International Journal of Innovative
Computing, Information and Control
Volume **10**, Number **2**, April **2014**

ICIC International ©2014 ISSN 1349-4198
pp. **729–740**

# A DISCRETE PARTICLE SWARM OPTIMIZATION ALGORITHM FOR JOB-SHOP SCHEDULING PROBLEM TO MAXIMIZING PRODUCTION

Zhigang Lian[1], Weitian Lin[2], Yejun Gao[2] and Bin Jiao[2]

[1]School of Electronic and Information Engineering
[2]School of Electrical Engineering
Shanghai Dianji University
No. 1350, Ganlan Road, Pudong New Dist., Shanghai 201306, P. R. China
{ lianzg; Linwt; Gaoyj; Jiaob }@sdju.edu.cn

Received March 2013; revised September 2013

Abstract. *This paper researches on a special job-shop scheduling problem meeting capabilities of equipments and different due-dates. Its model is developed, which can accord process capacity and different due-dates in numerous jobs to select some to process, and the objective is as far as possible to maximize output. In order to solve complex scheduling problems proposed in this paper, a discrete particle swarm optimization algorithm is presented to optimize it. To test the performance of the novel algorithm, we use some randomly generated instances to simulate solving the practical problems. Computational experiments show significant improvement over an existing particle swarm optimization algorithm.*

**Keywords:** Job-shop scheduling, Due-date, Maximizing, Production volume, Discrete particle swarm optimization algorithm

1. **Introduction.** Effective machine scheduling, which involves the allocation of machines to competing jobs over time, is critical to success in today's highly competitive global industries. In the past a huge amount of literature addressed the traditional scheduling problems as flow shop, job shop, single-machine scheduling, open shop scheduling and parallel machines shops. Optimization goals of different scheduling studied by researches are different. Many complex industrial problems are generally modeled as job-shop scheduling problems (JSSP). The classical JSSP is such a problem, which deals with the sequencing operation of a set of jobs on a set of machines with the objective to optimize some criterion or criteria. The job shop scheduling problem is one of the most difficult production scheduling problems in industry. A. Dolgui et al. in [1] consider single machine scheduling problems of minimizing the makespan in which the processing time of a job depends on its position. It considers scheduling models with positional deterioration or learning under precedence constraints that are built up iteratively from the prime partially ordered sets of a bounded width. Paper [2] considers single-machine scheduling problems in which the processing time of a job is a function of its starting time and its resource allocation. It researches minimizing a cost function containing makespan, total completion time, total absolute differences in completion times and total resource cost; minimizing a cost function containing makespan, total waiting time, total absolute differences in waiting times and total resource cost. D. C. Paraskevopoulos et al. propose a new solution representation and an evolutionary algorithm namely SAILS in [3], for solving the Resource Constrained Project Scheduling Problem (RCPSP). Paper [6] considers single-machine group scheduling problems with effects of earning and deterioration at the same

time. Its objective of scheduling problems is to minimize the makespan and the sum of completion times, respectively. J. Qian and G. Steiner in paper [7] researched scheduling problems with learning/deterioration effects and time-dependent processing times on a single machine, with or without due date assignment considerations. A hybrid estimation of distribution algorithm (HEDA) is proposed to solve the resource-constrained project scheduling problem (RCPSP) in paper [9]. Forward-backward iteration (FBI) and a permutation based local search method (PBLS) incorporated into the EDA based search to enhance the exploitation ability. Paper [10] proposes a neighborhood-windows technique for improving searching efficiency as well as a self-parameter updating technique for preventing trapping into a local optimum in high-dimensional problems. It compares PBA performance against BA and PSO performance in practical facility layout (FL) problem. M.-Z. Wang and J.-B. Wang in [11] consider the single-machine makespan minimization scheduling problem with nonlinear shortening processing times. By the nonlinear shortening processing times, it means that the processing times of jobs are non-increasing nonlinear functions of their starting times. Paper [14] researches on an inventory based two-objective job shop scheduling, in which both the make-span (the total completion time) and the inventory capacity were as objectives and were optimized simultaneously. S. K. Nayak et al. in [15] deal with the problem of dynamic task scheduling in grid environment of multi-processors. It formulates task scheduling as an optimization problem and then optimizes with a novel hybrid optimization algorithm, which combines the merits of Genetic Algorithm and Bacteria Foraging optimization. T.-C. Chiang and H.-J. Lin in [18] address the multiobjective flexible job shop scheduling problem (MOFJSP) regarding minimizing the makespan, total workload, and maximum workload. The problem is solved in a Pareto manner, whose goal is to seek for the set of Pareto optimal solutions. Paper [21] addresses the problem of minimizing the total weighted tardiness on a single-machine with a position-based learning effect. Several dominance properties are established to develop branch and bound algorithm and a lower bound is provided to derive the optimal solution. Paper [22] considers a single-machine common due window assignment and scheduling problem with batch delivery cost. Finished jobs are delivered in batches. The objective is to find a job sequence, a delivery date for each job, and a starting time and a size for the due window that jointly minimize the total cost comprising earliness, weighted number of tardy jobs, job holding, due window starting time and size, and batch delivery. In modern manufacturing and operations management, on-time delivery is a critical factor towards realizing customer satisfaction. Since then, a significant number of successful applications of GAs to JSSP have appeared in [4,16].

