Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 387082, 17 pages
http://dx.doi.org/10.1155/2015/387082

Research Article

A Mutualism Quantum Genetic Algorithm to Optimize the Flow Shop Scheduling with Pickup and Delivery Considerations

Jinwei Gu,[1] Manzhan Gu,[2] and Xingsheng Gu[3]

[1]*School of Economics and Management, Shanghai University of Electric Power, Shanghai 200090, China*
[2]*School of Mathematics, Shanghai University of Finance and Economics, Shanghai 200433, China*
[3]*Research Institute of Automation, East China University of Science and Technology, Shanghai 200237, China*

Correspondence should be addressed to Jinwei Gu; gujinwei1982@163.com

Received 5 June 2014; Revised 3 November 2014; Accepted 11 November 2014

Academic Editor: Kang Li

A mutualism quantum genetic algorithm (MQGA) is proposed for an integrated supply chain scheduling with the materials pickup, flow shop scheduling, and the finished products delivery. The objective is to minimize the makespan, that is, the arrival time of the last finished product to the customer. In MQGA, a new symbiosis strategy named mutualism is proposed to adjust the size of each population dynamically by regarding the mutual influence relation of the two subpopulations. A hybrid Q-bit coding method and a local speeding-up method are designed to increase the diversity of genes, and a checking routine is carried out to ensure the feasibility of each solution; that is, the total physical space of each delivery batch could not exceed the capacity of the vehicle. Compared with the modified genetic algorithm (MGA) and the quantum-inspired genetic algorithm (QGA), the effectiveness and efficiency of the MQGA are validated by numerical experiments.

1. Introduction

The coordination of logistics activities in a supply chain has received a lot of attention recently. From the manufacturer's point of view, the important problem in the supply chain scheduling is the coordination of the three stages including material supply, production scheduling, and product delivery. Traditionally, research on scheduling generally focuses on the models with various machine setting, job characteristics, and performance measures [1], and the transportation arrangements of the materials and the finished products are normally ignored in these models.

In the past decades, many researchers studied the machine scheduling problems with transportation under consideration. Some of the results consider the transportation of the jobs between machines in the flow shop model, which incorporates transport times when the jobs are transferred from one machine to another. Maggu et al. [2–4] firstly considered the scheduling problem with transportation between machines. In addition, Langston [5] designed algorithms for the problem of planning and coordinating movement within a deterministic flow shop system and analyzed the worst-case performances of the algorithms. Haouari and Ladhari [6] presented an effective branch and bound algorithm for the model proposed by Langston [5]. Hurink and Knust [7] considered the flow shop problems with transportation and a single robot with the objective of minimizing the makespan. They derived complexity results for the problems. Hurink and Knust [8] developed a tabu search algorithm for the problem with time windows. Hurink and Knust [9] extended some of the results in [8] to the job-shop scheduling models. For the model with a single robot, Lee and Strusevich [10] studied the two machine flow shop and open shop problems. They presented a best possible approximation algorithm for each of the two problems with some constraints. Naderi et al. [11] proposed simulated annealing algorithms for flow shop problems with the objective of minimizing the total weighted tardiness and makespan. For more new results on the scheduling problems with transportation between machines, please see [12–15], among others.

Another type of scheduling model with transportation focuses on the delivery of finished jobs to customers.

Lee and Chen [16] considered some scheduling models that incorporate the delivery decisions of the finished jobs. They researched the computational complexity of some problems and proposed polynomial or pseudopolynomial algorithms for them. Chang and Lee [17] extended one of the models in [16] to the environment in which each job has an individual amount of space. Li et al. [18] studied a single-machine scheduling problem with routing decisions of a vehicle that serves customers at different addresses. In addition, Chen and Vairaktarakis [19] researched eight problems with production and distribution under consideration and either designed a polynomial time algorithm or proved the NP-hardness for each of the problems. Geismar et al. [20] extended the Chen and Vairaktarakis model to the problem with a short shelf life and no inventory in the process. In addition, Soukhal et al. [21] investigated the scheduling models with constraints on both transportation and buffer capacities to minimize the makespan. They proved new complexity results for special cases of the problems considered. We refer to, for example, [22–25], for more recent results on the model with delivery operations for finished products.

This paper considers the scheduling model that integrates the pickup of materials, flow shop scheduling, and the delivery of finished jobs. In this model, the material warehouse, the factory, and the customer are located at three different places. There are two vehicles (namely, conveyor and truck) each with a limited capacity. One vehicle (conveyor) travels between the factory and the warehouse for material transportation, and the other vehicle (truck) travels between the customer and the factory for finished products delivery. This model applies to many situations in supply chain business activities. For example, a shoes manufacturer purchases materials from India and arranges the production in China. Finally, the finished products are delivered to the USA.

To the best of our knowledge, research on this model includes Hall and Potts [26], Li and Ou [27], and Wang and Cheng [28]. Hall and Potts [26] provided algorithms for models in which jobs are produced on machines and formed in batches for delivery. Li and Ou [27] studied the problem with a vehicle traveling between a warehouse and the factory. They proved the problem is NP-hard in strong sense and developed polynomial time and effective heuristic algorithms, respectively, for special case and the general setting. Wang and Cheng [28] studied the complexity of the model in which the warehouse, the factory, and the customer locations are different. They proved the problem is strongly NP-hard and proposed a heuristic with a tight bound of 2.

The model considered in this paper is different from that in [26]. Each job in our model has its own transportation time between the warehouse and the factory and between the factory and the customer. Our model also differs from Li and Ou's model in which the locations of the warehouse and the customer are the same. Furthermore, this paper considers flow shop during the production scheduling, while the model in [28] considers processing jobs in a single machine.

Quantum genetic algorithms (QGA) are heuristic search techniques inspired from the principles of survival of the fittest in natural genetic evolution and quantum theory. They are known to be efficient in a large search space, without explicitly requiring additional information (such as convexity or derivative information) about the objective. In addition, the Q-bit encoding method has the superposition and probability expression characteristics, and each individual can express more states to increase the diversity of populations. For these reasons, in the last few years, they have been applied to many combinatorial problems, including scheduling and vehicle routing applications that are partially related to our problem.

In order to solve the integrated supply chain scheduling model with the materials pickup, flow shop scheduling, and the finished products delivery, a technique called mutualism quantum genetic algorithm (MQGA) is developed, which differs from traditional QGA method in two aspects. *The first aspect* is the encoding method. With the encoding method in [29], a chromosome including two segments, the job sequence and the job-to-batch assignment, is designed, and this encoding method could equally and uniquely represent all possible solutions. Furthermore, in order to improve the encoding performance, a local speeding-up method is introduced to perform the deep search operation. After that, a checking routine is carried out to ensure the feasibility of each solution, that is, ensure that the total physical space of each delivery batch could not exceed the capacity of the vehicle. *The second aspect* is as follows: based on the relationship between two mutually symbiotic species, a population growth model (namely, mutualism) is introduced to improve the performance of the algorithm. In this model, a symbiosis strategy is developed to dynamically adjust the sizes of two subpopulations. This strategy helps to increase the diversity of genes and avoid premature convergence.

The remainder of this paper is organized as follows. In Section 2, the problem including the assumptions and notations is introduced. In Section 3, the two species growth model and the corresponding evolutionary strategy are proposed. The mechanism of MQGA is introduced in Section 4, and the experimental results are discussed in Section 5. Section 6 contains some concluding remarks.

2. Problem Description

In this paper, it is assumed that the material warehouse, the factory, and the customer are located at different addresses, and there is a set of jobs, $J = \{J_1, J_2, \ldots, J_n\}$, processed in a flow shop scheduling, which consists of m machines $M = \{M_1, M_2, \ldots, M_m\}$. All unprocessed jobs are initially located at the supplier's warehouse and need to be transported to the factory for processing by a vehicle (conveyor). The conveyor is located at the warehouse at the beginning and is available for transportation of jobs from warehouse to the factory and comes back. Each pickup journey takes a constant amount of time, and in a trip the conveyor loads a limited number of jobs due to the limitation of its space capacity. In addition, the finished jobs need to be delivered to a customer by another vehicle (truck), which is initially located at the factory. Because of the constraint of its space capacity, the truck transports limited finished jobs to the customer and returns to the factory. Each trip also takes a constant amount

of time. The objective is to minimize the makespan, that is, the arrival time of the last finished job to the customer. Figure 1 gives a description of the integrated supply chain problem including the pickup of materials, flow shop scheduling, and the delivery of finished products.

2.1. Assumptions. Before the introduction of the notations, a number of assumptions are given as follows:

(i) all the facilities including conveyor, machines, and truck are available from time zero;

(ii) there is no idle time between any consecutive two pickup journeys;

(iii) the time of loading and unloading jobs is included in the pickup time and delivery time;

(iv) the time of transporting jobs between machines is negligible;

(v) the storage or buffer capacities between successive machines are unlimited;

(vi) there is no priority among jobs;

(vii) machine failure is not considered.

2.2. Notations. The following notations are used in the problem model:

c_i: physical space of job J_i;

$S_1(S_2)$: physical space capacity of the conveyor (truck);

$T_{11}(T_{12})$: travel time from the warehouse to the factory (from the factory to the warehouse);

$T_{21}(T_{22})$: travel time from the factory to the customer (from the customer to the factory);

l_{1i}: the arrival time of J_i to the factory;

l_{2i}: the arrival time of J_i to the customer;

p_{ik}: processing time of J_i on machine M_k;

s_{ko}: the earliest start time of the oth position job on machine M_k;

u_{1j}: departure time of the jth batch from the warehouse to the workshop;

u_{2j}: departure time of the jth batch from the workshop to the customer;

q_{iko}: equal to 1 if job J_i is scheduled on the oth position on machine M_k; 0, otherwise;

L_j: the latest completion time of the jobs in the jth delivery batch;

U_{ij}: equal to 1 if job J_i is in the jth pickup batch; 0, otherwise;

V_{ij}: equal to 1 if job J_i is in the jth delivery batch; 0, otherwise;

M: very large positive constant.

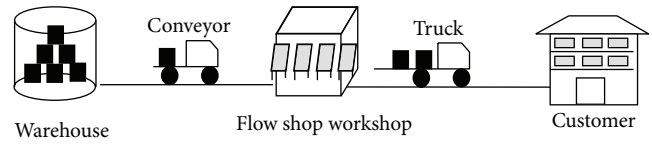

Figure 1: Schematic of the cooperative problem of flow shop production with pickup and delivery transportations.

