50×20_4	0.0348	0.0865	**0.0000**	0.0466	0.0321	**0.0001**	0.2420	0.2576	**0.0000**
Tai_50×20_5	0.0141	0.0830	**0.0016**	0.0325	0.0382	**0.0001**	0.1791	0.1837	**0.0007**
Tai_50×20_6	0.0444	0.0617	**0.0000**	0.0590	0.0151	**0.0006**	0.2868	0.0663	**0.0000**
Tai_50×20_7	0.0230	0.0684	**0.0007**	0.0147	0.0621	**0.0006**	0.0803	0.3219	**0.0028**
Tai_50×20_8	0.0522	0.0423	**0.0000**	0.0786	0.0238	**0.0002**	0.3895	0.1730	**0.0000**
Tai_50×20_9	**0.0007**	0.0560	0.0154	0.0679	0.0178	**0.0010**	0.3787	0.1106	**0.0000**
Tai_50×20_10	0.0061	0.0782	**0.0043**	0.0578	0.0451	**0.0001**	0.3079	0.2664	**0.0000**
Average	0.0235	0.0763	**0.0029**	0.0475	0.0377	**0.0004**	0.2508	0.2145	**0.0012**

Table 3. 5 The average relative error in C_{max} , MFT and MIT of problem Tai_100

	Makespan			MFT			MIT		
Problem	CDS	NEH	PSO	CDS	NEH	PSO	CDS	NEH	PSO
Tai_100×5_1	0.0047	0.1072	**0.0031**	0.0307	0.1104	**0.0000**	0.7611	3.3877	**0.0000**
Tai_100×5_2	0.0146	0.1278	**0.0007**	0.0228	0.0726	**0.0000**	1.5465	6.3127	**0.0000**
Tai_100×5_3	0.0212	0.0519	**0.0010**	0.0251	**0.0005**	0.0030	1.0418	3.3001	**0.0000**
Tai_100×5_4	**0.0018**	0.0931	0.0080	0.0017	0.0905	**0.0009**	0.0793	1.6482	**0.0186**
Tai_100×5_5	0.0989	**0.0000**	0.0886	0.1423	**0.0000**	0.1104	0.8203	1.1020	**0.0000**
Tai_100×5_6	**0.0006**	0.0441	0.0108	0.0095	**0.0008**	0.0042	0.2100	2.8709	**0.0175**
Tai_100×5_7	0.0089	0.1683	**0.0012**	0.0213	0.1649	**0.0000**	0.5958	3.9401	**0.0000**
Tai_100×5_8	0.0153	0.0590	**0.0013**	0.0317	0.0196	**0.0000**	2.5700	3.3516	**0.0000**
Tai_100×5_9	0.0124	0.0695	**0.0007**	0.0055	0.0095	**0.0003**	0.2062	0.3928	**0.0660**
Tai_100×5_10	0.0104	0.1217	**0.0013**	0.0262	0.1305	**0.0000**	0.8053	3.2704	**0.0000**
Average	0.0189	0.0843	**0.0117**	0.0317	0.0599	**0.0119**	0.8636	2.9577	**0.0102**
Tai_100×10_1	0.0598	**0.0001**	0.0238	0.0499	0.0625	**0.0000**	0.6604	0.2145	**0.0022**
Tai_100×10_2	0.0379	0.0265	**0.0000**	0.0319	0.0456	**0.0000**	1.3016	**0.0000**	0.6481
Tai_100×10_3	0.1283	**0.0000**	0.0953	0.1286	**0.0000**	0.0993	2.3663	**0.0000**	1.5710
Tai_100×10_4	0.1568	**0.0000**	0.1456	0.0672	**0.0000**	0.0468	1.7334	**0.0000**	1.1999
Tai_100×10_5	0.2324	**0.0000**	0.2222	0.2030	**0.0000**	0.1564	2.2895	**0.0000**	1.1552
Tai_100×10_6	0.0976	**0.0000**	0.0738	0.0497	**0.0000**	0.0272	0.9419	**0.0000**	0.5745
Tai_100×10_7	0.0608	0.0381	**0.0000**	0.0636	0.1082	**0.0000**	1.1585	**0.0000**	0.3347
Tai_100×10_8	0.1080	**0.0000**	0.1040	0.1284	**0.0000**	0.0906	5.3184	**0.0000**	3.5692
Tai_100×10_9	0.0338	**0.0000**	0.0582	0.0928	**0.0000**	0.0753	11.9100	**0.0000**	9.9398
Tai_100×10_10	0.0235	**0.0001**	0.0232	0.0253	0.0821	**0.0000**	0.8511	**0.0000**	0.4474
Average	0.0939	**0.0065**	0.0746	0.0840	**0.0298**	0.0496	2.8531	**0.0214**	1.9442
Tai_100×20_1	0.0488	**0.0001**	0.0232	0.0935	**0.0000**	0.0540	0.3083	**0.0051**	0.0572
Tai_100×20_2	0.0168	0.0591	**0.0003**	0.0190	0.0296	**0.0000**	0.1362	0.1685	**0.0000**
Tai_100×20_3	0.0147	0.0515	**0.0001**	0.0172	0.0284	**0.0001**	0.1139	0.1299	**0.0006**
Tai_100×20_4	0.0359	0.0362	**0.0000**	0.0583	**0.0010**	0.0047	0.3727	0.0143	**0.0196**
Tai_100×20_5	**0.0012**	0.0903	0.0071	0.0079	0.0531	**0.0009**	0.0571	0.1816	**0.0045**
Tai_100×20_6	0.0376	0.0613	**0.0000**	0.0814	0.0042	**0.0017**	0.5054	0.0439	**0.0048**
Tai_100×20_7	0.0295	0.0191	**0.0000**	0.0453	0.0136	**0.0003**	0.2969	0.1160	**0.0008**
Tai_100×20_8	0.0164	0.0539	**0.0002**	0.0339	0.0239	**0.0000**	0.2102	0.1705	**0.0000**
Tai_100×20_9	0.0073	0.0405	**0.0019**	0.0258	0.0128	**0.0001**	0.1775	0.0943	**0.0004**
Tai_100×20_10	0.0244	0.0202	**0.0003**	0.0345	0.0140	**0.0000**	0.2108	0.1138	**0.0000**
Average	0.0233	0.0432	**0.0033**	0.0417	0.0181	**0.0062**	0.2389	0.1038	**0.0088**

Table 3. 6 The average relative error in C_{max} , MFT and MIT of problem Tai_200 and Tai_500

	Makespan			MFT			MIT		
Problem	CDS	NEH	PSO	CDS	NEH	PSO	CDS	NEH	PSO
Tai_200×10_1	0.0100	0.0562	**0.0002**	0.0618	**0.0000**	0.0510	0.1367	0.0743	**0.0011**
Tai_200×10_2	0.0265	0.1715	**0.0000**	0.0142	0.1150	**0.0000**	0.1976	1.5751	**0.0000**
Tai_200×10_3	**0.0026**	0.0536	0.0036	0.0260	**0.0003**	0.0036	0.2743	0.5233	**0.0000**
Tai_200×10_4	**0.0016**	0.1253	0.0051	0.0104	0.1442	**0.0000**	0.1644	1.5938	**0.0002**
Tai_200×10_5	0.0034	0.0953	**0.0024**	0.0054	0.0941	**0.0004**	0.1018	0.7997	**0.0032**
Tai_200×10_6	0.0222	0.1332	**0.0000**	0.0332	0.1435	**0.0000**	0.4983	1.8804	**0.0000**
Tai_200×10_7	0.0378	0.1373	**0.0000**	0.0353	0.1060	**0.0000**	0.6398	0.3489	**0.0000**
Tai_200×10_8	**0.0001**	0.1628	0.0140	0.0135	0.1818	**0.0000**	0.2319	4.0845	**0.0000**
Tai_200×10_9	0.0364	0.1752	**0.0000**	0.0208	0.1410	**0.0000**	0.2960	0.7618	**0.0000**
Tai_200×10_10	0.0205	0.0780	**0.0000**	0.0241	0.0514	**0.0000**	0.3410	1.2791	**0.0000**
Average	0.0161	0.1188	**0.0025**	0.0245	0.0977	**0.0055**	0.2882	1.2921	**0.0005**

Tai_200×10_1	0.0218	0.1219	**0.0001**	0.0312	0.1105	**0.0000**	0.2688	0.6416	**0.0000**
Tai_200×10_2	**0.0022**	0.0880	0.0036	0.0098	0.0869	**0.0000**	0.0800	0.4945	**0.0004**
Tai_200×20_3	0.0047	0.0946	**0.0010**	0.0173	0.0940	**0.0000**	0.1606	0.6374	**0.0000**
Tai_200×20_4	**0.0009**	0.0349	0.0072	0.0107	0.0375	**0.0000**	0.1038	0.6611	**0.0000**
Tai_200×20_5	0.0338	0.0703	**0.0000**	0.0244	0.0871	**0.0000**	0.1953	0.8382	**0.0000**
Tai_200×20_6	0.0177	0.0367	**0.0003**	0.0539	0.0382	**0.0000**	0.4636	0.5675	**0.0000**
Tai_200×20_7	**0.0070**	0.0528	0.0008	0.0293	0.0611	**0.0000**	0.2523	0.5203	**0.0000**
Tai_200×20_8	0.0363	0.1009	**0.0000**	0.0370	0.1098	**0.0000**	0.3028	1.0376	**0.0000**
Tai_200×20_9	0.0351	0.0089	**0.0002**	0.0274	0.0066	**0.0002**	0.2270	0.2775	**0.0000**
Tai_200×20_10	0.0221	0.0804	**0.0000**	0.0276	0.0845	**0.0000**	0.2341	0.6931	**0.0000**
Average	0.0182	0.0689	**0.0013**	0.0269	0.0716	**0.0000**	0.2288	0.6369	**0.0000**

Tai_500×20_1	0.0223	0.0315	**0.0000**	0.0124	0.0305	**0.0000**	0.1482	0.3843	**0.0000**
Tai_500×20_2	0.0329	0.0416	**0.0000**	0.0164	0.0164	**0.0000**	0.1945	0.1943	**0.0000**
Tai_500×20_3	0.0155	0.0204	**0.0001**	0.0189	0.0115	**0.0000**	0.2272	0.1404	**0.0000**
Tai_500×20_4	0.0213	0.0481	**0.0000**	0.0246	0.0350	**0.0000**	0.2966	0.4243	**0.0000**
Tai_500×20_5	0.0059	0.0278	**0.0004**	0.0121	0.0213	**0.0000**	0.1455	0.2538	**0.0000**
Tai_500×20_6	0.0107	0.0339	**0.0000**	0.0113	0.0204	**0.0000**	0.1445	0.2644	**0.0000**
Tai_500×20_7	0.0043	0.0400	**0.0006**	0.0075	0.0285	**0.0000**	0.0919	0.2249	**0.0000**
Tai_500×20_8	0.0313	0.0306	**0.0000**	0.0260	0.0203	**0.0000**	0.3062	0.2444	**0.0000**
Tai_500×20_9	**0.0000**	0.0466	0.0121	**0.0002**	0.0281	0.0056	**0.0025**	0.3506	0.0678
Tai_500×20_10	0.0210	0.0429	**0.0000**	0.0269	0.0398	**0.0000**	0.3195	0.4827	**0.0000**
Average	0.0165	0.0363	**0.0013**	0.0156	0.0252	**0.0006**	0.1877	0.2964	**0.0068**

Table 3. 7 The aggregate performance of problem Tai20x5 to Tai50x20

Problem	Average			Problem	Average		
	CDS	NEH	PSO		CDS	NEH	PSO
Tai_20×5_1	3.9356	3.4314	**0.0323**	Tai_50×5_1	0.6289	0.0288	**0.0000**
Tai_20×5_2	8.7518	0.1787	1.0031	Tai_50×5_2	0.7907	0.0218	**0.0000**
Tai_20×5_3	3.2344	0.2217	**0.1933**	Tai_50×5_3	0.4696	0.0016	**0.0000**
Tai_20×5_4	4.3932	2.4289	**0.0096**	Tai_50×5_4	0.3395	0.0536	**0.0000**
Tai_20×5_5	2.3846	0.6656	**0.0504**	Tai_50×5_5	0.1631	0.0740	**0.0000**
Tai_20×5_6	0.4278	1.0193	**0.0001**	Tai_50×5_6	1.2894	0.0028	**0.0000**
Tai_20×5_7	16.225	1.2512	**0.6700**	Tai_50×5_7	1.5996	0.0021	**0.0000**
Tai_20×5_8	5.9324	3.9531	**0.0517**	Tai_50×5_8	0.9686	0.0000	**0.0000**
Tai_20×5_9	1.5895	1.3699	**0.0328**	Tai_50×5_9	0.1778	0.0806	**0.0000**
Tai_20×5_10	5.2287	5.4511	**0.0039**	Tai_50×5_10	1.3273	0.0200	**0.0000**
Average	5.2103	1.9971	**0.2047**	Average	0.7755	0.0285	**0.0000**
Tai_20×10_1	0.2728	0.4833	**0.0155**	Tai_50×10_1	0.7127	0.0000	**0.0000**
Tai_20×10_2	0.0170	0.8532	**0.1331**	Tai_50×10_2	0.3490	0.0054	**0.0000**
Tai_20×10_3	0.4886	0.4581	**0.0047**	Tai_50×10_3	0.6641	0.0002	**0.0000**
Tai_20×10_4	0.7027	0.5598	**0.0000**	Tai_50×10_4	0.8893	0.0000	**0.0000**
Tai_20×10_5	0.7397	0.6161	**0.0374**	Tai_50×10_5	0.8237	0.0304	**0.0000**
Tai_20×10_6	0.9313	0.4109	**0.0043**	Tai_50×10_6	0.3923	0.0474	**0.0000**
Tai_20×10_7	0.5393	0.1645	**0.1826**	Tai_50×10_7	0.5571	0.0000	**0.0000**
Tai_20×10_8	0.6053	0.5460	**0.0021**	Tai_50×10_8	0.1994	0.0080	**0.0000**
Tai_20×10_9	1.4812	0.4208	**0.0016**	Tai_50×10_9	0.4900	0.0002	**0.0000**
Tai_20×10_10	1.5617	0.8539	**0.0003**	Tai_50×10_10	0.2819	0.0152	**0.0000**
Average	0.7340	0.5367	**0.0382**	Average	0.5359	0.0107	**0.0000**
Tai_20×20_1	0.4160	0.0009	**0.0000**	Tai_50×20_1	0.3719	0.0005	**0.0000**
Tai_20×20_2	0.2208	0.0133	**0.0000**	Tai_50×20_2	0.3853	0.0010	**0.0000**
Tai_20×20_3	0.4482	0.0000	**0.0000**	Tai_50×20_3	0.4378	0.0157	**0.0000**
Tai_20×20_4	0.4649	0.0011	**0.0000**	Tai_50×20_4	0.3763	0.0001	**0.0000**
Tai_20×20_5	0.1179	0.0393	**0.0000**	Tai_50×20_5	0.3049	0.0024	**0.0000**
Tai_20×20_6	0.3131	0.0597	**0.0000**	Tai_50×20_6	0.1431	0.0006	**0.0000**
Tai_20×20_7	0.5909	0.0047	**0.0000**	Tai_50×20_7	0.4523	0.0040	**0.0000**
Tai_20×20_8	0.4420	0.0005	**0.0000**	Tai_50×20_8	0.2391	0.0002	**0.0000**
Tai_20×20_9	0.4203	0.0299	**0.0000**	Tai_50×20_9	0.1844	0.0164	**0.0000**
Tai_20×20_10	0.2493	0.0086	**0.0000**	Tai_50×20_10	0.3897	0.0045	**0.0000**
Average	0.3683	0.0158	**0.0000**	Average	0.3285	0.0045	**0.0000**

Table 3.7(Cont'd) The aggregate performance of problem Tai20×5 to Tai50×20

Problem	Average			Problem	Average		
	CDS	NEH	PSO		CDS	NEH	PSO
Tai_100×5_1	0.7965	3.6053	**0.0031**	Tai_200×10_1	0.2085	0.1305	**0.0524**
Tai_100×5_2	1.5839	6.5131	**0.0007**	Tai_200×10_2	0.2383	1.8617	**0.0000**
Tai_100×5_3	1.0881	3.3525	**0.0040**	Tai_200×10_3	0.3029	0.5773	**0.0073**
Tai_100×5_4	0.0827	1.8318	**0.0275**	Tai_200×10_4	0.1764	1.8634	**0.0053**
Tai_100×5_5	1.0615	1.1020	**0.1991**	Tai_200×10_5	0.1106	0.9891	**0.0060**
Tai_100×5_6	0.2201	2.9159	**0.0325**	Tai_200×10_6	0.5537	2.1572	**0.0000**
Tai_100×5_7	0.6260	4.2733	**0.0012**	Tai_200×10_7	0.7128	0.5923	**0.0000**
Tai_100×5_8	2.6171	3.4302	**0.0013**	Tai_200×10_8	0.2455	4.4290	**0.0140**
Tai_100×5_9	0.2241	0.4718	**0.0670**	Tai_200×10_9	0.3532	1.0780	**0.0000**
Tai_100×5_10	0.8419	3.5225	**0.0013**	Tai_200×10_10	0.3856	1.4085	**0.0000**
Average	0.9142	3.1018	**0.0338**	Average	0.3288	1.5087	**0.0085**
Tai_100×10_1	0.2771	0.0260	**0.0000**	Tai_200×20_1	0.8740	0.0001	**0.0000**
Tai_100×10_2	0.0720	0.6481	**0.0000**	Tai_200×20_2	0.6694	0.0040	**0.0000**
Tai_100×10_3	0.0000	1.7656	**0.0000**	Tai_200×20_3	0.8259	0.0010	**0.0000**
Tai_100×10_4	0.0000	1.3923	**0.0000**	Tai_200×20_4	0.7335	0.0072	**0.0000**
Tai_100×10_5	0.0000	1.5337	**0.0000**	Tai_200×20_5	0.9956	0.0000	**0.0000**
Tai_100×10_6	0.0000	0.6755	**0.0000**	Tai_200×20_6	0.6424	0.0003	**0.0000**
Tai_100×10_7	0.1463	0.3347	**0.0000**	Tai_200×20_7	0.6342	0.0008	**0.0000**
Tai_100×10_8	0.0000	3.7638	**0.0000**	Tai_200×20_8	1.2483	0.0000	**0.0000**
Tai_100×10_9	0.0000	10.0734	**0.0000**	Tai_200×20_9	0.2930	0.0004	**0.0000**
Tai_100×10_10	0.0822	0.4706	**0.0000**	Tai_200×20_10	0.8580	0.0000	**0.0000**
Average	0.0578	2.0684	**0.0000**	Average	0.7774	0.0014	**0.0000**
Tai_100×20_1	0.0051	0.1344	**0.0000**	Tai_500×20_1	0.1830	0.4463	**0.0000**
Tai_100×20_2	0.2572	0.0003	**0.0000**	Tai_500×20_2	0.2439	0.2524	**0.0000**
Tai_100×20_3	0.2098	0.0008	**0.0000**	Tai_500×20_3	0.2616	0.1723	**0.0001**
Tai_100×20_4	0.0515	0.0243	**0.0000**	Tai_500×20_4	0.3426	0.5074	**0.0000**
Tai_100×20_5	0.3250	0.0125	**0.0000**	Tai_500×20_5	0.1635	0.3030	**0.0004**
Tai_100×20_6	0.1093	0.0065	**0.0000**	Tai_500×20_6	0.1665	0.3187	**0.0000**
Tai_100×20_7	0.1487	0.0011	**0.0000**	Tai_500×20_7	0.1037	0.2934	**0.0006**
Tai_100×20_8	0.2482	0.0002	**0.0000**	Tai_500×20_8	0.3635	0.2953	**0.0000**
Tai_100×20_9	0.1475	0.0024	**0.0000**	Tai_500×20_9	0.0026	0.4253	**0.0855**
Tai_100×20_10	0.1480	0.0003	**0.0000**	Tai_500×20_10	0.3673	0.5653	**0.0000**
Average	0.1650	0.0183	**0.0000**	Average	0.2198	0.3579	**0.0086**