As due-date-related problems are usually computationally complex, most existing results are typically for problems with a simple setting. Paper [27] researched maximizing the profit with penalty of flow-shop scheduling problem from restrictive due-date. In production practice, to obtain more profit, factory does endeavor to process more production orders and that their complete time does not exceed due-date based on its resource restriction. Many approaches, especially artificial intelligence (AI) based meta-heuristics [17] are discussed. Particle swarm optimization (PSO) is an evolutionary computation technique developed by Dr. Eberhart and Dr. Kennedy in [5,12], inspired by social behavior of bird flocking or fish schooling. In the past decade, two versions of PSO, continuous and discrete, have been developed by Dr. Kennedy and Dr. Eberhart [8,13]. The continuous PSO has been applied successfully to many continuous and discrete optimization problems [19], while the applications of discrete PSO are relatively few. Paper [20] researched the parameter selection of particle swarm optimization. X. Jin et al. in [25] analyze the importance of the randomness in the PSO, and then give a PSO variant without randomness to show that traditional PSO cannot work without randomness. Paper [24]

proposes a novel Bayesian method based cognitive trust model, and then it proposes a trust dynamic level scheduling algorithm named Cloud-DLS by integrating the existing DLS algorithm. Moreover, a benchmark is structured to span a range of Cloud computing characteristics for evaluation of the proposed method. In recent years there have been some reported works focused on PSO that has been applied in many fields such as flow-shop [26] and job-shop [23] scheduling problems. Researchers proposed many algorithms to solve scheduling, but they have their limitations in practical application. Especially the PSO algorithm which was applied to solve the discrete problems reported very rare.

Relative to job-shop scheduling problems that optimize makespan or flow time, due-date-related problems are usually much more computationally complex. In this paper a job-shop scheduling problem to maximize production from restrictive different due-dates was developed. For the special scheduling optimization problem proposed in this paper, a discrete particle swarm optimization algorithm is researched to solve the special scheduling problem. The author has researched A Particle Swarm Optimization Algorithm of Inserting Particles for Flow-Shop Scheduling Problem to Maximizing Production Order Number, which has been published in Far East Journal of Dynamical Systems. It is deferent from this paper, the former study on flowshop scheduling, but this paper researches on jobshop scheduling, and the latter problem is more complex, the novel algorithm more easy to operate. This paper researches a special job-shop scheduling problem with capabilities of equipments and different due-dates, and it is as follows.

First, its model is developed, which can accord process capacity and different due-dates in numerous jobs to select some to process, and the objective is as many as possible to process production orders, and that is the maximum output. There is no documentation of the study reported on this scheduling problem.

Second, a discrete particle swarm optimization algorithm is presented for this special scheduling problem. Research on PSO algorithm reported more, but using it to solve complex scheduling problem in discrete is little. For the special scheduling optimization proposed in this paper, authors developed a discrete PSO algorithm. The novel PSO algorithm provides a new tool for the solution of the discrete problem.

Third, the novel algorithm is examined by using a set of benchmark instances with various sizes and levels of hardness and compared with other approaches reported in some existing literature works. Computation experiment shows that the efficiency of the proposed optimization approaches is efficacious.

The fourth is to investigate effect of various parameters for algorithm performance, and computational experiments are performed on different size scheduling problems. For solving different discrete scheduling problem and making its efficiency be more prominent, it can refer to the parameters debugging method of this paper to adjustment algorithm performance, and with the best performance is used to solve practical problems more widely.

In short, the literatures have not researched the special scheduling problem this paper presented and no effective algorithm to solve it. This paper researched above two problems.

## 2. Job-Shop Scheduling Problem to Maximize Production Restriction from Capacity and Different Due-Dates Formulation.

2.1. **The MPRCDJSSP description.** In practice, customers must be on time delivery, and the productive capacity of the enterprise is limited, and each order processing procedure is not the same, processing consuming hours and delivery time is different, so how to select processing orders to maximize capacity is a problem. Enterprise chooses combination different orders to process and will form different capacity bottleneck. This is

a complex selection problem of combinatorial optimization. This paper will solve the problem. The mathematical description of the problem is as follows.

The MPRCDJSSP description: One factory will choose some jobs ($j = 1, 2, \cdots, n$) from $n$ and they are to be processed on $m$ machines $M_1, M_2, \cdots, M_m$, and others are let contractor processing. Job $j$ consists of a sequence of $j_k$ ($k = 1, 2, \cdots, m$) operations $O_{1j}, O_{2j}, \cdots, O_{ij}, \cdots, O_{m_kj}$, which must be processed in this order. We have precedence constraints of the form $O_{kj} \to O_{k+1,j}$ ($k = 1, 2, \cdots, m-1$). There is a machine $\mu_{ij} \in \{M_1, M_2, \cdots, M_m\}$ and a processing time $T_{ij}$ ($i = 1, 2, \cdots, m, j = 1, 2, \cdots, n$) associated with each working procedure $O_{kj}$, $O_{kj}$ must be processed for $T_{ij}$ time units on machine $\mu_{ij}$. All jobs share different due-dates $d_j$, which are to be determined. Preemption of an operation is not allowed, thus, the operation of each job on a machine requires an uninterrupted period of time. Each job can be processed on one machine at the same time and each machine can handle only one production order at a time.

According to above constraint, when the profit of each job order win is equal, which production orders are selected to process and how to schedule can make the factory more outputs profit?

The above problem is called job-shop scheduling problem to maximize production restriction from capacity and different due-dates, and shortened form MPRCDJSSP, and this article will solve it.

2.2. **The backward sequence drawing GATT of MPRCDJSSP method.** MPRC DJSSP is not tardiness characteristic; if we use traditional former next sort method, often it will cause some tardiness, and resulted in a lot of invalid selection processing. To solve the MPRCDJSSP, a backward sequence drawing GATT of MPRCDJSSP method is proposed. The method produces every sort that is legal and valid, and a large extent improves scheduling efficiencies. Its steps are as follows.

From the final operator of last job order to the original operator of first job order in solution sequence backward sequence drawing GATT of MPRCDJSSP, while reverse order from final job order draw to anterior job order, if job order's first operator treating begin time is negative, choice tactics is from it to back all job orders (do not include the production order that first operator treating begin time is negative). For modeling convenience, we give the following hypothesis.

$C(i, j_k)$, $S(i, j_k)$ is working procedure $k$ ($k = 1, 2, \cdots, m$) of job order $j$ on machine $i$ processing complete and start time respectively.

