# 行政院國家科學委員會專題研究計畫 成果報告

## 粒子群最佳化於多目標排程問題之應用(第 3 年)
## 研究成果報告(完整版)

計 畫 類 別 ：個別型
計 畫 編 號 ：NSC 96-2221-E-216-052-MY3
執 行 期 間 ：98 年 08 月 01 日至 99 年 07 月 31 日
執 行 單 位 ：中華大學工業管理學系

計 畫 主 持 人：沙永傑

計畫參與人員：博士班研究生-兼任助理人員：林信宏

報 告 附 件 ：出席國際會議研究心得報告及發表論文

處 理 方 式 ：本計畫可公開查詢

中 華 民 國 99 年 09 月 24 日

行政院國家科學委員會補助專題研究計畫成果報告

# 粒子群最佳化於多目標排程問題之應用

計畫類別：■個別型計畫　　□整合型計畫

計畫編號：NSC　96－2221－E－216－052－MY3

執行期間：　96 年　8 月　1 日 至　99 年　7 月　31 日

計畫主持人：沙永傑

執行單位：中華大學工業工程與管理學系

中　華　民　國　九十九　年　九　月　十　日

# 行政院國家科學委員會專題研究計畫成果報告

# 粒子群最佳化於多目標排程問題之應用

# A Particle Swarm Optimization for Multi-Objective Scheduling Problems

計 畫 編 號：NSC 96-2221-E-216-052MY3
執 行 期 限：96 年 8 月 1 日至 99 年 7 月 31 日
主 持 人：沙永傑 中華大學工業工程與管理學系

## 一、中文摘要

粒子群最佳化(Particle Swarm Optimization, PSO)是一種以群體搜尋為基礎的最佳化演算法，其群體是由各自獨立的粒子所組成。PSO 模擬空間中的粒子運動，以搜尋空間(search space)表達出問題的解集合空間(solution space)。每個搜尋空間中的位置(position)對應到一個該問題解集合空間中的解。粒子群於搜尋空間中(解集合空間中)合作搜尋出最佳位置(最佳解)。PSO 最初應用於求解連續性最佳化問題。然而近年來也被應用於求解組合最佳化問題。

多數與排程問題有關的研究都只探討如何使單一目標最佳化，如：總完工時間、總延遲成本、延遲工件數...等。在現實情況下，決策者必需作出能夠將這些目標最佳化的決策，卻又面臨這些目標之間互相衝突的問題。若只追求單一目標的最佳化排程，則會造成另一目標的損失。本研究的主要目的在架構出能求解多目標排程問題之 PSO，幫助決策者在面對如此複雜排程問題時能夠做出合理決策。

本研究以三年時間分階段進行。第一年以流程型排程問題(flow shop scheduling problem)為研究對象；第二年以零工式排程問題(job shop scheduling problem)為研究對象；第三年以開放型排程問題(open shop scheduling problem)為研究對象。在這三種最基本的排程問題中，我們以完工時間(makespan)、總流程時間(total flow time)、總延遲時間(total tardiness)作為目標函數。本研究所提出的 PSO，應用於標竿測試問題與過去的啟發式演算法，在求解的速度與品質上有較佳的表現。

**關鍵詞：：粒子群最佳化、流程型排程、零工型排程、開放型排程、多目標排程**

## Abstract

Particle Swarm Optimization (PSO) is a population-based optimization algorithm. Each particle is an individual and the swarm is composed of particles. PSO mimics the particle movement in a space. In PSO, the problem solution space is formulated as a search space.

Each position in the search space is a correlated solution of the problem. Particles cooperate to find out the best position (best solution) in the search space (solution space). The original intention of PSO is to solve continuous optimization problems. However, it has been implemented to solve many combinatorial optimization problems in recent years.

In most previous research about scheduling,

there are only one objective function. For example, total complete time, total tardiness, and maximum makespan. In fact, the decision makers have to simultaneously optimize these objectives, but there are conflicts between these objectives. If we only optimize one of these objectives, we will loose another objective. In this research, we will construct a PSO to solve multi-objective scheduling problems. That can help decision makers to make a strategy to handle such complex scheduling problems.

We will execute this research in three years. In the first year, we focus on the flow shop scheduling problem. In the second year, we focus on the job shop scheduling problem. In the third year, we focus on the open shop scheduling problem. In these three scheduling problems, the objective functions are maximum completion time, total weighted completion time, and total weighted tardiness. Moreover, we will add some constraints into the scheduling problems, such as limited intermediate storage and dependent setup time.

**Keywords**: Particle swarm optimization, Multi-objective scheduling problem

## 二、研究目的

多數與排程問題有關的研究都只探討如何使單一目標最佳化，如：總完工時間、總延遲成本、延遲工件數...等。在現實情況下，決策者必需作出能夠將這些目標最佳化的決策，卻又面臨這些目標之間互相衝突的問題。若只追求單一目標的最佳化排程，則會造成另一目標的損失。

大多數排程問題皆為 NP 問題，也就是說，此問題無法在合理的計算時間內求出最佳解。因此，學者們發展出各類型啟發式解法試圖求出近似最佳解，而演化式計算(Evolution Computation)則是近十年來最被廣泛應用，也是求解效率最佳的一種啟發式解法。在演化式計算中又以粒子群最佳化(Particle Swarm Optimization, PSO)為最新、且為近 3 年來最被廣泛研究及討論的演算法。另於一般單目標的排程問題中，我們已將 PSO 應用於 JobShop 及 Open Shop 問題上。研究發現應用 PSO 求解會得到比其它方法更好的結果(Sha & Hsu, 2006a, 2006b)。因此，本研究的主要目的在架構出能求解多目標排程問題之 PSO，幫助決策者在面對此複雜的排程問題時能夠做出合理的決策。

因此本研究的主要目的是架構出適合求解多目標排程問題的 PSO。除此之外，由於 PSO 原本是用來求解連續性最佳化問題(continues optimization problems)，在組合最佳化方面的應用尚未成熟，還有許多可供研究及改良的空間。排程問題也是組合最佳化問題之一，因此本研究所發展之 PSO 將為後續 PSO 應用於組合最佳化問題研究之基礎。我們也將以本研究為基礎，將本研究所發展之 PSO 應用於求解其它組合最佳化問題。

## 三、粒子群演算法相關文獻

粒子群最佳化 (Particle Swarm Optimization, PSO) 由 Kennedy 和 Eberhart 於 1995 年提出。它是由模仿粒子於空間中運動的行為，而發展出的最佳化方法。PSO 和基因演算法(Genetic Algorithm,GA)相同，都是以群體為基礎的(population-based)最佳化演算法。在 PSO 中，群體(swarm)是由粒子(particle)所組成。群體(swarm)和粒子(particle)的關係，和在 GA 中母體(population) 和染色體(chromosome)的關係相似。

PSO 以搜尋空間(search space)表達出問題的解集合空間(solution space)。每個搜尋空間中的位置(position)對應到一個該問題解集合空間中的解。粒子群於搜尋空間中(解集合空間中)合作搜尋出最佳位置(最佳解)。而其中粒子運動主要受到三個因素影響： 慣性

(inertia)、個體最佳位置(pbest position)、群體最佳位置(gbest position)。慣性(inertia)為粒子於上一循環(iteration) 所殘留下來的速度(velocity)，它能夠由慣性權重(inertia weight)控制。慣性(inertia)的用意在於防止粒子停留在同一範圍內，而能跳出區域最佳解(local optima)。個體最佳位置(pbest solution)為粒子本身到目前為止所搜尋出的最佳位置(或最佳解)，因此每個粒子有它自己的個體最佳位置(pbest position)。群體最佳位置(gbest position)則是到目前為止群體(swarm)所搜尋出的最佳位置(最佳解)，而整個群體(swarm)只會有一個群體最佳位置(gbest position)。

在 PSO 中粒子速度(velocity)由一向量表示，而在每個循環(iteration)中粒子根據它所擁有的速度移動它的位置。在每個循環(iteration)裡，粒子朝著個體最佳位置(pbest position)以及群體最佳位置(gbest position)移動，而其速度也是根據個體最佳位置(pbest position)及群體最佳位置(gbest position)隨機求得。

## 四、研究成果

本研究已完成求解流程型排程問題(Flow-shop Scheduling Problem)之 PSO，為了證明所發展的 PSO 的適用性、求解品質與效率，我們將本研究的 PSO 與 TSP-GA(Ponnambalam, 2004)同時測試 21 個問題進行比較。目標式有三個，分別為最小完工時間(Makespan)、平均流程時間(Mean Flow Time)與機器閒置時間(Machine Idle Time)。運算的結果顯示，以平均相對誤差而言，針對最小完工時間，PSO 有 17 個問題表現優於 TSP-GA；平均流程時間，PSO 有 18 個問題表現優於 TSP-GA；機器閒置時間，PSO 有 19 個問題表現優於 TSP-GA。綜合而言，PSO 在 21 個案例中的 19 個，同時達到三個目標且優於 TSP-GA 。此結果已發表於 The 38th International Conference on Computers and Industrial Engineering (2008).

另一方面，亦將 PSO 與傳統的啟發式演算法 CDS 與 NEH 進行比較。同樣以最小完工時間(Makespan)、平均流程時間(Mean Flow Time)與機器閒置時間(Machine Idle Time)作為目標式。此次共測試 161 個標竿問題，包含 Rec01 至 Rec41，Tai20x5 至 Tai500x20。求解的結果顯示 PSO 比傳統的啟發式演算法 CDS 與 NEH 具備明顯的優勢。結果發表於 The 9th Asia Pacific Industrial Engineering & Management Systems Conference (2008).

PSO 應用於求解多目標流程型排程問題的結果已發表於 International Journal of Advanced Manufacturing Technology Vol. 45, No. 7, pp. 749-762. Vol. 37, No. 2, pp. 1065-1070, 2009 (SCI)(如附錄一).