Table 3. 8 The number and percentage of problems for different objective with superior results

	Makespan			MFT			MIT		
Problem	CDS	NEH	PSO	CDS	NEH	PSO	CDS	NEH	PSO
Tai_20×5	**7**	0	3	0	0	**10**	0	0	**10**
Tai_20×10	2	0	**8**	1	1	**8**	0	1	**9**
Tai_20×20	**1**	0	9	0	0	**10**	0	0	**10**
Tai_50×5	2	0	**8**	0	2	**8**	0	1	**9**
Tai_50×10	1	0	**9**	0	0	**10**	0	0	**10**
Tai_50×20	1	0	**9**	0	0	**10**	0	0	**10**
Tai_100×5	2	1	**7**	0	3	**7**	0	0	**10**
Tai_100×10	0	**8**	2	0	**6**	4	0	**9**	1
Tai_100×20	1	1	**8**	0	2	**8**	0	1	**9**
Tai_200×10	3	0	**7**	0	2	**8**	0	0	**10**
Tai_200×20	3	0	**7**	0	0	**10**	0	0	**10**
Tai_500×20	1	0	**9**	1	0	**9**	1	0	**9**
Sum	24	10	**86**	2	16	**102**	1	12	**107**
Percentage	20%	8.33%	**71.67%**	1.67%	13.33%	**85%**	0.83%	10%	**89.17%**

Table 3. 9 The number of problems for aggregate objectives with superior results

Problem	Aggregte			Problem	Aggregate		
	CDS	NEH	PSO		CDS	NEH	PSO
Tai_20×5	0	0	**10**	Tai_100×5	0	0	**10**
Tai_20×10	0	0	**10**	Tai_100×10	0	0	**10**
Tai_20×20	0	0	**10**	Tai_100×20	0	0	**10**
Tai_50×5	0	0	**10**	Tai_200×10	0	0	**10**
Tai_50×10	0	0	**10**	Tai_200×20	0	0	**10**
Tai_50×20	0	0	**10**	Tai_500×20	0	0	**10**
Sum	0	0	60	Sum	0	0	60

The proposed PSO algorithm was compared with five heuristic algorithms: CDS (1970), NEH (1983), RAJ (1994), GAN-RAJ (1993) and Laha (2008). We also coded these methods in Visual C++. The CDS heuristic (1970) takes its name from its three authors and is a heuristic generalization of Johnson's algorithm. The process generates a set of $m–1$ artificial two-machine problems, each of which is then solved by Johnson's rule. In this study, we modified the original CDS and compared the makespan, mean flow time, and machine idle time of all *m–1* generated problems. The non-dominated solution was selected to compare with the solutions obtained from our PSO algorithm. The other comparison was based on solutions determined by the NEH algorithm introduced by Nawaz et al. (1983). The NEH investigates $n(n+1)/2$ permutations to find near-optimal solutions. As we did for CDS, we modified the original NEH and compared the three objectives of all $n(n+1)/2$ sequences. We compared the non-dominated solution from these sequences with the solutions from our PSO.

The following two performance measures are used in this study: average-relative percentage deviation (ARPD) and maximum percentage deviation (MPD) where MS stands for makespan, TFT represents total flow time, MIT stands for machine idle time, H is the heuristic.

$$\mathrm{ARPD_{MS}} = \frac{100}{10}\sum_{i=1}^{10}\left(\frac{MS_{H,i} - BestMS_i}{BestMS_i}\right) \tag{3.20}$$

$$\mathrm{MPD_{MS}} = MAX_{i=1..10}\left(\frac{MS_{H,i} - BestMS_i}{BestMS_i}\right)\times 100 \tag{3.21}$$

$$\mathrm{ARPD_{TFT}} = \frac{100}{10}\sum_{i=1}^{10}\left(\frac{TFT_{H,i} - BestTFT_i}{BestTFT_i}\right) \tag{3.22}$$

$$\mathrm{MPD_{TFT}} = MAX_{i=1..10}\left(\frac{TFT_{H,i} - BestTFT_i}{BestTFT_i}\right)\times 100 \tag{3.23}$$

$$\mathrm{ARPD_{MIT}} = \frac{100}{10}\sum_{i=1}^{10}\left(\frac{MIT_{H,i} - BestMIT_i}{BestMIT_i}\right) \tag{3.24}$$

$$\text{MPD}_{\text{MIT}}= MAX_{i=1..10}\left(\frac{MIT_{H,i} - BestMIT_i}{BestMIT_i}\right)\times 100 \tag{3.25}$$

We tested our PSO on nine different problem sizes (n=20, 50, 100 and m=5, 10, 20) from Taillard's (1993) benchmarks. Table 3.10 compares the six methods using the ARPD and MPD. Table 4.10 shows that the proposed PSO outperforms for almost all problem instances in the makespan object. The comparison of TFT object is revealed in Table 3.11. It shows the ARPD and MPD of six heuristics and the Laha's algorithm performs better. We have given the comparison of MIT in Table 3.12 that indicates the proposed PSO can get better solution. At last, we aggregate the results of three objects in order to show the performance of the proposed PSO to solve the multi-objectives problems. We observed that the PSO performed better than other five heuristics. Table 3.13 shows the superior performance of the proposed PSO in terms of the three simultaneous objectives. The computation cost is demonstrated on Table 3.14. The proposed PSO spend more CPU time than other construct heuristic because of the proposed PSO is an evolutionary algorithm.

In addition, we compare TFT of benchmarks by more algorithms --- Liu and Reeves (2001) (LR), Chakravarthy-Rajendran (1999), simulated annealing-bases approach (SA) and Laha and Chakraborty (2008) (H-1 and H-2). The results show in Table 3.15 for ARPD and Table 3.16 for MPD. We can observe that the H-1 and H-2 perform better than other algorithms while only one object TFT is considered.

Table 3. 10 Comparison of makespan(MS) for different heuristics.

Problem size		NEH (1983)		CDS (1970)		RAJ (1994)		GAN-RAJ (1993)		Laha (2008)		PSO	
n	m	ARPD	MPD	ARPD	MPD	ARPD	MPD	ARPD	MPD	ARPD	MPD	ARPD	MPD
20	5	1.84	0.25	0.76	0.15	0.44	0.12	0.63	0.14	1.55	0.21	0.00	0.00
	10	1.78	0.23	0.71	0.12	0.85	0.17	0.83	0.14	1.50	0.20	0.00	0.00
	20	1.27	0.17	0.44	0.06	0.88	0.14	0.82	0.12	1.06	0.15	0.00	0.00
50	5	1.24	0.17	0.83	0.14	0.26	0.05	0.37	0.08	1.29	0.22	0.02	0.02
	10	1.28	0.19	0.59	0.08	0.48	0.09	0.53	0.10	1.29	0.18	0.01	0.01
	20	1.08	0.17	0.07	0.02	0.35	0.07	0.39	0.07	1.02	0.16	0.06	0.03
100	5	1.04	0.19	0.46	0.12	0.36	0.07	0.23	0.07	1.05	0.16	0.07	0.07
	10	0.28	0.06	0.47	0.07	0.29	0.06	0.24	0.04	0.89	0.13	0.01	0.01
	20	0.65	0.11	0.16	0.04	0.21	0.05	0.18	0.04	0.72	0.10	0.01	0.01

(NEH= Nawaz M, Enscore JR, Ham I (1983), CDS= Campbell HG, Dudek RA, Smith ML (1970), RAJ= Rajendran C (1994), GAN-RAJ= Gangadharan R, Rajendran C (1993), Laha= Laha & Chakraborty (2008), PSO= proposed PSO)

Table 3. 11 Comparison of total flow time (TFT) for different heuristics

Problem size		NEH (1983)		CDS (1970)		RAJ (1994)		GAN-RAJ (1993)		Laha (2008)		PSO	
n	m	ARPD	MPD	ARPD	MPD	ARPD	MPD	ARPD	MPD	ARPD	MPD	ARPD	MPD
20	5	0.65	0.17	1.71	0.27	1.70	0.31	1.88	0.34	4.43	0.61	1.28	0.20
	10	0.70	0.10	1.43	0.18	1.29	0.19	1.47	0.23	3.43	0.51	0.95	0.12
	20	0.59	0.14	1.23	0.18	1.27	0.21	1.31	0.24	2.29	0.30	0.82	0.12
50	5	0.11	0.07	2.48	0.56	2.56	0.51	2.58	0.53	5.86	0.94	2.48	0.44
	10	7.87	7.53	11.33	9.62	10.91	9.24	11.27	9.50	14.49	10.87	10.78	9.19
	20	0.39	0.09	1.55	0.20	1.58	0.20	1.60	0.19	3.18	0.40	1.44	0.17
100	5	0.27	0.27	2.24	2.24	3.59	3.59	3.00	3.00	5.56	5.56	2.60	2.60
	10	0.87	0.87	1.86	1.86	1.91	1.91	1.80	1.80	4.02	4.02	1.93	1.93
	20	1.39	1.39	1.65	1.65	1.73	1.73	1.65	1.65	2.83	2.83	1.59	1.59

(NEH= Nawaz M, Enscore JR, Ham I (1983), CDS= Campbell HG, Dudek RA, Smith ML (1970), RAJ= Rajendran C (1994), GAN-RAJ= Gangadharan R, Rajendran C (1993), Laha= Laha & Chakraborty (2008), PSO= proposed PSO)

Table 3. 12 Comparison of machine idle time (MIT) for different heuristics

Problem size		NEH (1983)		CDS (1970)		RAJ (1994)		GAN-RAJ (1993)		Laha (2008)		PSO	
n	m	ARPD	MPD	ARPD	MPD	ARPD	MPD	ARPD	MPD	ARPD	MPD	ARPD	MPD
20	5	4.54	2.94	43.56	20.33	3.20	1.03	5.04	1.38	10.79	4.70	1.50	0.43
	10	3.87	0.83	15.03	1.94	8.07	1.48	7.93	1.42	9.92	1.76	0.00	0.00
	20	11.37	1.55	19.19	2.40	14.88	2.01	14.46	1.85	15.29	2.10	0.00	0.00
50	5	67.77	26.95	208.65	108.95	17.11	11.76	17.08	11.76	52.70	23.48	2.95	2.82
	10	1.92	0.56	10.59	1.74	4.74	0.68	4.91	0.70	6.92	1.24	0.26	0.18
	20	2.26	0.36	8.02	0.97	5.75	0.83	5.80	0.87	7.47	0.96	0.00	0.00
100	5	18.18	4.94	40.24	7.65	4.41	1.40	2.00	0.76	15.47	3.34	3.51	1.69
	10	1.96	0.43	9.54	1.38	1.92	0.38	1.65	0.41	5.47	0.98	0.15	0.09
	20	1.03	0.26	4.26	0.52	2.79	0.40	2.64	0.35	3.77	0.45	0.00	0.00

(NEH= Nawaz M, Enscore JR, Ham I (1983), CDS= Campbell HG, Dudek RA, Smith ML (1970), RAJ= Rajendran C (1994), GAN-RAJ= Gangadharan R, Rajendran C (1993), Laha= Laha & Chakraborty (2008), PSO= proposed PSO)

Table 3. 13 Summation of MS, TFT and MIT for different heuristics

Problem size		NEH (1983)		CDS (1970)		RAJ (1994)		GAN-RAJ (1993)		Laha (2008)		PSO	
n	m	ARPD	MPD	ARPD	MPD	ARPD	MPD	ARPD	MPD	ARPD	MPD	ARPD	MPD
20	5	7.04	3.35	46.03	20.75	5.34	1.46	7.56	1.86	16.77	5.52	2.78	0.63
	10	6.36	1.16	17.18	2.25	10.21	1.83	10.23	1.79	14.85	2.46	0.95	0.12
	20	13.23	1.86	20.86	2.64	17.03	2.36	16.60	2.22	18.63	2.54	0.82	0.12
50	5	69.12	27.19	211.96	109.65	19.93	12.33	20.03	12.37	59.84	24.64	5.45	3.28
	10	11.08	8.28	22.51	11.44	16.13	10.00	16.71	10.30	22.70	12.29	11.04	9.38
	20	3.72	0.62	9.64	1.19	7.68	1.10	7.79	1.13	11.68	1.52	1.50	0.20
100	5	19.49	5.41	42.93	10.01	8.37	5.06	5.23	3.82	22.08	9.06	6.18	4.35
	10	3.11	1.36	11.87	3.32	4.12	2.35	3.69	2.25	10.38	5.13	2.08	2.02
	20	3.08	1.77	6.07	2.21	4.73	2.19	4.47	2.04	7.33	3.38	1.60	1.60

(NEH= Nawaz M, Enscore JR, Ham I (1983), CDS= Campbell HG, Dudek RA, Smith ML (1970), RAJ= Rajendran C (1994), GAN-RAJ= Gangadharan R, Rajendran C (1993), Laha= Laha & Chakraborty (2008), PSO= proposed PSO)

Table 3. 14 Average CPU time (in seconds)

n	m	NEH	CDS	RAJ	GANRAJ	Laha	PSO
20	5	0.0016	0.0031	0.0047	0.0014	0.0012	1.6641
	10	0.0015	0.0093	0.0094	0.0015	0.0015	2.0547
	20	0.0047	0.0109	0.0094	0.0031	0.0047	2.8078
50	5	0.0140	0.0016	0.0156	0.0047	0.0047	4.4906
	10	0.0234	0.0032	0.0297	0.0047	0.0063	5.3047
	20	0.0500	0.0078	0.0539	0.0078	0.0062	7.1593
100	5	0.0860	0.0016	0.0844	0.0047	0.0047	11.9094
	10	0.1750	0.0046	0.1750	0.0047	0.0078	13.4906
	20	0.3750	0.0078	0.3656	0.0079	0.0141	17.0079

(NEH= Nawaz M, Enscore JR, Ham I (1983), CDS= Campbell HG, Dudek RA, Smith ML (1970), RAJ= Rajendran C (1994), GAN-RAJ= Gangadharan R, Rajendran C (1993), Laha= Laha & Chakraborty (2008), PSO= proposed PSO)

Table 3. 15 Comparison of total flow time (TFT) for heuristics in ARPD

n	m	NEH	CDS	RAJ	GANRAJ	Laha	LR	SA	H-1	H-2	PSO
20	5	0.65	1.71	1.70	1.88	4.43	0.24	1.17	0.16	0.20	**1.28**
	10	0.70	1.43	1.29	1.47	3.43	0.09	0.72	0.01	0.01	**0.95**
	20	0.59	1.23	1.27	1.31	2.29	0.15	0.66	0.12	0.07	**0.82**
50	5	0.11	2.48	2.56	2.58	5.86	0.56	1.78	0.55	0.54	**2.48**
	10	7.87	11.33	10.91	11.27	14.49	8.06	1.24	7.97	7.89	**10.78**
	20	0.39	1.55	1.58	1.60	3.18	0.15	1.10	0.08	0.09	**1.44**
100	5	0.27	2.24	3.59	3.00	5.56	0.43	1.59	0.43	0.43	**2.60**
	10	0.87	1.86	1.91	1.80	4.02	0.04	1.24	0.03	0.03	**1.93**
	20	1.39	1.65	1.73	1.65	2.83	0.08	1.13	0.01	0.02	**1.59**

(NEH= Nawaz M, Enscore JR, Ham I (1983), CDS= Campbell HG, Dudek RA, Smith ML (1970), RAJ= Rajendran C (1994), GAN-RAJ= Gangadharan R, Rajendran C (1993), Laha= Laha & Chakraborty (2008), LR= Liu J, Reeves CR (2001), SA= Chakravarthy K, Rajendran C (1999), H-1 and H-2= Laha D, Chakraborty UK (2008),PSO= proposed PSO)

The heuristic TSP-GA algorithm proposed by Ponnambalam(2004) has been chosen to compare the performance of our PSO algorithm. The objectives considered in TSP-GA algorithm are minimization of makespan (C_{max}), minimization of mean flow time (MFT), and minimization of machine idle time (MIT). The best production sequence was chosen for each problem instance. The computational results of twenty-one problem tackled by TSP-GA heuristic are given in Table 3.17 .

Table 3. 16 Comparison of total flow time (TFT) for heuristics in MPD

n	m	NEH	CDS	RAJ	GANRAJ	Laha	LR	SA	H-1	H-2	PSO
20	5	0.17	0.27	0.31	0.34	0.61	0.12	0.21	0.11	0.12	**0.20**
	10	0.10	0.18	0.19	0.23	0.51	0.01	0.12	0.00	0.01	**0.12**
	20	0.14	0.18	0.21	0.24	0.30	0.05	0.12	0.05	0.05	**0.12**
50	5	0.07	0.56	0.51	0.53	0.94	0.25	0.38	0.25	0.25	**0.44**
	10	7.53	9.62	9.24	9.50	10.87	7.92	0.19	7.87	7.82	**9.19**
	20	0.09	0.20	0.20	0.19	0.40	0.04	0.16	0.04	0.04	**0.17**
100	5	0.27	2.24	3.59	3.00	5.56	0.43	1.59	0.43	0.43	**2.60**
	10	0.87	1.86	1.91	1.80	4.02	0.04	1.24	0.03	0.03	**1.93**
	20	1.39	1.65	1.73	1.65	2.83	0.08	1.13	0.01	0.02	**1.59**

(NEH= Nawaz M, Enscore JR, Ham I (1983), CDS= Campbell HG, Dudek RA, Smith ML (1970), RAJ= Rajendran C (1994), GAN-RAJ= Gangadharan R, Rajendran C (1993), Laha= Laha & Chakraborty (2008), LR= Liu J, Reeves CR (2001), SA= Chakravarthy K, Rajendran C (1999), H-1 and H-2= Laha D, Chakraborty UK (2008),PSO= proposed PSO)

Table 3. 17 The results of TSP_GA

Problem instance	Scale N × M*	C_{max}	*MFT*	*MIT*
Car1	11×5	8243	5746	2110
Car2	13×4	8458	5524	586
Car3	12×5	9010	6410	1485
Car4	14×4	8214	5416	1620
Car5	10×6	8633	5980	11666
Car6	8×9	10690	8125	7974
Car7	7×7	6681	5247	3587
Car8	8×8	8816	6605	8492
Hel2	20×10	169	114	143
Rec01	20×5	1505	1010	423
Rec03	20×5	1207	780	267
Rec05	20×5	1391	898	631
Rec07	20×10	1899	1269	3248
Rec09	20×10	1815	1164	3213
Rec11	20×10	1806	1196	2327
Rec13	20×15	2314	1582	5469
Rec15	20×15	2307	1655	4789
Rec17	20×15	2547	1710	7111
Rec19	30×10	2496	1599	2904
Rec21	30×10	2627	1628	3177
Rec23	30×10	2469	1570	3687

*N: number of jobs; M: number of machines

Though, the TSP-GA selected only one manufacturing permutation for each problem, the proposed PSO algorithm can find out a group of Pareto optimal solutions. All the solutions included in the Pareto optimal set are measured with the solution proposed by TSP-GA within each problem scenario. The relative evaluation method of two algorithms is introduced below. The sequence given by the PSO is noted S_{PSO} with makespan, mean flow time, and machine idle time as M_{PSO}, MFT_{PSO}, and MIT_{PSO}, respectively, and the sequence given by TSP-GA is noted S_{TSPGA} with makespan, mean flow time, and machine idle time as M_{TSPGA}, MFT_{TSPGA}, and MIT_{TSPGA}. The relative error in makespan, mean flow time, and machine idle time for schedule S_{PSO} are as follows.

$$\frac{M_{PSO} - \min(M_{PSO}, M_{TSPGA})}{\min(M_{PSO}, M_{TSPGA})} \tag{3.26}$$

$$\frac{MFT_{PSO} - \min(MFT_{PSO}, MFT_{TSPGA})}{\min(MFT_{PSO}, MFT_{TSPGA})} \tag{3.27}$$

$$\frac{MIT_{PSO} - \min(MIT_{PSO}, MIT_{TSPGA})}{\min(MIT_{PSO}, MIT_{TSPGA})} \tag{3.28}$$

Furthermore, the relative error in makespan, mean flow time, and machine idle time for schedule S_{TSPGA} could be derived using the following equations.

$$\frac{M_{TSPGA} - \min(M_{TSPGA}, M_{PSO})}{\min(M_{TSPGA}, M_{PSO})} \tag{3.29}$$

$$\frac{MFT_{TSPGA} - \min(MFT_{TSPGA}, MFT_{PSO})}{\min(MFT_{TSPGA}, MFT_{PSO})} \tag{3.30}$$

$$\frac{MIT_{TSPGA} - \min(MIT_{TSPGA}, MIT_{PSO})}{\min(MIT_{TSPGA}, MIT_{PSO})} \tag{3.31}$$

The average relative error of $C_{\max}$, MFT and MIT are given in Table 3.18. For each problem scenario, we sum up the average relative error of $C_{\max}$, MFT and MIT and also present in Table 3.18.