The complete time $C(i, j_k)$ of job order $j$ on machine $i$ is less than (or equal to) the minimum of start time $S(i_{k+1}, j_{k+1})$ of the working operation $k+1$ of this job order on machine $i_{k+1}$ and start time $S_{i(j+1)}$ of back job order $j+1$ on machine $i$. If working operation $k$ is last one of job order $j$, the complete time $C(i, j_k)$ of job order $j$ on machine $i$ is less than (or equal to) the minimum of start time $S_{(i,j+1)}$ of back job order $j+1$ on machine $i$ and due-date $d_j$ of job order $j$; The start time $S(i, j_k)$ of job order $j$ on machine $i$ is equal to its complete time $C(i, j_k)$ subtraction processing time $T_{ij}$ of production order $j$ on machine $i$; Its mathematic model is as follows:

$$C(i, j_k) = \min\{S(d_j), S_{i(j+1)}\} \tag{1}$$

$$S(i, j_k) = C(i, j_k) - Tij \quad j = (n, n-1, \cdots, n') \tag{2}$$

According to the backward sequence drawing GATT of MPRCDJSSP method, we have:

$$C(i, n_k) = S(i_{k+1}, n_{k+1}), \quad S(i, n_k) = C(i_{k+1}, n_{k+1}) - T_{in}, \quad \text{for } i = 1, \cdots, m; k = 1, \cdots, m-1;$$

$$\begin{aligned} &C(i_m, j_m) = \min\{d_j, S(i_m, j+1)\}, \quad S(i_m, j_m) = C(i_m, j_m) - T_{i_m j}; \\ &\qquad\qquad j = 1, 2, \cdots, n-1 \\ &C(i, j_k) = \min\{S(i_k, j), S(i, j+1)\}, \quad S(i, j_k) = C(i, j_k) - T_{ij} \\ &\qquad\qquad \text{for } j = n, n-1, \cdots, n'; i = 1, 2, \cdots, m. \end{aligned} \tag{3}$$

Selected job orders start processing time is as follows:

$$S_{\min} = S(i_1, n'_1) \tag{4}$$

So, selected all job orders satisfy following condition:

$$j = \{n, n-1, \cdots, n' | S_{\min} = S(i_1, n'_1) >= 0, \quad S_{\min} = S(i_1, (n-1)'_1) < 0\} \tag{5}$$

2.3. **The model of MPRCDJSSP.** On the basis of before description, the MPRCDJSSP involves the following constraints.

① *Due Dates Constraints*: These constraints ensure that the job order would be ready at its due date.

$$C_{i_m j_m} \le d_j \tag{6}$$

② *Duration Constraints*: These constraints ensure that the complete time $C(i, j_k)$ of job order $j$ on machine $i$ is less than(or equal to) the minimum of start time $S_{i_m j_{k+1}}$ of working operation $k+1$ of this job order and start time $S_{i(j+1)}$ of back job order $j+1$ on machine $i$; The beginning time of job order $j$ on machine $i$ is that its complete time subtracts processing time $T_{ij}$ on this machine. And its model can be seen in (6).

Based on the backward sequence drawing GATT of MPRCDJSSP method, the mathematic model of MPRCDJSSP is as follows:

$$\begin{aligned} &\max(n - n') \\ &\text{s.t.} \begin{cases} C(i_m, j_m) \le d_j \quad (j = n, n-1, \cdots, n'-1) \\ S(i_1, j_1) \ge 0 \quad (j = n, n-1, \cdots, n'-1) \\ C(i_1, n'_1) < 0 \end{cases} \end{aligned} \tag{7}$$

## 3. Discrete Particle Swarm Optimization Algorithm for MPRCDJSSP.

3.1. **Original particle swarm optimization algorithm.** The system of particle swarm optimization algorithm (PSO) is initialized with a population of random solutions and searches for optima by updating generations. Each individual or potential solution flies in the problem space with a velocity which is dynamically adjusted according to the flying experiences of its own and its colleagues. Suppose that the searching space is $N$-dimensional and $m$ particles form the colony. The $i$th-particle represents an $N$-dimensional vector $X_i$ $(i = 1, 2, \cdots, m)$, and it means that $i$th-particle locates at $X_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$ $(D = 1, 2, \cdots, N)$ searching space. The $i$th-particle's "flying" velocity is also an $N$-dimensional vector, denoted as $V_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$. Denote the best position of $i$th-particle as $P_i = (p_{i1}, p_{i2}, \cdots, p_{iD})$, and the best position of the colony as $P_g = (p_{g1}, p_{g2}, \cdots, p_{gD})$ respectively. The original particle swarm optimization algorithm (OPSO) could be performed by the following equations [5,12].

$$v_{id}(k+1) = v_{id}(k) + c_1 r_1 (p_{id}(k) - x_{id}(k)) + c_2 r_2 (p_{gd}(k) - x_{id}(k)) \tag{8}$$

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k+1) \tag{9}$$

where $k$ represents iterative number, $c_1$, $c_2$ are learning factors, usually $c_1 = c_2 = 2$. $r_1$, $r_2$ are random numbers between $(0, 1)$.

The OPSO algorithm generally is used to solve continuous optimization problems, and rarely to scheduling problem. This paper presents a discrete particle swarm optimization algorithm for MPRCDJSSP.