本研究已完成求解零工式排程問題(Job-shop Scheduling Problem)之 PSO，同樣的，我們選擇三個目標式，最小完工時間(Makespan)、總延遲時間(Total Tardiness)與機器閒置時間(Machine Idle Time)。比較的對象則是 MOGA (Ponnambalam, 2001)，在 23 個標竿問題中，PSO 在最小完工時間與總延遲時間，完全領先 MOGA；至於機器閒置時間則於 22 個問題中具競爭力。

PSO 應用於求解多目標零工型排程問題的結果已發表於 Expert Systems with Applications, Vol. 37, No. 2, pp. 1065-1070, 2010 (SCI) (如附錄二).

對於開放式排程問題，本研究亦完成 PSO 設計，我們以最小完工時間(Makespan)、總流程時間(Total Flow Time)與機器閒置時間(Machine Idle Time) 作為目標式。由於開放式排程問體文獻相對較少，本研究另外設計並撰寫基因演算法作為比較，並以 Guéret and Prins (1999)標竿問題進行測試，求解結果顯示本驗就提出的 PSO 整體績效大於 GA。此結果已投稿於 Journal of Industrial and Management Optimization.

## 五、參考文獻


Coello, C.A., & Lechuga, M.S. (2002). “MOPSO: a proposal for multiple objective particle swarm optimization,” Proceedings of the 2002 Congress on Evolutionary Computation, Vol. 2, 1051-1056.

Giffler J & Thompson G..L. (1960). “Algorithms for solving production scheduling problems,”Operations Research ,Vol. 8, 487-503.

Goldberg D.E. (1989). Genetic algorithms in search, optimization and machine learning. Reading, MA: Addison-Wesley.

Gonçalves J.F., Mendes .J.J de M., & Resende M.G.C. (2005). “A hybrid genetic algorithm for the job shop scheduling problem,” European Journal of Operational Research, Vol. 167, No. 1, 77-95.

Hu, X., & Eberhart, R.C. (2002). “Multiobjective optimization using dynamic neighborhood particle swarm optimization,” Proceedings of the 2002 Congress on Evolutionary Computation, Vol. 2,1677-1681.

Kennedy, J., & Eberhart, R.C. (1995). “Particle swarm optimization,” In: Proceedings of the 1995 IEEE International Conference on Neural Networks, 4 (pp. 1942-1948). Piscataway, NJ: IEEE Press.

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

Lourenço, H.R. (1995). “Local optimization and the job-shop scheduling problem,” European Journal of Operational Research, 83, 347-364.

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

Ponnambalam, S. G., H. Jagannathan, et al. (2004), "A TSP-GA multi-objective algorithm for flow-shop scheduling", International Journal of Advanced Manufacturing Technology 23(11), 909-915.

Sha, D.Y., & Hsu, C.-Y. (2006a). “A Hybrid particle swarm optimization for job shop scheduling problem,” Computers & Industrial Engineering, Vol. 51, No. 4, pp. 791-808.

Sha D.Y., & Hsu, C-Y. (2006b). “A modified parameterized active schedule generation algorithm for the job shop scheduling problem,” Proceedings of the 36th International Conference on Computers and Industrial Engineering (ICCIE 2006) (pp. 702-12).

Shi, Y., & Eberhart, R.C. (1998a). “Parameter selection in particle swarm optimization,” In V.W. Porto, N. Saravanan, D. Waagen, & A.E. Eiben (eds),

Proceedings of the 7th International Conference on Evolutionary Programming (pp. 591-600). New York: Springer-Verlag.

Shi, Y., & Eberhart, R.C. (1998b). “A modified particle swarm optimizer,” In: D. Fogel, Proceedings of the 1998 IEEE International Conference on Evolutionary Computation (pp. 69-73). Piscataway, NJ: IEEE Press.

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

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

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

Zhang, L.B., Zhou, C.G., Liu, X.H., Ma, Z.Q., Ma, M., & Liang, Y.C. (2003). “Solving multi objective optimization problems using particle swarm optimization,” Proceedings of the 2003 Congress on Evolutionary Computation, Vol. 4, 2400 – 2405.


## 六、附錄

1. A particle swarm optimization for multi-objective flow-shop scheduling
2. A multi-objective PSO for job-shop scheduling problems

Int J Adv Manuf Technol
DOI 10.1007/s00170-009-1970-6

ORIGINAL ARTICLE

# A particle swarm optimization for multi-objective flowshop scheduling

**D. Y. Sha · Hsing-Hung Lin**

Received: 5 September 2008 / Accepted: 6 February 2009


**Abstract** The academic approach of single-objective flowshop scheduling has been extended to multiple objectives to meet the requirements of realistic manufacturing systems. Many algorithms have been developed to search for optimal or near-optimal solutions due to the computational cost of determining exact solutions. This paper provides a particle swarm optimization-based multi-objective algorithm for flowshop scheduling. The proposed evolutionary algorithm searches the Pareto optimal solution for objectives by considering the makespan, mean flow time, and machine idle time. The algorithm was tested on benchmark problems to evaluate its performance. The results show that the modified particle swarm optimization algorithm performed better in terms of searching quality and efficiency than other traditional heuristics.

**Keywords** PSO · Multi-objective · Flowshop scheduling · Pareto optimal

## 1 Introduction

Production scheduling in real environments has become a significant challenge in enterprises maintaining their competitive positions in rapidly changing markets. Flowshop 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 flowshop scheduling problems and possible approaches to their solution over the last 50 years has been provided by Gupta and Stafford [5]. That survey indicated that most research on flowshop 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 flowshop 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 et al. [7]. Ruiz et al. [24] also presented a review and comparative evaluation of heuristics and meta-heuristics for permutation flowshop problems with the makespan criterion. The NEH algorithm [17] has been shown to be the best constructive heuristic for Taillard's benchmarks [28] while the iterated local search [27] method and the genetic algorithm (GA) [23] are better than other meta-heuristic algorithms.

Most studies of flowshop 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.

The English in this document has been checked by at least two professional editors, both native speakers of English. For a certificate, see: http://www.textcheck.com/cgi-bin/certificate.cgi?id=emRe2r

D. Y. Sha
Department of Industrial Engineering and System Management,
Chung Hua University,
Hsinchu, Taiwan, Republic of China

H.-H. Lin (✉)
Department of Industrial Engineering and Management,
National Chiao Tung University,
Hsinchu, Taiwan, Republic of China
e-mail: hsinhung@gmail.com

Published online: 24 February 2009

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 [15]. Ponnambalam et al. [19] 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 flowshop scheduling problem. Rajendran et al. [21] approached the problem of scheduling in permutation flowshop using two ant colony optimization (ACO) approaches, first to minimize the makespan, and then to minimize the sum of the total flow time. Yagmahan [30] was the first to apply ACO meta-heuristics to flowshop scheduling with the multiple objectives of makespan, total flow time, and total machine idle time.

The literature on multi-objective flowshop 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 [18]. 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 et al. [4] tackled a multi-criteria two-machine flowshop scheduling problem with minimization of the weighted sum of total completion time, total tardiness, and makespan.

Particle swarm optimization (PSO) is an evolutionary technique for unconstrained continuous optimization problems proposed by Kennedy et al. [10] The PSO concept is based on observations of the social behavior of animals such as birds in flocks, fish in schools, and swarm theory. To minimize the objective of maximum completion time (i.e., the makespan), Liu et al. [15] invented an effective PSO-based memetic algorithm for the permutation flowshop scheduling problem. Jarboui et al. [9] developed a PSO algorithm for solving the permutation flowshop scheduling problem; this was an improved procedure based on simulated annealing. PSO was recommended by Tasgetiren et al. [29] to solve the permutation flowshop scheduling problem with the objectives of minimizing makespan and the total flow time of jobs. Rahimi-Vahed et al. [22] tackled a bi-criteria permutation flowshop 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 flowshop scheduling problem, Lian et al. [14] addressed permutation flowshop scheduling with a minimized makespan using a novel PSO. All these approaches have demonstrated the advantages of the PSO method: simple structure, immediate applicability to practical problems, ease of implementation, quick solution, and robustness.

The aim of this paper is to explore the development of PSO for elaborate multi-objective flowshop scheduling problems. The original PSO was used to solve continuous optimization problems. Due to the discrete solution spaces of scheduling optimization problems, we modified the particle position representation, particle movement, and particle velocity in this study.

The remainder of this paper is organized as follows. Section 2 contains a formulation of the flowshop scheduling problem with two objectives. Section 3 describes the algorithm of the proposed PSO approach. Section 4 contains the simulated results of benchmark problems. Section 5 provides some conclusions and future directions.

## 2 Problem formulation

The problem of scheduling in flowshops has been the subject of much investigation. The primary elements of flowshop 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 flowshop scheduling problem using the following notation:

- $n$ is the total number of jobs to be scheduled,
- $m$ is the total number of machines in the process,
- $t(i,j)$ is the processing time for job $i$ on machine $j$ ($i$=1, 2,…n) and ($j$=1,2,…m), and
- $\{\pi_1, \pi_2, \ldots, \pi_n\}$ is the permutation of jobs.

The objectives considered in this paper can be calculated as follows:

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

$$
\begin{aligned}
&C(\pi_1,1) = t(\pi_1,1)\\
&C(\pi_i,1) = C(\pi_{i-1},1) + t(\pi_i,1) i = 2,\ldots,n\\
&C(\pi_1,j) = C(\pi_1,j-1) + t(\pi,j) j = 2,\ldots,m\\
&C(\pi_i,j) = \max\{C(\pi_{i-1},j), C(\pi_i,j-1)\} + t(\pi_i,j)\\
&\qquad i = 2,\ldots,n;\ j = 2,\ldots,m
\end{aligned}
$$

- Makespan, $f_{C\max} = C(\pi_n, m)$,
- Mean flow time, $f_{\mathrm{MFT}} = \left[\sum_{i=1}^{n} C(\pi_i, m)\right] \Big/ n$,

- Machine idle time, and
- $f_{MIT} = \{C(\pi_1, j-1) + \sum_{i=2}^{n} \{\max\{C(\pi_i, j-1) - C(\pi_{i-1}, j), 0\}\} | j = 2...m\}$

## 3 Basic PSO concept

PSO is an evolutionary technique (Kennedy et al. [10]) for solving unconstrained continuous optimization problems. The PSO concept is based on observations of the social behavior of animals. The population consisting of individuals (particles) is assigned a randomized initial velocity according each individual's own movement experience and that of the rest of the population. The relationship between the swarm and the particles in PSO is similar to the relationship between the population and the chromosomes in the GA.