2.3. The Optimization Model. Tang and Gong [30] proposed a mixed integer programming (MIP) model for a single-machine batch scheduling with pickup and delivery transportations. Based on it, this paper studies a different integrated supply chain problem, which consists of the pickup of materials, flow shop scheduling, and the delivery of finished products. For this problem, a MIP model is constructed below. This model is to decide the following policy: (1) how and when the materials are assigned to an appropriate delivery batch, (2) what the scheduling of n jobs on a series of m machines is, (3) how to dispatch the finished product as soon as possible to satisfy the customer's demand:

$$\min \quad \left(\max_{1\le i\le n} l_{2i}\right) \tag{1}$$

$$\text{Subject to} \sum_{o=1}^{n} q_{iko} = 1, \quad i = 1,2,\ldots,n,\ k = 1,2,\ldots,m, \tag{2}$$

$$\sum_{i=1}^{n} q_{iko} = 1, \quad o = 1,2,\ldots,n,\ k = 1,2,\ldots,m, \tag{3}$$

$$s_{ko} + \sum_{i=1}^{n} p_{ik} q_{iko} \le s_{k,o+1}, \quad k = 1,2,\ldots,m, \quad o = 1,2,\ldots,n-1 \tag{4}$$

$$s_{ko} + \sum_{i=1}^{n} p_{ik} q_{iko} \le s_{k+1,o}, \quad k = 1,2,\ldots,m-1, \quad o = 1,2,\ldots,n, \tag{5}$$

$$s_{1o} \ge l_{1i} - M\left(1 - q_{i1o}\right), \quad i = 1,2,\ldots,n,\ o = 1,2,\ldots,n, \tag{6}$$

$$L_j \ge s_{mo} + p_{im} - M\left(2 - V_{ij} - q_{imo}\right), \quad i = 1,2,\ldots,n, \quad o = 1,2,\ldots,n, \tag{7}$$

$$\sum_{i=1}^{n} c_i U_{ij} \le S_1, \quad j = 1,2,\ldots,n, \tag{8}$$

$$\sum_{i=1}^{n} c_i V_{ij} \le S_2, \quad j = 1,2,\ldots,n, \tag{9}$$

$$\sum_{j=1}^{n} U_{ij} = 1, \quad i = 1,2,\ldots,n, \tag{10}$$

$$\sum_{j=1}^{n} V_{ij} = 1, \quad i = 1, 2, \ldots, n, \tag{11}$$

$$\sum_{i=1}^{n} U_{i(j+1)} \le M \sum_{i=1}^{n} U_{ij}, \quad j = 1, 2, \ldots, n-1, \tag{12}$$

$$\sum_{i=1}^{n} V_{i(j+1)} \le M \sum_{i=1}^{n} V_{ij}, \quad j = 1, 2, \ldots, n-1, \tag{13}$$

$$u_{21} = L_1, \tag{14}$$

$$u_{2j} \ge L_j, \quad j = 2, 3, \ldots, n, \tag{15}$$

$$u_{1j} \ge u_{1(j-1)} + T_{11} + T_{12}, \quad j = 2, 3, \ldots, n, \tag{16}$$

$$u_{2j} \ge u_{2(j-1)} + T_{21} + T_{22}, \quad j = 2, 3, \ldots, n, \tag{17}$$

$$l_{1i} \ge u_{1j} + T_{11} - M\left(1 - U_{ij}\right), \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, n, \tag{18}$$

$$l_{2i} \ge u_{2j} + T_{21} - M\left(1 - V_{ij}\right), \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, n, \tag{19}$$

$$l_{2i} \ge 0, \quad i = 1, 2, \ldots, n, \tag{20}$$

$$L_j \ge 0, \quad j = 1, 2, \ldots, n, \tag{21}$$

$$u_{1j} \ge 0, \quad u_{2j} \ge 0, \quad j = 1, 2, \ldots, n, \tag{22}$$

$$s_{ko} \ge 0, \quad k = 1, 2, \ldots, m, \ o = 1, 2, \ldots, n, \tag{23}$$

$$U_{ij}, V_{ij} \in \{0, 1\}, \quad i = 1, 2, \ldots, n, \ j = 1, 2, \ldots, n, \tag{24}$$

$$q_{iko} \in \{0, 1\}, \quad i = 1, 2, \ldots, n, \quad k = 1, 2, \ldots, m, \quad o = 1, 2, \ldots, n. \tag{25}$$

Objective function (1) represents the arrival time of the last finished product to the customer. Constraints (2) ensure that job J_i should be placed in one and only one position on each machine. Constraints (3) guarantee that there is one and only one job processed on the oth position on each machine. Constraints (4) define that the earliest start time of the job on the $(o+1)$th position should not be earlier than the completion time of the oth position job on the same machine. Constraints (5) ensure that, for the same job, the earliest start time on machine M_{k+1} should not be earlier than its completion time on machine M_k. Constraints (6) define that the start processing time of a job should not be earlier than its release time. Constraints (7) indicate that the ready time for delivering the oth job should not be earlier than its completion time on the last machine. Constraints (8) and (9) define that the number of jobs in each batch should satisfy the space capacity of the conveyor or the truck. Constraints (10) and (11) determine each job to be assigned to one and only one batch. Constraints (12) and (13) indicate that if no job is assigned to batch j, then no job will be assigned to batch $(j+1)$. Constraints (14) and (15) define the departure time u_{1j}. Constraints (16) and (17) define that the departure time of the jth batch should not be earlier than the returning time of the $(j-1)$th batch. Constrains (18) and (19) specify the nonnegativity of l_{1i} and l_{2i}. Equations (20)–(25) define the variables.

3. The Mutualism Strategy for Population Growth

As the long evolutionary process of nature, the relationship between living beings is complicated. The phenomenon that two species live together is generally referred to as the symbiosis. This paper studies a kind of symbiosis called mutualism, which refers to two species living together and depending on each other over a long period of time. The nutrition of one species is the food source of the other one.

Mutualism brings two advantages to our algorithm. Firstly, there is no need to design the fitness function, and the fitness value of an individual is obtained by the cooperation between two populations, which could reduce the dependence on the domain knowledge. Secondly, different from the traditional framework in which the evolution of populations depends on the fitness values of individuals, mutualism incorporates the cooperative behavior between an individual and its surroundings, so as to postpone the premature convergence and improve the convergence speed.

3.1. The Mutualism Population Growth Model. A differential equation of mutualism population growth model is introduced in this subsection. Assume two species A and B live in the same environment and they do not take each other as food, but the existence of one species can promote the population growth of the other one. For example, algae and fungi are two species living in lichen. Algae provide nutrient to fungi through photosynthesis, and fungi offer algae water and inorganic substance. If algae and fungi are separated in lichen, both of them will die.

Let $x(t)$ and $y(t)$, respectively, be the numbers of individuals of species A and B at the tth generation. Define the maximal individual number of species A (B) by $K_1(K_2)$, and the cooperation degree is $a_1(a_2)$. In this way, the differential equation of the cooperation system can be written as follows:

$$\frac{dx(t)}{dt} = r_1\left(1 - \frac{x(t)}{K_2} + \frac{a_1 y(t)}{K_1}\right)x(t), \tag{26}$$

$$\frac{dy(t)}{dt} = r_2\left(1 + \frac{a_2 x(t)}{K_2} - \frac{y(t)}{K_1}\right)y(t), \tag{27}$$

where r_i $(i = 1, 2)$ denotes the accrual rate, that is, birth rate subtracting death rate. In time interval $[t, t+\Delta t]$, define $x(t+\Delta t) - x(t) = r_1 \cdot \Delta t \cdot x(t)$ as the modified number of species A. So $\Delta x/\Delta t = r_1 x(t)$. Once the population number reaches the maximal value, species A will stop growing, and the accrual rate r_1 will become zero. $1 - x(t)/K_2$ denotes the block function on the population size of species A. Suppose the total amount of food is 1; then $x(t)/K_2$ is the amount of food consumed by species A. When species A and B live together in the same environment, they cooperate with each other and bring direct advantages for population growth,

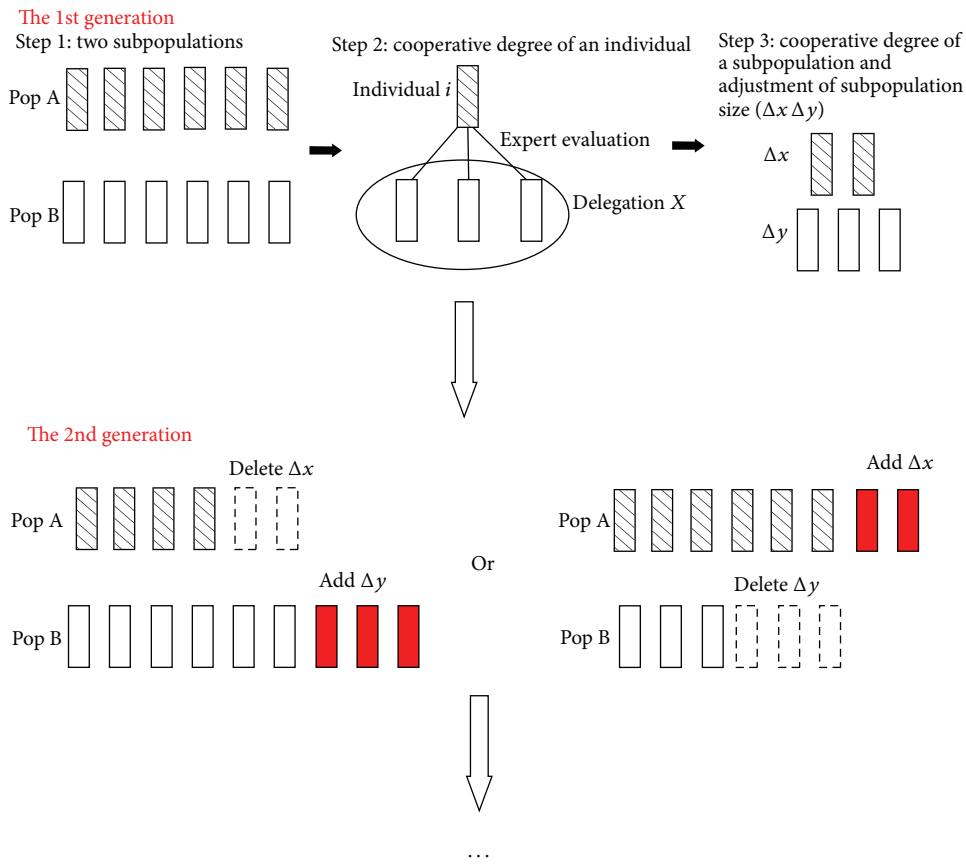

Figure 2: Description of the mutualism strategy for population growth.