### 3.2. **The discrete particle swarm optimization algorithm.**

3.2.1. *Model of DPSO algorithm.* This subsection presents a discrete PSO algorithm to solve MPRCDJSSP, shortened form DPSO. Suppose that searching space is $n$-dimensional and $m$ particles form the colony. The $i$th-particle represents an $n$-dimensional vector $X_i$ $(i = 1, 2, \cdots, m)$, and the position of each particle is a potential result. We could calculate the particle's fitness by putting its position into a designated objective function. The $i$th-particle's "flying" velocity is also an $n$-dimensional vector, denoted as $V_i = (v_{i1}, v_{i2}, \cdots, v_{in})$. The best position of $i$th-particle pbest is denoted as $P_i = (p_{i1}, p_{i2}, \cdots, p_{in})$, and the best position of global colony gbest as $P_g = (p_1, p_2, \cdots, p_n)$ respectively. Introduce it into $q_M$ new particles, at the same time with $r_N$ and $s_W$ best particle of every generation and global colony respectively displace random selecting particles in every generation. Its model expression is as follows:

$$v_{id}(k+1) = wv_{id}(k) + c_1 r_1(p_{id}(k) - x_{id}(k)) + c_2 r_2(p_{gd}(k) - x_{id}(k)) \tag{10}$$

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k+1), \quad (i = 1, 2, \cdots, m; d = 1, 2, \cdots, n) \tag{11}$$

$$X(k+1) = D(X_{q_1}, X_{q_2}, \cdots, X_{q_M}, X_{r_1}, X_{r_2}, \cdots, X_{r_N}, X_{s_1}, X_{s_2}, \cdots, X_{s_W}) \tag{12}$$

where $k$ represents the number of iterative generation, and $q_M$, $r_N$, $s_W$ $(0 \le q_M, r_N, s_W < psize)$ is random integer which denote numbers of displaced particles. $D(X_q, X_r, X_s)$ is denoted using $q_M$ new particles, $r_N$ best particle in every generation population and $s_W$ best particle of global colony displace $M+N+W$ particles of $X$, and $M+N+W$ denote whole number of every generation displacing particle. And others parameters are same as in (2) and (3). In DPSO algorithm, each particle of the swarm benefits from discoveries and previous experiences of all other colleagues during the search process.

3.2.2. *Coding scheme in DPSO algorithm.* For solving discrete MPRCDJSSP, the smallest position value (SPV) rule is employed in DPSO algorithm. In DPSO algorithm this working procedure coding is used, which gives birth to all children is flexible scheduling.

1) Sorting the weight $X = [x_1, x_2, \cdots, x_n]$ from small to large get $X' = [x'_1, x'_2, \cdots, x'_n]$, and using $y_i$ to denote the position of $x'_i$ in $X = [x_1, x_2, \cdots, x_n]$. Obtain the new sequence $Y' = [y'_1, y'_2, \cdots, y'_n]$.

2) Every weight of sequence $Y$ divides working procedure $M$, and then round towards plus infinity to get final one feasible solution sequence $Y = [y_1, y_2, \cdots, y_n]$.

3) Working procedure coding: suppose $n \times m$ JSSP, randomly give birth to following sequence:

$$se(sequence) = [1, 3, \cdots, n, 5, 3, \cdots, n, \cdots, 1, 2, \cdots, n],$$

where the number indicates the job and the same number denotes different working procedure of same job according to their order of appearance in $se$, so the same number of job $1, 2, \cdots, n$ is $m$ in se respectively.

In the following we will use one $2 \times 4$ (four jobs and every job has two working procedures) JSSP example to explain above working procedure coding.

| Dim, $n$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $X$ | 0.8147 | 0.9058 | 0.1270 | 0.9134 | 0.6324 | 0.0975 | 0.2785 | 0.5469 |
| $X'$ | 0.0975 | 0.1270 | 0.2785 | 0.5469 | 0.6324 | 0.8147 | 0.9058 | 0.9134 |
| $Y'$ | 6 | 3 | 7 | 8 | 5 | 1 | 2 | 4 |
| $Y$ | 3 | 2 | 4 | 4 | 3 | 1 | 1 | 2 |

Above working procedure coding makes real coding $X = [0.8147, 0.9058, 0.1270, 0.9134, 0.6324, 0.0975, 0.2785, 0.5469]$ transform natural number $Y = [3, 2, 4, 4, 3, 1, 1, 2]$. For

above $2 \times 4$ JSSP, solution sequence $se = [3, 2, 4, 4, 3, 1, 1, 2]$, which denotes to in turn process working procedure 1 of job 3, working procedure 1 of job 2, working procedure 1 of job 4, working procedure 2 of job 4, working procedure 2 of job 3, working procedure 1 of job 1, working procedure 2 of job 1, lastly process working procedure 2 of job 2. Using this working procedure coding scheme make every particle be feasible scheduling.

3.2.3. *Step of DPSO algorithm.* The detailed steps of DPSO algorithm are as follows.

Step 0: Let swarm population size be PS; iterative number iter be 0; terminate generation number be G and initialize others parameters that will be used in DPSO algorithm. Use (10), (11) and SPV to generate initialization particle population pop and velocity vel;

Step 1: Use (2) backward sequence drawing GATT of MPRCDJSSP (backward sequence schedule). When all production order start processing time is positive, use (7) to evaluate each particle's fitness (selected production order number); Initialize gbest position with the best fitness particle in the swarm; Initialize pbest position with a copy of particle itself;

Step 2: Generate next swarm population with Equations (10)-(12) and SPV; iter = iter + 1;

Step 3: Use (7) to compute each particle's fitness in the swarm; find new $P_g$ of the swarm and $P_i$ of each particle by comparison; update $P_i$ and $P_g$, introduce it into $q_M$ new particles and use $r_N$ best particle $P_l$ of every generation population and $s_W$ best particle $P_g$ of global colony displaced random selecting particles in population; if iter = = G, go to Step 4, or else go to Step 2;

Step 4: Output optimization result $P_g$.

The searching is a repeat process, and the stop criteria are that the end iteration number is reached or the minimum error condition is satisfied. The stop condition depends on the problem to be optimized.

In next section, the above DPSO algorithm will be used to solve a random MPRCDJSSP example and validate it effective.

4. **Computation Experiments.** To testing the efficiency of DPSO algorithm applied to solve MPRCDJSSP, we use a set of benchmark instances with various sizes and levels of hardness and compare with other approaches reported in some existing literature works. A MPRCDJSSP example is randomly produced as follows.