Table 3. 18 The average relative error of PSO and TSP-GA

Problem instance	Average relative error in C_{max}		Average relative error in MFT		Average relative error in MIT		Sum of relative errors in C_{max}, MFT, MIT	
	PSO	TSP-GA	PSO	TSP-GA	PSO	TSP-GA	PSO	TSP-GA
Car1	**0.0019**	0.0624	**0**	0.1134	**0.1539**	0.1666	**0.1559**	0.3426
Car2	**0.0036**	0.0493	**0.0019**	0.1167	0.8397	**0.1360**	0.8451	**0.3021**
Car3	**0.0003**	0.1116	**0.0011**	0.1706	0.6444	**0.0755**	0.6458	**0.3578**
Car4	0.0617	**0.0006**	0.0299	**0.0101**	**0.0160**	1.0439	**0.1076**	1.0547
Car5	**0.0028**	0.0664	**0.0134**	0.0701	**0**	2.5296	**0.0162**	2.6662
Car6	**0**	0.1774	**0**	0.1472	**0.0218**	0.1502	**0.0218**	0.4749
Car7	0.0563	**0.0013**	0.0356	**0.0165**	**0.0651**	0.8770	**0.1570**	0.8950
Car8	0.0188	**0.0076**	**0.0165**	0.0026	**0.0002**	0.2678	0.9124	**0.2781**
Hel2	**0**	0.0956	**0**	0.1012	**0.0062**	0.2418	**0.0062**	0.4387
Rec01	**0**	0.0649	0.1119	**0**	**0**	4.2732	**0.1119**	4.3382
Rec03	0.0427	**0.0018**	0.2474	**0**	**0.0125**	0.6474	**0.3028**	0.6493
Rec05	**0.0011**	0.0372	0.2210	**0**	**0**	5.4157	**0.2221**	5.4529
Rec07	**0.0004**	0.0655	**0.0001**	0.0634	**0**	1.1898	**0.0006**	1.3187
Rec09	**0.0005**	0.0334	0.0228	**0.0128**	**0**	1.1366	**0.0233**	1.1830
Rec11	**0**	0.0915	**0**	0.0717	**0**	0.7475	**0**	0.9108
Rec13	**0.0005**	0.0379	**0**	0.2253	**0**	0.6488	**0.0005**	0.9121
Rec15	**0**	0.0503	**0**	0.3085	**0**	0.4447	**0**	0.8037
Rec17	**0**	0.1531	**0**	0.3481	**0**	1.3679	**0**	1.8693
Rec19	**0**	0.0274	0.1035	**0**	**0**	0.6042	**0.1035**	0.6316
Rec21	**0**	0.1064	0.0309	**0**	**0**	1.3329	**0.0309**	1.4394
Rec23	**0.0002**	0.0526	0.0992	**0**	**0**	0.6914	**0.0994**	0.7441

CHAPTER 4 PSO for Multi-objective JSSP

4.1 Problem Formulation

A typical job shop scheduling problem could be formulated as follows. There are *n* jobs to be processed through *m* machines. Each job must pass through each machine once and only once. Each job should be processed through the machines in a particular order, and there are no precedence constraints among different job operations. Each machine can process only one job at a time, and it cannot be interrupted. Besides, the operation time is fixed and known in advanced. The most objective of JSSP is to find a schedule to minimize the time required to complete all jobs, that is, makespan (C_{max}). In this study, we attempt to reach the three objectives (makespan, machine idle time and total tardiness) simultaneously. We formulate the object function of job shop scheduling problem as follows.

Makespan, $f_{C\max} = C(\pi_n, m)$ (4.1)

Total tardiness, $f_{total\ tardiness} = \sum_{i=1}^{n} \max[0, L_i]$ (4.2)

Total idle time,

$$f_{total\ idle\ time} = \{C(\pi_1, j-1) + \sum_{i=2}^{n} \{\max\{C(\pi_i, j-1) - C(\pi_{i-1}, j), 0\}\} \mid j = 2 \ldots m\} \tag{4.3}$$

4.2 Particle Position Representation

In the study of job shop scheduling, we randomly generated a group of particles positions whose value represents the associated operation priority. For an *n*-job *m*-machine problem, the position of particle *k* can be represented by an $m \times n$ matrix, i.e.

$$X^k = \begin{bmatrix} x_{11}^k & x_{12}^k & \cdots & x_{1n}^k \\ x_{21}^k & x_{22}^k & \cdots & x_{2n}^k \\ \vdots & \vdots & & \vdots \\ x_{m1}^k & x_{m2}^k & \cdots & x_{mn}^k \end{bmatrix}$$, where x_{ij}^k denotes the priority of operation o_{ij} which means the operation of job *j* that need to be processed on machine *i*. The particle positions are decoded into an active schedule by Giffler and Thompson's(1960) heuristic.

The G&T algorithm is briefly described as follows.

Notation:

(i,j): the operation of job *j* that needs to be processed on machine *i*.

S: the partial schedule that contains scheduled operations.

Ω: the set of schedulable operations.

$s_{(i,j)}$: the earlist time at which operation *(i,j)* belongs to *Ω* could be started.

$p_{(i,j)}$: the processing time of operation *(i,j)*.

$f_{(i,j)}$: the earlist time at which operation *(i,j)* belongs to *Ω* could be finished, $f_{(i,j)} = s_{(i,j)} + p_{(i,j)}$.

G&T algorithm

Step 1: Initialize $S = \phi$; Ω is initialized to contain all operations without predecessors.

Step 2: Determine $f^* = \min_{(i,j)\in\Omega}\{f_{(i,j)}\}$ and the machine m^* on which f^* could be realized.

Step 3: (1) Identify the operation set $(i', j') \in \Omega$ such that (i', j') requires machine m^*, and $s_{(i',j')} < f^*$

(2)Choose (i, j) from the operation set identified in (1) with the largest priority.

(3)Add (i, j) to S.

(4)Assign $s_{(i,j)}$ as the starting time of (i, j).

Step 4: If a complete schedule has been generated, stop. Else, delete (i, j) from Ω and include its immediate successor in Ω, then go to Step 2.

We demonstrated the mechanism of G&T algorithm by the 2×2 example shows on Table 3.1, and the position of particle k is $X^k = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$.

Table 4. 1 An 2×2 example

Jobs	Machine sequence	Processing times
1	1, 2	$p_{(1,2)}$=5; $p_{(2,1)}$=4
2	2, 1	$p_{(2,2)}$=4; $p_{(1,2)}$=3

Initialization

Step 1: $S = \phi$;Ω={(1, 1), (2, 2)}.

Iteration 1

Step 2: $s_{(1,1)}$=0, $s_{(2,2)}$=0, $f_{(1,1)}$=5, $f_{(2,2)}$=4; f^*=min{$f_{(1,1)}$,$f_{(2,2)}$}=4, m^*=2.

Step 3: Identify the operation set {(2, 2)}; choose operation (2, 2), which has the largest priority, and add it into schedule S.

Step 4: Update Ω={(1,1), (1,2)}, go to Step 2.

Iteration 2

Step 2: $s_{(1,1)}$=0, $s_{(1,2)}$=4, $f_{(1,1)}$=5, $f_{(1,2)}$=7; f^*=min{$f_{(1,1)}$,$f_{(1,2)}$}=5, m^*=1.

Step 3: Identify the operation set {(1, 1), (1, 2)}; choose operation (1, 2), which has the largest priority, and add it into schedule S.

Step 4: Update Ω={(1, 1)} , go to Step 2.

Iteration 3

Step 2: $s_{(1,1)}$=7, $f_{(1,1)}$=12; f^*=min{$f_{(1,1)}$}=12, m^*=1.

Step 3: Identify the operation set {(1, 1)}; choose operation (1, 1), which has the largest priority, and add it into schedule *S*.

Step 4: Update Ω={(2, 1)} , go to Step 2.

Iteration 4

Step 2: $s_{(2,1)}$=12, $f_{(2,1)}$=16; f^*=min{$f_{(2,1)}$}=16, m^*=2.

Step 3: Identify the operation set {(2, 1)}; choose operation (2, 1), which has the largest priority, and add it into schedule *S*.

Step 4: A complete schedule has been generated, and then stops.

The proposed PSO differs from the original PSO in the information stored in the *pbest* and *gbest* solution. While the original PSO keeps the best positions found so far, the proposed PSO holds the best schedule generated by G&T algorithm. In the previous example, the schedule S^k rather than the position X^k is retained in the *pbest* and *gbest* solutions, where S^k is $\begin{bmatrix} 2 & 1 \\ 2 & 1 \end{bmatrix}$. Based on the insertion operator the movement of particles is modified in accordance to the representation of particle position.

4.3 Particle Velocity

In the proposed PSO for job shop scheduling, the velocity of operation o_{ij} of particle *k* is denoted by v_{ij}^k, $v_{ij}^k \in \{0,1\}$, where o_{ij} is the operation of job *j* that needs to be processed on machine i. When v_{ij}^k equals 1, it means that operation o_{ij} in the preference list of particle k (the position matrix, X^k) has just been moved to the current location, and we should not move it in this iteration. On the other hand, if operation o_{ij} is moved to a new location in this iteration, we set $v_{ij}^k \leftarrow 1$, indicating that o_{ij} has been moved in this iteration and should not been moved in the next few iterations.

Just as the original PSO is applied to a continuous space, inertia weight w is used to control particle velocities. We randomly update velocities at the beginning of the iteration. For each particle k and operation o_{ij}, if v_{ij}^k equals 1, v_{ij}^k will be set to 0 with probability $(1-w)$. This means that if operation o_{ij} is fixed on the current location in the preference list of particle k, o_{ij} is allowed to move in this iteration with probability $(1-w)$. The newly moved operations will then be fixed for more iteration with larger inertia weight, and fixed for less iterations with smaller inertia weight.

4.4 Particle Movement

The particle movement of job shop scheduling is based on the swap operator proposed by D.Y. Sha et al. (2006).

Notations:

x_i^k is the schedule list of machine i of particle k.

$pbest_i^k$ is the schedule list of machine i of k-th *pbest* solution.

$gbest_i$ is the schedule list of machine i of *gbest* solution.

c_1 and c_2 are constant between 0 and 1, $c_1 + c_2 \le 1$.

The swap procedure is accounted as below.

Step 1: Randomly choose a position ζ from x_i^k.

Step 2: Mark the job on position ζ of x_i^k by Λ_1.

Step 3: If the random number *rand* < c_1 then seek the position of Λ_1 in $pbest_i^k$, otherwise seek the position of Λ_1 in $gbest_i$. Denote the position that has been found in $pbest_i^k$ or $gbest_i$ by ζ′, and job in position ζ′ of x_i^k by Λ_2.

Step 4: If Λ_2 has been denoted, $v_{iJ_1}^k = 0$ and $v_{iJ_2}^k = 0$, then swap Λ_1 and Λ_2 in x_i^k, $v_{iJ_1}^k \leftarrow 1$.

Step 5: If all the position of x_i^k have been considered, then stop. Otherwise, if ζ < *n*, then ζ←ζ+1, else ζ←1, go to Step 2.

We take a 6-job problem for example where x_i^k=[4 2 1 3 6 5], $pbest_i^k$=[1 5 4 2 6 3], $gbest_i$=[3 2 6 4 5 1], v_i^k=[0 0 1 0 0 0], c_1=0.6 and c_2=0.2.

Step 1: The position of x_i^k is randomly chose, ζ=3.

Step 2: The job in the 3rd position of x_i^k is job 1, namely Λ_1=1.

Step 3: A random number *rand* is generated, say *rand*=0.7. Since *rand* > c_1, we compare each position of $gbest_i$ with Λ_1 and the matched position ζ′=6. The job in the 6th position of x_i^k is job 5, namely Λ_2=5.

Step 4: Since $v_{i4}^k = 0$ and $v_{i5}^k = 0$, swap job 1 and job 5 in x_i^k, then x_i^k=[4 2 5 3 6 1], and let $v_{i4}^k \leftarrow 1$ then v_i^k=[0 0 1 1 0 0].

Step 5: Let $\zeta \leftarrow 4$ and go to Step2. Repeat the process until all positions of x_i^k have been considered.

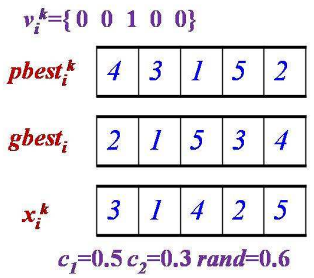

Figure 4. 1 Example of JSSP

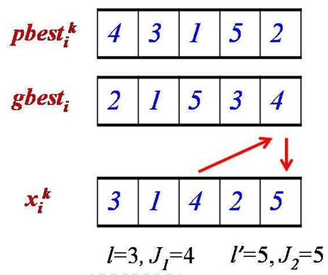

Figure 4. 2 Finding the location to exchange

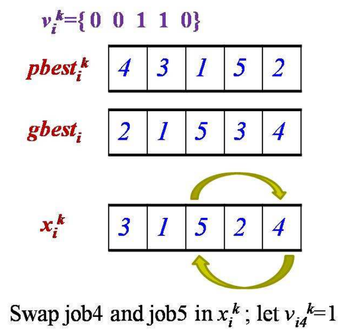

Figure 4. 3 Exchange operation of PSO

4.5 Diversification strategy

If all the particles have the same non-dominated solutions, they will be trapped in local optima. To prevent this from happening, a diversification strategy is proposed to keep the non-dominated solutions different. Once any new solution is generated by particles, the non-dominating solution set will be updated in these three situations:

(1)If the solution of the particle dominates the *gbest* solution, assign the particle solution to the *gbest*.

(2)If the solution of the particle equals to any solution in the non-dominated solution set, replace the non-dominated solution with the particle solution.

If the solution of the particle is dominated by the worst solution and not equal to any non-dominated solution, set the worst solution equal to the particle solution.

4.6 Computational Results

The proposed multi-objective PSO (MOPSO) algorithm was tested on benchmark problems obtained from the OR-Library. The program was coded in Visual C++ and run 40 times on each problem on a Pentium 4 3.0-GHz computer with 1 GB of RAM running Windows XP.

The Taguchi methods employ the loss function for measuring product or process quality as well as for determining manufacturer's tolerance limits (Taguchi 1986). Basically, the objective is to improve product or process quality by reducing the mean squared deviation. Taguchi also proposes signal-to-noise (S/N) ratio to the nominal-the-best (NTB), the smaller-the-better (STB), and the larger-the-better (LTB) problems, which are used when quality characteristics are static, to evaluate the robustness of a system performance. In this study, we focus on the minimization of the objective function with the STB characteristic. Therefore, the definition of the S/N ratio is as follow.

$$S/N = -10 \cdot \mathrm{Log}\left(\frac{1}{\mathrm{n}}\sum_{i=1}^{n} y_i^2\right) \tag{4.4}$$

where n denotes the number of repetition, y_i represents the experimental data.

The parameter of PSO includes weight, learning factors (c_1, c_2), swarm size and iteration numbers. This study considers four factors with four levels each. The parameter settings of four factors are as Table 4.2. We choose the orthogonal array L_{16} to execute the experiments.

Table 4. 2 The parameter of PSO

Factors	Level			
	1	2	3	4
A (w)	0.1	0.3	0.6	0.9
B(c_1, c_2)	0.1, 0.9	0.3, 0.7	0.5, 0.5	0.7, 0.3
C(Swarm size)	60	80	100	120
D(Iteration)	50	100	150	200

According to the L_{16}，the experimental data and S/N ratio of 15×15 problems are given in Table 4.3. According to Table 4.3, the factors response of S/N ratio is showed in Table 4.4. The factors response diagram of S/N ratio shows as Figure4.4. Table 4.5 shows the best level of factors.