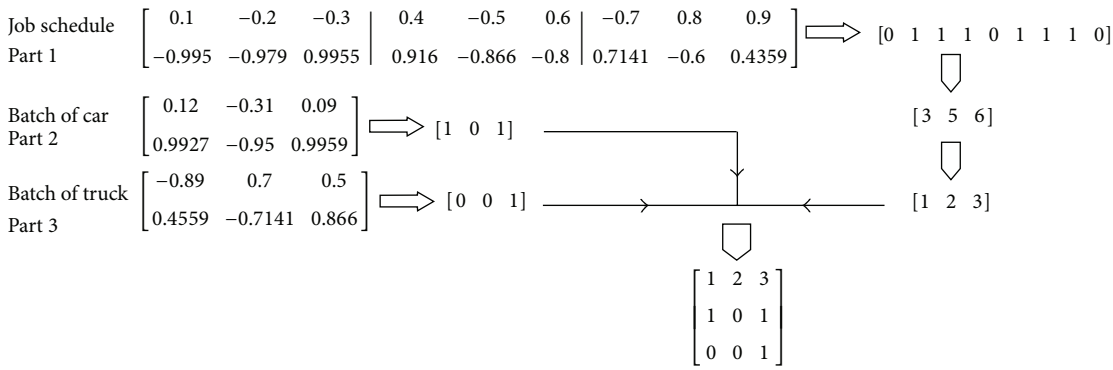

Figure 3: The encoding and decoding mode of Q-bit.

and $\alpha_1 y(t)/K_1$ denotes the influence degree of species B on the population size of species A. We have similar explanation for (27).

3.2. The Mutualism Strategy. In the standard genetic algorithm, the fitness function will be given as an input, and the fitness is easy to reach the peak value. However, in the real environment the adaptability of a species is dynamic and is affected by its surroundings. Therefore, in MQGA, the fitness value of an individual is replaced by its cooperation degree with individuals in other species, and the details on the computation of the cooperation degree are given below.

3.2.1. Expert Evaluation Method. Given two individuals i and j, respectively, from populations A and B, a method is proposed to evaluate the cooperation degree of individual i. Let ob_i and ob_j be the objective values of i and j. Denote, respectively, by $\mathrm{ob}_{\min}$, $\mathrm{ob}_{\max}$, and $\overline{\mathrm{ob}}$ the minimum value, the maximum value, and the average objective value of all individuals in A and B. By cooperating with individual j, define Score_{ij} as the cooperative degree of individual i.

Case 1. Consider $\mathrm{ob}_i \in [\mathrm{ob}_{\min}, \overline{\mathrm{ob}}]$, $\mathrm{ob}_j \in [\mathrm{ob}_{\min}, \overline{\mathrm{ob}}]$. In this case, each of the two individuals has a high survival rate and

Table 1: The cooperation degree of individual i (with individual j) Score_{ij}.

Condition	Score
$\text{ob}_i \in [\text{ob}_{\min}, \overline{\text{ob}}]$, $\text{ob}_j \in [\text{ob}_{\min}, \overline{\text{ob}}]$	4
$\text{ob}_i \in [\text{ob}_{\min}, \overline{\text{ob}}]$, $\text{ob}_j \in [\overline{\text{ob}}, \text{ob}_{\max}]$	2
$\text{ob}_i \in [\overline{\text{ob}}, \text{ob}_{\max}]$, $\text{ob}_j \in [\text{ob}_{\min}, \overline{\text{ob}}]$	0
$\text{ob}_i \in [\overline{\text{ob}}, \text{ob}_{\max}]$, $\text{ob}_j \in [\overline{\text{ob}}, \text{ob}_{\max}]$	1

Table 2: The processing times of the 8 jobs on 2 machines.

Processing time	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
M_1	3	1	2	1	2	3	3	4
M_2	4	1	2	3	4	3	1	2

the "matching" ability to cooperate. They can offer nutrition to each other, and the collaborative effect is the best. So, the cooperative degree of individual i is set to be "4."

Case 2. Consider $\text{ob}_i \in [\text{ob}_{\min}, \overline{\text{ob}}]$, $\text{ob}_j \in [\overline{\text{ob}}, \text{ob}_{\max}]$. In this case, compared to individual j, individual i has a smaller objective value, which indicates that individual i is more capable of growing and developing. If the two individuals work together, individual i cannot obtain enough nutrition from individual j, and the cooperation is a "mismatch." Therefore, the cooperative degree of individual i is set to be "2."

Case 3. Consider $\text{ob}_i \in [\overline{\text{ob}}, \text{ob}_{\max}]$, $\text{ob}_j \in [\text{ob}_{\min}, \overline{\text{ob}}]$. Known from Case 2, in this case individual j is more capable of growing and developing, and individual j will absorb more nutrition from individual i. That will eventually impede the development and growth of individual i. Therefore, the cooperative degree of individual i is defined to be "0."

Case 4. Consider $\text{ob}_i \in [\overline{\text{ob}}, \text{ob}_{\max}]$, $\text{ob}_j \in [\overline{\text{ob}}, \text{ob}_{\max}]$. In this case, the two individuals have the similar low levels of growing, and a long period of time is needed for the development of the individuals. But they have the "matching" abilities and can cooperate with each other. Therefore, the cooperative degree of individual i is set to be "1." The four cases are summarized in Table 1.

3.2.2. Cooperative Degree of Individual. Given two populations A and B, denote by $c\text{Fitness}_i$ the cooperative degree of individual i in A. Firstly choose some individuals from B to form a delegation X, and X can be chosen by ways including random selection, greedy selection, or tournament. Then let individual i cooperate with each individual in X with the expert evaluation method. The fitness of i is calculated as follows:

$$c\text{Fitness}_i = \sum_{j \in X} \text{Score}_{ij}, \tag{28}$$

where Score_{ij} denotes the cooperative degree of individual i by cooperating with individual j.

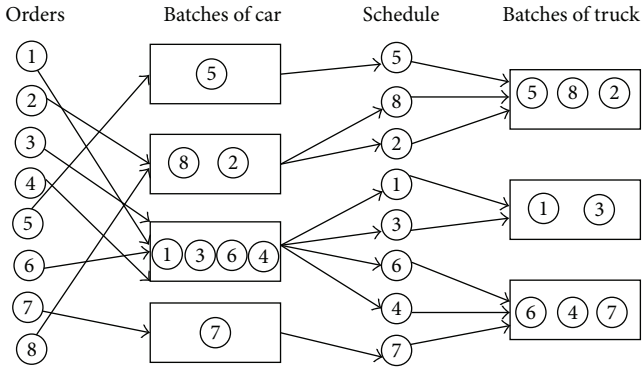

Figure 4: Job sequence and the job-to-batch assignment.

3.2.3. Cooperative Degree of a Population. The cooperative degree of a population stands for the cooperative level of the population, and it can be calculated as follows:

$$a_1 = \frac{\sum_{i \in \text{Pop1}} c\text{Fitness}_i}{\sum_{i \in \text{Pop2}} c\text{Fitness}_i}, \qquad a_2 = \frac{1}{a_1}, \tag{29}$$

where Pop1 (Pop2) is the delegation of population A (B). If $a_1 > 1$, A is more cooperative and obtains more advantage or nutrition from B. Conversely, if $a_2 > 1$, B is more cooperative than A. The two parameters a_1 and a_2 show the mutual influence degree on population sizes.

3.2.4. Adjustment of Population Sizes. In our mutualism population growth model, assume

$$r_1 = \frac{K_1 - x(t)}{s_1}, \qquad r_2 = \frac{K_2 - y(t)}{s_2}, \tag{30}$$

where s_1, s_2 are constants. Based on (26) and (27), at each generation t, the population sizes of A and B will be modified by Δx and Δy, respectively, which are given as follows:

$$\begin{aligned} \Delta x &= \frac{K_1 - x}{s_1}\left(1 - \frac{x}{K_2} + \frac{a_1 y}{K_1}\right)x, \\ \Delta y &= \frac{K_2 - y}{s_2}\left(1 + \frac{a_2 x}{K_2} - \frac{y}{K_1}\right)y. \end{aligned} \tag{31}$$

The newly added individuals are generated randomly, and the deleted individuals are the ones with the smallest cooperative degree values.

If the cooperative degree of a population is high, its population size will increase. Thus the growth of a population size mostly depends on its ability to coordinate. If the ability is strong, the size will become bigger and bigger till the maximum value. Otherwise, the size will become small and eventually extinct. In order to survive, each population will try its best to gain good genes and cooperate with others. In this way, the quality of solution is improved and the performance of algorithm is enhanced.

3.3. Framework of the Mutualism Strategy for Population Growth. A description of the mutualism strategy for population growth is given in Figure 2.

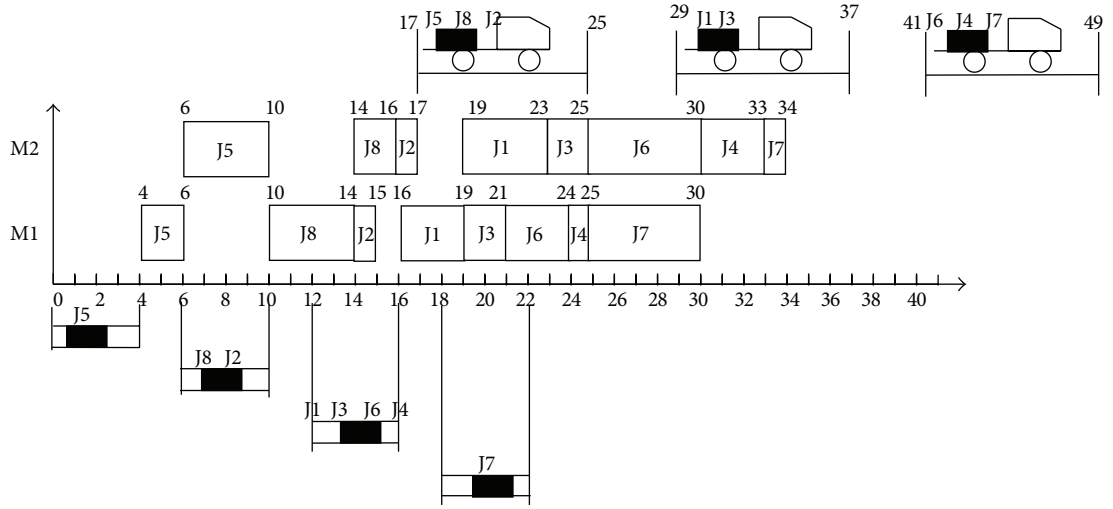

Figure 5: The decoding process and the Gantt chart.