There are selective production orders $j$ ($j = 1, 2, \cdots, 20$) and is performed by machine $i$ ($i = 1, 2, \cdots, 5$) with fixed processing times. Production order $j$'s due-date is $d_j$, and $d_j = [450, 500, 400, 500, 250, 400, 450, 500, 450, 500, 350, 500, 400, 350, 450, 400, 500, 450, 350, 200]$. The working procedure $k$ ($k = 1, 2, 3, 4, 5$) of production order $j$ ($j = 1, 2, \cdots, 20$) process to take how much time $T_{ij}$ and by which machine is as follows:

$$O_{ij} = \begin{pmatrix} 5 & 1 & 4 & 1 & 5 & 2 & 3 & 2 & 3 & 1 & 3 & 3 & 2 & 5 & 5 & 1 & 5 & 5 & 1 & 4 \\ 1 & 3 & 3 & 5 & 3 & 1 & 2 & 1 & 4 & 5 & 5 & 4 & 3 & 4 & 1 & 4 & 2 & 3 & 2 & 1 \\ 3 & 5 & 5 & 3 & 2 & 4 & 1 & 3 & 5 & 3 & 1 & 1 & 5 & 2 & 4 & 3 & 1 & 1 & 5 & 2 \\ 4 & 2 & 2 & 4 & 1 & 3 & 4 & 4 & 2 & 4 & 4 & 5 & 4 & 1 & 3 & 5 & 4 & 2 & 4 & 3 \\ 2 & 4 & 1 & 2 & 4 & 5 & 5 & 5 & 1 & 2 & 2 & 2 & 1 & 3 & 2 & 2 & 3 & 4 & 3 & 5 \end{pmatrix}$$

$$T_{ij} = \begin{pmatrix} 16 & 30 & 4 & 32 & 38 & 20 & 14 & 8 & 6 & 22 & 28 & 12 & 24 & 18 & 10 & 40 & 36 & 26 & 34 & 2 \\ 8 & 22 & 2 & 10 & 14 & 20 & 38 & 12 & 18 & 4 & 32 & 26 & 30 & 36 & 16 & 40 & 24 & 6 & 34 & 28 \\ 4 & 30 & 20 & 22 & 26 & 18 & 34 & 8 & 16 & 12 & 28 & 40 & 6 & 24 & 32 & 14 & 10 & 2 & 38 & 36 \\ 16 & 24 & 18 & 14 & 30 & 36 & 2 & 20 & 34 & 28 & 40 & 8 & 12 & 38 & 32 & 22 & 10 & 6 & 26 & 4 \\ 24 & 38 & 22 & 10 & 28 & 16 & 14 & 34 & 30 & 36 & 6 & 32 & 20 & 2 & 18 & 8 & 26 & 4 & 12 & 40 \end{pmatrix}$$

According to factory's production capacity and different production order's due-dates, which will be chosen and what sequence can process more production? On the mathematical theory, the problem exists in $\sum_{i=1}^{20} C(20,i) = 1048575$ select methods to process. There are $\sum_{i=1}^{20} C(20,i) * (i*5)! = 9.3326 * 10^{157}$ organization production schemes of small size selected production scheduling problem. Handwork cannot solve the practical problems. We use the novel algorithm proposed earlier to completely resolve the complex issues.

For testing the efficiency of DPSO algorithm, we compare their effectiveness of DPSO, OPSO and GA to solve above random MPRCDJSSP. In GA we use mutation operation M1∼M9 and crossover operation C1, because these operations' efficiency is better shown in paper [28]. The probability displaced randomly select particles with new particles, the best particle of every generation population and the best particle of global colony is 1/3 respectively in DPSO algorithm. The crossover and mutation probability is 0.85, 0.15 respectively. To get the average performance of each algorithm twenty running on above instance are performed and the solution quality is averaged, and let the best of them be the solution of this algorithm. The paper uses the upper bound, lower bound, arithmetic mean of fitness value in twenty and running one times taking time (microsecond) four indexes to estimate these three algorithms. The computation comparing results of DPSO, OPSO and GA to solve MPRCDJSSP are as the following Table 1, in which value in form takes one radix point respectively.

**Remark 4.1.** *In Table 1, when DPSO, OPSO algorithm and GA optimize MPRCDJSSP obtain the best fitness and value use black bold to show; Max/Min/Mean/Time denote the upper bound, lower bound and arithmetic mean of fitness value respectively. Time shows running one times taking time. V denotes every generation displacing particle whole number.*

This section investigates the feasibility and effectiveness of the DPSO algorithm to solve MPRCDJSSP. Through the simulation experiment, from Table 1 we can see that

Table 1. The comparisons of DPSO, OPSO algorithm and GA for MJNRDDJSSP

| Algorithm (population size: 150, terminate generation number: 800) | | | | | | |
|---|---|---|---|---|---|---|
| Max/Min/Mean/Time | | | | | | |
| w | DPSO (with the $P_l$ displace) | | | DPSO (with the $P_g$ displace) | | |
| | V=psize/8 | V=psize/12 | V=psize/16 | V=psize/10 | V=psize/15 | V=psize/20 |
| .1 | 17/14/15.5/20.5 | 16/14/15.1/20.6 | 16/12/15.1/20.8 | 17/11/14.4/20.1 | 17/13/14.7/20.7 | 16/14/15.1/23 |
| .2 | 17/14/15.4/20.6 | 17/13/15.3/20 | 17/15/15.8/19.9 | **18/13/15.3/20.2** | 17/15/15.6/21.6 | 17/13/15.3/20.8 |
| .3 | **18/14/15.3/20** | 17/15/15.8/21.2 | 17/13/14.9/23.3 | 17/15/15.7/20.7 | 17/15/15.9/22.7 | 17/15/16.1/21.9 |
| .4 | 17/14/15.6/20.8 | 17/13/15.8/21.8 | 17/14/15.8/21.8 | **18/14/15.9/20.9** | 17/14/15.6/23.5 | 17/15/16/20.5 |
| .5 | **18/14/15.5/20.9** | 17/14/15.9/22.2 | 17/14/15.7/20 | 16/13/14.9/20.9 | 17/14/15.5/20 | 17/14/15.5/20 |
| .6 | 15/13/14.3/23.5 | 16/11/14.2/21.7 | 16/12/14.2/21.1 | 14/10/11.9/21 | 16/12/13.5/25.8 | 17/12/14.4/20.1 |
| .7 | 15/11/12.6/19.8 | 15/10/12.4/21.6 | 15/11/12.5/20.9 | 13/9/11.2/21.2 | 13/10/11/20.3 | 14/10/11.8/21.2 |