The PSO 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 determine the best position (solution) in the search space (solution space).

Suppose that the search space is $D$-dimensional and there are $m$ particles in the swarm. Each particle is located at position $X_i=\{x_{i1}, x_{i2}, \ldots, x_{iD}\}$ with velocity $V_i=\{v_{i1}, v_{i2}, \ldots, v_{iD}\}$, where $i=1, 2, \ldots, m$. In the PSO algorithm, each particle moves toward its own best position (pbest) denoted as $\text{Pbest}_i=\{\text{pbest}_{i1}, \text{pbest}_{i2}, \ldots, \text{pbest}_{in}\}$. The best position of the whole swarm (gbest) denoted as $\text{Gbest}=\{\text{gbest}_1, \text{gbest}_2, \ldots, \text{gbest}_n\}$ with each iteration. Each particle changes its position according to its velocity, which is randomly generated toward the pbest and gbest positions. For each particle $r$ and dimension $s$, the new velocity $v_{rs}$ and position $x_{rs}$ of particles can be calculated by the following equations:

$$v_{rs}^{t} = w \times v_{rs}^{t-1} + c_1 \times rand_1 \times \left(pbest_{rs}^{t-1} - x_{rs}^{t-1}\right) + c_2 \times rand_2 \times \left(gbest_{s}^{t-1} - x_{rs}^{t-1}\right) \quad (1)$$

$$x_{rs}^{t} = x_{rs}^{t-1} + v_{rs}^{t-1} \quad (2)$$

where $t$ *is* the iteration number. The inertial weight $w$ is used to control exploration and exploitation. A large value of $w$ keeps particles at high velocity and prevents them from becoming trapped in local optima. A small value of $w$ maintains particles at low velocity and encourages them to exploit the same search area. The constants $c_1$ and $c_2$ are acceleration coefficients that determine whether particles prefer to move closer to the pbest or gbest positions. The $rand_1$ and $rand_2$ are independent random numbers uniformly distributed between 0 and 1. The termination criterion of the PSO algorithm includes the maximum number of generations, the designated value of pbest, and no further improvement in pbest. The standard PSO process outline is as follows.

Step 1: initialize a population of particles with random positions and velocities on $D$ dimensions in the search space.
Step 2: update the velocity of each particle according to Eq. (1).
Step 3: update the position of each particle according to Eq. (2).
Step 4: map the position of each particle into the solution space and evaluate its fitness value according to the desired optimization fitness function. Simultaneously update the pbest and gbest positions if necessary.
Step 5: loop to Step 2 until an exit criterion is met, usually a sufficient goodness of fitness or a maximum number of iterations.

The original PSO was designed for a continuous solution space. We modified the PSO position representation, particle velocity, and particle movement as described in the next section to make PSO suitable for combinational optimization problems.

## 4 Formation of the proposed PSO

There are two different representations of particle position associated with a schedule. Zhang [31] 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 as described in Sections 4.2 and 4.3. We have also included the maintenance of Pareto optima and local search procedures to achieve better performance.

### 4.1 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. The following example is a permutation sequence for a six-job permutation flowshop 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.,

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

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

Then, we convert the permutation list to a priority list. The$x_i^k$is a value randomly initialized to some value between ($p$–0.5) and ($p$ + 0.5). This means $x_i^k \leftarrow p + \text{rand} - 0.5$, 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

$$X^k = [2.7 \;\; 5.2 \;\; 1.8 \;\; 0.6 \;\; 6.3 \;\; 3.9]$$

### 4.2 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

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

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 $\text{pbest}_i^k(\text{gbest}_i)$, our PSO considers whether the value of $x_i^k$ is larger or smaller than $\text{pbest}_i^k(\text{gbest}_i)$ If $x_i^k$ has decreased in the present iteration, this means that $\text{pbest}_i^k(\text{gbest}_i)$ is smaller than $x_i^k$, and $x_i^k$ is set moving toward $\text{pbest}_i^k(\text{gbest}_i)$ 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 $\text{pbest}_i^k(\text{gbest}_i)$ is larger than $x_i^k$, and $x_i^k$ is set moving toward $\text{pbest}_i^k(\text{gbest}_i)$ 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 each iteration. 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−$w$).

### 4.3 Particle movement

The particle movement is based on the insertion operator proposed by Sha et al. [25, 26]. 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 $x_i^k \leftarrow 3 + \text{rand} - 0.5$ 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 $x_i^k \leftarrow p + \text{rand} - 0.5$. The location of operation $j_i$ in the permutation sequence of the $k$th pbest and gbest solutions are $\text{pbest}_i^k$ and $\text{gbest}_i$, respectively. As particle $k$ moves, if $v_i^k$ equals 0 for all $j_i$, then $x_i^k$ is set to $\text{pbest}_i^k + \text{rand} - 0.5$ with probability $c_1$ and set to $\text{gbest}_i + \text{rand} - 0.5$ 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 $c_1 + c_2 \leq 1$. We explain this concept by assuming specific values for $V^k$, $X^k$, $\text{pbest}^k$, gbest, $c_1$, and $c_2$.

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

- 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 $\text{rand}_1$=0.6. Since $\text{rand}_1 \leq c_1$, then the generated random number $\text{rand}_2$=0.3. Since $\text{pbest}_2^k \leq x_2^k$, set $v_2^k \leftarrow -1$ and $x_2^k \leftarrow \text{pbest}_2^k + \text{rand}_2 - 0.5$, i.e., $x_2^k = 0.8$.
- For $j_3$, since $v_3^k = 0$, the generated random number $\text{rand}_1$=0.93. Since $\text{rand}_1 > c_1 + c_2$, $x_3^k$ 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 $\text{rand}_1$=0.85. Since $c_1 < \text{rand}_1 \leq c_1 + c_2$, the generated random number $\text{rand}_2$=0.7. Since $\text{gbest}_5 \leq x_5^k$, set $v_5^k \leftarrow -1$. Then $x_5^k \leftarrow \text{gbest}_5 + \text{rand}_2 - 0.5$, i.e., $x_5^k = 1.2$.
- For $j_6$, since $v_6^k = 0$, the generated random number $rand_1$=0.95. Since $\text{rand}_1 > c_1 + c_2$, $x_6^k$ 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 $v_i^k \leftarrow -1$.

### 4.4 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.

$$\text{Minimize } F(x) = (f_1(x), f_2(x), \ldots, f_n(x))$$
$$\text{where}, x \in \Re^m, F(x) \in \Re^n$$

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

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

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 Pareto optimalPareto optimal 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 et al., [11]; Zitzler et al. [32]). 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.

### 4.5 Diversification strategy

If all the particles have the same non-dominated solutions, they will be trapped in the local optimal. To prevent this, a diversification strategy is proposed to keep the non-dominated solutions different. Once any new solution is generated by the particles, the non-dominated solution set is updated according to one of three situations.

1. If the solution of the particle is dominated by the gbest solution, assign the particle solution to gbest.
2. If the solution of the particle equals any solution in the non-dominated solution set, replace the non-dominated solution with the particle solution.
3. If the solution of the particle is dominated by the worst non-dominated solution and not equal to any non-dominated solution, set the worst non-dominated solution equal to the particle solution.

## 5 Computational results

The proposed PSO algorithm was verified by benchmark problems obtained from the OR-Library that were contributed by Carlier [2], Heller [8], and Reeves [23]. 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 proposed PSO algorithm was compared with five heuristic algorithms: CDS[1], NEH[17], RAJ[20], GAN-RAJ[6] and Laha[13]. We also coded these methods in Visual C++. The CDS heuristic [1] 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. [17]. 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.

$$\mathrm{ARPD_{MS}} = \frac{100}{10}\sum_{i=1}^{10}\left(\frac{MS_{H,i} - \mathrm{BestMS}_i}{\mathrm{BestMS}_i}\right) \tag{3}$$

$$\mathrm{MPD_{MS}} = \mathrm{MAX}_{i=1..10}\left(\frac{MS_{H,i} - \mathrm{BestMS}_i}{\mathrm{BestMS}_i}\right) \times 100 \tag{4}$$

$$\mathrm{ARPD_{TFT}} = \frac{100}{10}\sum_{i=1}^{10}\left(\frac{\mathrm{TFT}_{H,i} - \mathrm{BestTFT}_i}{\mathrm{BestTFT}_i}\right) \tag{5}$$

$$\mathrm{MPD_{TFT}} = \mathrm{MAX}_{i=1..10}\left(\frac{\mathrm{TFT}_{H,i} - \mathrm{BestTFT}_i}{\mathrm{BestTFT}_i}\right) \times 100 \tag{6}$$

$$\mathrm{ARPD_{MIT}} = \frac{100}{10}\sum_{i=1}^{10}\left(\frac{\mathrm{MIT}_{H,i} - \mathrm{BestMIT}_i}{\mathrm{BestMIT}_i}\right) \tag{7}$$

$$\mathrm{MPD_{MIT}} = \mathrm{MAX}_{i=1..10}\left(\frac{\mathrm{MIT}_{H,i} - \mathrm{BestMIT}_i}{\mathrm{BestMIT}_i}\right) \times 100 \tag{8}$$

We tested our PSO on nine different problem sizes ($n$= 20, 50, 100 and $m$=5, 10, 20) from Taillard's [28] benchmarks. Table 1 compares the six methods using the

**Table 1** Comparison of makespan(MS) for different heuristics

| Problem size | | NEH [17] | | CDS [1] | | RAJ [20] | | GAN-RAJ [6] | | Laha [13] | | 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 | 0.2 | 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 et al. [17], *CDS* Campbell et al. [1], *RAJ* Rajendran C [20], *GAN-RAJ* Gangadharan and Rajendran [6], *Laha* Laha and Chakraborty [12, 13], *PSO* proposed PSO)