4. The Mutualism Quantum Genetic Algorithm

The genetic algorithm (GA) proposed by Holland has achieved a great success for solving complex combinational problem. To improve GA, Han and Kim [31] recently introduced a quantum-inspired genetic algorithm, which attracted many researchers' attention. However, the experiment results are still not satisfactory for large-scale problems. In this section, a method named mutualism quantum genetic algorithm (MQGA) is introduced to solve the integrated supply chain problem including the pickup of materials, the flow shop scheduling, and the delivery of finished jobs. The symbiosis evolutionary strategy is embedded in the quantum genetic algorithm (QGA).

4.1. Hybrid Q-Bit Coding. The encoding in [29] is adopted in this paper, as it has obtained the success on the coordinated scheduling problem with batching machine scheduling and two-stage transportation. Their encoding divided a chromosome into two segments, the job sequence and the job-to-batch assignment, which can represent all possible solutions. Besides, the Q-bit representation is combined in our algorithm for its advantage of linear superposition of solutions and increase of diversity of genes. The details of the encoding are illustrated as follows.

Firstly, based on the concept and principles of quantum computing, the smallest unit of information stored in a two-state quantum computer is called a Q-bit. A Q-bit may be in the "1" state, "0" state, or any superposition of the two. A Q-bit can be represented as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers that specify the probability amplitudes of the corresponding states. $|\alpha|^2$ gives the probability that the Q-bit will be found in the "0" state and $|\beta|^2$ gives the probability that the Q-bit will be found in the "1" state. Normalization of the state to unity guarantees $|\alpha|^2 + |\beta|^2 = 1$. Therefore, in MQGA, a population $P_Q(t) = \{P_1^t, \ldots, P_N^t\}$ in Q-bit representation is initially randomly generated, and an individual is defined as a string of m Q-bits $\left[\begin{smallmatrix}\alpha_1\\ \beta_1\end{smallmatrix}\middle|\begin{smallmatrix}\alpha_2\\ \beta_2\end{smallmatrix}\middle|\cdots\middle|\begin{smallmatrix}\alpha_m\\ \beta_m\end{smallmatrix}\right]$, where $|\alpha_i|^2 + |\beta_i|^2 = 1, i = 1, 2, \ldots, m$.

Table 3: Parameters of the conveyor and truck.

Problem	Parameter					
	$C1$	Tr1	$R1$	$C2$	Tr2	$R2$
N(FT06)	6	10	3	10	20	10
N(FT10)	6	40	20	10	100	30
N(FT20)	6	20	10	10	60	30
N(ABZ3)	6	100	100	10	300	100
N(ABZ6)	6	100	100	10	300	100
N(ABZ7)	6	30	30	10	100	100
N(ABZ8)	6	30	30	10	100	100
N(ABZ9)	6	30	30	10	100	100
N(TA1)	6	30	30	10	100	100
N(TA2)	6	100	30	10	300	200
N(TA3)	6	100	30	10	300	200

Note: $C1$ and $C2$ are the capacities of conveyor and truck, respectively. Tr1 and Tr2 are the pickup and delivery times. $R1$ and $R2$ are the returning times of conveyor and truck.

Table 4: Physical space of jobs.

Jobs 1–20
2 6 1 0.3 3 1.3 2 3.3 1.2 4 2.8 3.2 0.7 0.1 0.9 2 6 1 0.3 3
Jobs 21–40
1.3 2 3.3 1.2 4 2.8 3.2 0.7 0.1 0.9 2 6 1 0.3 3 1.3 2 3.3 1.2 4
Jobs 41–60
2.8 3.2 0.7 0.1 0.9 3 4 3 6 1 2 6 1 0.3 3 1.3 2 3.3 1.2 4
Jobs 61–80
2.8 3.2 0.7 0.1 0.9 2 6 1 0.3 3 1.3 2 3.3 1.2 4 2.8 3.2 0.7 0.1 0.9
Jobs 81–100
2 6 1 0.3 3 1.3 2 3.3 1.2 4 2.8 3.2 0.7 0.1 0.9 3 4 3 6 1

Table 5: AOV results for the experiment on tuning the parameters.

Source	Sum of squares	Df	Mean square	F value
A: P_c	0.0083	4	0.0021	3.3334
B: P_m	0.0072	4	0.0018	4.3732
Interactions AB	0.0438	16	0.0029	7.3679

Secondly, the individual is converted into binary representation in the following way. For each Q-bit, let η_i be a random number generated in $[0, 1]$. If $|\alpha_i|^2 > \eta_i$, the Q-bit is converted to be 1; otherwise it is 0.

Thirdly, the Q-bit representation needs to be changed into two segments, the job sequence and the job-to-batch assignment, which include the comprehensive information of production and delivery. Thus we give a converting rule to form three parts, flow shop part (the code size is $(\lceil \log_2^n \rceil + 1) \times n$), material pickup part (the code size is $1 \times n$), and product delivery part (the code size is $1 \times n$), in which we set the last Q-bit of part 2 and part 3 as "1" to ensure the last job belongs to the last batch.

For example, given a 3-job, 3-machine problem, let 15 Q-bits represent a code, and every three Q-bits form one group. So a code contains five groups, which are then separated into 3 parts. The first three groups are defined as Part 1, the fourth group is Part 2, and the fifth group is Part 3.

For Part 1, suppose the binary representation is [0 1 1 | 1 0 1 | 1 1 0], which is already converted from Q-bit representation; then get [3 5 6]. This job permutation is regarded as a random key representation. If values of two elements in the representation are different, let the smaller one denote the job with smaller index; otherwise, let the first one denote the job with smaller index. So, the above random key representation is corresponding to job permutation [1 2 3].

For Part 2 and Part 3, suppose the binary representations are [1 0 1] and [0 0 1], which represent the job-to-batch assignment of the conveyor and the truck, respectively. The final code is the combination of Part 1, Part 2, and Part 3. An example is given in Figure 3.

The final code for another problem with 8 jobs and 2 machines is presented in (32) which is the individual structure of the MQGA, where the first line represents a permutation of 8 jobs and the second (third) line denotes the job-to-batch assignment in the conveyor (truck):

$$\begin{pmatrix} 5 & 8 & 2 & 1 & 3 & 6 & 4 & 7 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}. \tag{32}$$

The rule is that the jobs in the positions after each "1" to the next "1" belong to a new batch. Therefore, in the pickup transportation, job 5 is in the first batch, job 8 and job 2 belong to the second batch, job 1, job 3, job 6, and job 4 are in the third batch, and job 7 is in the last batch. The third line is the delivery batch of truck. Similarly, in the delivery transportation, job 5, job 8, and job 2 are in the same batch, job 1 and job 3 belong to the next batch, and job 6, job 4, and job 7 are in the last batch. Figure 4 illustrates the job sequence and the job-to-batch assignment.

Assume the processing times of the 8 jobs on the two machines are given in Table 2. The pickup time and returning time of the conveyor are 4 time units and 2 time units, respectively. The delivery time and returning time of truck are 8 time units and 4 time units, respectively. Figure 5 gives the visual assignment of jobs to be transported and produced.

4.2. Further Processing Mechanisms Solutions

4.2.1. The Speeding-Up Method of Inserting Neighborhood. When applying local search techniques to search better solutions, the neighborhood structure is very important, since it will directly affect the results. For the flow shop problem, the results in [32–34] show that the inserting operation is better than exchanging when searching in the local neighborhood. Therefore, this paper employs the inserting operation during the process of search in the neighborhood. The inserting neighborhood of a permutation is a set of solutions, that is, the permutations generated by deleting a job from the original position and inserting the job into a new position.

Denote by $\pi(i)$ the job in the ith position in the permutation π, and define $\omega_{\pi(i)}$ as the subpermutation after removing job $\pi(i)$. Assuming job $\pi(i)$ is inserted in the hth position of ω_i, let $\omega_{\pi(i)}(h)$ be the new permutation and σ the position where job $\pi(i)$ is inserted to achieve the minimal makespan; that is,

$$\omega_{\pi(i)}(\sigma) = \min_{1 \le h \le n} \omega_{\pi(i)}(h). \tag{33}$$

Take the two-machine flow shop problem as an example. Let $C_{\max}(\pi; M_1, M_2)$ be the makespan for a given permutation π, and let $T_{M_1}(\pi(j))$ and $T_{M_2}(\pi(j))$, respectively, be the earliest possible completion times of job $\pi(j)$ on machines M_1 and M_2 in the *forward channel*. In addition, define $T'_{M_1}(\pi(j))$, $T'_{M_2}(\pi(j))$, respectively, denoting the earliest possible completion times of job $\pi(j)$ on machines M_1 and M_2 in the *backward channel*. The iterative formulas are given as follows:

$$T_{M_1}(\pi(j)) = T_{M_1}(\pi(j-1)) + p_{1\pi(j)}, \quad j = 1, 2, \ldots, n,$$

$$T_{M_2}(\pi(j)) = \max\left[T_{M_1}(\pi(j)), T_{M_2}(\pi(j-1))\right] + p_{2\pi(j)}, \quad j = 1, 2, \ldots, n,$$

$$T_{M_1}(\pi(0)) = T_{M_2}(\pi(0)) = 0,$$

$$T'_{M_2}(\pi(j)) = T'_{M_2}(\pi(j+1)) + p_{2\pi(j)}, \quad j = n-1, n-2, \ldots, 1,$$

$$T'_{M_1}(\pi(j)) = \max\left[T'_{M_2}(\pi(j)), T'_{M_1}(\pi(j+1))\right] + p_{1\pi(j)}, \quad j = n-1, n-2, \ldots, 1,$$

Begin
$i \leftarrow 1$
While $(i \leq n)$ **do**
Begin
Obtain the subpermutation $\omega_{\pi(i)}$ by removing job $\pi(i)$ in the original permutation π.
For $k = 1, 2, \ldots, m-1$
Calculate $T_{M_k}(\omega_{\pi(i)}(j))$, $T_{M_{k+1}}(\omega_{\pi(i)}(j))$, $T'_{M_{k+1}}(\omega_{\pi(i)}(j))$ and $T'_{M_k}(\omega_{\pi(i)}(j))$.
End
For $h = 1, 2, \ldots, n$ and $h \neq i$
For $k = 1, 2, \ldots, m-1$
Calculate $C_{\max}(\omega_{\pi(i)}(h); M_k, M_{k+1})$.
End
Calculate $C_{\max}(\omega_{\pi(i)}(h)) = C_{\max}(\omega_{\pi(i)}(h); M_{m-1}, M_m)$.
End
Denote by σ the position where job $\pi(i)$ is inserted to get the minimal objective value.
$i = i + 1$;
End
End
Output the local optimal individual.

Procedure 1: The speeding-up method of inserting neighborhood.