| w | DPSO (randomly with the new particle and $P_l$, $P_g$ displace) | | | OPSO | GA | |
|---|---|---|---|---|---|---|
| .1 | V=psize/10 | V=psize/14 | V=psize/18 | 15/12/14.2/21.2 | C1, M1 | 16/12/14/4 |
| | 17/13/15.2/21.7 | 17/14/15.5/22.8 | 17/15/15.6/20.2 | | C1, M2 | 16/14/14.6/4.1 |
| .2 | 17/14/15.7/20.4 | **18/14/15.8/20.6** | 17/15/15.6/20.8 | 17/12/15.3/23.2 | C1, M3 | 16/13/14.6/4.2 |
| .3 | 17/15/15.7/21.1 | **18/14/15.7/20.5** | 17/14/15.7/21.1 | **17/14/15.4/20.5** | C1, M4 | 16/13/14.6/4.2 |
| .4 | 17/15/15.7/22.2 | **18/15/15.8/20.8** | 17/14/15.5/20.1 | **17/14/15.4/20.8** | C1, M5 | 16/13/14.3/4 |
| .5 | 17/13/15.6/20.6 | 17/14/15.7/21.2 | 17/14/15.6/20.5 | 17/13/15.3/22.2 | C1, M6 | **17/13/14.4/4.1** |
| .6 | 17/12/14/20.2 | 16/12/14.3/20.1 | 16/12/13.8/20.4 | 16/12/14.8/21.6 | C1, M8 | 16/13/14.7/4.1 |
| .7 | 15/10/11.4/21.1 | 14/11/11.8/19.8 | 15/11/12.6/20.5 | 14/11/12.3/21.1 | C1, M9 | 16/13/14.4/4.2 |

using DPSO algorithm displaced randomly selected particles with new particles, the best particle of every generation population and the best particle of global colony to solve MPRCDJSSP obtained the upper bound, average and lower bound of production order number is better. The effectiveness of DPSO algorithm displaced randomly selected particles with the best particle of every generation population or the best particle of global colony takes second place, the OPSO takes third place; GA is the worst when we do not think taking time. The effectiveness of DPSO and OPSO algorithm optimizing MPRCDJSSP with inertia coefficient $w \in [0.1\ 0.5]$ is the best. In DPSO algorithm, the number of displaced particles should be $N \in [PS/15\ PS/10]$. MOPSO and OPSO algorithms optimize MPRCDJSSP that running one times taking time is more than GA, because they coding process from real to integer takes much time, and the GA using natural number taking time is small.

Use DPSO algorithm to optimize above MPRCDJSSP and obtained the best product scheme is choice 18 production orders to process. The best scheduling is 9 as the following

