Knowledge-Based Systems 90 (2015) 185–198

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Intelligent algorithms for a new joint replenishment and synthetical delivery problem in a warehouse centralized supply chain

Ligang Cui [a,*], Lin Wang [b], Jie Deng [c], Jinlong Zhang [b]

[a] School of Economics and Management, Chongqing Jiaotong University, No.66, Xuefu Ave., Chongqing 400074, China
[b] School of Management, Huazhong University of Science and Technology, No.1037, Luoyu Road, Wuhan 430074, China
[c] Intellectual Property Institution of Chongqing, Chongqing University of Technology, No.69 Hongguang Rd. Chongqing 400054, China

ARTICLE INFO

Article history:
Received 27 March 2015
Revised 14 August 2015
Accepted 20 September 2015
Available online 8 October 2015

Keywords:
Supply chain management
Joint replenishment
Delivery strategy
Beta-heuristic

ABSTRACT

In this paper, a novel joint replenishment and synthetical delivery (JRD) model is proposed to improve the coordination of replenishment and delivery processes. Traditional, in a warehouse centralized supply chain, orders from online customers are usually delivered independently after multiple items have been jointly replenished. To decrease the outbound delivery cost, a new delivery strategy considering synthetical dispatched orders, orders and customers matching, and customer visiting sequence is proposed. Three new beta-heuristic algorithms, namely, quantum evolution algorithm (QEA), differential evolution algorithm (DE) and quantum differential evolution algorithm (QDE), are utilized to solve the proposed JRD. The most important parts of each algorithm (initialization, reproduction and mutation, and selection) are redesigned according to the structure of the decision variables. Numerical experiments are conducted to find the best parameter settings and potential searching abilities of each algorithm. Finally, experimental results show the superiorities of the proposed DE and QDE in terms of searching speed, accuracy, and robustness.

1. Introduction

Joint replenishment is one of the most favored considerations in the replenishment process under multi-product environments. Especially, it is very critical to multinational corporations that aim to establish stable worldwide supply/procurement systems. In recent years, joint replenishment strategy and the delivery strategy are considered simultaneously [1,2], but the delivery process is assumed with a consolidated one-to-one service mode, which loses some generality comparing that in real delivery practices. Therefore, it leaves us spaces to make improvements in delivery process of JRDs. To begin with, before unfolding the model constructing process, the development of JRD researches are reviewed in the following contents.

Initially, JRPs are the problems that pertain to coordinating the replenishment of a group of items that may be jointly ordered from a single supplier [3,4]. The main feature of these classic JRPs is that they assume constant demand [5]. Other JRPs are mainly extensions of the classic JRPs [6], such as JRP under dynamic demand [5], and JRP under stochastic demand [7], which shows that JRPs consistently attract the attention of numerous researchers, although JRP has been discussed for several years.

The savings from group replenishment are so significant that practitioners are motivated to adjust their replenishment strategies regularly to save cost. For example, for a higher ratio of major to minor setup cost, the savings in the replenishment scheme of Chan et al. [4] range from 36% to 70% in different routes. According to the estimation of Porras and Dekker [8], for a 20-item problem with the major setup cost in the range of 25–75 units, approximately 5.4–13.2% savings can be achieved with respect to independent ordering compared with the cost obtained through the economic order quantity (EOQ) approach. It has been reported that 2.3% extra savings are obtained using joint replenishment strategy supported by a novel differential evolution algorithm (DE) comparing with the cost obtained through the EOQ strategy [9].

Typically speaking, of all the new JRP models, two extensions of JRP should be noted if the warehouse is assumed as the center of a supply chain. One extension is in the supply end, and the other is in the selling end. For both extensions, delivery considerations are taken into account, such considerations are called the joint replenishment-delivery problem, given that inventory and scheduling decisions are integrated in JRDs [10–12]. For the former type of JRD, the original JRPs with one supplier are usually extended to multi-supplier problems. After that, the delivery process is included after the orders are placed. Hsu [13] investigated a joint replenishment decision for a