$i = 1$
For batch B_j
If $\mathrm{Vol}(B_j) > \mathrm{TVol}$,
Find the job $J_{j,b}$ in B_j such that
$\mathrm{Vol}(J_{j,1} + J_{j,2} + \cdots + J_{j,b}) \leq \mathrm{TVol}$ and $\mathrm{Vol}(J_{j,1} + J_{j,2} + \cdots + J_{j,s} + J_{j,b+1}) > \mathrm{TVol}$.
Let $B_j = \{J_{j,1}, \ldots, J_{j,b}\}$ and $B_{j+1} = B_{j+1} \cup \{J_{j,b+1}, \ldots, J_{j,b_j}\}$. Update the index of jobs in B_{j+1}.
$i = i + 1$.
End

Procedure 2: The modified mechanism of a solution.

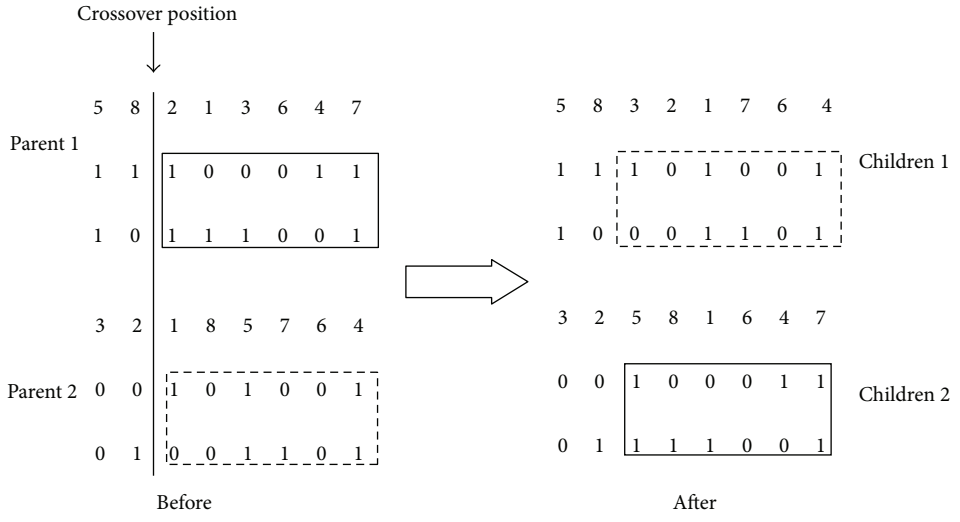

Figure 6: Crossover operation of MQGA.

Table 6: Detailed comparison results of MQGA, QGA, and MGA on N(TA1) problem.

RUN	500 × 10			1000 × 10			2000 × 20		
	MQGA	MGA	QGA	MQGA	MGA	QGA	MQGA	MGA	QGA
1	3166	3329	3199	3139	3326	3194	3152	3308	3183
2	3182	3336	3204	3166	3328	3193	3147	3327	3184
3	3165	3382	3229	3154	3351	3209	3154	3345	3206
4	3172	3277	3218	3162	3266	3210	3147	3210	3195
5	3164	3307	3220	3151	3323	3208	3148	3285	3187
6	3179	3330	3215	3165	3319	3170	3155	3324	3131
7	3164	3267	3222	3144	3253	3222	3128	3253	3220
8	3145	3371	3221	3138	3218	3191	3167	3313	3185
9	3154	3297	3226	3163	3293	3221	3140	3305	3226
10	3163	3298	3206	3140	3285	3145	3135	3276	3146

Begin
$t \leftarrow 0$
Step 1. Set parameters: crossover probability P_c, mutation probability P_m, iterative generation GN, two sub-populations sizes PS_1, PS_2.
Step 2. Initialize subpopulations in hybrid Q-bit representation.
While ($t \neq$ GN) **do**
Begin
Step 3. If (subpopulation is not empty)
Apply mutualism strategy, rescale the size of sub-populations according to population growth model.
End.
Step 4. Perform selection, crossover, mutation operations in each subpopulation;
Step 5. If (catastrophe condition is satisfied)
Perform catastrophe operation.
Else
Apply quantum rotation operation.
End
Step 6. Calculate objective value and fitness value of each individual in the current generation, and update the best solution if possible.
$t \leftarrow t + 1$
End
End
Output the global best result.

Algorithm 1: MQGA (mutualism quantum genetic algorithm).

$$T'_{M_1}(\pi(n)) = T'_{M_2}(\pi(n)) = 0,$$

$$C_{\max}(\pi) = T_{M_2}(\pi(n)) = T'_{M_1}(\pi(1)),$$

$$\begin{aligned} &C_{\max}\left(\omega_{\pi(i)}(h); M_1, M_2\right) \\ &= \max\Big[T_{M_1}(\omega(h-1)) + p_{1\pi(i)} + T'_{M_1}(\omega(h)), \\ &\qquad T_{M_2}(\omega(h-1)) + p_{2\pi(i)} + T'_{M_2}(\omega(h)), \\ &\qquad T_{M_1}(\omega(h-1)) + p_{1\pi(i)} + p_{2\pi(i)} + T'_{M_2}(\omega(h))\Big]. \end{aligned} \tag{34}$$

For the flow shop problem with more than two machines, the details of obtaining the local optimal solution are given in Procedure 1.

4.2.2. A Modified Mechanism of the Solution. Because the total physical space of all jobs in a batch may exceed the capacity of the vehicle, our encoding could not ensure a feasible solution, and a modified mechanism is needed to update the solution, which is an important step, not only in encoding process, but also in the steps of crossover and mutation, as an unfeasible solution may affect the search process in local neighborhood.

Let TVol be the capacity of the vehicle and Vol(B_j) the total space of all jobs in batch B_j. For simplicity, denote by $J_{j,1}, J_{j,2}, \ldots J_{j,b_j}$ the jobs in batch B_j, where b_j is the number of jobs in B_j. The modified mechanism is described in Procedure 2.

4.3. Other Operations. The operations introduced below are performed at each generation of the quantum genetic algorithm (QGA). The fitness of each individual can be

Table 7: Comparison results of MQGA, QGA, and MGA on N(TA1) problem.

Size	MQGA			MGA			QGA		
	BV	WV	AV	BV	WV	AV	BV	WV	AV
500 × 10	3155	3192	3165.4	3267	3382	3325.7	3199	3229	3216
1000 × 10	3138	3166	3150.2	3218	3351	3296.2	3145	3222	3196.3
2000 × 20	3128	3167	3147.3	3210	3345	3294.6	3131	3226	3186.3

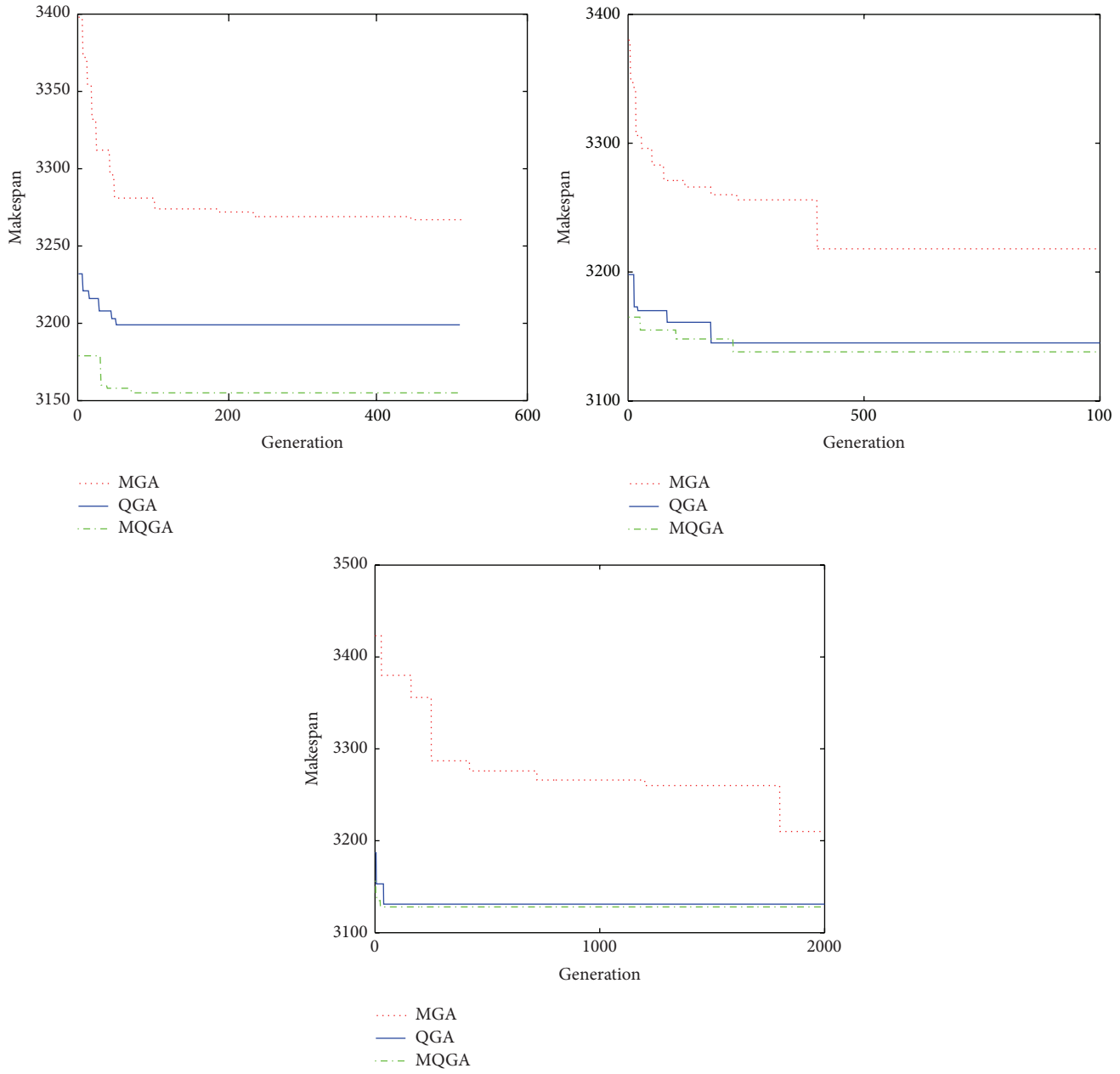

Figure 7: The convergence curves of MQGA, QGA, and MGA.

calculated by the mutualism strategy in Section 3.2. In order to ensure the individual with good genes can be chosen in the offspring, the roulette selection is applied here, which could maintain the diversity of genes. In the crossover operation, two decoded individuals are selected as parents to operate single-point crossover. See Figure 6 for the illustration of this operation, in which the corresponding genes of Q-bit parents are swapped to produce two new offspring.