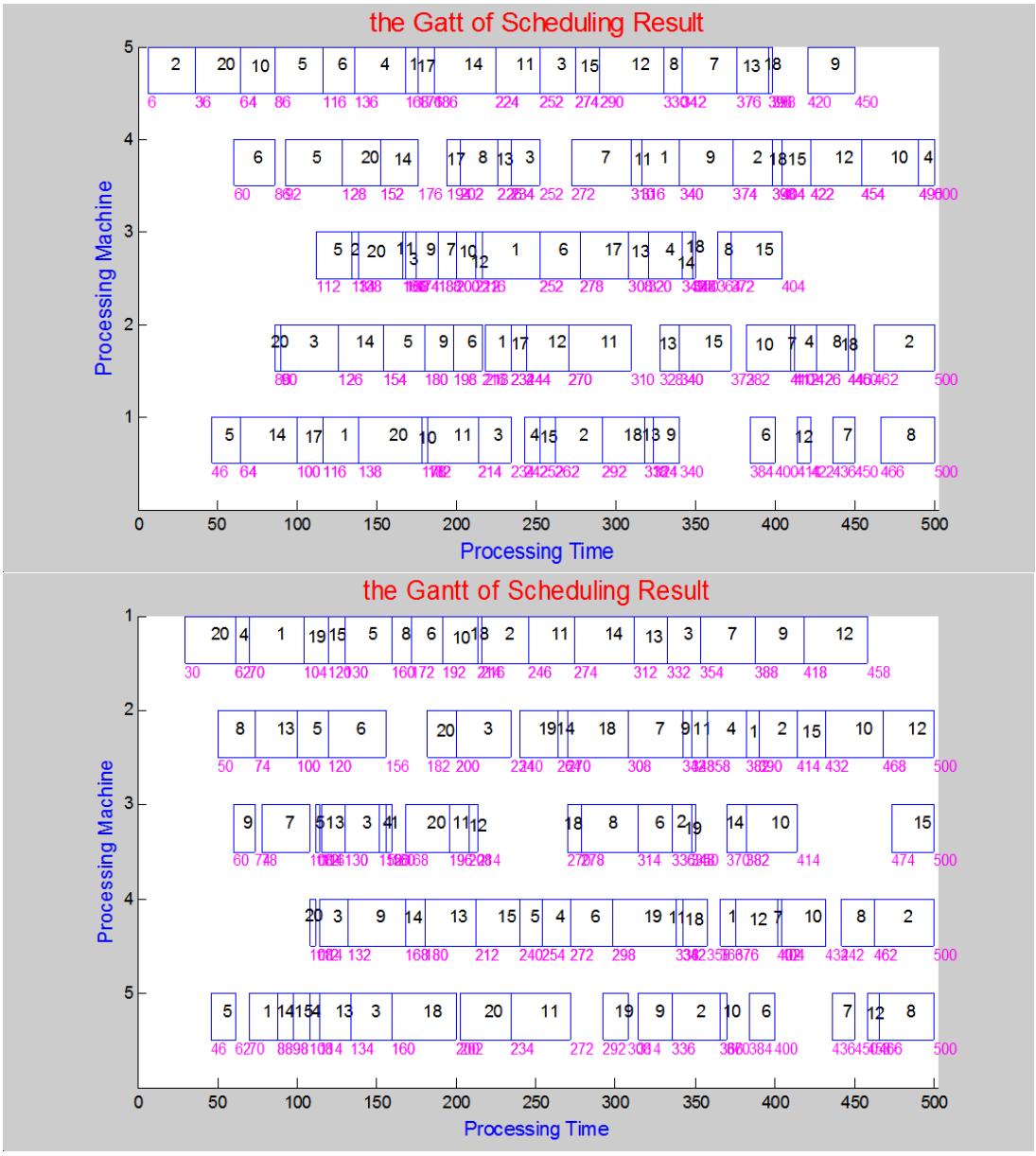


FIGURE 1. GATT of MPRCDJSSP select and scheduling

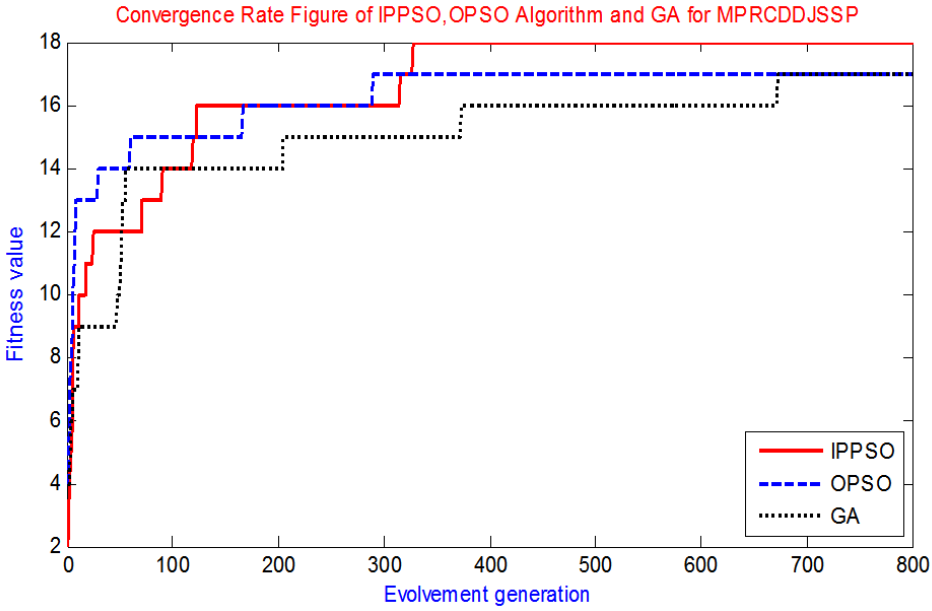


FIGURE 2. Convergence figure of DPSO, OPSO algorithm and GA for MPRCDJSSP

respectively: $\{20, 9, 8, 5, 7, 13, 1, 14, 4, 6, 15, 19, 3, 18, 11, 12, 10, 2\}$; $\{5, 14, 17, 2, 20, 6, 3, 10, 1, 11, 9, 8, 7, 13, 4, 12, 15, 18\}$; $\{11, 20, 1, 5, 10, 7, 8, 9, 2, 14, 3, 4, 13, 6, 15, 18, 17, 12\}$; $\{16, 4, 5, 14, 8, 18, 20, 12, 13, 1, 9, 7, 11, 15, 6, 3, 17, 10\}$; $\{20, 10, 1, 11, 15, 3, 5, 6, 13, 12, 17, 9, 7, 8, 4, 14, 2, 18\}$; $\{5, 6, 20, 15, 11, 2, 17, 1, 19, 8, 12, 9, 13, 7, 18, 4, 3, 10\}$; $\{5, 12, 10, 7, 6, 14, 20, 3, 4, 9, 17, 13, 1, 8, 16, 18, 15, 2\}$; $\{13, 10, 5, 20, 2, 19, 14, 11, 3, 9, 1, 6, 12, 15, 7, 8, 18, 17\}$;$\{20, 7, 16, 5, 14, 15, 18, 6, 8, 3, 10, 13, 2, 1, 12, 9, 4, 17\}$. The two best schedulings with DPSO algorithm to optimize above the MPRCDJSSP see Gantt in Figure 1. According to the optimal scheduling results to organize production, otherwise 18 the same orders of production selected to process will also cause the tardiness.

DPSO algorithm the convergence contrast of DPSO, OPSO algorithm and GA with their best parameter and operation respectively solving above MPRCDJSSP is shown by Figure 2.

From simulation solution we can see clearly that the optimal processing production order number got by DPSO algorithm is more than OPSO algorithm and GA. The convergence rate of former to optimize MPRCDJSSP is faster than both in the latter. Computational experiments show the DPSO algorithm for MPRCDJSSP is very valid, especially for many production orders and few machines problem.

5. **Conclusion.** In many research scheduling literatures, they usually study minimizing the makespan and obtaining great harvest. In this paper we present a job-shop scheduling problem to maximize production restriction from capacity and different due-dates, whose objective is to process as much as possible production orders according to factory's production capacity and different due-dates. The model of this special problem is set up, which selectively produces production order and cannot exceed each production order's due-dates. According to its special, the DPSO algorithm is proposed to solve it. Experiments show the high performance of DPSO algorithm and the efficiency of the proposed solving approaches.

There are a number of research directions that can be considered as useful extensions of this research. Future works will investigate applying the developed DPSO algorithm to other optimization problems such as discrete scheduling problem and TSP.