Table 4. 3 The L_{16} orthogonal array and S/N ration of 15×15 problem

No. of Experiment	Level of Factors				S/N ratio
	A(w)	B(c_1, c_2)	C(Swarm size)	D(Iteration)	
1	1	1	1	1	-79.4583
2	1	2	2	2	-79.1078
3	1	3	3	3	-78.8201
4	1	4	4	4	-78.9164
5	2	1	2	3	-78.8372
6	2	2	1	4	-79.1162
7	2	3	4	1	-79.4006
8	2	4	3	2	-79.1026
9	3	1	3	4	-78.9947
10	3	2	4	3	-79.2078
11	3	3	1	2	-79.5239
12	3	4	2	1	-80.0762
13	4	1	4	2	-80.3084
14	4	2	3	1	-80.7562
15	4	3	2	4	-80.1928
16	4	4	1	3	-80.4881

Table 4. 4 The factors response of 15×15 problem

Level	Factors			
	A(w)	B(c_1, c_2)	C(Swarm size)	D(Iteration)
1	**-79.0826**	**-79.4384**	-79.6775	-79.9582
2	-79.1187	-79.6066	-79.5934	-79.5393
3	-79.4700	-79.5118	-79.4926	-79.3945
4	-80.4416	-79.6955	**-79.4904**	**-79.3371**

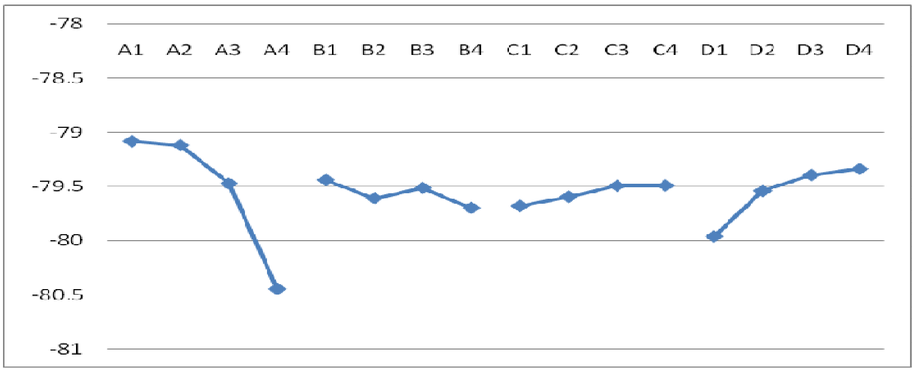

Figure 4. 4 The factor response diagram of S/N ratio diagram of 15x15 problem

Table 4. 5 The best level of factors of 15x15 problem

Level	Factors			
	w	c_1, c_2	Swarm size	Iteration
	1	1	4	4
	0.1	0.1, 0.9	120	200

According to the L_{16}，the experimental data and S/N ratio of 20x15 problems are given in Table 4.6. According to Table 4.6, the factors response of S/N ratio is showed in Table 4.7. The factors response diagram of S/N ratio shows as Figure4.5. Table 4.8 shows the best level of factors.

Table 4. 6 The L_{16} orthogonal array and S/N ration of 20x15 problem

No. of Experiment	Level of Factors				S/N ratio
	w	c_1, c_2	Swarm size	Iteration	
1	1	1	1	1	-79.6904
2	1	2	2	2	-79.3892
3	1	3	3	3	-79.5379
4	1	4	4	4	-79.1821
5	2	1	2	3	-79.4004
6	2	2	1	4	-79.5285
7	2	3	4	1	-80.5712
8	2	4	3	2	-79.9816
9	3	1	3	4	-79.6734
10	3	2	4	3	-79.8359
11	3	3	1	2	-80.3399
12	3	4	2	1	-81.1149
13	4	1	4	2	-81.2241
14	4	2	3	1	-81.5864
15	4	3	2	4	-81.0604
16	4	4	1	3	-81.2293

Table 4. 7 The factors response of 20x15 problem

Level	Factors			
	w	c_1, c_2	Swarm size	Iteration
1	**-79.4540**	**-80.0597**	-80.2499	-80.7967
2	-79.8950	-80.1805	-80.3232	-80.2853
3	-80.2782	-80.4116	-80.2769	-80.0649
4	-81.2794	-80.4571	**-80.2709**	**-79.9229**

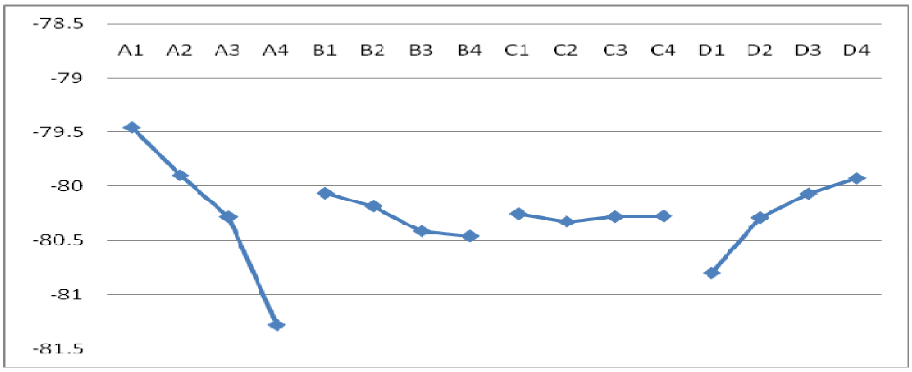

Figure 4. 5 The factor response diagram of S/N ratio diagram of 20x15 problem

Table 4. 8 The best level of factors of 20x15 problem

Level	Factors			
	w	c_1, c_2	Swarm size	Iteration
	1	1	4	4
	0.1	0.1, 0.9	120	200

According to the L_{16}, the experimental data and S/N ratio of 20x20 problems are given in Table 4.9. According to Table 4.9, the factors response of S/N ratio is showed in Table 4.10. The factors response diagram of S/N ratio shows as Figure4.6. Table 4.11 shows the best level of factors of 20x20 problem.

Table 4. 9 The L_{16} orthogonal array and S/N ration of 20x20 problem

No. of Experiment	Level of Factors				S/N ratio
	w	c_1, c_2	Swarm size	Iteration	
1	1	1	1	1	-85.2975
2	1	2	2	2	-85.0583
3	1	3	3	3	-85.0206
4	1	4	4	4	-84.6681
5	2	1	2	3	-84.8432
6	2	2	1	4	-84.8625
7	2	3	4	1	-85.7203
8	2	4	3	2	-85.128
9	3	1	3	4	-84.9865
10	3	2	4	3	-85.2221
11	3	3	1	2	-85.4592
12	3	4	2	1	-86.1118
13	4	1	4	2	-86.2683
14	4	2	3	1	-86.5763
15	4	3	2	4	-86.174
16	4	4	1	3	-86.1046

Table 4. 10 The factors response of 20x20 problem

Level	Factors			
	w	c_1, c_2	Swarm size	Iteration
1	**-85.0169**	**-85.3857**	-85.4541	-85.9522
2	-85.1533	-85.4848	-85.5882	-85.5058
3	-85.4656	-85.6137	**-85.4816**	-85.3256
4	-86.2846	-85.5477	-85.5100	**-85.2145**

Figure 4. 6 The factor response diagram of S/N ratio diagram of 20x20 problem

Table 4. 11 The best level of factors of 20x20 problem

Level	Factors			
	w	c_1, c_2	Swarm size	Iteration
	1	1	3	4
	0.1	0.1, 0.9	100	200

According to the L_{16}, the experimental data and S/N ratio of 30×15 problems are given in Table 4.12. According to Table 4.12, the factors response of S/N ratio is showed in Table 4.13. The factors response diagram of S/N ratio shows as Figure4.7. Table 4.14 shows the best level of factors of 30×15 problem.

Table 4. 12 The L_{16} orthogonal array and S/N ration of 30×15 problem

No. of Experiment	Level of Factors				S/N ratio
	w	c_1, c_2	Swarm size	Iteration	
1	1	1	1	1	-80.9651
2	1	2	2	2	-80.6913
3	1	3	3	3	-80.4876
4	1	4	4	4	-80.2728
5	2	1	2	3	-80.2116
6	2	2	1	4	-80.4169
7	2	3	4	1	-82.1322
8	2	4	3	2	-81.3419
9	3	1	3	4	-80.5676
10	3	2	4	3	-81.1456
11	3	3	1	2	-81.9082
12	3	4	2	1	-82.4581
13	4	1	4	2	-82.2892
14	4	2	3	1	-82.5693
15	4	3	2	4	-82.2314
16	4	4	1	3	-82.3951

Table 4. 13 The factors response of 30×15 problem

Level	Factors			
	w	c_1, c_2	Swarm size	Iteration
1	**-80.6117**	**-81.0826**	-81.4900	-82.0755
2	-81.0946	-81.2888	-81.5040	-81.5989
3	-81.5794	-81.7436	**-81.3253**	-81.1448
4	-82.3731	-81.7055	-81.5340	**-80.9489**

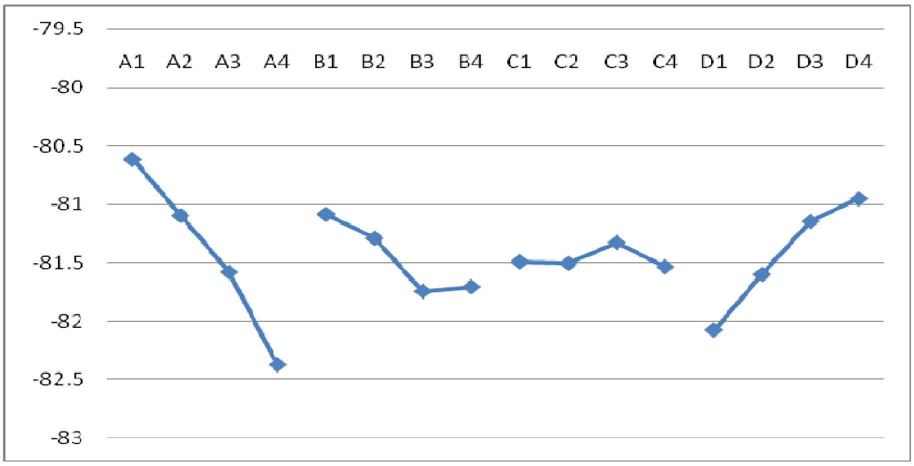

Figure 4. 7 The factor response diagram of S/N ratio diagram of 30x15 problem

Table 4. 14 The best level of factors of 30x15 problem

Level	Factors			
	w	c_1, c_2	Swarm size	Iteration
	1	1	3	4
	0.1	0.1, 0.9	100	200

According to the L_{16}, the experimental data and S/N ratio of 30x20 problems are given in Table 4.15. According to Table 4.15, the factors response of S/N ratio is showed in Table 4.16. The factors response diagram of S/N ratio shows as Figure4.8. Table 4.17 shows the best level of factors of 30x20 problem.

Table 4. 15 The L_{16} orthogonal array and S/N ration of 30x20 problem

No. of Experiment	Level of Factors				S/N ratio
	w	c_1, c_2	Swarm size	Iteration	
1	1	1	1	1	-85.9203
2	1	2	2	2	-85.2966
3	1	3	3	3	-85.3185
4	1	4	4	4	-85.0323
5	2	1	2	3	-85.1811
6	2	2	1	4	-85.2356
7	2	3	4	1	-86.7576
8	2	4	3	2	-86.0746
9	3	1	3	4	-85.5114
10	3	2	4	3	-85.8642
11	3	3	1	2	-86.4892
12	3	4	2	1	-86.9177
13	4	1	4	2	-86.962
14	4	2	3	1	-87.3283
15	4	3	2	4	-86.8707
16	4	4	1	3	-86.93

Table 4. 16 The factors response of 30×20 problem

Level	Factors			
	w	c_1, c_2	Swarm size	Iteration
1	**-85.4043**	**-85.9470**	-86.1896	-86.7605
2	-85.8617	-86.0174	-86.1451	-86.2479
3	-86.2298	-86.4007	**-86.1320**	-85.8799
4	-87.0265	-86.3056	-86.2201	**-85.7249**

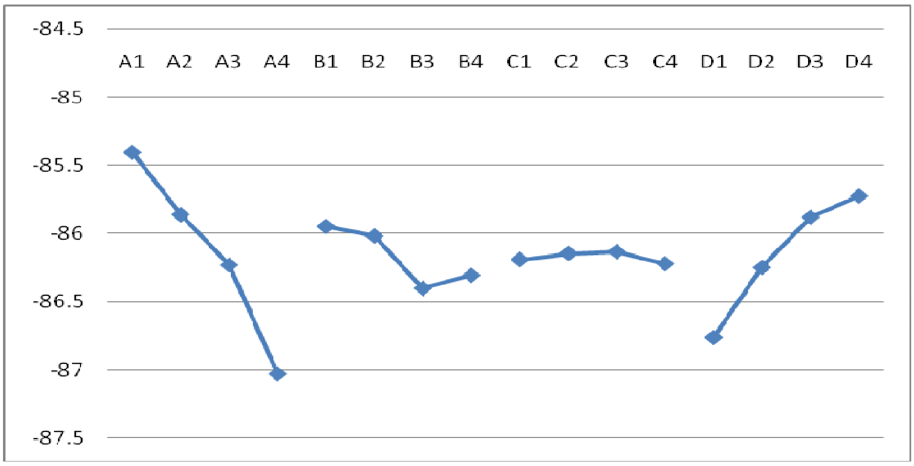

Figure 4. 8 The factor response diagram of S/N ratio diagram of 30×20 problem

Table 4. 17 The best level of factors of 30×20 problem

Level	Factors			
	w	c_1, c_2	Swarm size	Iteration
	1	1	3	4
	0.1	0.1, 0.9	100	200

According to the L_{16} , the experimental data and S/N ratio of 50×15 problems are given in Table 4.18. According to Table 4.18, the factors response of S/N ratio is showed in Table 4.19. The factors response diagram of S/N ratio shows as Figure4.9. Table 4.20 shows the best level of factors of 50×15 problem.

Table 4. 18 The L_{16} orthogonal array and S/N ration of 50×15 problem

No. of Experiment	Level of Factors				S/N ratio
	w	c_1, c_2	Swarm size	Iteration	
1	1	1	1	1	-82.4901
2	1	2	2	2	-82.8781
3	1	3	3	3	-83.4561
4	1	4	4	4	-82.5799
5	2	1	2	3	-82.0682
6	2	2	1	4	-82.4229
7	2	3	4	1	-84.2093
8	2	4	3	2	-83.2563
9	3	1	3	4	-82.5724
10	3	2	4	3	-83.2699
11	3	3	1	2	-83.8847
12	3	4	2	1	-84.0546
13	4	1	4	2	-83.8385
14	4	2	3	1	-84.3872
15	4	3	2	4	-83.8774
16	4	4	1	3	-83.9298

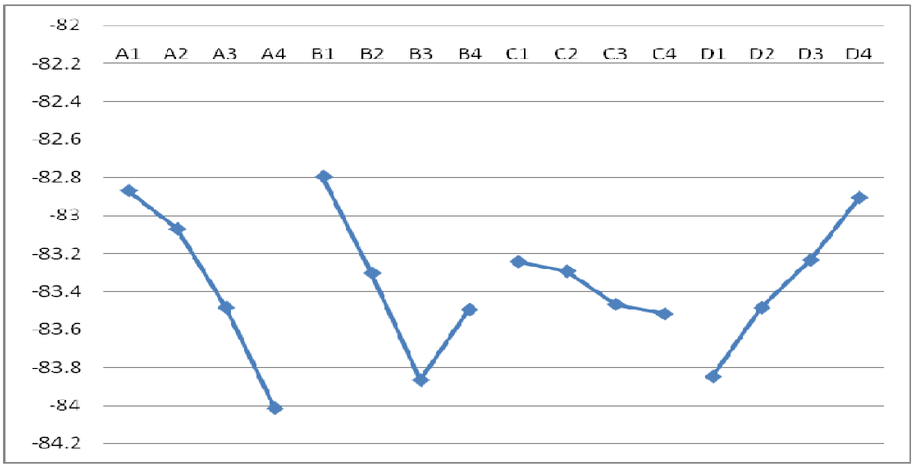

Figure 4. 9 The factor response diagram of S/N ratio diagram of 50×15 problem

Table 4. 19 The factors response of 50×15 problem

Level	Factors			
	w	c_1, c_2	Swarm size	Iteration
1	**-82.8678**	**-82.7947**	**-83.2423**	-83.8471
2	-83.0693	-83.3023	-83.2918	-83.4845
3	-83.4836	-83.8650	-83.4670	-83.2330
4	-84.0140	-83.4943	-83.5173	**-82.9050**

Table 4. 20 The best level of factors of 50×15 problem

Level	Factors			
	w	c_1, c_2	Swarm size	Iteration
	1	1	1	4
	0.1	0.1, 0.9	60	200

According to the L_{16}，the experimental data and S/N ratio of 50×20 problems are given in Table 4.21. According to Table 4.21, the factors response of S/N ratio is showed in Table 4.22. The factors response diagram of S/N ratio shows as Figure4.10. Table 4.23 shows the best level of factors of 50×15 problem.

Table 4. 21 The L_{16} orthogonal array and S/N ration of 50×20 problem

No. of Experiment	Level of Factors				S/N ratio
	w	c_1, c_2	Swarm size	Iteration	
1	1	1	1	1	-87.1114
2	1	2	2	2	-87.0564
3	1	3	3	3	-87.589
4	1	4	4	4	-86.2943
5	2	1	2	3	-86.4336
6	2	2	1	4	-86.4534
7	2	3	4	1	-88.1888
8	2	4	3	2	-87.3232
9	3	1	3	4	-86.5846
10	3	2	4	3	-87.2789
11	3	3	1	2	-87.9287
12	3	4	2	1	-88.1861
13	4	1	4	2	-87.9479
14	4	2	3	1	-88.349
15	4	3	2	4	-87.895
16	4	4	1	3	-88.0081

Table 4. 22 The factors response of 50×20 problem

Level	Factors			
	w	c_1, c_2	Swarm size	Iteration
1	**-87.0371**	**-87.0608**	**-87.4212**	-87.9857
2	-87.1615	-87.3397	-87.4470	-87.5811
3	-87.5381	-87.9055	-87.5072	-87.3649
4	-88.0537	-87.5140	-87.4874	**-86.8558**

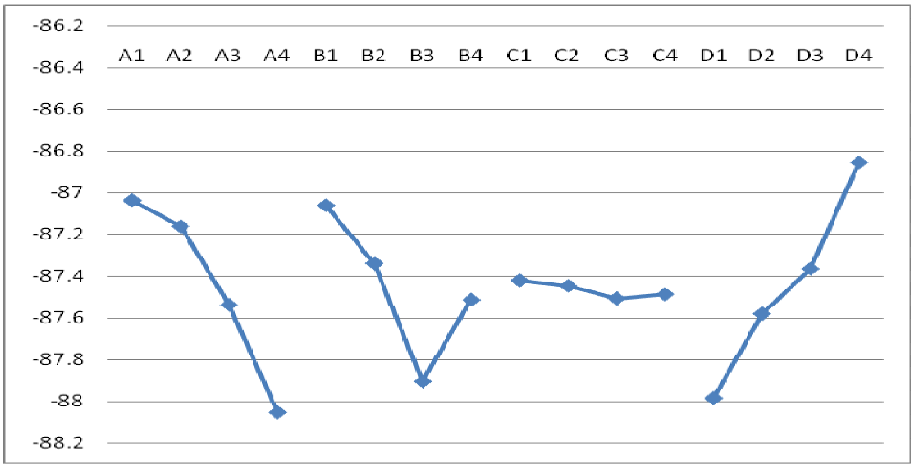

Figure 4. 10 The factor response diagram of S/N ratio diagram of 50×20 problem

Table 4. 23 The best level of factors of 50×20 problem

Level	Factors			
	w	c_1, c_2	Swarm size	Iteration
	1	1	1	4
	0.1	0.1, 0.9	60	200

The results of the experiments showed that the best level of parameter w, c_1, c_2 and iteration numbers are the same even in different scale of problems. The inertia weight w is 0.1 which means the particles prefer to move slowly in the searching progress. The possibilities of moving back to original position are existed. However, the learning factors c_1 and c_2 are fixed 0.1 and 0.9, no matter the scale of the problems are changed. We can say that the particles are intended to learn from the global solution more than the local best solution. That is the particles learning more from swarm experience than individual experience.

During the pilot experiment, we used four swarm sizes N (60, 80, 100, and 120) to test the algorithm. The outcome of N=120 was best, so that value was used in all further tests. Parameters c_1 and c_2 were tested at various values in the range 0.1–0.7 in increments of 0.2. The inertial weight w was reduced from w_{max} to w_{min} during iterations, where w_{max} was set to 0.5, 0.7, and 0.8, and w_{min} was set to 0.1, 0.3, and 0.5.