In the mutation operation, a NOT Gate is used as the mutation operator. Firstly, select individuals with mutation

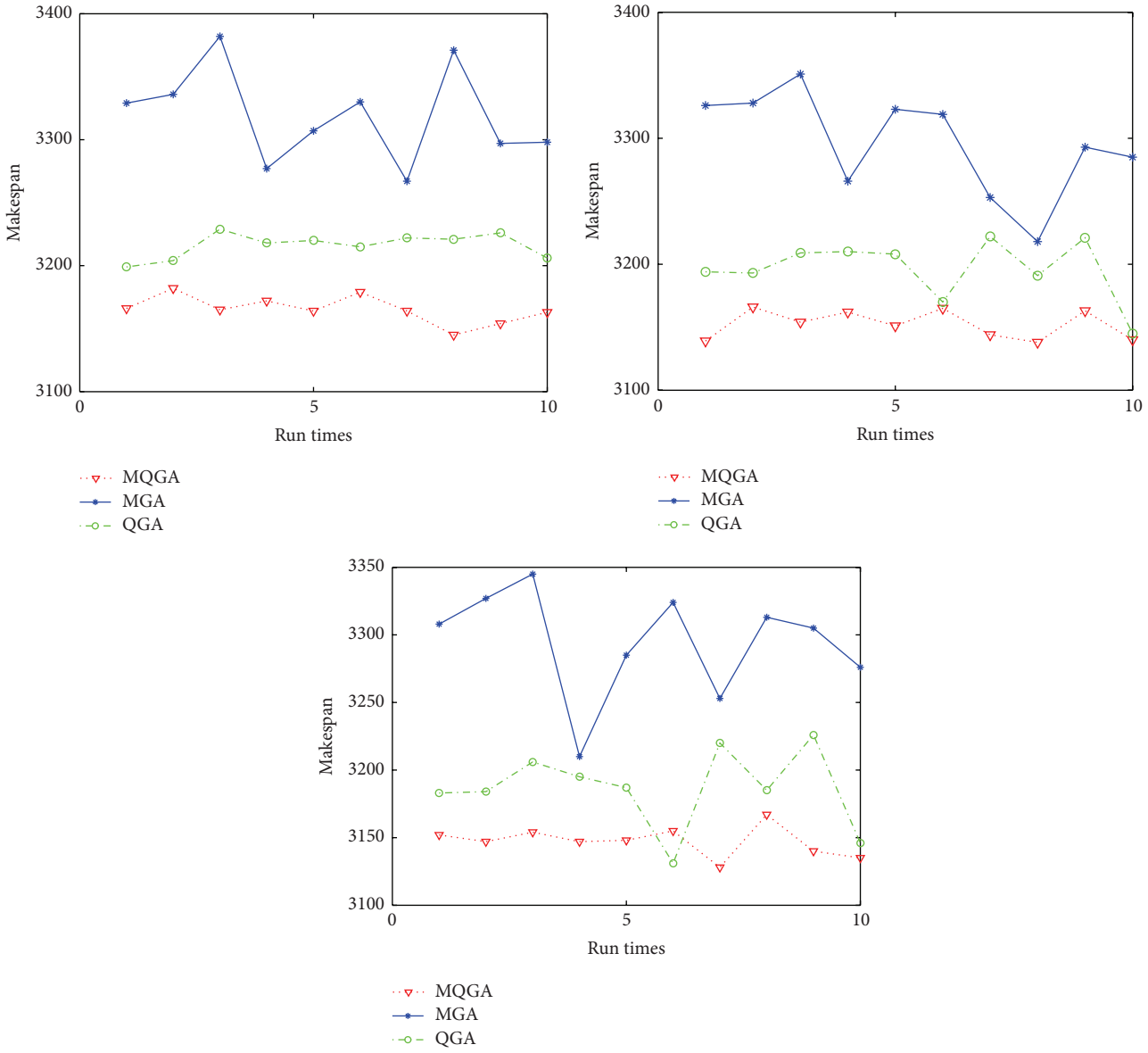

Figure 8: Distributions of solutions of MQGA, QGA, and MGA.

probability P_m; then randomly generate a mutation position. Quantum rotation gate is used to maintain diversity of genes, which could change the probability amplitude of quantum states. See more details in our previous work [35]. Besides, a catastrophe operator is used to avoid premature convergence. If the solution does not change in two consecutive generations, it could be regarded to be trapped in local optimal solution, and the local best solution will be reserved while others will be replaced by solutions generated randomly.

4.4. The Main Procedure of MQGA. In this subsection, the details of the MQGA are introduced in Algorithm 1.

5. Experiment

In this paper, all algorithms are programmed with MATLAB language, and all the computations are conducted on a Pentium PC 1.66 GHZ with 512 MB memory. In order to evaluate the performance of MQGA, an extensive set of instances with different characteristics are generated based on the flow shop benchmark problems, including FT benchmark and ABZ benchmark. In order to test the relative large-scale problem, three test instances are designed, where there are 50 jobs and 5 machines, 50 jobs and 10 machines, and 100 jobs and 5 machines, seen in the Appendix. Table 3 shows the transporting times of conveyor and truck in different problems. Table 4 shows the physical spaces of the 100 jobs. These test instances are named N(FT06), N(FT10), N(FT20),

Table 8: Computational results of MQGA and QGA, MGA.

Problem	Approach	$\overline{\text{Ob}}$	Improved%	$\overline{\text{AG}}$	$\overline{\text{AT}}$
N(FT06) (6 ∗ 6)	MGA	103.3	—	73.3	13
	QGA	103.3	0	33.6	30
	MQGA	***103***	0.48%	9.8	113
N(FT10) (10 ∗ 10)	MGA	1282.2	—	300.8	29
	QGA	1280.7	0.12%	97.3	98
	MQGA	***1273.6***	0.31%	76.2	163
N(FT20) (20 ∗ 5)	MGA	1330	—	26.7	23
	QGA	1328.6	0.09%	18.3	36
	MQGA	***1323.1***	0.32%	36.7	133
N(ABZ3) (10 ∗ 10)	MGA	2088.6	—	430.8	29
	QGA	2079.4	0.44%	146.6	101
	MQGA	***2071.3***	0.83%	83.4	164
N(ABZ6) (10 ∗ 10)	MGA	1883.2	—	363.6	29
	QGA	1874.4	0.47%	111.3	101
	MQGA	***1861.4***	1.16%	123.3	164
N(ABZ7) (15 ∗ 20)	MGA	1276.2	—	140.3	68
	QGA	1233.8	3.32%	104.3	172
	MQGA	***1228.2***	3.76%	86.8	310
N(ABZ8) (15 ∗ 20)	MGA	1290.3	—	114.4	68
	QGA	1277.2	1.03%	17.7	172
	MQGA	***1268.4***	1.71%	19.8	310
N(ABZ9) (15 ∗ 20)	MGA	1273.3	—	143.9	68
	QGA	1238.3	1.18%	29.7	171
	MQGA	***1241.3***	2.3%	11.3	310
N(TA1) (50 ∗ 5)	MGA	3286.2	—	71.7	84
	QGA	3173.3	3.37%	17	426
	MQGA	***3134.2***	4.02%	21.1	778
N(TA2) (50 ∗ 10)	MGA	3346.6	—	91.2	130
	QGA	4917	11.33%	9.1	480
	MQGA	***4803***	13.4%	19.3	814
N(TA3) (100 ∗ 5)	MGA	10706.3	—	63.3	171
	QGA	9289.33	13.23%	16.2	632
	MQGA	***9126.4***	14.76%	17.8	963

Note: $\overline{\text{AG}}$ represents the converged generation and $\overline{\text{AT}}$ represents the average computation time.

N(ABZ3), N(ABZ6), N(ABZ7), N(ABZ8), N(ABZ9), N(TA1), N(TA2), and N(TA3).

5.1. Parameter Settings. Two parameters including crossover rate P_c and mutation rate P_m are discussed in this section. Assume the two rates are set with the following levels:

$$\begin{aligned} P_m &: 0.2, 0.4, 0.6, 0.8, 1; \\ P_c &: 0.2, 0.4, 0.6, 0.8, 1. \end{aligned} \tag{35}$$

In this way, there are a total of 25 combinations. For each combination, MGA [13] is used to test the 11 testing instances in Table 3. After the experiment, the data is transformed to be the relative percentage deviation (RPD), which is calculated as follows:

$$\text{RPD} = \frac{\text{WT}_A - \text{WT}_{\text{ref}}}{\text{WT}_{\text{ref}}} \times 100\%, \tag{36}$$

where WT_A is the corresponding objective value of the solution achieved by proposed algorithm and WT_{ref} is the minimum objective value under multiple simulation of the same test problem. In addition, for a combination of P_c and P_m values, the analysis of variance (AOV) model is employed to eliminate the different RPD values caused by the block (each test problem is regarded as a block). The main experiment results of the AOV are shown in Table 5.

According to the theory of statistical analysis, the greater the F value is, the more significantly the corresponding factors affect. Table 5 shows the interaction between the two parameters has a significant effect on the performance of the algorithm. Based on the experiment, we find the best value of P_c is 0.8 and the best value of P_m is 0.2, since they could find the most times of optimal objective value.

5.2. Simulation Work

5.2.1. The Detailed Simulation Result of N(TA1) Problem. In order to validate the performance of MQGA, MGA [13] and QGA [36] are compared. Firstly problem N(TA1) is chosen as the representative instance to analyze the performance of algorithms. Three combinations of iterative generations and population sizes are 500 × 10 (i.e., the maximum number of iterations is 500, population size is 10, and subpopulation size is 5), 1000 × 10, and 2000 × 20. For each combination, the results are presented in Table 6, and more results are given in Table 7, where "BV" and "WV," respectively, denote the best and worst objective values in the 10 run times. "AV" is the average objective value.

Known from the results in Table 6, for each calculation, MQGA always obtains a better solution than that obtained by MGA and QGA. When the iteration is 500, the range of objective values of MQGA is (3155, 3192) and (3267, 3382) and (3199, 3229) for MGA and QGA. When the iteration is 2000, the ranges of objective values are, respectively, (3128, 3167), (3210, 3345), and (3131, 3226). Therefore, the search ability of MQGA increases faster than that of MGA and QGA as the iteration increases, and MGA and QGA are more likely to be trapped in local optimal solutions.

The convergence curves of MQGA, MGA, and QGA obtained for each combination are depicted in Figure 7, which indicates the convergence speed of MQGA is faster than that of MGA and QGA, especially for the combination with larger iteration.