**Table 2** Comparison of total flow time (TFT) for different heuristics

| Problem size | | NEH [17] | | CDS [1] | | RAJ [20] | | GAN-RAJ [6] | | Laha [13] | | 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 et al. [17], *CDS* Campbell et al. [1], *RAJ* Rajendran C [20], *GAN-RAJ* Gangadharan and Rajendran [6], *Laha* Laha and Chakraborty [12, 13], *PSO* proposed PSO)

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

| Problem size | | NEH [17] | | CDS [1] | | RAJ [20] | | GAN-RAJ [6] | | Laha [13] | | 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 et al. [17], *CDS* Campbell et al. [1], *RAJ* Rajendran C [20], *GAN-RAJ* Gangadharan and Rajendran [6], *Laha* Laha and Chakraborty [12, 13], *PSO* proposed PSO)

**Table 4** Summation of MS, TFT and MIT for different heuristics

| Problem size | | NEH [17] | | CDS [1] | | RAJ [20] | | GAN-RAJ [6] | | Laha [13] | | 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 et al. [17], *CDS* Campbell et al. [1], *RAJ* Rajendran C [20], *GAN-RAJ* Gangadharan and Rajendran [6], *Laha* Laha and Chakraborty [12, 13], *PSO* proposed PSO)

**Table 5** Average CPU time (in seconds)

| n | m | NEH | CDS | RAJ | GANRAJ | Laha [12] | 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 et al. [17], *CDS* Campbell et al. [1], *RAJ* Rajendran C [20], *GAN-RAJ* Gangadharan and Rajendran [6], *Laha* Laha and Chakraborty [12, 13], *PSO* proposed PSO)

**Table 6** Comparison of total flow time (TFT) for different 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 et al. [17], *CDS* Campbell et al [1], *RAJ* Rajendran C [20], *GAN-RAJ* Gangadharan R, Rajendran C [6], *Laha* Laha and Chakraborty [12, 13], *LR* Liu J, Reeves CR [16], *SA* Chakravarthy K, Rajendran C [3], *H-1 and H-2* Laha D, Chakraborty UK [12, 13], *PSO* proposed PSO)

**Table 7** Comparison of total flow time (TFT) for different 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 et al. [17], *CDS* Campbell et al [1], *RAJ* Rajendran C [20], *GAN-RAJ* Gangadharan R, Rajendran C [6], *Laha* Laha and Chakraborty [12, 13], *LR* Liu J, Reeves CR [16], *SA* Chakravarthy K, Rajendran C [3], *H-1 and H-2* Laha D, Chakraborty UK [12, 13], *PSO* proposed PSO)

ARPD and MPD. Table 1 show that the proposed PSO outperforms for almost all problem instances in the makespan object. The comparison of TFT object is revealed in Table 2. 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 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 4 shows the superior performance of the proposed PSO in terms of the three simultaneous objectives. The computation cost is demonstrated on Table 5. The proposed PSO spends 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[16] (LR), Chakravarthy-Rajendran [3], simulated annealing-bases approach (SA) and Laha and Chakraborty [12] (H-1 and H-2). The results are shown in Table 6 for ARPD and Table 7 for MPD. We can observe that the H-1 and H-2 perform better than other algorithms while only one object TFT is considered.

## 6 Conclusion

Many flowshop scheduling problem studies have been conducted in the past. However, the objective of most of these has been the minimization of the maximum completion time (i.e., the makespan). In the real world, there exist other objectives, such as minimization of machine idle time that might help improve efficiency and reduce production costs. PSO, which was inspired by the behavior of birds and fish, has certain advantages, including simple structure, easy implementation, immediate accessibility, short search time, and robustness. However, there has been limited study of PSO to address the multiple objectives found in the flowshop scheduling problem. We have therefore presented a PSO method for solving a flowshop scheduling problem with multiple objectives, including minimization of makespan, mean flow time, and machine idle time.

PSO was originally proposed for continuous optimization problems. We modified the representation of particle position, particle movement, and particle velocity to make PSO suitable for flowshop scheduling, which is a combinational problem. In addition, we used a mutation operator in our PSO algorithm. We also incorporated the concept of Pareto optimality to measure the performance of multiple objectives rather than using a weighted fitness function. Another necessary adjustment to the original PSO, required to maintain the Pareto optimal solution, was the external Pareto optimal set used to produce a limited size of non-dominated solutions. We also used a diversification strategy in our PSO algorithm. The results demonstrated that the proposed PSO could produce more optimal solutions than other heuristics (CDS, NEH, RAJ, GAN-RAJ, and Laha). The ARPD and MPD of each problem scenario in our PSO algorithm were less than those methods. The results of our performance measurement also revealed that the proposed PSO algorithm outperformed the heuristics in minimizing the makespan, mean flow time, and total machine idle time.

In future research, we will attempt to apply our PSO to other shop scheduling problems with multiple objectives. Possible topics for further study include modification of the particle position representation, particle movement, and particle velocity. Issues related to Pareto optimality, such as a solution maintenance strategy and performance measurement, are also topics worthy of future study.

## References


1. Campbell HG, Dudek RA, Smith ML (1970) A heuristic algorithm for the n-job m-machine sequencing problem. Manage Sci 16:B630–B637. doi:10.1287/mnsc.16.10.B630
2. Carlier J (1978) Ordonnancements à contraintes disjonctives. RAIRO Rech Oper. Oper Res 12:333–351
3. Chakravarthy K, Rajendran C (1999) A heuristic for scheduling in a flowshop with the bicriteria of makespan and maximum tardiness minimization. Prod Plann Contr 10:707–714. doi:10.1080/095372899232777
4. Eren T, Güner E (2007) The tricriteria flowshop scheduling problem. Int J Adv Manuf Technol 36:1210–1220. doi:10.1007/s00170-007-0931-1
5. Gupta JND, Stafford JEF (2006) Flowshop scheduling research after five decades. Eur J Oper Res 169:699–711. doi:10.1016/j.ejor.2005.02.001
6. Gangadharan R, Rajendran C (1993) Heuristic algorithms for scheduling in no-wait flow shop. Int J Prod Econ 32:285–290. doi:10.1016/0925-5273(93) 90042-J
7. Hejazi SR, Saghafian S (2005) Flowshop- scheduling problems with makespan criterion: a review. Int J Prod Res 43:2895–2929. doi:10.1080/0020754050056417
8. Heller J (1960) Some numerical experiments for an MxJ flow shop and its decision- theoretical aspects. Oper Res 8:178–184. doi:10.1287/opre.8.2.178
9. Jarboui B, Ibrahim S, Siarry P, Rebai A (2008) A combinational particle swarm optimisation for solving permutation flowshop problems. Comput Ind Eng 54:526–538. doi:10.1016/j.cie.2007.09.006
10. Kennedy J, Eberhart R (1995) Particle swarm optimization. Proc IEEE Int Conf Neural Netw 1995:1942–1948. doi:10.1109/ICNN.1995.488968
11. Knowles JD, Corne DW (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
12. Laha D, Chakraborty UK (2008) An efficient heuristic approach to total flowtime minimization in permutation flowshop scheduling. Int J Adv Manuf Technol 38:1018–1025. doi:10.1007/s00170-007-1156-z
13. Laha D, Chakraborty UK (2009) A constructive heuristic for minimizing makespan in no-wait flow shop scheduling. Int J Adv Manuf Technol 41:97–109. doi:10.1007/s00170-008-1545-0

14. Lian Z, Gu X, Jiao B (2008) A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan. Chaos Solitons Fractals 35:851–861. doi:10.1016/j.chaos.2006.05.082
15. Liu B, Wang L, Jin YH (2007) An effective PSO-based memetic algorithm for flow shop scheduling. IEEE Trans Syst Man Cybern C 37:18–27. doi:10.1109/TSMCB.2006.883272
16. Liu J, Reeves CR (2001) Constructive and composite heuristic solutions to the P//∑Ci scheduling problem. Eur J Oper Res 132:439–452. doi:10.1016/S0377-2217(00) 00137-5
17. Nawaz M, Enscore JR, Ham I (1983) A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 11:91–95. doi:10.1016/0305-0483(83) 90088-9
18. Pasupathy T, Rajendran C, Suresh RK (2006) A multi-objective genetic algorithm for scheduling in flow shops to minimize the makespan and total flow time of jobs. Int J Adv Manuf Technol 27:804–815. doi:10.1007/s00170-004-2249-6
19. Ponnambalam SG, Jagannathan H, Kataria M (2004) A TSP-GA multi-objective algorithm for flow-shop scheduling. Int J Adv Manuf Technol 23:909–915. doi:10.1007/s00170-003-1731-x
20. Rajendran C (1994) A no-wait flow shop scheduling heuristic to minimize makespan. J Oper Res Soc 45:472–478
21. Rajendran C, Ziegler H (2004) Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. Eur J Oper Res 155:426–438. doi:10.1016/S0377-2217(02) 00908-6
22. Rahimi-Vahed A, Mirghorbani S (2007) A multi-objective particle swarm for a flow shop scheduling problem. J Comb Optim 13:79–102. doi:10.1007/s10878-006-9015-7
23. Reeves CR (1995) A genetic algorithm for flowshop sequencing. Comput Oper Res 22:5–13. doi:10.1016/0305-0548(93) E0014-K
24. Ruiz R, Maroto C (2004) A comprehensive review and evaluation of permutation flowshop heuristics. Eur J Oper Res 165:479–494. doi:10.1016/j.ejor.2004.04.017
25. Sha DY, Hsu CY (2006) A hybrid particle swarm optimization for job shop scheduling problem. Comput Ind Eng 51:791–808. doi:10.1016/j.cie.2006.09.002
26. Sha DY, Hsu CY (2008) A new particle swarm optimization for the open shop scheduling problem. Comput Oper Res 35:3243–3261. doi:10.1016/j.cor.2007.02.019
27. Stützle T (1998) Applying iterated local search to the permutation flow shop problem. Tech Rep, AIDA-98-04, FG Intellektik, TU Darmstadt.
28. Taillard E (1993) Benchmarks for basic scheduling problems. Eur I Oper Res 64:278–285. doi:10.1016/0377-2217(93) 90182-M
29. Tasgetiren MF, Liang YC, Sevkli M, Gencyilmaz G (2007) A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. Eur J Oper Res 177:1930–1947. doi:10.1016/j.ejor.2005.12.024
30. Yagmahan B, Yenisey MM (2008) Ant colony optimization for multi-objective flow shop scheduling problem. Comput Ind Eng 54:411–420. doi:10.1016/j.cie.2007.08.003
31. Zhang H, Li X, Li H, Huang F (2005) Particle swarm optimization-based schemes for resource-constrained project scheduling. Auto Const 14:393–404. doi:10.1016/j.autcon.2004.08.006
32. Zizter E, Laumanns M, Thiele L (2001) SPEA2: Improving the strength Pareto evolutionary algorithm. Computer Engineering and Networks Laboratory (TIK) – Report 103 Sept 2001.