The combination of c_1=0.1, c_2=0.8, w_{max}=0.5 and w_{min}=0.1 gave the best results. The maximum iteration limit was set to 200 and the maximum archive size was set to 120.

The MOGA proposed by Ponnambalam et al. (2001) was chosen as a baseline against which to compare the performance of our PSO algorithm. The objectives considered in the MOGA algorithm are minimization of makespan, minimization of total tardiness, and minimization of machine idle time. The MOGA methodology is based on the machine-wise priority dispatching rule (pdr) and the G&T procedure (1960). The each gene represents a pdr code. The G&T procedure was used to generate an active feasible schedule. The MOGA fitness function is the weighted sum of makespan, total tardiness, and total idle time of machines with random weights.

The computation results showed that the relative error of the solution for C_{max} and total idle time determined by the proposed MOPSO was better in 23 out of 23 problems than the MOGA. In 22 of the 23 problems, the proposed PSO performed better for the solution considering total tardiness. Overall, the proposed MOPSO was superior to the MOGA in solving the JSP with multiple objectives.

Table 4. 24 Comparison of MOGA and MOPSO for Makespan

Benchmark	n	m	Makespan (MOGA)	Makespan (MOPSO)	% Deviation
abz5	10	10	1587	1338	0
abz6	10	10	1369	1046	0
ft06	6	6	76	56	0
ft10	10	10	1496	1045	0
la01	10	5	1256	709	0
la02	10	5	1066	713	0
la03	10	5	821	671	0
la04	10	5	861	631	0
la05	10	5	893	593	0
la16	10	10	1452	1040	0
la17	10	10	1172	889	0
la19	10	10	1251	938	0
la20	10	10	1419	985	0
orb01	10	10	1704	1181	0
orb02	10	10	1284	1029	0
orb03	10	10	1643	1114	0
orb04	10	10	1543	1122	0
orb05	10	10	1323	1013	0
orb06	10	10	1645	1144	0
orb07	10	10	583	302	0
orb08	10	10	1340	1000	0
orb09	10	10	1462	1044	0
orb10	10	10	1382	1077	0

Table 4. 25 Comparison of MOGA and MOPSO for Total idle time

Benchmark	n	m	Total idle time(MOGA)	Total idle time(MOPSO)	% Deviation
abz5	10	10	8097	3978	0
abz6	10	10	7744	2937	0
ft06	6	6	259	100	0
ft10	10	10	9851	1999	0
la01	10	5	3431	571	0
la02	10	5	2687	573	0
la03	10	5	1722	633	0
la04	10	5	1798	557	0
la05	10	5	2182	473	0
la16	10	10	9169	2718	0
la17	10	10	7044	3365	0
la19	10	10	7164	2796	0
la20	10	10	8745	2883	0
orb01	10	10	11631	3909	0
orb02	10	10	7585	3539	0
orb03	10	10	11138	3788	0
orb04	10	10	9802	3921	0
orb05	10	10	8322	3727	0
orb06	10	10	10836	3478	0
orb07	10	10	3423	1381	0
orb08	10	10	8840	3542	0
orb09	10	10	9439	4224	0
orb10	10	10	8271	4177	0

Table 4. 26 Comparison of MOGA and MOPSO for Total tardiness

Benchmark	N	m	Total tardiness (MOGA)	Total tardiness (MOPSO)	% Deviation
abz5	10	10	1948	611	0
abz6	10	10	1882	339	0
ft06	6	6	31	3	0
ft10	10	10	3459	1534	0
la01	10	5	3324	721	0
la02	10	5	2081	425	0
la03	10	5	1926	373	0
la04	10	5	3194	673	0
la05	10	5	1716	736	0
la16	10	10	1127	1417	0
la17	10	10	1779	53	0
la19	10	10	1581	733	0
la20	10	10	1451	407	0
orb01	10	10	3052	191	0
orb02	10	10	1565	137	0
orb03	10	10	4140	247	0
orb04	10	10	4951	221	0
orb05	10	10	2195	30	0
orb06	10	10	2601	0	0
orb07	10	10	699	0	0
orb08	10	10	3498	253	0
orb09	10	10	2029	0	0
orb10	10	10	1806	0	0

Table 4. 27 Comparison of MOGA and MOPSO with three objectives

Problem	n	m	Makespan				Total machine idle time				Total tardiness			
			MOGA	MOPSO			MOGA	MOPSO			MOGA	MOPSO		
			best	best	average	worst	best	best	average	worst	best	best	average	worst
abz5	10	10	1587	1399	1460	1521	8097	3911	4429.6	5441	1948	90	372.2	725
abz6	10	10	1369	1049	1102.6	1162	7744	2868	3203.1	3875	1882	90	232.85	385
ft10	10	10	1496	1055	1123.6	1166	9851	1630	2204.45	2762	3459	848	1231.95	1663
la16	10	10	1452	1015	1077	1152	9169	2740	3157.65	3679	1127	340	517.95	813
la17	10	10	1172	840	898.6	976	7044	2643	2997.8	3279	1779	277	392.9	552
la19	10	10	1251	923	998.15	1047	7164	2288	3023.2	3476	1581	49	264.45	567
la20	10	10	1419	980	1051.65	1123	8745	2758	3246.7	3779	1451	204	357.1	439
orb01	10	10	1704	1234	1274	1377	11631	3700	4125.8	4812	3052	769	1098.85	1629
orb02	10	10	1284	999	1066	1135	7585	3352	3768.8	4561	1565	64	249.65	436
orb03	10	10	1643	1165	1256.05	1354	11138	3620	4277.85	4839	4140	571	1071.45	1552
orb04	10	10	1543	1134	1208.7	1327	9802	3682	4451.6	5482	4951	443	809.4	1267
orb05	10	10	1323	1009	1066.1	1118	8322	3328	3923.05	4253	2195	136	413.25	697
orb06	10	10	1645	1124	1211.75	1272	10836	3192	3718.8	4177	2601	558	914.15	1390
orb07	10	10	583	271	290.45	318	3423	233	344.15	580	699	63	82.9	112
orb08	10	10	1340	976	1067.3	1123	8840	3349	3810.55	4202	3498	745	1026.15	1365
orb09	10	10	1462	1024	1106.65	1196	9439	3762	4279.3	4658	2029	445	642.1	765
orb10	10	10	1382	1123	1172.65	1243	8271	3863	4531.35	4954	1806	45	479.5	774
la01	10	5	1256	715	770.55	819	3431	479	661.3	1032	3324	453	599.75	861
la02	10	5	1066	713	758.45	804	2687	411	549.5	688	2081	296	447.75	706
la03	10	5	821	663	703.55	757	1722	648	776.55	902	1926	381	684.5	926
la04	10	5	861	601	669.85	720	1798	345	582.55	727	3194	389	563.45	768
la05	10	5	893	593	609.55	669	2182	390	517.65	665	1716	477	630.55	900
ft06	6	6	76	58	60.65	68	259	93	118.7	163	31	0	0.75	9

Table 4. 28 The results of solving FT, ABZ, ORB and YN with MOPSO

Problem	n	m	Makespan			MFT			MIT		
			best	average	worst	best	average	worst	best	average	worst
ft06	6	6	55	55.24	57	49	50.29	51	54	60.90	90
ft10	10	10	973	997.48	1033	852	885.62	938	1116	1707.24	2131
ft20	20	5	1247	1280.19	1315	883	951.00	1032	166	361.90	551
abz5	10	10	1249	1276.62	1329	1134	1173.86	1236	3124	3531.10	4223
abz6	10	10	948	971.24	996	889	910.24	933	2370	2688.14	3069
abz7	20	15	779	791.00	814	676	693.10	714	3132	3452.52	3665
abz8	20	15	776	803.81	835	681	708.52	732	3193	3551.24	3973
abz9	20	15	786	823.19	843	667	696.71	739	3225	3660.62	4321
orb01	10	10	1093	1136.95	1185	992	1026.43	1076	1286	1780.62	2677
orb02	10	10	921	939.24	967	867	897.19	925	2185	2600.33	2925
orb03	10	10	1064	1101.05	1148	962	1015.52	1072	1186	1627.24	2366
orb04	10	10	1031	1070.95	1106	994	1029.81	1079	2179	2645.14	3342
orb05	10	10	896	946.81	1003	828	870.86	915	2331	2734.57	3351
orb06	10	10	1028	1071.76	1135	955	985.24	1068	1439	1666.90	1980
orb07	10	10	403	420.71	438	381	402.24	425	919	1094.00	1195
orb08	10	10	937	957.90	1025	882	904.48	946	1123	1606.57	1990
orb09	10	10	958	981.10	1028	903	942.90	1004	2291	2699.14	3127
orb10	10	10	967	1023.00	1065	944	991.67	1029	2606	2927.95	3338
yn1	20	20	999	1030.52	1058	889	908.38	931	6481	7002.14	7456
yn2	20	20	1043	1073.52	1127	940	966.62	1003	7116	7598.67	8363
yn3	20	20	1021	1044.00	1072	912	938.29	961	6640	7039.33	7880
yn4	20	20	1108	1141.86	1160	973	1005.29	1033	7223	7752.76	8387

Table 4. 29 The results of solving LA with MOPSO

Problem	n	m	Makespan			MFT			MIT		
			best	average	worst	best	average	worst	best	average	worst
la01	10	5	666	666.10	668	561	584.90	604	242	336.81	435
la02	10	5	665	682.19	706	525	560.29	591	223	401.57	548
la03	10	5	608	626.86	657	508	540.62	594	431	518.24	579
la04	10	5	593	605.48	617	516	537.43	571	209	314.48	426
la05	10	5	593	593.00	593	483	517.05	559	422	492.67	652
la06	15	5	926	926.00	926	762	789.57	832	393	488.86	584
la07	15	5	890	894.95	906	672	714.52	745	532	623.48	676
la08	15	5	863	865.95	884	710	740.10	783	239	333.71	450
la09	15	5	951	951.05	952	805	818.95	849	273	359.57	445
la10	15	5	958	958.00	958	798	835.62	865	433	523.90	624
la11	20	5	1222	1222.00	1222	960	1014.10	1072	344	496.71	654
la12	20	5	1039	1039.00	1039	840	881.43	926	346	393.38	454
la13	20	5	1150	1151.52	1162	926	984.81	1043	334	452.52	555
la14	20	5	1292	1292.00	1292	1010	1056.10	1094	544	822.52	1014
la15	20	5	1210	1236.57	1255	926	986.57	1041	371	588.86	726
la16	10	10	979	992.90	1008	798	847.48	882	2644	2962.62	3265
la17	10	10	784	801.19	832	725	745.05	777	2333	2555.43	2909
la18	10	10	853	892.14	942	760	788.48	829	2417	2660.52	2920
la19	10	10	847	875.10	902	753	782.71	805	2000	2329.05	2625
la20	10	10	907	922.48	942	789	811.19	852	2328	2597.76	2997
la21	15	10	1136	1177.67	1229	965	1006.81	1047	2230	2584.67	3157
la22	15	10	1000	1026.67	1049	879	909.43	963	2056	2323.43	2756
la23	15	10	1040	1080.19	1111	934	967.81	1002	1826	2129.10	2345
la24	15	10	1004	1034.33	1072	900	929.10	961	1741	2039.14	2313
la25	15	10	1042	1076.57	1122	906	939.05	979	2004	2512.86	2909
la26	20	10	1347	1376.48	1417	1145	1195.62	1263	1932	2425.86	2725
la27	20	10	1378	1428.43	1480	1163	1225.95	1295	1979	2521.57	3074
la28	20	10	1373	1400.24	1425	1187	1217.33	1289	2154	2568.48	2863
la29	20	10	1345	1382.67	1428	1130	1183.24	1252	2846	3106.90	3474
la30	20	10	1443	1488.05	1529	1175	1255.24	1305	2530	3032.29	3443
la31	30	10	1850	1880.52	1918	1528	1593.81	1643	2654	2923.14	3292
la32	30	10	1969	2013.57	2056	1705	1733.29	1771	2425	2765.76	3186
la33	30	10	1767	1834.19	1887	1520	1572.81	1648	2424	2783.14	3342
la34	30	10	1846	1893.43	1924	1564	1623.24	1682	2375	2824.62	3110
la35	30	10	1946	2020.24	2111	1600	1651.24	1710	3295	3917.29	4550

Table 4.19(cont'd) The results of solving LA with MOPSO

la36	15	15	1351	1395.33	1447	1211	1256.33	1321	6595	7075.29	7914
la37	15	15	1504	1548.24	1617	1280	1315.29	1351	6909	7622.57	8405
la38	15	15	1272	1334.10	1378	1130	1158.24	1217	5819	6796.67	7999
la39	15	15	1331	1367.43	1404	1141	1185.43	1217	5875	6404.76	7103
la40	15	15	1293	1322.67	1367	1160	1193.10	1248	5607	6227.33	7030

Table 4. 30 The results of solving SWV with MOPSO

Problem	n	m	Makespan			MFT			MIT		
			best	average	worst	best	average	worst	best	average	worst
swv01	20	10	1694	1724.285714	1761	1442	1507.238095	1577	2375	2965.142857	3703
swv02	20	10	1710	1758.52381	1805	1490	1558.761905	1622	2265	2961.619048	3652
swv03	20	10	1672	1720.047619	1781	1483	1540.333333	1606	2323	2883.666667	3421
swv04	20	10	1734	1802.666667	1860	1504	1560.52381	1644	1967	2602.809524	3091
swv05	20	10	1749	1787.428571	1824	1498	1575.571429	1630	2094	2571.047619	3881
swv06	20	15	2099	2141.666667	2220	1714	1785.809524	1928	4559	5697.333333	7070
swv07	20	15	1957	2003.095238	2057	1631	1705.333333	1806	4872	5427.380952	6718
swv08	20	15	2155	2210.190476	2260	1718	1800.428571	1880	5353	6335.333333	7728
swv09	20	15	2048	2114.952381	2164	1644	1739.142857	1871	5005	6113.47619	7360
swv10	20	15	2138	2183.809524	2227	1742	1805.380952	1916	5297	6266.142857	7290
swv11	50	10	3815	3865	3944	2902	3006.285714	3145	3755	4884.571429	6071
swv12	50	10	3742	3881.714286	3987	2885	2993.333333	3164	4097	5032.809524	5931
swv13	50	10	3884	3937.52381	3990	2888	2992.428571	3069	4655	5961.380952	7658
swv14	50	10	3658	3743.142857	3855	2686	2841.571429	2997	3305	4400.619048	5821
swv15	50	10	3681	3752.714286	3844	2725	2839.52381	2923	3978	5279.380952	6326
swv16	50	10	2924	2954.047619	3043	2446	2517.142857	2614	2970	3437.238095	4299
swv17	50	10	2839	2880.857143	2927	2344	2421.095238	2512	3235	3605.380952	3905
swv18	50	10	2879	2902.190476	2938	2377	2437.904762	2482	3205	3588.238095	3889
swv19	50	10	2965	3013.380952	3065	2421	2504.52381	2575	3186	3667.380952	4092
swv20	50	10	2829	2879.238095	2907	2352	2404.952381	2480	2800	3243.047619	3572

CHAPTER 5 PSO for Multi-objective OSSP

5.1 Problem Formulation

The common characteristics of shop scheduling problems are as follows. A set of *n* jobs must be processed on a set of *m* machines. Each job consists of *m* operations, each of which must be processed on a different machine for a given process time. At any time, at most one operation can be processed on each machine, and at most one operation of each job can be processed. Unlike flow-shop and job-shop scheduling problems, the exceptional condition of the open-shop scheduling problem is that the operations of each job can be processed in any order.

The aim of the openshop scheduling problems are to assign jobs to machines so that the completion time, also called the makespan, total flow time, and machine idle time are minimized simultaneously. To minimize the makespan, we must minimize the maximum total processing time on all machines. The total flow time refers to the sum of the completion times of all jobs. The idle times of each machine during the work cycle are summed to obtain the total machine idle time. The object functions of makespan, total flow time and machine idle time are described as chapter 3.

5.2 Particle Position Representation

In this study, we randomly generated a group of particles (solutions) represented by a permutation sequence that is an ordered list of operations. For an *n*-job *m*-machine problem, the position of particle *k* can be represented by an $m \times n$ matrix, i.e.,

$X^k = \begin{bmatrix} x_{11}^k & x_{12}^k & \cdots & x_{1n}^k \\ x_{21}^k & x_{22}^k & \cdots & x_{2n}^k \\ \vdots & \vdots & & \vdots \\ x_{m1}^k & x_{m2}^k & \cdots & x_{mn}^k \end{bmatrix}$, where x_{ij}^k denotes the priority of operation o_{ij}, which means the operation of job j that must be processed on machine i.

The Giffler and Thompson (G&T) algorithm is briefly described below.

Notation:

(i,j) is the operation of job j that must be processed on machine i

S is the partial schedule that contains scheduled operations

Ω is the set of operations that can be scheduled

$s_{(i,j)}$ is the earliest time at which operation (i,j) belonging to Ω can be started.

$p_{(i,j)}$ is the processing time of operation (i,j).

$f_{(i,j)}$ is the earliest time at which operation (i,j) belonging to Ω can be finished, $f_{(i,j)} = s_{(i,j)} + p_{(i,j)}$.

G&T algorithm:

Step 1: Initialize $S = \phi$; Ω to contain all operations without predecessors.

Step 2: Determine $f^* = \min_{(i,j)\in\Omega}\{f(i,j)\}$ and the machine m^* on which f^* can be realized.

Step 3:

(1)Identify the operation set $(i',j')\in\Omega$ such that (i',j') requires machine m^*, and $S(i',j') < f^*$.

(2) Choose (i, j) from the operation set identified in Step 3(1) with the largest priority.

(3) Add (i, j) to S.

(4) Assign $s_{(i,j)}$ as the starting time of (i, j).

Step 4: If a complete schedule has been generated, stop. Otherwise, delete (i, j) from Ω, include its immediate successor in Ω, and then go to Step 2.

The movement of particles is modified in accordance with the representation of particle position based on the insertion operator.

5.3 Particle Velocity

The original PSO velocity concept is that each particle moves according to the velocity determined by the distance between the previous position of the particle and the *gbest* (*pbest*) solution. The two major purposes of the particle velocity are to move the particle toward the *gbest* and *pbest* solutions, and to maintain the inertia to prevent particles from becoming trapped in local optima.

In the proposed PSO, we concentrated on preventing particles from becoming trapped in local optima rather than moving them toward the *gbest* (*pbest*) solution. If the priority value increases or decreases with the present velocity in this iteration, we maintain the priority value increasing or decreasing at the beginning of the next iteration with probability w, which is the PSO inertial weight. The larger the value of w is, the greater the number of iterations over which the priority value keeps increasing or decreasing, and the greater the difficulty the particle has returning to the current position. For an n-job problem, the velocity of particle k can be represented as

$V^k = \begin{bmatrix} v_{11}^k & v_{12}^k & \cdots & v_{1n}^k \\ v_{21}^k & v_{22}^k & \cdots & v_{2n}^k \\ \vdots & \vdots & \cdots & \vdots \\ v_{m1}^k & v_{m2}^k & \cdots & v_{mn}^k \end{bmatrix}$, where v_{ij}^k is the velocity of the operation o_{ij} of particle k, $v_{ij}^k \in \{-1,0,1\}$.