**Acknowledgment.** This work is supported by the Humanities and Social Sciences Youth Fund Project of Ministry of Education (Grant No. 09yjc630151); Innovation Project of Shanghai Science and Technology Commission (Grant No. 11YZ268); Shanghai Dianji University Computer application technology disciplines (Grant No. 13XKJ01). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

## REFERENCES

[1] A. Dolgui, V. Gordon and V. Strusevich, Single machine scheduling with precedence constraints and appositionally dependent processing times, *Computers & Operations Research*, vol.39, pp.1218-1224, 2012.

[2] C.-M. Wei, J.-B. Wang and P. Ji, Single-machine scheduling with time-and-resource-dependent processing times, *Applied Mathematical Modelling*, vol.36, pp.792-798, 2012.

[3] D. C. Paraskevopoulos, C. D. Tarantilis and G. Ioannou, Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm, *Expert Systems with Applications*, vol.39, pp.3983-3994, 2012.

[4] H. Zhou et al., Minimizing weighted tardiness of job-shop scheduling using a hybrid genetic algorithm, *European Journal of Operational Research*, 2009.

[5] J. Kennedy and R. Eberhart, Particle swarm optimization, *IEEE Int. Conf. on Neural Networks*, Perth, Australia, pp.1942-1948, 1995.

[6] J. Bai, Z.-R. Li and X. Huang, Single-machine group scheduling with general deterioration and learning effects, *Applied Mathematical Modelling*, vol.36, pp.1267-1274, 2012.

[7] J. Qian and G. Steiner, Fast algorithms for scheduling with learning effects and time-dependent processing times on a single machine, *European Journal of Operational Research*, vol.225, pp.547-551, 2013.

[8] J. Kennedy and R. C. Eberhart, A discrete binary version of the particle swarm algorithm, *Proc. of the World Multiconference on Systemics, Cybernetics and Informatics*, pp.4104-4109, 1997.

[9] L. Wang and C. Fang, A hybrid estimation of distribution algorithm for solving the resource-constrained project scheduling problem, *Expert Systems with Applications*, vol.39, pp.2451-2460, 2012.

[10] M.-Y. Cheng and L.-C. Lien, A hybrid AI-based particle bee algorithm for facility layout optimization, *Engineering with Computers*, vol.28, pp.57-69, 2012.

[11] M.-Z. Wang and J.-B. Wang, Single-machine makespan minimization scheduling with nonlinear shortening processing times, *Computers & Operations Research*, vol.39, pp.659-663, 2012.

[12] R. C. Eberhart and J. Kennedy, A new optimizer using particle swarm theory, *Proc. of the 6th International Symposium on Micro Machine and Human Science*, Nagoya, Japan, pp.39-43, 1995.

[13] R. C. Eberhart and Y. Shi, Particle swarms optimization: Developments, applications and resources, *Proc. of Congress on Evolutionary Computation*, Seoul, Korea, 2001.

[14] Q. Ren, Y. Wang and X. Wang, Inventory based two-objective job shop scheduling model and its hybrid genetic algorithm, *Applied Soft Computing*, vol.13, pp.1400-1406, 2013.

[15] S. K. Nayak, S. K. Padhy and S. P. Panigrahi, A novel algorithm for dynamic task scheduling, *Future Generation Computer Systems*, vol.28, pp.709-717, 2012.

[16] S.-H. Chen, P.-C. Chan, T. C. E. Cheng and Q. Zhang, A self-guided genetic algorithm for permutation flowshop scheduling problems, *Computers & Operations Research*, vol.39, pp.1450-1457, 2012.

[17] T. K. Liu et al., Modified genetic algorithm for the job-shop scheduling problem, *Int. J. Adv. Manuf. Technol.*, vol.27, no.9/10, pp.1021-1029, 2006.

[18] T.-C. Chiang and H.-J. Lin, A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling, *Int. J. Production Economics*, vol.141, pp.87-98, 2013.

[19] F. Van den Bergh and A. P. Engelbrecht, Cooperative learning in neural network using particle swarm optimizers, *S. Afr. Comput. J.*, vol.26, pp.84-90, 2000.

[20] Y. Shi and R. C. Eberhart, Parameter selection in particle swarm optimization, *Proc. of the 7th Annual Conference on Evolutionary Programming*, New York, pp.591-600, 1998.

[21] Y. Yin, C.-C. Wu, W.-H. Wu and S.-R. Cheng, The single-machine total weighted tardiness scheduling problem with position-based learning effects, *Computers & Operations Research*, vol.39, pp.1109-1116, 2012.

[22] Y. Yin, T. C. E. Cheng, J. Wang and C.-C. Wu, Single-machine common due window assignment and scheduling to minimize the total cost, *Discrete Optimization*, vol.10, pp.42-53, 2013.
[23] W. J. Xia and Z. M. Wu, A hybrid particle swarm optimization approach for the job-shop scheduling problem, *Int. J. Adv. Manuf. Technol.*, vol.29, no.3/4, pp.360-366, 2006.
[24] W. Wang, G. Zeng, D. Tang and J. Yao. Cloud-DLS: Dynamic trusted scheduling for cloud computing, *Expert Systems with Applications*, vol.39, pp.2321-2329, 2012.
[25] X. Jin et al., Particle swarm optimization using dimension selection methods, *Applied Mathematics and Computation*, vol.219, pp.5185-5197, 2013.
[26] Z. G. Lian et al., A similar particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan, *Applied Mathematics and Computation*, vol.175, no.1, pp.773-785, 2006.
[27] Z. G. Lian et al., A maximizing the profit with penalty of flow-shop scheduling problem from restrictive due-date, *Proc. of Asia Simulation Conf./the 6th Int. Conf. on System Simulation and Scientific Computing*, Beijing, pp.1387-1392, 2005.
[28] Z. G. Lian et al., A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan, *Applied Mathematics and Computation*, vol.183, pp.1008-1017, 2006.
[29] Z. G. Lian et al., A particle swarm optimization algorithm of inserting particles for flow-shop scheduling problem to maximizing production order number, *Far East Journal of Dynamical Systems*, vol.13, no.2, pp.97-111, 2010.