The solutions distributions of MQGA, QGA, and MGA are given in Figure 8, which shows that there is no overlap in the solution space of each algorithm. Moreover, the change in the solutions achieved by MQGA is smaller than that of MGA and QGA, which implies that the quality of solution obtained by MQGA is better than that of MGA and QGA.

5.2.2. The Comparison Results of MQGA with MGA and QGA. The maximum number of iterations is 1000, and the size of

Table 9: (a) N(TA1): 50 jobs ∗ 5 machines. (b) N(TA2): 50 jobs ∗ 10 machines. (c) TA3: 100 jobs ∗ 5 machines.

(a)

Machines	Processing times
M_1	73 87 13 11 41 43 93 69 80 13 24 72 38 81 83 88 26 6 89 67 70 30 89 30 68 21 78 46 99 10 17 23 83 47 86 18 67 46 4 14 4 20 88 30 84 38 93 76 30 30
M_2	26 37 23 93 49 12 39 17 46 20 32 44 92 73 93 33 10 43 2 62 62 82 29 29 94 20 42 80 94 33 8 41 63 4 71 30 14 32 30 30 27 98 39 84 63 12 38 43 49 13
M_3	48 4 92 92 72 43 3 98 93 17 79 11 16 89 81 92 43 61 39 28 94 87 23 1 33 91 67 91 4 60 38 23 90 93 13 63 23 34 47 98 91 11 46 30 77 3 14 47 80 43
M_4	26 67 4 14 93 34 21 20 6 18 73 23 16 77 28 24 13 77 36 16 32 46 21 81 28 70 89 34 96 62 46 60 19 97 13 7 44 7 73 13 66 70 97 33 97 64 73 28 4 87
M_5	77 94 9 37 29 79 33 73 63 86 23 39 76 24 38 3 91 29 22 27 39 31 46 18 93 38 83 38 97 10 79 93 2 87 17 18 10 30 8 26 14 21 13 10 83 46 42 18 36 2

(b)

Machines	Processing times
M_1	46 32 79 43 97 10 44 24 83 73 66 49 93 61 19 47 84 13 11 19 98 2 83 44 7 73 19 69 12 73 83 23 33 16 88 8 26 42 38 63 7 2 44 38 24 76 83 61 32 90
M_2	61 87 31 23 73 93 28 90 94 39 64 2 16 33 33 40 81 26 83 4 4 10 63 96 33 71 66 94 7 13 11 99 37 30 36 69 22 36 67 63 96 74 4 42 40 30 93 36 23 87
M_3	31 38 83 33 71 38 36 64 43 48 69 96 33 82 33 64 11 61 36 33 87 88 10 32 38 23 24 90 7 11 49 2 76 17 32 39 9 83 69 67 28 88 23 91 71 3 26 41 96
M_4	31 24 21 37 69 31 30 31 21 19 63 91 11 6 31 63 36 39 37 47 36 63 39 4 10 12 62 43 49 34 87 29 2 18 73 39 77 69 13 78 68 37 22 41 92 67 24 87 91 31
M_5	37 16 42 47 94 14 94 34 72 36 88 31 41 71 94 99 11 97 44 77 69 91 38 23 87 7 66 34 86 49 3 48 44 93 37 82 31 39 78 33 36 3 38 10 98 6 44 62 24 94
M_6	79 93 68 73 37 44 34 39 76 62 74 28 78 43 98 83 91 27 6 82 60 44 43 76 99 66 11 33 32 8 40 62 23 24 30 1 73 27 16 91 33 11 99 2 60 90 36 62 13 3
M_7	83 87 38 38 86 67 23 19 97 78 66 67 7 23 67 8 77 71 83 29 49 3 94 76 93 48 4 37 82 37 61 6 97 3 27 93 46 92 46 32 8 11 7 34 72 37 83 22 87 63
M_8	22 29 99 23 98 33 80 82 33 68 47 74 26 61 93 33 11 42 72 14 8 98 90 36 73 69 26 24 33 98 86 30 92 94 66 47 3 41 41 47 89 28 39 80 47 37 74 38 39 3
M_9	27 92 73 94 18 41 37 38 36 20 2 39 91 81 33 14 88 22 36 63 79 23 66 3 13 31 2 81 12 40 39 32 16 87 78 41 43 94 1 93 22 93 62 33 30 34 27 30 34 77
M_{10}	24 47 39 66 41 46 24 23 68 30 93 22 64 81 94 97 34 82 11 91 23 32 26 22 12 23 34 87 39 2 38 84 62 10 11 93 37 81 10 40 62 49 90 34 11 81 31 21 39 27

(c)

Machines	Processing times						
	84.6317	68.0327	97.8144	20.8223	22.0642	31.3817	40.4331
	32.3333	4.4926	9.2277	31.4611	18.0836	83.0344	63.7339
	83.9330	27.7676	37.3838	30.3118	33.6348	32.3347	22.3726
	17.1233	22.6226	6.0628	74.2341	33.4834	39.4332	46.8303
	83.6232	12.0133	9.4702	27.0917	37.0777	36.4394	61.8333
	70.0938	30.8900	30.2302	43.1238	41.1160	31.3372	39.7248
	64.9842	21.4270	93.8391	42.2748	27.2131	37.3729	39.4439
M_1	18.3346	31.9796	79.8378	11.1263	39.2043	28.3831	82.1932
	93.3273	27.8467	63.8107	48.3132	67.9713	30.4670	47.9016
	41.2603	34.3209	42.7133	4.6217	63.8433	23.6188	76.6038
	77.1361	60.7312	7.4211	13.2976	66.3300	4.0213	31.2134
	30.9947	33.1613	77.0320	42.0403	7.9610	74.3028	99.9878
	93.4269	19.8363	30.6913	83.1429	10.3026	89.1798	93.6973
	96.6370	67.6072	1.3964	30.3368	62.9276	29.0398	60.9821
	66.8269	44.3931					

(c) Continued.

Machines	Processing times						
	3.8077	68.7184	67.7863	31.3860	4.0038	92.7664	82.9002
	33.9639	32.9803	63.0667	88.7886	72.8946	80.2037	60.6660
	43.8637	26.0218	93.0429	64.2378	93.1670	44.7731	47.8723
	29.7237	24.9201	19.3493	17.8681	33.8913	8.7417	28.9312
	63.8303	89.6336	11.3332	40.3292	74.7622	86.1109	90.3164
	64.3916	74.1080	8.2019	90.6283	94.3113	92.2433	42.3333
	32.9836	37.3679	87.3114	71.4010	93.0243	39.3819	42.0138
M_2	36.9339	31.2930	46.0303	2.3923	43.4493	22.6322	18.8634
	3.2671	88.2232	17.1819	19.3328	31.4467	6.9719	33.8983
	79.9701	33.3020	79.0346	73.3813	21.1483	68.0938	22.2373
	31.0398	37.3793	31.1383	38.7069	36.4293	33.9889	2.4179
	37.6434	21.2294	23.1638	39.8660	88.1233	18.2311	34.4082
	39.3800	34.1221	42.7137	86.3060	97.8039	23.4139	40.0631
	34.7736	73.7374	38.2443	73.9441	79.1348	31.9843	93.7063
	11.4333	43.9139					
	37.0079	47.6363	86.7381	36.0169	23.9910	9.3497	91.3933
	19.0191	92.3208	78.0443	77.9396	74.4131	9.1036	3.1612
	69.8134	32.7903	10.2037	63.0130	43.4309	62.3464	49.9032
	76.9814	23.6630	68.1342	24.8938	74.0296	77.4290	73.3663
	94.1423	91.3789	22.3412	34.2217	18.2232	99.8933	63.0020
	33.6438	89.0716	22.0123	10.2334	97.3328	32.2667	88.2888
	94.8882	22.4733	61.8833	80.0331	30.4777	2.3938	86.1700
M_3	3.4326	24.9428	33.2202	76.1338	43.0818	91.6040	66.1194
	79.3679	48.9789	12.8067	94.3034	86.6166	13.3706	93.9793
	6.0822	37.3802	18.7386	21.6410	11.0119	83.0266	49.0443
	83.1172	27.6230	33.3916	92.4096	33.0112	39.0013	30.3430
	3.9994	74.3970	32.0863	77.0834	33.4703	29.9367	87.0861
	36.1939	38.6870	2.3033	41.1629	31.4037	18.4714	60.3132
	61.0278	23.8668	31.3109	16.4060	93.9227	30.7267	98.7219
	31.2407	13.8634					
	79.3039	32.9897	47.6910	43.9032	36.0006	82.3696	66.3923
	13.4142	93.8438	30.0931	20.0777	14.9284	8.6927	43.8123
	37.7344	9.7682	18.3971	7.6113	76.3946	63.8904	94.7271
	47.9722	47.8333	93.3017	48.3228	83.3002	71.6867	39.6977
	37.0047	94.8397	44.0043	28.4989	73.2636	70.9429	31.1982
	31.7673	67.9373	43.9669	90.9212	27.1023	33.3360	76.3692
	37.1149	13.7774	99.9414	72.7029	69.0932	18.7311	32.1909
M_4	99.8103	93.3610	84.7230	97.3806	10.7734	78.1100	44.3464
	10.0022	44.4773	16.0306	46.4031	20.4337	93.7849	1.8234
	81.3674	43.6923	38.7263	37.3334	96.4237	18.6704	11.2710
	81.2803	27.3601	39.3646	22.3763	92.6382	43.3897	38.6866
	36.8882	99.7779	12.2373	16.3636	30.3478	89.7477	33.2003
	4.2014	4.7337	74.4997	17.4811	60.4477	38.0942	63.8836
	93.1060	93.7823	68.7374	78.9328	16.2863	31.7426	7.9919
	9.7908	63.7969					
	46.0182	70.8827	42.3334	23.9494	78.7140	11.2288	87.7309
	1.1682	73.2947	74.3247	33.3868	80.9084	30.3979	29.9039
	10.0888	83.4943	8.1367	31.3683	93.0997	38.3136	64.6483
	70.0836	73.4314	86.4972	34.8031	18.2930	30.8623	20.3774
	23.2021	20.4869	93.3820	74.4314	82.4732	34.6946	94.8308
	13.8809	36.4386	43.4073	32.8384	68.6438	99.1667	6.2637
	33.4487	29.1977	42.6223	39.6733	19.3706	13.3877	64.6711
M_5	63.4906	63.7133	41.8206	93.1286	73.4143	87.1880	31.7227
	43.9343	63.0724	39.6809	33.3300	99.3787	21.3336	92.2013
	76.3623	23.4309	24.2814	81.0009	34.4622	46.7340	8.9193
	23.3668	99.9983	11.6722	92.1918	37.6619	93.3813	36.9370
	43.4437	40.4863	90.1080	90.7238	33.8413	39.7471	47.7907
	18.2683	44.6693	10.9124	22.2803	23.8896	72.8488	74.6871
	63.8373	44.3997	79.0272	34.2089	18.3843	13.0348	14.6130
	47.6381	9.2464					

population is 100 (in MQGA, there are two subpopulations, and the size of each subpopulation is 50). To be fair, each algorithm runs 10 times for each instance. The experiment results are shown in Table 8, in which each item is an average value. The "Improved%" column represents the percentage difference between the average objective values obtained by the current algorithm with MGA and is calculated by the following formula:

$$\text{Improved\%} = \frac{\left(\overline{\text{Ob}} - \overline{\text{MGA}}\right)}{\overline{\text{MGA}}} \times 100\%, \tag{37}$$

where $\overline{\text{Ob}}$ represents the objective value achieved by the current algorithm and $\overline{\text{MGA}}$ represents the objective value achieved by MGA. Boldface and italic indicate the best results for each problem.