* Corresponding author. Tel.: +86 17782262750.
E-mail addresses: cuiligangdj@gmail.com, cligang@126.com (L. Cui), wanglin982@gmail.com (L. Wang), mabelduo@hotmail.com (J. Deng), prozhangjl@gmail.com, jlzhang@hust.edu.cn (J. Zhang).

http://dx.doi.org/10.1016/j.knosys.2015.09.019
0950-7051/

central factory and its satellite factories. In his model, raw materials are grouped together from different satellite factories to form a large shipment and then delivered to the central factory. For the latter type of JRD, Chan et al. [14] focused on a new multi-buyer situation of a firm who owns multi-branches. The firm prepares the ordered items and schedules the deliveries to his branches. A joint economic procurement-production-delivery policy was proposed to find the production sequences of multi-items, the common production cycle length, and the delivery frequencies [15]. Another JRD provided by Cha et al. [1] linked joint replenishment and delivery for a one-warehouse, *n*-retailer system. Both replenishment frequency and outbound frequency are decision variables. Based on the work of Cha et al., [1], Qu et al. [2] proposed JRD with a grouping constraint.

For current JRDs, the delivery assumptions are sometimes too strong such that generality is lost. For example, all the outbound deliveries are assumed as one-to-one delivery in Moon et al. [16]. Sometimes, delivery assumptions are complex to implement, such as in Kim et al. [15]. In reality, the outbound delivery strategy largely depends on the number of customers who order items. For example, if several customers order a specific product, the conventional one-to-one delivery strategy in Çetinkaya and Lee[11], Çetinkaya et al. [17] and Moon et al. [16] show higher distribution cost. Therefore, based on the work of Moon et al. [16], Qu et al. [2] and Cui et al. [18], we improve the delivery end of JRD under stationary policy and construct a new joint replenishment and delivery model with a varying number of customers who order multiple items.

Another great challenge in the study of JRDs is finding effective and efficient approaches for researchers [9]. JRPs [8] and JRDs [1] have been proven as NP-hard problems. Current approaches for JRPs can be classified as heuristics and meta-heuristics [6]. Available approaches to JRPs include an iterative algorithm, RAND algorithm [19], power-of-two policy [20], genetic algorithm (GA) [1], and DE [21]. The most well-known heuristic procedure for JRP is *RAND* for an equally divided search space. Other heuristics are mainly developed by using the RAND approach, such as the *QD-RAND* [22], *SP-RAND* [16]. In *RAND*, a continuous solution space must be searched so that small spaces can be equally split to find global optimal solutions, which constrains its ability to find solutions distributed in discrete spaces.

Current solving methods for JRDs originated from JRPs. Certain researchers opt to find proper meta-heuristic algorithms to solve different JRDs, such as the heuristic and evolutionary algorithm. Some approaches obtain close or even better results compared with those obtained by heuristic approaches. For example, GAs [1] and DEs [23] show good comprehensive performance in handling this NP-hard problem. However, finding a proper algorithm to solve a specific JRD is different because many candidate algorithms are available. *A more promising approach is to take advantage of the superior settings of currently adopted evolution algorithms and develop a new high performance algorithm*.

This study aims to contribute to literature in JRD research by providing an improved JRD model and effective intelligent algorithms. Our research differs from previous research and the main contributions are given in the following aspects:

(1) A new JRD model is constructed, in which the outbound schedule/scheme is improved from a fixed one-to-one service mode to a touring mode that dynamically changes according to a specific number of served customers.
(2) Three beta-heuristic algorithms named QEA, DE and QDE are specifically designed and utilized to solve the new JRD model. The evolutionary processes of these algorithms are redesigned according to the structure of the decision variables.
(3) Parameter sensitivities, most effective parameter settings, and the robustness of threes algorithms in solving small scale JRP, benchmark functions, and JRDs are vigorously compared and analyzed.
(4) Two supplement experiments are conducted to explore the performance of DE and QDE in solving the large-scale JRD and JRD with resource limitations.

The remainder of this paper is organized as follows. Section 2 presents the notations and the model formulation. Most importantly, the improvements to the conventional JRD and the steps for processing delivery strategy are introduced. Section 3 discusses the detail designed three algorithms, QEA, DE and QDE for solving the new JRD. Parameter sensitivities and robustness of three algorithms are tested and the computational results are intensively discussed in Section 4. The conclusions and future research are given in Section 5.