# A multi-objective PSO for job-shop scheduling problems

D. Y. Sha[a] and Hsing-Hung Lin[b*]

[a]*Department of Industrial Engineering and System Management, Chung Hua University,, Hsin Chu, Taiwan R.O.C.;*
[b]*Department of Industrial Engineering and Management, National Chiao Tung University, Hsin Chu, Taiwan R.O.C.*

Abstract

Most previous research into the job-shop scheduling problem has concentrated on finding a single optimal solution (e.g., makespan), even though the actual requirement of most production systems requires multi-objective optimization. The aim of this paper is to construct a particle swarm optimization (PSO) for an elaborate multi-objective job-shop scheduling problem. The original PSO was 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 was used to solve various benchmark problems. Test results demonstrated that the modified PSO performed better in search quality and efficiency than traditional evolutionary heuristics.

**Keywords*:*** job-shop scheduling; particle swarm optimization; multiple objectives

*Corresponding author. Email: hsinhung@gmail.com

## Introduction

The job-shop scheduling problem (JSP) has been studied for more than 50 years in both academic and industrial environments. Jain et al. provided a concise overview of JSPs over the last few decades and highlighted the main techniques [10]. The JSP is the most difficult class of combinational optimization. Garey et al. demonstrated that JSPs are non-deterministic polynomial-time hard (NP-hard) [7]; hence we cannot find an exact solution in a reasonable computation time. The single-objective JSP has attracted wide research attention. Most studies of single-objective JSPs result in a schedule to minimize the time required to complete all jobs, i.e., to minimize the makespan ($C_{max}$). Many approximate methods have been developed to overcome the limitations of exact enumeration techniques. These approximate approaches include simulated annealing (SA) [17], tabu search [18][19][24] and genetic algorithms (GA) [1][9][12][27]. However, real-world production systems require simultaneous achievement of multiple objective requirements. This means that the academic concentration of objectives in the JSP must been extended from single to multiple. Recent related JSP research with multiple objectives is summarized as below.

Ponnambalam has offered a multi-objective GA to derive optimal machine-wise priority dispatching rules for resolving job-shop problems with objective functions that consider minimization of makespan, total tardiness, and total machine idle time[20]. Ponnambalam's multi-objective genetic algorithm (MOGA) has been tested with various published benchmarks, and is capable of providing optimal or near-optimal solutions. A Pareto front provides a set of best solutions to determine the tradeoffs between the various objects, and good parameter settings and appropriate representations can enhance the behavior of an evolution algorithm. Esquivel et al. studied the influence of distinct parameter combinations as well as different chromosome representations [5]. Initial results showed that:

(i) larger numbers of generations favor the building of a Pareto front because the search process does not stagnate, even though it may be rather slow,

(ii) multi-recombination helps to speed the search and to find a larger set size when seeking the Pareto optimal set, and

(iii) operation-based representation is better than priority-list and job-based representation selected for contrast under recombination methods.

The Pareto archived simulated annealing (PASA) method, a meta-heuristic procedure based on the SA algorithm, was developed by Suresh to find non-dominated solution sets for the JSP with the objectives of minimizing the makespan and the mean flow time of jobs[25]. The superior performance of the PASA can be attributed to the mechanism it uses to accept the candidate solution. Candido et al. addressed JSPs with numbers of more realistic constraints, such as jobs with several subassembly levels, alternative processing plans for parts and alternative resources of operations, and the requirement for multiple resources to process an operation [3]. The robust procedure worked well in all problem instances and proved to be a promising tool for solving more realistic JSPs. Lei first designed a crowding-measure-based multi-objective evolutionary algorithm (CMOEA) that makes use of the crowding measure to adjust the external population and assign different fitness for individuals [14]. Compared to the strength Pareto evolutionary algorithm, CMOEA performs well in job-shop scheduling with two objectives including minimization of makespan and total tardiness.

One of the latest evolutionary techniques for unconstrained continuous optimization is particle swarm optimization (PSO) proposed by Kennedy et al. [11]. PSO has been successfully used in different fields due to its ease of implementation and computational efficiency. Even so, application of PSO to the combination optimization problem is rare. Coello et al. provided an approach in which Pareto dominance is incorporated into PSO to allow the heuristic to handle problems with several object functions [4]. The algorithm uses a secondary repository of particles to guide particle flight. That approach was validated using several test functions and metrics drawn from the standard literature on evolutionary multi-objective optimization. The results show that the approach is highly competitive. Liang et al. invented a novel PSO-based algorithm for JSPs[16]. That

algorithm effectively exploits the capability of distributed and parallel computing systems, with simulation results showing the possibility of high-quality solutions for typical benchmark problems. Lei presented a PSO for the multi-objective JSP to minimize makespan and total job tardiness simultaneously [15]. Job-shop scheduling can be converted into a continuous optimization problem by constructing the corresponding relationship between a real vector and a chromosome obtained using the priority rule-based representation method. The global best position selection is combined with crowding-measure-based archive maintenance to design a Pareto archive PSO. That algorithm is capable of producing a number of high-quality Pareto optimal scheduling plans.

Hybrid algorithms that combine different approaches to build on their strengths have led to another branch of research. Wang et al. combined GA with SA in a hybrid framework, in which the GA was introduced to present a parallel search architecture, and SA was used to increase the probability of escape from local optima at high temperatures [27]. Computer simulation results showed that the hybrid strategy was very effective and robust, and could find optima for almost all benchmark instances. Xia et al. developed an easily implemented approach for the multi-objective flexible JSP based on the combination of PSO and SA [28]. They demonstrated that their proposed algorithm was a viable and effective approach to the multi-objective flexible JSP, especially for large-scale problems. Ripon extended the idea in the jumping genes genetic algorithm, a hybrid approach capable of searching for near-optimal and non-dominated solutions with better convergence by simultaneously optimizing criteria [21].

Previous literature indicates that there has been little study of the JSP with multiple objectives. In this study, we use a new evolutionary PSO technique to solve the JSP with multiple objectives.

**Job-shop scheduling problem**

A typical JSP can 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 the different job operations. Each machine can perform only one job at a time, and it cannot be interrupted. In addition, the operation time is fixed and known in advance. The objective of the JSP is to find a schedule to minimize the time required to complete all jobs, that is, to minimize the makespan $C_{max}$. In this study, we attempt to attain the three objectives (i.e., minimizing makespan, machine idle time, and total tardiness) simultaneously. We formulate the multi-objective JSP using the following notation:

*n* is the total number of jobs to be scheduled

*m* is the total number of machines in the process

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

$L_i$ is the lateness of job *i*

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

The objectives considered in this paper are formulated as follows:

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

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

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

$$C(\pi_1, j) = C(\pi_1, j-1) + t(\pi, j) \;\; j = 2,...,m \tag{3}$$

$$C(\pi_i, j) = \max\{C(\pi_{i-1}, j), C(\pi_i, j-1)\} + t(\pi_i, j) \;\; i = 2,...,n; \;\; j = 2,...,m \tag{4}$$

$$\text{Makespan, } \; f_{C\max} = C(\pi_n, m) \tag{5}$$

$$\text{Total tardiness, } \; f_{total\ tardiness} = \sum_{i=1}^{n} \max[0, L_i] \tag{6}$$

Total idle time, $f_{total\ idle\ time} = \{C(\pi_1, j-1) + \sum_{i=2}^{n}\{\max\{C(\pi_i, j-1) - C(\pi_{i-1}, j), 0\}\} \mid j = 2...m\}$ (7)

**PSO background**

PSO is based on observations of the social behavior of animals, such as birds in flocks or fish in schools, as well as on swarm theory. The population consisting of individuals or particles is initialized randomly. Each particle is assigned with a randomized velocity according to its own movement experience and that of the rest of the population. The relationship between the swarm and particles in PSO is similar to the relationship between the population and chromosomes in a GA.

In PSO, the problem solution space is formulated as a search space. Each particle position in the search space is a correlated solution to the problem. Particles cooperate to determine the best position (solution) in the search space (solution space).