The initial particle velocities are generated randomly. Instead of considering the distance from x_{ij}^k to $pbest_{ij}^k(gbest_{ij})$, our PSO considers whether the value of x_{ij}^k is larger or smaller than $pbest_{ij}^k(gbest_{ij})$ If x_{ij}^k has decreased in the present iteration, this means that $pbest_{ij}^k(gbest_{ij})$ is smaller than x_{ij}^k, and x_{ij}^k is set moving toward $pbest_{ij}^k(gbest_{ij})$ by letting $v_{ij}^k \leftarrow -1$. Therefore, in the next iteration, x_{ij}^k is kept decreasing by one (i.e., $x_{ij}^k \leftarrow x_{ij}^k - 1$) with probability *w*. Conversely, if x_{ij}^k has increased in this iteration, this means that $pbest_{ij}^k(gbest_{ij})$ is larger than x_{ij}^k, and x_{ij}^k is set moving toward $pbest_{ij}^k(gbest_{ij})$ by letting $v_{ij}^k \leftarrow 1$. Therefore, in the next iteration, x_{ij}^k is kept increasing by one (i.e. $x_{ij}^k \leftarrow x_{ij}^k + 1$) with probability *w*.

The inertial weight *w* influences the velocity of particles in PSO. We randomly update velocities at the beginning of each iteration. For each particle *k* and operation o_{ij}, if v_{ij}^k is not equal to 0, v_{ij}^k is set to 0 with probability (1–*w*). This ensures that x_{ij}^k stops increasing or decreasing continuously in this iteration with probability (1–*w*).

5.4 Particle Movement

In our PSO, the particle movement is based on the insert operator proposed by Sha and Hsu. We set $x_{ij}^k \leftarrow p + rand_2 - 0.5$ if we want to insert o_{ij} into the *p*th location in the permutation list. In addition, the location of operation o_{ij} in the operation sequence of *k*th *pbest* and *gbest* solution are $pbest_{ij}^k$ and $gbest_{ij}$. When particle *k* moves, for all o_{ij}, if v_{ij}^k equals 0, the x_{ij}^k will be set to $pbest_{ij}^k + rand_2 - 0.5$ with probability c_1 and set to be $gbest_{ij} + rand_2 - 0.5$ with probability c_2, where $rand_2$ is a random variable between 0 and 1, and c_1 and c_2 are constants between 0 and 1, and $c_1+c_2 \leqq 1$. For example, assume that V^k, X^k, $pbest^k$, $gbest$, c_1, and c_2 are as follows:

$$V^k = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix},\ X^k = \begin{bmatrix} 2.5 & 3.3 \\ 1.3 & 4.2 \end{bmatrix},\ pbest^k = \begin{bmatrix} 1 & 4 \\ 3 & 2 \end{bmatrix}, gbest = \begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix},\ c_1 = 0.7,\ c_2 = 0.1$$

For o_{11}:

Because $v_{11}^k \neq 0, x_{11}^k \leftarrow x_{11}^k + v_{11}^k$, that is, $x_{11}^k = 1.5$.

For o_{12}:

Because $v_{12}^k = 0$, randomly generate $rand_1 = 0.6$.

Because $rand_1 \leq c_1$, randomly generate $rand_2 = 0.3$.

Because $pbest_{12}^k \geq x_{12}^k$, set $v_{12}^k \leftarrow 1$, and then

$x_{12}^k \leftarrow pbest_{12}^k + rand_2 - 0.5$, that is, $x_{12}^k = 3.8$.

For o_{21}:

Because $v_{21}^k = 0$, randomly generate $rand_1 = 0.9$.

Because $rand_1 > c_1 + c_2$, x_{21}^k does not be changed.

For o_{22}:

Because $v_{22}^k = 0$, randomly generate $rand_1 = 0.75$.

Because $c_1 < rand_1 \leq c_1 + c_2$, generate $rand_2 = 0.8$.

Because $gbest_{21}^k < x_{22}^k$, set $v_{22}^k \leftarrow -1$, and then

$x_{22}^k \leftarrow gbest_{22}^k + rand_2 - 0.5$, that is, $x_{22}^k = 2.3$.

Finally, after the particle moved, the V^k and Xk are:

$$V^k = \begin{bmatrix} -1 & 1 \\ 0 & -1 \end{bmatrix} \text{ and } X^k = \begin{bmatrix} 1.5 & 3.8 \\ 1.3 & 2.3 \end{bmatrix}.$$

5.5 Computational Results

The proposed multi-objective PSO (MOPSO) algorithm was tested on benchmark problems obtained from the Guéret and Prins (1999). The program was coded in Visual C++ and run 20 times on each problem on a Pentium 4 3.0-GHz computer with 1 GB of RAM running Windows XP. During the pilot experiment, we used four swarm sizes N (30, 60, 80, and 100) to test the algorithm. The outcome of N=80 was best, so that value was used in all further tests. Parameters c_1 and c_2 were tested at various values in the range 0.1–0.7 in increments of 0.2. The inertial weight w was reduced from w_{max} to w_{min} during iterations, where w_{max} was set to 0.5, 0.7, and 0.9, and w_{min} was set to 0.1, 0.3, and 0.5. The combination of c_1=0.7, c_2=0.1, w_{max}=0.7 and w_{min}=0.3 gave the best results. The maximum iteration limit was set to 60 and the maximum archive size was set to 80.

In the first experiment, we have assigned the Pareto set as *Pbest* solutions which considered four different conditions. In the first scenario, we took all three objectives into consideration. The two objectives including makespan and total flow time are considered in the second scenario. The third and fourth scenario considered makespan, machine idle time and total flow time, machine idle time, respectively. The results of the first experiment are as Table 5.1-5.4.

Table 5. 1 The results of the first experiment considering three objectives as Pareto set

	makespane		total flow time		machine idle time	
	best	average	best	average	best	average
1	1100	1109.3	10636	10717	624	712.15
2	1097	1101.8	10489	10551	590	659.75
3	1090	1101.8	10563	10661	589	648.4
4	1089	1091.6	10561	10606	498	625.1
5	1084	1094.7	10495	10595	558	616.7
6	1071	1082.1	10530	10560	493	513.95
7	1081	1083.3	10519	10569	549	594.85
8	1098	1103.1	10675	10722	671	714.6
9	1117	1128.3	10662	10738	681	763.6
10	1097	1098	10621	10715	673	777.2
	1092.4	1099.4	10575	10643	592.6	662.63

Table 5. 2 The results of the first experiment considering makespan and total flow time as Pareto set

	makespane		total flow time		machine idle time	
	best	average	best	average	best	average
1	1106	1108.85	10604	10687.1	637	727
2	1097	1101.4	10499	10542.1	612	681.6
3	1087	1099.5	10560	10645.7	597	670.8
4	1089	1093.7	10527	10591.4	550	661.85
5	1087	1097.6	10516	10597.6	608	668.75
6	1071	1076.25	10527	10558.6	501	528.05
7	1081	1082.65	10500	10541.4	585	605
8	1098	1102.4	10656	10719.6	692	755.4
9	1122	1129.2	10667	10757.6	734	838.1
10	1097	1098.25	10704	10751.6	800	839.65
	1093.5	1098.98	10576	10639.2	631.6	697.62

Table 5. 3 The results of the first experiment considering makespan and machine idle time as Pareto set

	makespane		total flow time		machine idle time	
	best	average	best	average	best	average
1	1097	1111.85	10733	10785.4	668	717.6
2	1100	1109.2	10602	10683.1	594	666.7
3	1087	1094.45	10606	10682.15	580	638.85
4	1089	1093.6	10557	10634.45	482	599.25
5	1075	1092.75	10552	10659.35	520	616.95
6	1071	1077.9	10554	10577.2	496	504.7
7	1081	1082.75	10537	10576.7	521	564.75
8	1098	1101.9	10696	10750.85	654	700.9
9	1116	1127	10722	10854.15	681	761.8
10	1094	1098.2	10635	10747.3	656	742.15
	1091	1098.96	10619	10695.065	585	651.365

Table 5. 4 The results of the first experiment considering total flow time and machine idle time as Pareto set

	makespane		total flow time		machine idle time	
	best	average	best	average	best	average
1	1112	1114.2	10636	10733.8	617	679.95
2	1101	1111.3	10470	10556.35	582	673.05
3	1100	1113.5	10558	10668.65	591	662.45
4	1096	1099.45	10547	10597	476	576.5
5	1085	1097.65	10523	10596	514	603
6	1071	1098	10490	10558.3	473	525.45
7	1081	1083.5	10501	10556.85	491	553.65
8	1099	1106.05	10634	10711.15	628	708.6
9	1129	1134.6	10644	10685.55	685	765.6
10	1096	1100.2	10642	10717.7	627	721.45
	1097	1105.845	10565	10638.135	568	646.97

Table 5. 5 Summary of the results of the first experiment

Optimized Objectives	makespane		total flow time		machine idle time	
	best	average	best	average	best	average
All	1092.4	1099.38	10575.1	10643.31	592.6	662.63
MS+TFT	1093.5	1098.98	10576	10639.25	631.6	697.62
MS+MIT	1090.8	1098.96	10619.4	10695.07	585.2	651.365
TFT+MIT	1097	1105.845	10564.5	10638.14	568.4	646.97

In the second experiment, we have divided the swarm into sub-swarm to search for the solutions. At first we use three groups (sub-swarm) for three objects as (i) in Table 5.13. In (ii), (iii) and (iv), only one particle swarm is applied to search single object. In the last part of this experiment, two sub-swarm are used to search the solutions. In (v) of Table 5.13, the two sub-swarm, one is searched for the object makespan while the other is searched for total flow time. In (vi) of Table 5.13, the two sub-swarm, one is searched for the object makespan while the other is searched for machine idle time. In (vii) of Table 5.13, the two sub-swarm, one is searched for the object total flow time while the other is searched for machine idle time.

Table 5. 6 The results of the second experiment considering three objectives with three sub-swarms

	makespane		total flow time		machine idle time	
	best	average	Best	average	best	average
1	1106	1116.85	10632	10731.45	627	714.05
2	1101	1113.85	10492	10568.75	619	740.55
3	1109	1115.45	10548	10705.75	608	678.9
4	1091	1099.7	10550	10617.9	476	600.25
5	1088	1101	10509	10601.15	495	608.3
6	1071	1089.9	10488	10533.65	451	515.05
7	1081	1085.65	10493	10569.2	492	562.1
8	1099	1112.35	10655	10747.05	684	720.65
9	1131	1136.65	10697	10746.2	705	791.95
10	1097	1101.15	10665	10772.3	640	724.95
	1097	1107.255	10573	10659.34	580	665.675

Table 5. 7 The results of the second experiment considering makespan with one swarm

	makespane		total flow time		machine idle time	
	best	average	best	average	best	average
1	1095	1100.55	10725	10777.85	704	749.65
2	1097	1100.1	10565	10652.05	621	705.65
3	1087	1094.1	10566	10682.4	606	665.25
4	1089	1091.1	10615	10650.5	663	679.2
5	1084	1091.2	10550	10631.8	586	650.6
6	1071	1080.7	10522	10577.9	503	564.8
7	1081	1081.6	10561	10607.55	560	615.8
8	1098	1100.4	10660	10723.45	665	736.3
9	1116	1125.65	10765	10853.95	802	874.9
10	1092	1095.1	10679	10736.4	751	798.05
	1091	1096.05	10621	10689.385	646	704.02

Table 5. 8 The results of the second experiment considering total flow time with one swarm

	makespane		total flow time		machine idle time	
	best	average	best	average	best	average
1	1109	1117.5	10636	10749.8	637	820.3
2	1101	1112.85	10495	10547.55	620	766.75
3	1103	1114.7	10577	10700.2	660	704.35
4	1090	1099.2	10550	10609.1	587	686.3
5	1087	1096.9	10524	10589.5	564	668.45
6	1072	1087.3	10481	10565.2	522	564.6
7	1081	1089.3	10508	10577.65	591	660.2
8	1100	1111.1	10700	10757.45	739	783.55
9	1131	1141.55	10678	10783.4	825	903.05
10	1097	1106	10700	10782.95	742	854.3
	1097	1107.64	10585	10666.28	649	741.185

Table 5. 9 The results of the second experiment considering machine idle time with one swarm

	makespane		total flow time		machine idle time	
	best	average	best	average	best	average
1	1112	1117.9	10780	10853.35	625	705.7
2	1101	1118.65	10646	10746.35	633	739.2
3	1103	1114.75	10688	10740.85	678	690.15
4	1096	1100.15	10596	10675.6	473	556.5
5	1095	1103.1	10644	10717.25	569	622.8
6	1072	1111.1	10540	10599.6	494	533.75
7	1081	1088.45	10537	10629.4	492	573.7
8	1100	1113.65	10734	10808.45	679	723.2
9	1132	1137.95	10745	10877.85	720	818.5
10	1099	1107.4	10814	10856.7	649	722.55
	1099	1111.31	10672	10750.54	601	668.605

Table 5. 10 The results of the second experiment considering makespan and TFT with two sub-swarms

	makespane		total flow time		machine idle time	
	best	average	best	average	best	average
1	1100	1106.7	10609	10683.65	631	728.75
2	1097	1101.4	10478	10523.75	582	674.8
3	1082	1098.75	10515	10660.55	579	680.8
4	1089	1091.45	10543	10577	595	652.75
5	1087	1092.5	10512	10555.05	505	620
6	1071	1079.45	10490	10542.95	499	536.15
7	1081	1081.5	10490	10533.6	551	598.05
8	1097	1100.3	10632	10679.7	692	733.25
9	1116	1128.2	10647	10721.3	703	859.05
10	1092	1095.1	10629	10671.15	728	771.35
	1091	1097.535	10555	10614.87	607	685.495

Table 5. 11 The results of the second experiment considering makespan and MIT with two sub-swarms

	makespane		total flow time		machine idle time	
	best	average	best	average	best	average
1	1095	1105.35	10705	10784.85	658	705.65
2	1097	1104.6	10570	10678.4	554	639
3	1087	1098.45	10557	10682.4	544	653.3
4	1089	1093.6	10597	10637.6	508	583.3
5	1087	1094.3	10560	10629.95	532	603.4
6	1071	1087.7	10529	10576.4	473	507.1
7	1081	1082.2	10548	10592.45	489	557.25
8	1097	1099.4	10700	10744.7	651	698.05
9	1116	1125.5	10726	10817.7	666	780.2
10	1092	1097.35	10702	10760.4	642	711.6
	1091	1098.845	10619	10690.485	572	643.885

Table 5. 12 The results of the second experiment considering TFT and MIT with two sub-swarms

	makespane		total flow time		machine idle time	
	best	average	best	average	best	average
1	1106	1116.85	10632	10731.45	627	714.05
2	1101	1113.85	10492	10568.75	619	740.55
3	1109	1115.45	10548	10705.75	608	678.9
4	1091	1099.7	10550	10617.9	476	600.25
5	1088	1101	10509	10601.15	495	608.3
6	1071	1089.9	10488	10533.65	451	515.05
7	1081	1085.65	10493	10569.2	492	562.1
8	1099	1112.35	10655	10747.05	684	720.65
9	1131	1136.65	10697	10746.2	705	791.95
10	1097	1101.15	10665	10772.3	640	724.95
	1097	1107.255	10573	10659.34	580	665.675

Table 5. 13 Summary of the results of the second experiment

Optimized	makespane		total flow time		machine idle time	
Objectives	best	average	best	average	best	average
All	1091.6	1099.08	10562.3	10633.97	567.6	648.77
MS	1091	1096.05	10620.8	10689.39	646.1	704.02
TFT	1097.1	1107.64	10584.9	10666.28	648.7	741.185
MIT	1099.1	1111.31	10672.4	10750.54	601.2	668.605
MS+ TFT	1091.2	1097.535	10554.5	10614.87	606.5	685.495
MS +MIT	1091.2	1098.845	10619.4	10690.49	571.7	643.885
TFT +MIT	1097.4	1107.255	10572.9	10659.34	579.7	665.675

In order to compare the performance of our PSO with traditional meta-heuristic algorithm, we code the GA algorithm to program with C++ language in addition. At first, we apply PSO to solve the hardest benchmark problem generated by Guéret and Prins (1999). The program runs 20 times on each problem on a Pentium 4 3.0-GHz computer with 1 GB of RAM running Windows XP. During the pilot experiment, we used four swarm sizes N (50, 100, 150, and 200) to test the algorithm. The outcome of N=150 was best, so that value was used in all further tests. Parameters c_1 and c_2 were tested at various portfolios in the range 0.1–0.7 in increments of 0.2. The inertial weight w was reduced from 0.9 to 0.1 during iterations. The combination of c_1=0.1, c_2=0.8, w=0.1 gave the best results. The maximum iteration limit was set to 60 and the maximum archive size was set to 150. The results of MOPSO for notorious open shop scheduling problems are demonstrated in Table 5.14

Table 5. 14 The results of MOPSO for benchmark problems gp03-gp10

Problem	Makespan		MIT		TFT		CPU Time
	Best	Average	Best	Average	Best	Average	
gp03-01	1168	1168	0	0	3174	3408.9	7.016
gp03-02	1170	1170	0	0	3340	3437.8	7.015
gp03-03	1168	1168	0	0	3336	3418.0	7.063
gp03-04	1166	1166	0	0	3170	3380.6	7.000
gp03-05	1170	1170	0	24	3181	3387.7	7.110
gp03-06	1169	1169	0	0	3177	3386.3	7.187
gp03-07	1165	1165	0	0	3166	3444.8	7.188
gp03-08	1167	1167	0	0	3334	3398.8	7.047
gp03-09	1162	1162	0	7.9	3167	3386.8	7.094
gp03-10	1165	1165	0	0	3330	3401.1	7.063
Average	1167	1167	0	3.2	3238	3405.1	7.078
gp04-01	1281	1281	274	455	4326	4534.75	14.297
gp04-02	1270	1270	0	257	4346	4872.85	14.250
gp04-03	1288	1288	240	393	4574	4778.4	14.110
gp04-04	1261	1261	0	187	4530	4820.7	14.125
gp04-05	1289	1289	277	405	4305	4783.3	14.156
gp04-06	1269	1269	179	301	4539	4937.4	15.281
gp04-07	1267	1267	0	175	4568	4722.85	14.563
gp04-08	1259	1259	191	368	4524	4751.7	14.406
gp04-09	1280	1280	278	512	4304	4545.2	14.375
gp04-10	1263	1263	188	228	4549	5005	14.250
Average	1272.7	1272.7	162.7	328	4456.5	4775.215	14.381
gp05-01	1245	1245	456	489.25	5593	5802.4	24.844
gp05-02	1247	1247	243	596.25	5418	5828.1	24.813
gp05-03	1265	1265	260	364.1	5797	6024.4	24.734
gp05-04	1258	1258.2	471	553.4	5713	5851.6	25.000
gp05-05	1280	1280	291	632.55	5318	5824.65	24.500
gp05-06	1269	1269.05	268	326.25	5589	5618.35	24.156
gp05-07	1269	1269	0	317.95	5550	5879.3	24.187
gp05-08	1287	1287	294	704.8	5526	5733.2	24.657
gp05-09	1262	1262	302	480.35	5630	6030.6	23.937
gp05-10	1254	1254.95	271	539.8	5618	5885.75	23.984
Average	1263.6	1263.72	285.6	500.47	5575.2	5847.835	24.481