In the performance of solution, from Table 8, MQGA is superior to MGA and QGA for all the test instances, and QGA always outperforms MGA. The improvement becomes more obvious as the scale of the problem increases. For example, for medium-scale problem N(FT20), the optimal ability of MQGA improves 0.32%. For the large-scale problem N(TA3), the MQGA improves 14.76%. The reason is that when the number of jobs increases the transportation of vehicle will become the bottleneck for MGA and QGA. Therefore, MGA and QGA are not applicable to the relative large-scale problem instances. In addition, in the performance of convergence, MQGA can obtain a better solution than MGA and QGA with less iterative generations. For the relative large-scale problems like N(TA1), N(TA2), and N(TA3), MGA and QGA converge faster to the local optimal solution. However, MQGA could jump out of the local optimum with more iterative generations. Finally, in the performance of time, MQGA needs more computation time than MGA and QGA but obtains an improved solution.

6. Conclusions and Future Research

This paper studies an integrated scheduling with the materials pickup, flow shop scheduling, and the finished products delivery. The objective is to find a coordinated schedule to minimize the arrival time of the last completed product to the customer. In order to solve the problem, a biologically inspired quantum genetic algorithm is proposed with a new mutualism strategy. The experiment results demonstrate that MQGA can find a satisfactory solution with an acceptable amount of computation time.

Future research could address problems with multiple customers or multiple transport vehicles or different shop environments, including flexible scheduling and job-shop. Problems with other performance measures, including minimum mean tardiness, and multimeasures should also be studied.

Appendix

See Table 9.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant nos. 11201282 and 61304209), Humanity and Social Science Youth Foundation of Ministry of Education of China (Grant no. 10YJCZH032), Innovation Program of Shanghai Municipal Education Commission (14YZ127), and the Fund of Scheme for Training Young Teachers in Colleges and Universities in Shanghai (ZZCD12006).

References

[1] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Prentice-Hall, Upper Saddle River, NJ, USA, 2nd edition, 2002.

[2] P. L. Maggu and G. Das, "On 2*n* sequencing problem with transportation times of jobs," *Pure and Applied Mathematica Sciences*, vol. 12, pp. 1–6, 1980.

[3] P. L. Maggu, G. Das, and R. Kumar, "On equivalent-job for job-block in $2 \times n$ sequencing problem with transportation-times," *Journal of the Operations Research Society of Japan*, vol. 24, no. 2, pp. 136–146, 1981.

[4] P. L. Maggu, M. L. Singhal, N. Mohammad, and S. K. Yadav, "On *n*-job, 2-machine flow-shop scheduling problem with arbitrary time lags and transportation times of jobs," *Journal of the OR Society of Japan*, vol. 23, pp. 219–227, 1982.

[5] M. A. Langston, "Interstage transportation planning in the deterministic flow-shop environment," *Operations Research*, vol. 35, no. 4, pp. 556–564, 1987.

[6] M. Haouari and T. Ladhari, "Minimising maximum lateness in a two-machine flowshop," *Journal of the Operational Research Society*, vol. 51, no. 9, pp. 1100–1106, 2000.

[7] J. Hurink and S. Knust, "Makespan minimization for flow-shop problems with transportation times and a single robot," *Discrete Applied Mathematics*, vol. 112, no. 1–3, pp. 199–216, 2001.

[8] J. Hurink and S. Knust, "A tabu search algorithm for scheduling a single robot in a job-shop environment," *Discrete Applied Mathematics*, vol. 119, no. 1-2, pp. 181–203, 2002.

[9] J. Hurink and S. Knust, "Tabu search algorithms for job-shop problems with a single transport robot," *European Journal of Operational Research*, vol. 162, no. 1, pp. 99–111, 2005.

[10] C.-Y. Lee and V. A. Strusevich, "Two-machine shop scheduling with an uncapacitated interstage transporter," *IIE Transactions*, vol. 37, no. 8, pp. 725–736, 2005.

[11] B. Naderi, R. Tavakkoli-Moghaddam, and M. Khalili, "Electromagnetism-like mechanism and simulated annealing algorithms for flowshop scheduling problems minimizing the total weighted tardiness and makespan," *Knowledge-Based Systems*, vol. 23, no. 2, pp. 77–85, 2010.

[12] M. Khalili and R. Tavakkoli-Moghaddam, "A multi-objective electromagnetism algorithm for a bi-objective flowshop scheduling problem," *Journal of Manufacturing Systems*, vol. 31, no. 2, pp. 232–239, 2012.

[13] J. Behnamian, S. M. T. Fatemi Ghomi, F. Jolai, and O. Amirtaheri, "Minimizing makespan on a three-machine flowshop

batch scheduling problem with transportation using genetic algorithm," *Applied Soft Computing Journal*, vol. 12, no. 2, pp. 768–777, 2012.

[14] L. Tang, J. Guan, and G. Hu, "Steelmaking and refining coordinated scheduling problem with waiting time and transportation consideration," *Computers & Industrial Engineering*, vol. 58, no. 2, pp. 239–248, 2010.

[15] A. Elmi and S. Topaloglu, "A scheduling problem in blocking hybrid flow shop robotic cells with multiple robots," *Computers and Operations Research*, vol. 40, no. 10, pp. 2543–2555, 2013.

[16] C.-Y. Lee and Z.-L. Chen, "Machine scheduling with transportation considerations," *Journal of Scheduling*, vol. 4, no. 1, pp. 3–24, 2001.

[17] Y.-C. Chang and C.-Y. Lee, "Machine scheduling with job delivery coordination," *European Journal of Operational Research*, vol. 158, no. 2, pp. 470–487, 2004.

[18] C.-L. Li, G. Vairaktarakis, and C.-Y. Lee, "Machine scheduling with deliveries to multiple customer locations," *European Journal of Operational Research*, vol. 164, no. 1, pp. 39–51, 2005.

[19] Z.-L. Chen and G. L. Vairaktarakis, "Integrated scheduling of production and distribution operations," *Management Science*, vol. 51, no. 4, pp. 614–628, 2005.

[20] H. N. Geismar, G. Laporte, L. Lei, and C. Sriskandarajah, "The integrated production and transportation scheduling problem for a product with a short lifespan," *INFORMS Journal on Computing*, vol. 20, no. 1, pp. 21–33, 2008.

[21] A. Soukhal, A. Oulamara, and P. Martineau, "Complexity of flow shop scheduling problems with transportation constraints," *European Journal of Operational Research*, vol. 161, no. 1, pp. 32–41, 2005.

[22] M. M. Mazdeh and M. Rostami, "A branch-and-bound algorithm for two-machine flow-shop scheduling problems with batch delivery costs," *International Journal of Systems Science: Operations and Logistics*, vol. 1, no. 2, pp. 94–104, 2014.

[23] P. J. Kalczynski and J. Kamburowski, "An empirical analysis of heuristics for solving the two-machine flow shop problem with job release times," *Computers & Operations Research*, vol. 39, no. 11, pp. 2659–2665, 2012.

[24] M. Rasti-Barzoki, S. R. Hejazi, and M. Mahdavi Mazdeh, "A branch and bound algorithm to minimize the total weighted number of tardy jobs and delivery costs," *Applied Mathematical Modelling*, vol. 37, no. 7, pp. 4924–4937, 2013.

[25] C. Low, C.-M. Chang, R.-K. Li, and C.-L. Huang, "Coordination of production scheduling and delivery problems with heterogeneous fleet," *International Journal of Production Economics*, vol. 153, pp. 139–148, 2014.

[26] N. G. Hall and C. N. Potts, "Supply chain scheduling: batching and delivery," *Operations Research*, vol. 51, no. 4, pp. 566–584, 2003.

[27] C.-L. Li and J. Ou, "Machine scheduling with pickup and delivery," *Naval Research Logistics*, vol. 52, no. 7, pp. 617–630, 2005.

[28] X. Wang and T. C. Cheng, "Production scheduling with supply and delivery considerations to minimize the makespan," *European Journal of Operational Research*, vol. 194, no. 3, pp. 743–752, 2009.

[29] C.-H. Liu, "Using genetic algorithms for the coordinated scheduling problem of a batching machine and two-stage transportation," *Applied Mathematics and Computation*, vol. 217, no. 24, pp. 10095–10104, 2011.

[30] L. Tang and H. Gong, "A hybrid two-stage transportation and batch scheduling problem," *Applied Mathematical Modelling*, vol. 32, no. 12, pp. 2467–2479, 2008.

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

[32] G. I. Zobolas, C. D. Tarantilis, and G. Ioannou, "Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm," *Computers & Operations Research*, vol. 36, no. 4, pp. 1249–1267, 2009.

[33] J. Grabowski and J. Pempera, "Some local search algorithms for no-wait flow-shop problem with makespan criterion," *Computers and Operations Research*, vol. 32, no. 8, pp. 2197–2212, 2005.

[34] G. L. Deng and X. S. Gu, "A hybrid discrete differential evolution algorithm for the no-idle permutation flow shop scheduling problem with makespan griterion," *Computers and Operations Research*, vol. 39, pp. 2132–2160, 2012.

[35] J. Gu, X. Gu, and M. Gu, "A novel parallel quantum genetic algorithm for stochastic job shop scheduling," *Journal of Mathematical Analysis and Applications*, vol. 355, no. 1, pp. 63–81, 2009.

[36] L. Wang, H. Wu, F. Tang, and A. Z. Zheng, "A hybrid quantum-inspired genetic algorithm for flow shop scheduling," in *Advances in Intelligent Computing*, vol. 3654 of *Lecture Notes in Computer Science*, pp. 636–644, Springer, 2005.