Suppose that the search space is *D*-dimensional and there are $\rho$ particles in the swarm. Particle *i* is located at position $X^i=\{x_1^i, x_2^i, \ldots, x_D^i\}$ and has velocity $V^i=\{v_1^i, v_2^i, \ldots, v_D^i\}$, where $i$=1, 2, …,$\rho$. Based on the PSO algorithm, each particle move towards its own best position (*pbest*), denoted as $Pbest^i=\{pbest_1^i, pbest_2^i, \ldots, pbest_n^i\}$, and the best position of the whole swarm (*gbest*) is 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 the *pbest* and *gbest* positions. For each particle *r* and dimension *s*, the new velocity $v_s^r$ and position $x_s^r$ of particles can be calculated by the following equations:

$$v_s^r(\tau) = w \times v_s^r(\tau-1) + c_1 \times rand_1 \times [pbest_s^r(\tau-1) - x_s^r(\tau-1)] + c_2 \times rand_2 \times [gbest_s^r(\tau-1) - x_s^r(\tau-1)] \quad (8)$$

$$x_s^r(\tau) = x_s^r(\tau-1) + v_s^r(\tau-1) \quad (9)$$

In Eqs. (8) and (9), $\tau$ is the iteration number. The inertial weight $w$ is used to control exploration and exploitation. A large $w$ value keeps the particles moving at high velocity and prevents them from becoming trapped in local optima. A small $w$ value ensures a low particle velocity and encourages particles to exploit the same search area. The constants $c_1$ and $c_2$ are acceleration coefficients to determine whether particles prefer to move closer to the *pbest* or *gbest* positions. 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 a maximum number of generations, a designated value of *pbest*, and lack of further improvement in *pbest*. The standard PSO process is outlined as follows:

Step 1: Initialize a population of particles with random positions and velocities in a $D$-dimensional search space.
Step 2: Update the velocity of each particle using Eq. (8).
Step 3: Update the position of each particle using Eq. (9).
Step 4: Map the position of each particle into the solution space and evaluate its fitness value according to the desired optimization fitness function. Simultaneously update the *pbest* and *gbest* positions if necessary.
Step 5: Loop to Step 2 until the termination criterion is met, usually after a sufficient good fitness or a maximum number of iterations.

The original PSO was designed for a continuous solution space. We must modify the PSO position representation, particle velocity, and particle movement so they work better with combinational optimization problems. These changes are described in next section.

## Proposed method

There are four types of feasible schedules in JSPs, 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 [8] heuristic. There are two different representations of particle position associated with a schedule. The results of Zhang [29] demonstrated that permutation-based position representation outperforms priority-based representation. While choosing to implement permutation-based position presentation, we must also adjust the particle velocity and particle movement. In addition, we also propose the maintenance of Pareto optima and a diversification procedure to achieve better performance.

### *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*×*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

$\Omega$ is the set of operations that can be scheduled

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

G&T algorithm:

Step 1: Initialize $S = \phi$; $\Omega$ 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 $\Omega$, include its immediate successor in $\Omega$, and then go to Step 2.

Table 1 shows the mechanism of the G&T algorithm using a 2×2 example. The position of particle $k$ is $X^k = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$.

*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) that 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) that 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) that 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) that has the largest priority, and add it into schedule *S*.

Step 4: A complete schedule has been generated, so stop the process.

The proposed PSO differs from the original PSO in the information stored in the *pbest* and *gbest* solutions. While the original PSO keeps the best positions found so far, the proposed PSO maintains the best schedule generated by the 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 $\begin{bmatrix} 2 & 1 \\ 2 & 1 \end{bmatrix}$. The movement of particles is modified in accordance with the representation of particle position based on the insertion operator.

***Particle velocity***

The original PSO velocity concept assumes 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 keep the particle moving toward the *gbest* and *pbest* solutions, and to maintain inertia to prevent particles from becoming trapped in local optima.

In the proposed PSO, we concentrate on preventing particles from becoming trapped in local optima rather than moving them toward the *gbest* (*pbest*) solution. If the priority value is increased or decreased by the present velocity in the current iteration, we keep the priority value increasing or decreasing at the beginning of the next iteration with probability *w*, which is the inertial weight in PSO. The larger the value of *w*, the more the iteration priority value keeps increasing or decreasing, and the more the difficult it is for the particle to return to its current position. For an *n*-job problem, the velocity of particle *k* can be represented as

$V^k = [v_1^k \; v_2^k \; ... \; v_n^k], v_i^k \in \{-1,0,1\}$, where $v_i^k$ is the velocity of $j_i$ of particle $k$.

The initial velocity of particles is 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 $x_i^k$ decreases in the present iteration, this mean 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 $x_i^k$ increases in this iteration, then $pbest_i^k$ ($gbest_i$) is larger than $x_i^k$, and $x_i^k$ is set moving toward $pbest_i^k$ ($gbest_i$) by setting $v_i^k \leftarrow 1$. 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 the particles in the PSO. We randomly update velocities at the beginning of the iteration. For each particle *k* and operation $j_i$, if $v_i^k$ does not equal to 0, $v_i^k$ will be set to 0 with probability (1–*w*). This forces $x_i^k$ to stop increasing or decreasing continuously in this iteration with probability (1–*w*) while $x_i^k$ keeps increasing or decreasing.

***Particle movement***

The particle movement is based on the swap operator proposed by Sha et al. [22][23].

Notation:

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

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

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

$c_1$ and $c_2$ are constants between 0 and 1 such that $c_1 + c_2 \leq 1$.

The swap procedure occurs as shown below.

Step 1: Randomly choose a position $\zeta$ from $x_i^k$.

Step 2: Mark the job on position $\zeta$ of $x_i^k$ by $\Lambda_1$.

Step 3: If the random number *rand* < $c_1$ then seek the position of $\Lambda_1$ in $pbest_i^k$; otherwise, seek the position of $\Lambda_1$ in $gbest_i$. Denote the position that has been found in $pbest_i^k$ or $gbest_i$ by $\zeta'$, and the job in position $\zeta'$ of $x_i^k$ by $\Lambda_2$.

Step 4: If $\Lambda_2$ has been denoted, $v_{iJ_1}^k = 0$, and $v_{iJ_2}^k = 0$, then swap $\Lambda_1$ and $\Lambda_2$ in $x_i^k$, $v_{iJ_1}^k \leftarrow 1$.

Step 5: If all the positions of $x_i^k$ have been considered, then stop. If not, and if $\zeta$ < *n*, then $\zeta \leftarrow \zeta$+1; otherwise, $\zeta \leftarrow 1$. Go to Step 2.

For example, consider the 6-job problem 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 chosen: $\zeta$=3.

Step 2: The job in the 3rd position of $x_i^k$ is job 1, i.e., $\Lambda_1$=1.

Step 3: A random number *rand* is generated; assume *rand*=0.7. Since *rand* > $c_1$, we compare each position of $gbest_i$ with $\Lambda_1$ and the matched position $\zeta'$=6. The job in the 6th position of $x_i^k$ is job 5, i.e., $\Lambda_2$=5.

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

Step 5: Let $\zeta \leftarrow 4$ and go to Step 2. Repeat the process until all positions of $x_i^k$ have been considered.

***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:

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

(ii) 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.

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

**Computational results**

The proposed multi-objective PSO (MOPSO) algorithm was tested on benchmark problems obtained from the OR-Library [2][26]. 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. During the pilot experiment, we used four swarm sizes $N$ (10, 30, 60, and 80) 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.

The MOGA proposed by Ponnambalam et al. [19] 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 [8]. 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.

## Conclusion

While there has been a large amount of research into the JSP, 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 JSPs can be found in the literature. Therefore, we presented a MOPSO method for solving the JSP 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.

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

## Acknowledgments

This study was supported by a grant from the National Science Council of Taiwan (NSC-96-2221-E-216-052MY3).

## Appendices

Pseudo-code of the PSO for the multi-objective JSP is as follows.

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 equal 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

## References


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

[2]Beasley J.E., 1990. "OR-Library: distributing test problems by electronic mail", Journal of the Operational Research Society 41(11) pp1069-1072.

[3]Candido, M. A. B., Khator, S.K. & Barcia, R.M., 1998. “A genetic algorithm based procedure for more realistic job shop scheduling problems,” *International Journal of Production Research*, **36**(12), 3437–3457.

[4]Coello, C.A., Plido, G.T. & Lechga, M.S., 2004. “Handling multiple objectives with particle swarm optimization,” *IEEE Transactions on Evolutionary Computation*, **8**(3), 256–278.

[5]Esquivel, S.C., Ferrero, S.W. & Gallard, R.H., 2002. “Parameter settings and representations in Pareto-based optimization for job shop scheduling,” *Cybernetics and Systems: An international Journal*, **33**, 559–578.

[6]Fisher, H. & Thompson, G. L., 1963. *Industrial Scheduling*, Englewood Cliffs, NJ: Prentice-Hall.

[7]Garey, M. R., Johnson, D. S. & Sethi, R., 1976. “The complexity of flowshop and jobshop scheduling,” *Mathematics of Operations Research*, **1**, 117–129.

[8]Giffler, J. & Thompson, G. L., 1960. “Algorithms for solving production scheduling problems,” *Operations Research*, **8**, 487–503.

[9]Gonçalves, 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.

[10]Jain, A.S. & Meeran, S., 1999. “Deterministic job-shop scheduling: Past, present and future,” *European Journal of Operational Research*, **113**, 390–434.

[11]Kennedy, J. and R. Eberhart (1995), Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks 1995, 1942-1948.

[12]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* (pp. 506–511). San Francisco, CA: Morgan Kaufman Publishers.

[13]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.

[15]Lei, D. & Wu, Z., 2006. “Crowding-measure-based multi-objective evolutionary algorithm for job shop scheduling,” *International Journal of Advanced Manufacturing Technology*, **30**, 112–117.

[14]Lei, D., 2008. “A Pareto archive particle swarm optimization for multi-objective job shop scheduling,” *Computers & Industrial Engineering*, **54**(4), 960–971.

[16]Liang Y.C., Ge, H.W., Zho, Y. & Guo, X.C., 2005. “A particle swarm optimization-based algorithm for job-shop scheduling problems,” *International Journal of Computational Methods*, **2**(3), 419–430.

[17]Lourenço, H. R., 1995. “Local optimization and the job-shop scheduling problem,” *European Journal of Operational Research*, **83**, 347–364.

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

[19]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.

[20]Ponnambalam S. G., Ramkumar, V. & Jawahar, N., 2001. “A multi-objective genetic algorithm for job shop scheduling.” *Production Planning and Control*, **12**(8), 764–774.

[21]Ripon, K. S. N., 2007. “Hybrid evolutionary approach for multi-objective job-shop scheduling problem,” *Malaysian Journal of Computer Science*, **20**(2), 183–198.

[22]Sha, D.Y. & Hsu, C.-Y., 2006, "A hybrid particle swarm optimization for job shop scheduling problem," *Computers & Industrial Engineering*, **51**(4), 791–808.