Table 5.14(Cont'd) The results of MOPSO for benchmark problems gp03-gp10

gp06-01	1265	1265	332	432	6858	7053.9	41.812
gp06-02	1285	1285.45	409	677	7003	7111.85	40.485
gp06-03	1256	1256.75	44	545	6811	7149.25	42.000
gp06-04	1275	1275.05	525	821	6857	7046.25	41.422
gp06-05	1299	1299.4	82	670	7042	7215.4	38.703
gp06-06	1284	1284.85	282	619	6687	7181.8	41.250
gp06-07	1290	1290	317	684	6601	7077.85	41.813
gp06-08	1265	1265.7	352	626	7047	7194.7	39.641
gp06-09	1243	1245.8	252	536	6401	6955.7	41.047
gp06-10	1254	1254.25	486	593	6580	6878.75	40.859
Average	1271.6	1272.225	308.1	620	6788.7	7086.545	40.903

gp07-01	1159	1162.3	319	482	7799	7980.15	58.547
gp07-02	1185	1185	152	533	7749	7885.75	58.141
gp07-03	1237	1237.65	57	676	8042	8316.15	58.922
gp07-04	1167	1168.75	197	502	7783	8016.5	60.656
gp07-05	1158	1158.3	417	493	7793	7868.25	64.469
gp07-06	1193	1194.25	346	613	7771	7979.95	65.594
gp07-07	1185	1185.1	372	549	7767	7916	62.766
gp07-08	1181	1181.2	46	569	7869	8019.25	60.062
gp07-09	1220	1220.15	306	549	7780	7995.95	62.078
gp07-10	1270	1270	276	614	8023	8274.05	61.079
Average	1195.5	1196.27	249	558	7837.6	8025.2	61.231

gp08-01	1147	1160.25	29	347	9005	9127.15	84.156
gp08-02	1137	1143.85	247	404	8923	9018.7	93.281
gp08-03	1115	1119.55	67	285	8739	8813.65	93.031
gp08-04	1154	1159.6	267	410	8960	9071.3	91.860
gp08-05	1218	1219.35	214	625	8864	9157.15	97.609
gp08-06	1116	1130.85	51	321	8777	8928.4	93.375
gp08-07	1129	1135.95	132	339	8892	8957.2	92.766
gp08-08	1148	1158.55	7	358	8928	9113.45	93.375
gp08-09	1115	1118.95	159	245	8838	8891.85	93.109
gp08-10	1162	1162.5	225	590	8982	9052.8	95.781
Average	1144.1	1150.94	140	392	8890.8	9013.165	92.834

Table 5.14(Cont'd) The results of MOPSO for benchmark problems gp03-gp10

gp09-01	1138	1146.75	255	419	10050	10118.25	133.42
gp09-02	1114	1120	0	205	9857	9969.00	137.37
gp09-03	1118	1120.4	232	422	9991	10042.65	136.17
gp09-04	1140	1145.35	186	430	10014	10114.50	137.15
gp09-05	1180	1180.3	344	572	10029	10186.15	150.39
gp09-06	1097	1113.9	0	394	9819	9936.90	166.68
gp09-07	1098	1114.75	82	319	9792	9875.10	165.82
gp09-08	1110	1117.25	0	219	9749	9919.35	167.46
gp09-09	1126	1130.05	124	341	9829	9964.70	164.56
gp09-10	1124	1137	213	317	9862	9947.05	163.62
Average	1124.5	1132.575	144	364	9899.2	10007.36	152.26

gp10-01	1100	1113.6	0	201	10835	11008.05	220.45
gp10-02	1102	1116.7	82	351	10816	10994.45	228.82
gp10-03	1093	1113.1	74	199	10813	10935.10	224.67
gp10-04	1087	1100.75	51	266	10760	10884.35	212.71
gp10-05	1093	1101.3	0	125	10731	10900.35	208.71
gp10-06	1074	1104.3	0	186	10637	10900.05	214.31
gp10-07	1084	1093.6	0	142	10632	10787.55	185.40
gp10-08	1098	1105.8	101	261	10779	10924.30	187.01
gp10-09	1117	1138.8	61	424	10955	11182.60	173.26
gp10-10	1095	1115.5	136	279	10824	10991.15	176.79
Average	1094.3	1110.345	51	243	10778.2	10950.79	203.21

The GA program also runs 20 times on each problem on a Pentium 4 3.0-GHz computer with 1 GB of RAM running Windows XP. The parameter setting of GA algorithm is described as follows. During the pilot experiment, we used four population sizes *N* (50, 100, 150, and 200) to test the algorithm. The outcome of *N*=150 was best, so that value was used in all further tests. The crossover and mutation rate is test in the range of 0.1-0.9. The combination of cross rate equals 0.5, mutation rate equals 0.1 gave the best results. The maximum iteration limit was set to

60 and the maximum archive size was set to 150. The results of MOGA for notorious open shop scheduling problems are demonstrated in Table 5.15.

Table 5. 15 The results of MOGA for benchmark problems gp03-gp10

Problem	Makespan		MIT		TFT		CPU
	Best	Average	Best	Average	Best	Average	Time
gp03-01	1168	1168.15	0	0	3174	3384.9	3.59
gp03-02	1170	1170	0	32.6	3177	3397.05	3.75
gp03-03	1168	1168	0	0	3336	3426.2	3.91
gp03-04	1166	1166	0	0	3170	3413	3.75
gp03-05	1170	1170	0	15.9	3181	3403.6	3.75
gp03-06	1169	1169	0	16.1	3177	3394.35	3.90
gp03-07	1165	1165.05	0	16.4	3166	3379.25	3.60
gp03-08	1167	1167	0	0	3172	3366.4	3.90
gp03-09	1162	1162	0	7.85	3167	3402.5	3.60
gp03-10	1165	1165	0	7.9	3172	3385.3	3.90
Average	1167	1167.02	0	9.675	3189.2	3395.255	3.76
gp04-01	1281	1283.7	0	204.1	4325	4705.8	4.22
gp04-02	1270	1271.75	0	147.5	4309	4798.6	4.69
gp04-03	1288	1290.7	0	321.6	4574	4889.9	4.37
gp04-04	1261	1261	233	252	4527	4651.85	4.69
gp04-05	1289	1290.25	0	362.2	4309	4881.25	4.53
gp04-06	1269	1270.85	179	339.7	4539	4848.3	5.00
gp04-07	1271	1277.8	0	194	4582	4845.2	4.53
gp04-08	1259	1259	191	496	4524	4549.3	4.69
gp04-09	1280	1284.5	0	320.4	4316	4642.45	4.69
gp04-10	1263	1263.45	188	219.8	4785	5005.45	4.69
Average	1273.1	1275.3	79.1	285.7	4479	4781.81	4.61

Table 5.15(Cont'd) The results of MOGA for benchmark problems gp03-gp10

gp05-01	1245	1253.8	454	716.1	5824	5960	6.25
gp05-02	1247	1267.2	270	672	5587	6004.5	6.25
gp05-03	1265	1265	260	339.6	5588	6043.9	6.25
gp05-04	1263	1275.95	231	516.8	5633	5890.1	6.41
gp05-05	1281	1285.5	274	406.1	5574	6019.5	6.25
gp05-06	1270	1282.15	228	484.7	5589	5884.25	6.09
gp05-07	1269	1269.65	0	493.9	5552	5786.8	6.25
gp05-08	1288	1294.45	295	383.7	5836	6048.45	6.40
gp05-09	1262	1274.15	262	589.4	5573	5973.75	6.25
gp05-10	1257	1274.5	237	610.7	5675	6013.65	6.25
Average	1264.7	1274.235	251.1	521.3	5643.1	5962.49	6.26

gp06-01	1266	1284.95	271	671.1	7076	7256.45	8.75
gp06-02	1289	1289.85	0	670.5	6914	7254.25	8.12
gp06-03	1257	1261.8	0	670.9	6842	7233.65	8.60
gp06-04	1275	1283.35	240	902.5	6903	7066.35	9.21
gp06-05	1301	1302.6	105	787.7	6910	7283.45	8.44
gp06-06	1285	1294.95	0	914.4	6918	7251.8	8.44
gp06-07	1292	1295.7	294	640	6826	7454.2	8.44
gp06-08	1268	1271.75	426	864.7	6652	7015.5	8.28
gp06-09	1246	1254.9	467	714.7	6985	7024	8.75
gp06-10	1258	1267.55	504	892.4	6585	7007	8.75
Average	1273.7	1280.74	230.7	772.9	6861.1	7184.665	8.57

gp07-01	1189	1190.55	594	641.3	7974	8025.8	23.44
gp07-02	1186	1190.7	618	668.6	7811	7854.7	23.12
gp07-03	1239	1264.95	253	866.65	8001	8482.15	23.59
gp07-04	1173	1188.15	407	758	7882	8022.65	23.75
gp07-05	1188	1202.85	416	593.5	7991	8188.85	23.60
gp07-06	1200	1236.75	394	903.6	7740	8147.6	23.75
gp07-07	1186	1211.6	362	614	7668	8069.9	23.29
gp07-08	1191	1191	626	735.05	7902	7966.5	23.28
gp07-09	1222	1222	510	618.3	7829	7859.3	23.28
gp07-10	1271	1273.65	556	1145.65	8091	8466.7	23.91
Average	1204.5	1217.22	473.6	754.465	7888.9	8108.415	23.50

Table 5.15(Cont'd) The results of MOGA for benchmark problems gp03-gp10

gp08-01	1182	1203.1	457	869.4	9072	9283.6	33.12
gp08-02	1166	1184.05	453	867.2	9101	9184.9	33.28
gp08-03	1148	1180.85	416	644	8987	9073.8	32.97
gp08-04	1181	1189	656	768	8955	9041.75	32.81
gp08-05	1224	1227.65	482	899.5	8824	9185.45	32.97
gp08-06	1170	1183.3	700	908.7	8983	9092.35	33.60
gp08-07	1169	1199.15	560	746.2	9028	9271.2	32.97
gp08-08	1182	1191.5	455	921.7	9210	9348.45	33.43
gp08-09	1152	1190.5	471	803.8	8810	9216.15	32.97
gp08-10	1187	1202.8	368	811.7	8910	9250.45	32.50
Average	1176.1	1195.19	501.8	824	8988	9194.81	33.06

gp09-01	1166	1190.05	664	983.6	10109	10275.05	46.56
gp09-02	1158	1173.85	255	733.5	9907	10194.45	45.63
gp09-03	1157	1209.5	689	1177	10113	10537.95	45.78
gp09-04	1164	1181.55	478	786.6	10071	10323.45	46.72
gp09-05	1199	1206.75	346	978.8	10209	10355.3	46.40
gp09-06	1139	1159.6	591	967.7	10071	10221.55	46.10
gp09-07	1153	1159.05	507	615	9870	10105	46.41
gp09-08	1151	1179	561	762.6	10044	10186.85	45.78
gp09-09	1176	1180.2	614	753.4	9877	10129.1	46.25
gp09-10	1142	1165.95	526	750.5	9880	10051.6	46.72
Average	1160.5	1180.55	523.1	850.9	10015.1	10238.03	46.23

gp10-01	1157	1163.9	617	849.55	11165	11292.55	63.75
gp10-02	1155	1170	691	1041.85	11151	11397.55	63.44
gp10-03	1141	1157.8	476	775.95	11064	11212.95	63.75
gp10-04	1113	1136.4	484	692.15	10871	11082.85	63.60
gp10-05	1145	1160.05	522	706.75	11155	11271.35	64.38
gp10-06	1148	1190.2	460	815.15	11181	11488.2	63.44
gp10-07	1139	1160.3	519	855.2	11084	11264.6	63.90
gp10-08	1146	1182.8	474	888.95	11142	11529.45	64.07
gp10-09	1147	1164.2	480	779.75	11070	11326.95	63.90
gp10-10	1163	1180.3	491	741.55	11123	11354	63.60
Average	1145.4	1166.595	521.4	814.685	11100.6	11322.04	63.78

The comparison of MOPSO and MOGA for objectives makespan, machine idle time and total flow time are showed in Table 5.16, 5.17, 5.18 respectively.

Table 5. 16 The comparison of MOPSO and MOGA for makespan

	PSO		GA		Error Ratio	
	Makespan		Makespan		Makespan	
	Best	Average	Best	Average	Best	Avgerage
gp03	1167.0	1167.0	1167.0	1167.0	**0**	**0**
gp04	1272.7	1272.7	1273.1	1275.3	**0**	**0**
gp05	1263.6	1263.7	1264.7	1274.2	**0**	**0**
gp06	1271.6	1272.2	1273.7	1280.7	**0**	**0**
gp07	1195.5	1196.2	1204.5	1217.2	**0**	**0**
gp08	1144.1	1150.9	1176.1	1195.1	**0**	**0**
gp09	1124.5	1132.5	1160.5	1180.5	**0**	**0**
gp10	1094.3	1110.3	1145.4	1166.5	**0**	**0**

Table 5. 17 The comparison of MOPSO and MOGA for machine idle time

	PSO		GA		Error Ratio	
	Machine Idle Time		Machine Idle Time		Machine Idle Time	
	Best	Average	Best	Average	Best	Avgerage
gp03	0.0	3.1	0	9.6	**0**	**0**
gp04	162.7	328.1	79.1	285.7	1.05	0.14
gp05	285.6	500.4	251.1	521.2	0.13	**0**
gp06	308.1	620.2	230.7	772.8	0.33	**0**
gp07	248.8	558.0	473.6	754.4	**0**	**0**
gp08	139.8	392.3	501.8	823.9	**0**	**0**
gp09	143.6	363.8	523.1	850.8	**0**	**0**
gp10	50.5	243.3	521.4	814.6	**0**	**0**

Table 5. 18 The comparison of MOPSO and MOGA for total flow time

	PSO		GA		Error Ratio	
	Total Flow Time		Total Flow Time		Total Flow Time	
	Best	Average	Best	Average	Best	Avgerage
gp03	3237.5	3405.0	3189.2	3395.2	0.015	0.0028
gp04	4456.5	4775.2	4479.0	4781.8	**0**	**0**
gp05	5575.2	5847.8	5643.1	5962.4	**0**	**0**
gp06	6788.7	7086.5	6861.1	7184.6	**0**	**0**
gp07	7837.6	8025.2	7888.9	8108.4	**0**	**0**
gp08	8890.8	9013.1	8988.0	9194.8	**0**	**0**
gp09	9899.2	10007.3	10015.1	10238.0	**0**	**0**
gp10	10778.2	10950.7	11100.6	11322.0	**0**	**0**

In order to compare the convergence degree of GA and PSO, the scatter diagrams are plot as Figure 5.1-5.3. The solutions found by the PSO are more condensed than the GA.

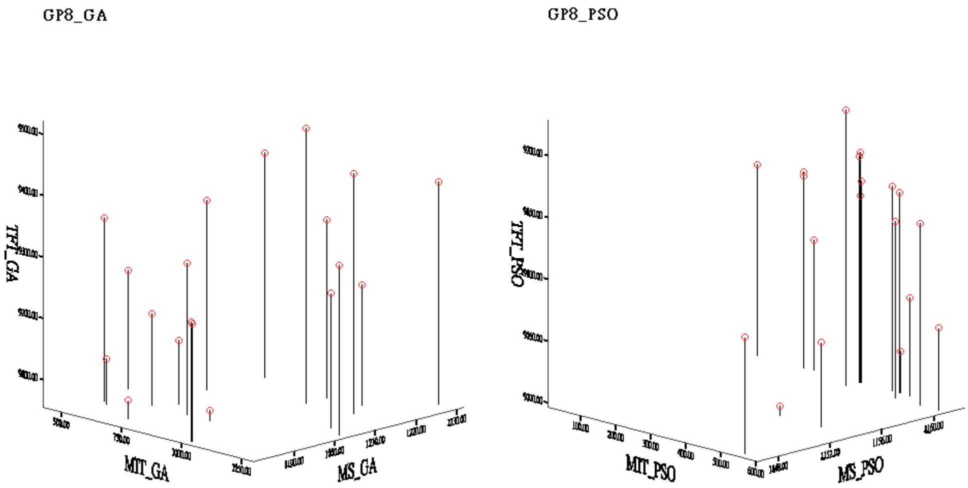

Figure 5. 1 The scatter diagrams of gp8

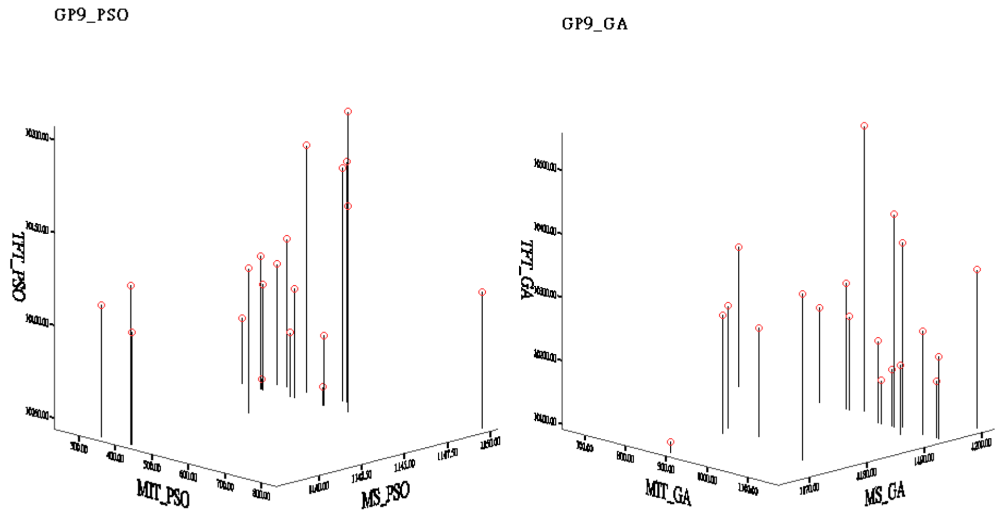

Figure 5. 2 The scatter diagrams of gp9

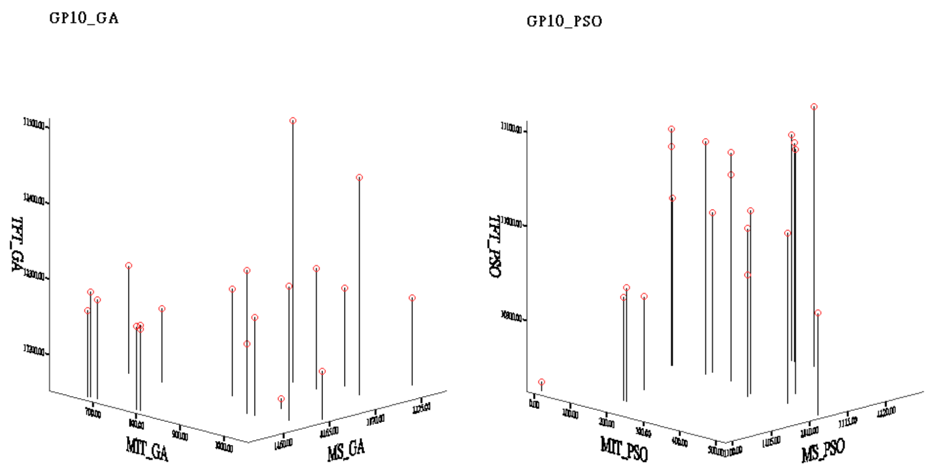

Figure 5. 3 The scatter diagrams of gp10

CHAPTER 6 CONCLUSIONS AND FUTURE STUDIES

6.1 Conclusions

Many studies focused on flowshop scheduling problem could be found. However, the objective of most research focused on minimization of maximum completion time (i.e. makespan). In real world, there exist other objectives such as minimization of machine idle time that might help improve efficiency and reduce production costs. Particle swarm optimization inspired by the spirit of bird flocking and fish schooling behaviors consists with advantages including simple structure, easy implementation, immediate accessibility, short searching time, and robustness. However, limited study of flowshop scheduling problem with multi-objectives addressed by PSO could be found from the literature. We have presented a PSO method for solving flowshop scheduling problem with multiple objectives including minimization makespan, minimization mean flow time and machine idle time.