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

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

[25]Suresh R.K. & Mohanasndaram, K.M. 2006., "Pareto archived simulated annealing for job shop scheduling with multiple objectives," *International Journal of Advanced Manufacturing Technology*, **29**, 184–196.

[26]Taillard, E.D., 1993. "Benchmarks for basic scheduling problems," *European Journal of Operational Research*, **64**, 278–285.

[27]Wang, L. & Zheng, D.-Z., 2001. "An effective hybrid optimization strategy for job-shop scheduling problems," *Computers & Operations Research*, **28**, 585–596.

[28]Xia, W. & Wu, Z., 2005. "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems," *Computers & Industrial Engineering*, **48**, 409–425.

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

Table 1 2×2 example

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

Table 2 Comparison of MOGA and MOPSO with three objectives.

| Benchmark | *N* | *m* | Makespan (MOGA) | Makespan (MOPSO) | % Deviation | Machine idle time (MOGA) | Machine idle time (MOPSO) | % Deviation | Total tardiness (MOGA) | Total tardiness (MOPSO) | % Deviation |
|---|---|---|---|---|---|---|---|---|---|---|---|
| abz5 | 10 | 10 | 1587 | 1338 | 0 | 8097 | 3978 | 0 | 1948 | 611 | 0 |
| abz6 | 10 | 10 | 1369 | 1046 | 0 | 7744 | 2937 | 0 | 1882 | 339 | 0 |
| ft06 | 6 | 6 | 76 | 56 | 0 | 259 | 100 | 0 | 31 | 3 | 0 |
| ft10 | 10 | 10 | 1496 | 1045 | 0 | 9851 | 1999 | 0 | 3459 | 1534 | 0 |
| orb01 | 10 | 10 | 1704 | 1181 | 0 | 11631 | 3909 | 0 | 3052 | 191 | 0 |
| orb02 | 10 | 10 | 1284 | 1029 | 0 | 7585 | 3539 | 0 | 1565 | 137 | 0 |
| orb03 | 10 | 10 | 1643 | 1114 | 0 | 11138 | 3788 | 0 | 4140 | 247 | 0 |
| orb04 | 10 | 10 | 1543 | 1122 | 0 | 9802 | 3921 | 0 | 4951 | 221 | 0 |
| orb05 | 10 | 10 | 1323 | 1013 | 0 | 8322 | 3727 | 0 | 2195 | 30 | 0 |
| orb06 | 10 | 10 | 1645 | 1144 | 0 | 10836 | 3478 | 0 | 2601 | 0 | 0 |
| orb07 | 10 | 10 | 583 | 302 | 0 | 3423 | 1381 | 0 | 699 | 0 | 0 |
| orb08 | 10 | 10 | 1340 | 1000 | 0 | 8840 | 3542 | 0 | 3498 | 253 | 0 |
| orb09 | 10 | 10 | 1462 | 1044 | 0 | 9439 | 4224 | 0 | 2029 | 0 | 0 |
| orb10 | 10 | 10 | 1382 | 1077 | 0 | 8271 | 4177 | 0 | 1806 | 0 | 0 |
| la01 | 10 | 5 | 1256 | 709 | 0 | 3431 | 571 | 0 | 3324 | 721 | 0 |
| la02 | 10 | 5 | 1066 | 713 | 0 | 2687 | 573 | 0 | 2081 | 425 | 0 |
| la03 | 10 | 5 | 821 | 671 | 0 | 1722 | 633 | 0 | 1926 | 373 | 0 |
| la04 | 10 | 5 | 861 | 631 | 0 | 1798 | 557 | 0 | 3194 | 673 | 0 |
| la05 | 10 | 5 | 893 | 593 | 0 | 2182 | 473 | 0 | 1716 | 736 | 0 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| la16 | 10 | 10 | 1452 | 1040 | 0 | 9169 | 2718 | 0 | 1127 | 1417 | 0.25732 |
| la17 | 10 | 10 | 1172 | 889 | 0 | 7044 | 3365 | 0 | 1779 | 53 | 0 |
| la19 | 10 | 10 | 1251 | 938 | 0 | 7164 | 2796 | 0 | 1581 | 733 | 0 |
| la20 | 10 | 10 | 1419 | 985 | 0 | 8745 | 2883 | 0 | 1451 | 407 | 0 |

---

The English in this document has been checked by at least two professional editors, both native speakers of English. For a certificate, see:

http://www.textcheck.com/cgi-bin/certificate.cgi?id=LbNsng

# 國科會補助專題研究計畫項下出席國際學術會議心得報告

日期：98 年 7 月 15 日

| | | | |
|---|---|---|---|
| 計畫編號 | NSC96－2221－E－216－052－MY3 | | |
| 計畫名稱 | 粒子群最佳化於多目標排程問題之應用 | | |
| 出國人員姓名 | 沙永傑 | 服務機構及職稱 | 中華大學工業工程與管理學系 |
| 會議時間 | 98 年 7 月 6 日至 98 年 7 月 8 日 | 會議地點 | 法國 Troyes |
| 會議名稱 | (中文)第 39 屆計算機與工業工程國際研討會<br>(英文)39th International Conference on Computers & Industrial Engineering (CIE39) | | |
| 發表論文題目 | (中文)零工型排程問題的多目標粒子群最佳化演算法<br>(英文) A Multi-objective PSO for Job-shop Scheduling Problems | | |

一、參加會議經過

第三十九屆計算機與工業工程國際研討會，主要是由國際期刊:計算機與工業工程委託 University of Technology of Troyes (UTT)舉辦。Troyes 是法國香檳區的一座古老城鎮，位於巴黎東南方 150 公里處，以擁有 16 世紀的文化遺產聞名。在年中獲悉第三十九屆計算機與工業工程國際研討會的訊息，便積極準備投稿，於截稿時間 1 月 31 日前，將論文全文投出，三月初收到接受通知。之後即開始後續報名與行程安排。於 7 月 5 日搭機前往巴黎參加 CIE39 研討會。

我們的報告歸類為 Operational Research / meta-heuristic 的 session, 在第三天下午 16:10 開始報告，原本是在 break 之後的第二篇，但是由於第一篇的報告者未出席，因此我們的報告便提前。報告十五分鐘，之後有與會學者提問，提問的內容皆相當有深度且有意義。主要是因為在這個 session 中的四篇論文，都是採用同樣的演算法為基礎，解決不同領域的問題，因此大家很容易得以產生共鳴，並引發熱烈的討論。透過自己與其他學者的報告還有討論互動，可以充分了解我們的研究內容中仍可進行修正之處，另一方面也了解到，我們所應用的最佳化演算法，在國際上研究的熱潮與主要研究的方向。

二、與會心得

本屆計算機與工業工程國際會議，主題涵蓋近年來學術與產業界的研究與發展，包含的主軸有：Applied Operations Research、Probabilistic and Statistical Models、Communications & Networking、Data Mining, Knowledge Discovery and Computational Intelligence、Multi-Criteria Decision Making and Decision Analysis、System Simulation and Forecasting、Information Technology、Supply Chain Management & Logistics、Web-based Applications, E-Business and E-Commerce、Quality Management/Engineering, Reliability and Maintenance 、 Facilities Layout Design, Warehousing, Material Handling 、 Production/Manufacturing Systems and Processes; Agile Manufacturing; ERP/APS 、 Design for Manufacturing, Robust Design, Reverse Engineering、Group Technology & Cellular Manufacturing、Environmentally Conscious Manufacturing、Human Factors, Industrial Ergonomics and Safety、Project Development & Management、Global Economy Engineering & IE、Manufacturing、 Technologies、Artificial Intelligence & Expert Systems、Industrial Engineering Education and E-learning、Systematic Innovation、Ethics in IE Education Research & Practice。從上述的主題中不難發現，工業工程領域除了傳統的範疇之外已逐步拓展到科技管理與服務業管理等非傳統工業工程的領域。

此次會議中，針對每位演講者/論文發表者的講題，各場聽眾皆有相當熱烈的回應與討論，於此當中，不僅瞭解目前國外學者的研究方向，亦學習到如何與國外聽眾作互動，同時也更深刻了解到學術溝通的重要性。透過此次會議的參與，讓我吸收了許多寶貴的經驗，也藉由與國外學者的密切互動，讓自己更加瞭解未來工業工程發展的趨勢，並對工業工程領域相關的學術研究有更全盤性的認識與了解。

## 三、攜回資料名稱及內容

1. 大會手冊一本
2. 大會論文摘要集一本
3. 大會論文光碟一片
4. 環保背包一只紀念品(雷射指引筆)一份
5. 與在場學者交流之名片

# 國科會補助專題研究計畫項下出席國際學術會議心得報告

日期：98 年 12 月 25 日

| | | | |
|---|---|---|---|
| 計畫編號 | NSC96－2221－E－216－052－MY3 | | |
| 計畫名稱 | 粒子群最佳化於多目標排程問題之應用 | | |
| 出國人員姓名 | 沙永傑 | 服務機構及職稱 | 中華大學工業工程與管理學系 |
| 會議時間 | 98 年 12 月 14 日至 98 年 12 月 16 日 | 會議地點 | 日本北九州 |
| 會議名稱 | (中文)第十屆亞太工業工程與管理系統會議<br>(英文) The 10th Asia Pacific Industrial Engineering & Management Systems Conference | | |
| 發表論文題目 | (中文)一種於多目標零工型排程的新型粒子群最佳化<br>(英文) A Novel Particle Swarm Optimization for Multi-objective Job-shop Scheduling | | |

一、參加會議經過

第十屆亞太工業工程與管理系統研討會，主要是由國際期刊:工業工程與管理系統與早稻田大學(Waseda University)共同舉辦。APIEMS 主要提供工業工程與管理系統領域的學術與產業界的研究者與工程師交換最新發展與異教交流的論壇。在年中獲悉第十屆工業工程與管理系統國際研討會的訊息，便積極準備投稿，於摘要截稿時間 8 月 15 日前，將論文摘要投出，九月初收到接受通知，九月十五日將論文全文投出，十月十五日收到全文接受通知。之後即開始後續報名與行程安排。於 12 月 13 日搭機前往日本參加 APIEMS 研討會。

我們的報告歸類為 Swarm Intelligience and Neural Network 的 session, 在第一天下午 16:30 開始報告，我們的報告是此 session 的第一篇。報告十五分鐘，之後有與會學者提問，提問的內容皆相當有深度且有意義。主要是因為在這個 session 中的五篇論文，都是採用粒子群演算法為基礎，解決不同領域的問題，因此引發熱烈的討論。透過自己與其他學者的報告還有問題討論的互動，可以充分了解我們的研究內容中仍可進行修正之處，另一方面也了解到，我們所應用的最佳化演算法，在國際上研究的熱潮與主要研究的方向。

二、與會心得

本屆亞太工業工程與管理系統研討會，主題涵蓋近年來學術與產業界的研究與發展，包含的主題有：Applied Statistics & Data Mining, CAD/CAM, Computational Intelligence in IE, Decision Making Models, Decision Support Systems, Enterprise Information Systems/ERP, Facilities Design and Location, Green Design/Green Manufacturing, Healthcare Management, Human Factors/Industrial Ergonomics, Human Resource Management, Industrial Engineering Education, Inventory Systems and Management, Lean Manufacturing/Logistics, Manufacturing/Industrial Automation, Operations Research/Optimization, Product Design/Development, Production Systems Design, Planning and Control, Productivity and Business Strategies, Project Management, Quality Engineering, Research Methods in IE, Safety Management, Service Systems and Management, Soft Computing/Meta-Heuristics, Supply Chain and Logistics, Systems Engineering and Management, Systems Simulation, Technology Management, Total Quality Management。除了工業工程的主題外，尚包括健康管理、服務系統與管理、系統工程與管理、科技管理等領域。

透過參與此次會議，對於演講者/論文發表者的講題，不僅可以瞭解到目前國外學者的研究方向，於此當中，亦強化與國外學者間的互動，同時也更深刻體認到學術交流的重要性，當然，參與的場次中聽眾皆有相當熱烈的回應與討論。透過此次會議的參與，吸收了更多寶貴的經驗，也藉由與國外學者的密切互動，讓自己更加瞭解未來工業工程與管理發展的趨勢。

## 三、攜回資料名稱及內容

1. 大會手冊一本
2. 大會論文摘要集一本
3. 大會論文光碟一片
4. 環保背包一只
5. 與在場學者交流之名片

# 國科會補助專題研究計畫項下出席國際學術會議心得報告

日期：99 年 3 月 26 日

| | | | |
|---|---|---|---|
| 計畫編號 | NSC96－2221－E－216－052－MY3 | | |
| 計畫名稱 | 粒子群最佳化於多目標排程問題之應用 | | |
| 出國人員姓名 | 沙永傑 | 服務機構及職稱 | 中華大學工業工程與管理學系 |
| 會議時間 | 99 年 3 月 17 日至<br>99 年 3 月 19 日 | 會議地點 | 香港 |
| 會議名稱 | (中文) 2010 IAENG International Conference on Industrial Engineering<br>(英文)2010 年工業工程國際會議 | | |
| 發表論文題目 | (中文)一種於多目標開放型排程的修訂粒子群最佳化<br>(英文) A Modified Particle Swarm Optimization for Multi-objective Open Shop Scheduling | | |

## 一、參加會議經過

2010 年 IAENG 工業工程國際會議，主要是由 IAENG（Interantional Association of Engineers）非營利國際組織於香港舉辦。該研討會聚焦於工程理論與應用以及計算機科學領域的主題。2009 年的會議吸引超過來自五十個國家及 1000 名學者專家與會。本次投稿於 1 月 12 日截止，於是便積極準備於截止日前將投稿論文送出，二月份收到接受通知，於三月十七日前往香港參加開幕與報告。

我們的報告被安排在第一天下午 15:45 開始報告，原本是在 break 之後的第二篇，但是由於第一篇的報告者未出席，因此我們的報告便提前。報告十五分鐘，之後有與會學者提問，提問的內容皆相當有深度且有意義。主要是因為在這個 session 中的幾篇論文，都是採用啟發式演算法，解決不同領域的問題，因此大家很容易得以產生共鳴，並引發熱烈的討論。透過自己與其他學者的報告還有討論互動，可以充分了解我們的研究內容中仍可進行修正之處，另一方面也了解到，我們所應用的最佳化演算法，在國際上研究的熱潮與主要研究的方向。

## 二、與會心得

本年度國際工業工程會議，主題涵蓋近年來學術與產業界的研究與發展，包含的主軸 Engineering Physiology, Biomedical Instrumentation, Engineering Statistics, Quality Management Systems, Maintenance

Engineering, Reliability and Quality Control, Engineering Experimental Design, Integrated Product Engineering, Engineering Risk and Decision Analysis, Computer Supported Collaborative Engineering, Human Factors and Ergonomics, Computer-Aided Design, Computer Aided Manufacturing, Computer Simulation MethodsFacilities Design and Logistics, Manufacturing Processes and Methods, Information Systems for the Manufacturing, Quality and Productivity Management, Optimization Methods, Intelligent Engineering Systems, Engineering Management and Leadership 等。由於會議是由多領域組成，從上述的主題中不難發現，工業工程領域除了傳統的範疇之外已逐步拓展到科技管理與服務業管理等非傳統工業工程的領域。

此次會議中，針對每位演講者/論文發表者的講題，聽眾皆有相當熱烈的回應與討論，不僅能夠瞭解目前國外學者的研究方向，亦學習到如何與國外聽眾作互動，同時也更深刻了解到學術溝通的重要性。透過此次會議的參與，吸收了許多寶貴的經驗。

## 三、攜回資料名稱及內容

1. 大會手冊一本
2. 大會論文摘要集一本
3. 大會論文光碟一片
4. 背包一只
5. 與在場學者交流之名片

無研發成果推廣資料

# 96 年度專題研究計畫研究成果彙整表

| 計畫主持人：沙永傑 | 計畫編號：96-2221-E-216-052-MY3 |
|---|---|
| 計畫名稱：粒子群最佳化於多目標排程問題之應用 | |

| 成果項目 | | | 量化 | | | 單位 | 備註（質化說明：如數個計畫共同成果、成果列為該期刊之封面故事...等） |
|---|---|---|---|---|---|---|---|
| | | | 實際已達成數（被接受或已發表） | 預期總達成數(含實際已達成數) | 本計畫實際貢獻百分比 | | |
| 國內 | 論文著作 | 期刊論文 | 0 | 0 | 100% | 篇 | |
| | | 研究報告/技術報告 | 0 | 0 | 100% | | |
| | | 研討會論文 | 0 | 0 | 100% | | |
| | | 專書 | 0 | 0 | 100% | | |
| | 專利 | 申請中件數 | 0 | 0 | 100% | 件 | |
| | | 已獲得件數 | 0 | 0 | 100% | | |
| | 技術移轉 | 件數 | 0 | 0 | 100% | 件 | |
| | | 權利金 | 0 | 0 | 100% | 千元 | |
| | 參與計畫人力（本國籍） | 碩士生 | 0 | 0 | 100% | 人次 | |
| | | 博士生 | 0 | 0 | 100% | | |
| | | 博士後研究員 | 0 | 0 | 100% | | |
| | | 專任助理 | 0 | 0 | 100% | | |
| 國外 | 論文著作 | 期刊論文 | 2 | 2 | 100% | 篇 | |
| | | 研究報告/技術報告 | 0 | 0 | 100% | | |
| | | 研討會論文 | 5 | 3 | 100% | | |
| | | 專書 | 0 | 0 | 100% | 章/本 | |
| | 專利 | 申請中件數 | 0 | 0 | 100% | 件 | |
| | | 已獲得件數 | 0 | 0 | 100% | | |
| | 技術移轉 | 件數 | 0 | 0 | 100% | 件 | |
| | | 權利金 | 0 | 0 | 100% | 千元 | |
| | 參與計畫人力（外國籍） | 碩士生 | 0 | 0 | 100% | 人次 | |
| | | 博士生 | 0 | 0 | 100% | | |
| | | 博士後研究員 | 0 | 0 | 100% | | |
| | | 專任助理 | 0 | 0 | 100% | | |

| | |
|---|---|
| 其他成果<br>(無法以量化表達之成果如辦理學術活動、獲得獎項、重要國際合作、研究成果國際影響力及其他協助產業技術發展之具體效益事項等，請以文字敘述填列。) | 無 |

| | 成果項目 | 量化 | 名稱或內容性質簡述 |
|---|---|---|---|
| 科教處計畫加填項目 | 測驗工具(含質性與量性) | 0 | |
| | 課程/模組 | 0 | |
| | 電腦及網路系統或工具 | 0 | |
| | 教材 | 0 | |
| | 舉辦之活動/競賽 | 0 | |
| | 研討會/工作坊 | 0 | |
| | 電子報、網站 | 0 | |
| | 計畫成果推廣之參與（閱聽）人數 | 0 | |

# 國科會補助專題研究計畫成果報告自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估

■達成目標

□未達成目標（請說明，以 100 字為限）

□實驗失敗

□因故實驗中斷

□其他原因

說明：

2. 研究成果在學術期刊發表或申請專利等情形：

論文：■已發表 □未發表之文稿 □撰寫中 □無

專利：□已獲得 □申請中 ■無

技轉：□已技轉 □洽談中 ■無

其他：(以 100 字為限)

3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）(以 500 字為限)

本研究已完成求解多目標流程型、零工型與開放型排程問題之粒子群最佳化演算法，此研究成果的學術意義在於，順利將粒子群最佳化演算法轉換為可應用於離散型問題之演算法，另一方面則是以求解 Pareto set 的方式找出多目標最佳解，最可貴的是，以標竿問題測試，驗證粒子群演算法再效能上優於基因演算法與其他啟發式演算法。基於上述的的成果，我們相信粒子群最佳化演算法可進一步應用於其他離散型的 NP 問題，而且其成果也會優於現有已知的演算法，值得進一步後續發展與研究。