The original PSO was proposed for the continuous optimization problems. In order to make it suitable for flowshop scheduling (i.e. a combinational problem), we modified the representation of particle position, particle movement, and particle velocity. In addition, a mutation operator was adopted in our PSO algorithm. We also incorporated the concept of Pareto optimal to measure the performance of multiple objectives rather than weighted fitness function. Another necessary adjustment of original PSO to keep Pareto optimal solution is the external Pareto optimal set that is cooperated to deposit a limited size of non-dominated solutions. At last, we utilized a diversification strategy in our PSO algorithm. The results demonstrated that the proposed PSO can obtain more optimal solutions than GA heuristic. The relative error ratios of each problem scenario in our PSO algorithm are less than the GA. The results of performance measure also revealed that the proposed PSO algorithm outperformed GA in minimizing makespan, mean flow time and total machine idle time.

While there has been a large amount of research into the JSSP, most of this has focused on minimizing the maximum completion time (i.e., makespan). There exist other objectives in the real world, such as the minimization of machine idle time that might help improve efficiency and reduce production costs. PSO, inspired by the behavior of birds in flocks and fish in schools, has the advantages of simple structure, easy implementation, immediate accessibility, short search time, and robustness. However, few applications of PSO to multi-objective JSSPs can be found in the literature. Therefore, we presented a MOPSO method for solving the JSSP with multiple objectives, including minimization of makespan, total tardiness, and total machine idle time.

The original PSO was proposed for continuous optimization problems. To make it suitable for job-shop scheduling (i.e., a combinational problem), we modified the representation of particle position, particle movement, and particle velocity. We also introduced a mutation operator and used a diversification strategy. The results demonstrated that the proposed MOPSO could obtain more optimal solutions than the MOGA. The relative error ratios of each problem scenario in our MOPSO algorithm were less than in the MOGA. The performance measure results also revealed that the proposed MOPSO algorithm outperformed MOGA in simultaneously minimizing makespan, total tardiness, and total machine idle time.

Although a large amount of research has addressed the open-shop scheduling problem, most of this has focused on minimizing the maximum completion time (i.e., makespan). Other objectives exist in the real world, such as minimizing the machine idle time, that might help improve efficiency and reduce production costs. PSO, inspired by the behavior of flocks of birds and schools of fish, has the advantages of a simple structure, easy implementation, immediate accessibility, short search time, and robustness. However, few applications of PSO to multi-objective open-shop scheduling problems can be found in the literature. Therefore, we proposed a MOPSO algorithm to solve the open-shop scheduling problem with multiple objectives, including minimization of makespan, total flow time, and machine idle time.

The algorithm was tested to verify different scenarios, using different Pareto sets with different combinations of objectives. Different swarm sizes with varied objective combinations were also evaluated. The results demonstrated that the algorithm performed better when only one swarm was used for all three objectives compared to the case where the swarm was divided into three sub-swarms for each objective.

6.2 Future Studies

For further research, we will attempt to apply our PSO to other shop scheduling problems with multiple objectives. Possible topics for further study include the modification of particle position representation, particle movement, and particle velocity. In addition, issues related to Pareto optimal such as solution maintenance strategy and performance measurement are also worth to be investigated in future.

We will also attempt to apply MOPSO to other shop scheduling problems with multiple objectives in future research. Other possible topics for further study include modification of the particle position, particle movement, and particle velocity representation. Issues related to Pareto optimization, such as solution maintenance strategy and performance measurement, also merit future investigation.

Appendix

<u>The pseudo-code of the PSO for MO-FSSP is as follow.</u>

Initialize a population of particles with random positions.

for each particle *k* ***do***

    Evaluate X^k (the position of particle k)

    Save the $pbest^k$ to optimal solution set S

end for

Set *gbest* solution equals to the best $pbest^k$

repeat

    Updates particles velocities

    for each particle *k* ***do***

        Move particle *k*

Evaluate X^k

Update *gbest*, *pbest* and S

    end for

until maximum iteration limit is reached

<u>The pseudo code of the PSO for MO-JSSP is given below:</u>

Initialize a population of particles with random positions.

for each particle *k* ***do***

    Apply G&T algorithm to decode X^k into a schedule S^k.

    set the k^{th} *pbest* solution ($pbest^k$) equal to S^k, $pbest^k \leftarrow S^k$.

end for

set *gbest* solution equal to the best $pbest^k$.

repeat

update velocities

for each particle *k* ***do***

move particle *k*

apply G&T algorithm to decode x^k into S^k.

update *pbest* solutions and *gbest* solution

end for

until maximum iterations is attained

The pseudo code of the PSO for MO-OSSP is given below:

Initialize a population of particles with random positions.

for each particle *k* ***do***

Apply G&T algorithm to decode X^k into a schedule S^k.

set the k^{th} *pbest* solution ($pbest^k$) equal to S^k, $pbest^k \leftarrow S^k$.

end for

set *gbest* solution equal to the best $pbest^k$.

repeat

update velocities

for each particle *k* ***do***

move particle *k*

apply G&T algorithm to decode x^k into S^k.

update *pbest* solutions and *gbest* solution

end for

until maximum iterations is attained

References

Adulbjan P., M. T. Tabucanon (1980). Multicriterion Optimization in Industrial Systems. Chapter 9, *Decision Models for Industrial System Engineers and Managers*, Asian Institute of Technology, Bangkok

Bean, J. (1994). Genetic algorithms and random keys for sequencing and optimization. *Operations Research Society of America (ORSA) Journal on Computing*, 6, 154–160.

Bllzewicz J., Pesch E., Sterna M., Werner F. (2004). Open shop scheduling problems with late work criteria. *Discrete Applied Mathematics*, 134 (1-3), 1-24.

Blum C. (2005). Beam-ACO—hybridizing ant colony optimization with beam search: an application to open shop scheduling. *Computers & Operations Research*, 32, 1565–1591.

Campbell H.G., Dudek R.A. & Smith M.L. (1970). A heuristic algorithm for the n-job m,_machine sequencing problem. *Management Science*, 16, B630–B637.

Candido M. A. B., S. K. Khator & R.M. Barcia. (1998). A genetic algorithm based rocedre for more realistic job shop scheduling problems. *International Journal of Production Research*, 36(12), 3437-3457.

Carlier J. (1978). Ordonnancements à contraintes disjonctives. RAIRO Rech Oper/Oper Res 12, 333–351.

Chakravarthy K. & Rajendran C. (1999). A heuristic for scheduling in a flowshop with the bicriteria of makespan and maximum tardiness minimization. *Production Planning and Control*, 10, 707-714.

Coello Coello Carlos A., Gregorio Toscano Plido & Maximino Salazar Lechga. (2004). Handling Mltiple Objectives With article Swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3), 256-278.

Eren T. & Güner E. (2007). The tricriteria flowshop scheduling problem. *International Journal of Advanced Manufacturing Technology*, 36, 1210–1220.

Esquivel S. C., S. W. Ferrero & R. H. Gallard, (2002). Parameter Settings and Representations in Pareto-based Optimization for Job Shop Scheduling. *Cybernetics and Systems: An international Journal*, 33, 559-578.

Fisher, H., & Thompson, G. L., (1963). Industrial scheduling. Englewood Cliffs. *NJ:*

Prentice-Hall.

Gangadharan R. & Rajendran C. (1993) Heuristic algorithms for scheduling in no-wait flow shop. *International Journal of Production Economy,* 32, 285-290.

Garey, M. R., Johnson, D. S., & Sethi, R., (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research,* 1, 117–129.

Giffler, J., Thompson, G. L., (1960). Algorithms for solving production scheduling problems. *Operations Research,* 8, 487–503.

Gonc¸alves, J. F., Mendes, J. J. M., & Resende, M. G. C., (2005). A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research,* 167(1), 77–95.

Gonzalez T, Sahni S. (1976). Open shop scheduling to minimize finish time. *Journal of the ACM*, 23(4), 665–79.

Gupta J.N.D. & Stafford J.E.F. (2006). Flowshop scheduling research after five decades. *European Journal of Operational Research*, 169, 699–711.

Hejazi S.R. & Saghafian S. (2005). Flowshop scheduling problems with makespan criterion: a review. *International Journal of Production Research*, 43, 2895–2929.

Heller J. (1960). Some numerical experiments for an MxJ flow shop and its decision-theoretical aspects. *Operations Research*, 8, 178–184.

Hwang C. L., A. S. M. Masud, S. R. Paidy & K. Yoon (1982). Mathematical Programming with Multiple Objectives: A Tutorial. *Computers and Operations Research: A Special Issue on Mathematical Programming with Multiple Objective*, 7(1-2).

Jain A. S. & S. Meeran, (1999). Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research*, 113, 390-434.

Jarboui B., Ibrahim S., Siarry P. & Rebai A. (2008). A combinational particle swarm optimisation for solving permutation flowshop problems. *Computers & Industrial Engineering,* 54, 526–538.

Kennedy J. & Eberhart R. (1995). Particle swarm optimization. *Proceedings of the IEEE International Conference on Neural Networks 1995,* 1942–1948.

Knowles J.D. & Corne D.W. (1999). The Pareto archived evolution strategy: a new baseline algorithm for multi-objective optimization. In: *Congress on*

Evolutionary Computation, Washington, DC, IEEE Service Center, 98–105

Kobayashi, S., Ono, I., & Yamamura, M., 1995, "An efficient genetic algorithm for job shop scheduling problems", In L. J. Eshelman (Ed.), *Proceedings of the sixth international conference on genetic algorithms,* San Francisco, CA: Morgan Kaufman Publishers, 506–511.

Laha D. & Chakraborty U.K. (2008) An efficient heuristic approach to total flowtime minimization in permutation flowshop scheduling. *International Journal of Advanced Manufacturing Technology,* 38, 1018-1025.

Laha D. & Chakraborty U.K. (2008) A constructive heuristic for minimizing makespan in no-wait flow shop scheduling. *International Journal of Advanced Manufacturing Technology,* DOI 10.1007/s00170-008-1545-0

Lawrence, S. (1984). Resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques. Graduate School of Industrial Administration (GSIA), Carnegie Mellon University, Pittsburgh, PA.

Lei Deming (2008). A Pareto archive particle swarm optimization for multi-objective job shop scheduling. *Computers & Industrial Engineering,* 54(4), 960-971.

Lei Deming & Zhiming Wu (2006). Crowding-measure-based multiobjective evolutionary algorithm for job shop scheduling. *International Journal of Advanced Manufacturing Technology,* 30, 112-117.

Lian Z., Gu X. & Jiao B. (2008). A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan. *Chaos, Solutions and Fractals,* 35, 851–861.

Liang Y. C., H. W. GE, Y. Zho & X. C. GUO (2005). A Particle Swarm Optimization-based Algorithm for Job-shop Schedling Problems. *International Journal of Computational Methods,* 2(3), 419-430.

Liaw C-F. (1999) Applying simulated annealing to the open shop scheduling problem. *IIE Transactions*, 31, 457–65.

Liaw C-F. (1999) A tabu search algorithm for the open shop scheduling problem. *Computers & Operations Research*, 26, 109–26.

Liaw C-F. (2000) A hybrid genetic algorithm for the open shop scheduling problem. *European Journal of Operational Research*, 124, 28–42.

Liu B., Wang L. & Jin Y.H. (2007). An effective PSO-based memetic algorithm for flow shop scheduling. *IEEE Transaction on System Man and Cybernetics- Part C,* 37, 18–27.

Liu J. & Reeves C.R. (2001). Constructive and composite heuristic solutions to the P//∑Ci scheduling problem. *European Journal of Operational Research*, 132, 439-452.

Lourenc¸o, H. R. (1995). Local optimization and the job-shop scheduling problem. *European Journal of Operational Research*, 83, 347–364.

Nawaz M., Enscore J.R. & Ham I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11, 91–95.

Nowicki, E. & Smutnicki, C., (1996). A fast taboo search algorithm for the job shop problem. *Management Science,* 42(6), 797–813.

Pasupathy T., Rajendran C., & Suresh R.K. (2006). A multi-objective genetic algorithm for scheduling in flow shops to minimize the makespan and total flow time of jobs. *International Journal of Advanced Manufacturing Technology,* 27, 804–815.

Pezzella F. & Merelli E. (2000). A tabu search method guided by shifting bottleneck for the job shop scheduling problem. *European Journal of Operational Research*, 120(2), 297–310.

Ponnambalam S.G., Jagannathan H. & Kataria M. (2004). A TSP-GA multi-objective algorithm for flow-shop scheduling. *International Journal of Advanced Manufacturing Technology,* 23, 909–915.

Ponnambalam S. G., V. Ramkumar & N. Jawahar, (2001). A multiobjective genetic algorithm for job shop scheduling. *Production Planning and Control,* 12(8), 764-774.

Rahimi-Vahed A. & Mirghorbani S. (2007). A multi-objective particle swarm for a flow shop scheduling problem. *Journal of Combination Optimzation,* 13, 79–102.

Rajendran C. (1994). A no-wait flow shop scheduling heuristic to minimize makespan. *Journal of the Operational Research Society,* 45, 472-478.

Rajendran C. & Ziegler H. (2004). Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. *European Journal of Operational Research*, 155, 426–438.

Reeves C.R. (1995). A genetic algorithm for flowshop sequencing. *Computers & Operations Research,* 22, 5–13.

Ripon Kazi Shah Nawaz (2007). Hybrid Evolutionary Approach for Multi-objective Job-shop Scheduling Problem. *Malaysian journal of Computer Science,* 20(2), 183-198.

Prins C. (2000) Competitive genetic algorithms for the open-shop scheduling problem. *Mathematical Methods of Operations Research*, 52, 389–411.

Ruiz R. & Maroto C. (2004). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, 165, 479–494.

Senthilkumar P., Shahbudeen P. (2006) GA based heuristic for the open job shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, 30, 297–301.

Seo Fumiko & Masatoshi Sakawa (1988). *Multiple Criteria Decision Analysis in Regional Planning : Concepts, Methods and Applications*, Reidel, Boston.

Sha D.Y. & Hsu C.Y. (2006). A hybrid particle swarm optimization for job shop scheduling problem. *Computers & Industrial Engineering,* 51, 791–808.

Sha, D.Y. & Hsu C.Y. (2008). A new particle swarm optimization for the open shop scheduling problem. *Computers & Operations Research*, 35, 3243–3261.

Stützle T. (1998). Applying iterated local search to the permutation flow shop problem. *Tech Rep, AIDA-98-04, FG Intellektik,* TU Darmstadt.

Sun, D., Batta, R., & Lin, L., (1995), Effective job shop scheduling through active chain manipulation. *Computers & Operations Research*, 22(2), 159–172.

Suresh R. K. & K. M. Mohanasndaram (2006). Pareto archived simulated annealing for job shop scheduling with multiple objectives. *International Journal of Advanced Manufacturing Technology,* 29, 184-196.

Taillard E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64, 278-285.

Tang Lixin, Bai Danyu. (2010). A new heuristic for open shop total completion time problem. *Applied Mathematical Modeling* , 34, 735-743.

Tasgetiren M.F., Liang Y.C., Sevkli M. & Gencyilmaz G. (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. *European Journal of Operational*

Research, 177, 1930–1947.

Wang Ling & Da-Zhong Zheng (2001). An effective hybrid optimization strategy for job-shop scheduling problems. *Computers & Operations Research*, 28, 585-596.

Xia Weijn & Zhiming Wu, (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, 48, 409-425.

Yagmahan B. & Yenisey M.M. (2008). Ant colony optimization for multi-objective flow shop scheduling problem. *Computers & Industrial Engineering* 54, 411–420.

Young M., The Techincal Writers Handbook. Mill Valley, CA: University Science, 1989.

Zhang H., Li X., Li H. & Huang F. (2005). Particle swarm optimization-based schemes for resource-constrained project scheduling. *Automation in Construction*, 14(3), 393–404.

Zitzler E., Laumanns M. & Thiele L. (2001) SPEA2: Improving the strength Pareto evolutionary algorithm. Computer Engineering and Networks Laboratory (TIK) – Report 103 Sept 2001.

作者簡介

姓名：林信宏 (Hsing-Hung Lin)

學歷：

學士	東海大學	資訊科學系	(79.9~83.6)
碩士	中華大學	工業工程與管理研究所	(85.9~87.6)
博士	交通大學	工業工程與管理研究所	(93.9~ 99.6)

著作：

一、 期刊論文

1. D.Y. Sha, Hsing-Hung Lin, 2009, "A particle swarm optimization for multi-objective flowshop scheduling," *International Journal of Advanced Manufacturing Technology*, 45(7), 749-762. (SCI)
2. D.Y. Sha, Hsing-Hung Lin, 2010, "A multi-objective PSO for job-shop scheduling problem," *Expert Systems with Applications*, 37(2), 1065-1070. (SCI)

二、 審查中論文

1. D.Y. Sha, Hsing-Hung Lin, "A multi-objective PSO for open-shop scheduling problem," submit to *Journal of Information & Optimization Sciences*
2. D.Y. Sha, Hsing-Hung Lin, "A novel particle swarm optimization for multi-objective job-shop scheduling ," submit to *Journal of Industrial and Management Optimization*
3. D.Y. Sha, Hsing-Hung Lin, "Scheduling multi-objective jobshops using particle swarm optimization," submitted to *Journal of Intelligent Manufacturing*

三、 研討會論文

1. D.Y. Sha, Hsing-Hung Lin, 2008, "A Multi-objective Particle Swarm Optimization for Flow Shops Scheduling Problem," *Proceedings of the 38th International Conference on Computers and Industrial Engineering*, pp. 233-241, Beijing, China.
2. D.Y. Sha, Hsing-Hung Lin, 2008, "A Pareto Archive Particle Swarm Optimization for Multi-objective Flowshop Scheduling," *Proceedings of the*

2008 Asia Pacific Industrial Engineering & Management Systems Conference, pp.2269-2277, Bali, Indonesia.

3. D.Y. Sha, Hsing-Hung Lin, 2009, "A Multi-objective PSO for Job-shop Scheduling Problem," *Proceedings of the 39th International Conference on Computers and Industrial Engineering*, Troyes, France.
4. D.Y. Sha, Hsing-Hung Lin, 2009, "A Novel Particle Swarm Optimization for Multi-objective Job-shop Scheduling," *Proceedings of the 2009 Asia Pacific Industrial Engineering & Management Systems Conference*, Kitakyushu, Japan.
5. D.Y. Sha, Hsing-Hung Lin, 2010, "A Modified Particle Swarm Optimization for Multi-objective Open-shop Scheduling," *Proceedings of the 2010 AENG International Conference on Industrial Engineering*, Hong Kong.