2. Improved JRD model

We reconsider the situation in Moon et al. [16], and a third party warehouse is considered in the center of a worldwide supply chain. Specifically, the warehouse replenishes multiple items from suppliers abroad, and then sells the items to the customers who sent the orders through the e-market. We also assume that the warehouse is located in the center of the local area and that the customers are randomly scattered over this area and around the warehouse. Therefore, the warehouse should decide the inbound schedule and delivery schemes considering the replenishment schedule of multi-item. The *objective* is to determine the *replenishment frequencies of multi-item*, the *basic cycle time*, and the *customer visiting sequence*, simultaneously. A sketch map of the improved JRD problem is illustrated in Fig. 1.

The basic notations below are employed in the following contents:

Basic notations	
i	the index of ith items, $i = 1,2,...,n$.
TC	the total cost of the consolidated JRD per unit time
D_i	annual demand of item i
S	major ordering cost of each order
s_i	minor ordering cost of each item
s_i^c	fixed outbound transportation cost of each item
h_i	holding cost of item i per unit, per time
w_i	customer waiting cost of item i per unit, per time
Q_i	order quantity of item i
T	basic cycle time (*decision variable*)
k_i	replenishment schedule of item i (*decision variable, positive integer*)
K	$n \times 1$ vector that consists of k_i
f_i	outbound delivery schedule of item i (*decision variable, positive integer*)
F	$n \times 1$ vector that consists of f_i

2.1. JRD with the stationary policy

JRD with the stationary policy has two typical features, one is that intervals between successive deliveries must remain the same throughout the planning horizon. The other is that the consolidated freight order quantity also remains unchanged. The specific process is as follows: First, the warehouse replenishes item i at each integer multiple k_i of the basic cycle time T. Then, the warehouse delivers the ordered items to the customers according to the order information, inventory level and outbound schedule f_i of item i. Fig. 2 provides a good understanding of JRD with the stationary policy.

The total cost per unit time (or *total cost* for short) is given by

$$TC(T,K,F) = \frac{1}{T}\left(S + \sum_{i=1}^{n}\frac{s_i}{k_i}\right) + \sum_{i=1}^{n}\frac{(f_i-1)k_iTD_ih_i}{2f_i} + \sum_{i=1}^{n}\frac{f_is_i^c}{k_iT} + \sum_{i=1}^{n}\frac{k_iTD_iw_i}{2f_i} \quad (1)$$

The total cost is the summation of four terms: major and minor ordering cost, inventory holding cost, outbound delivery cost and customer waiting cost. T, K and F are the decision variables that minimize

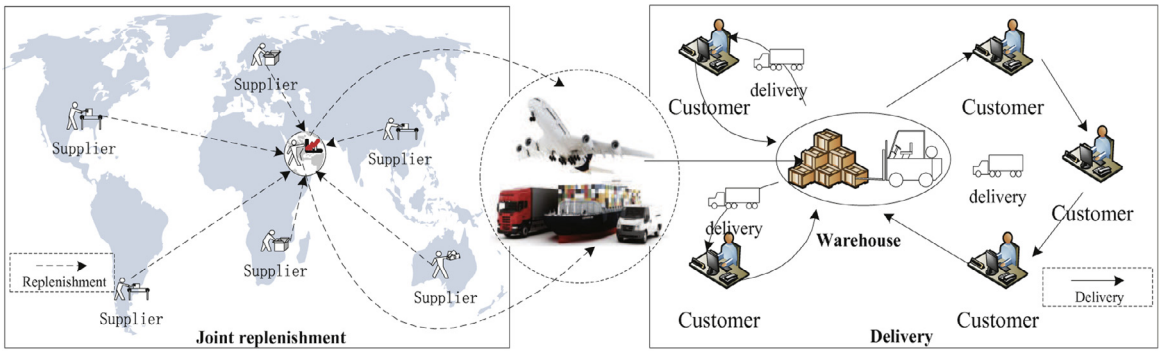

Fig. 1. Sketch map of the proposed JRD.

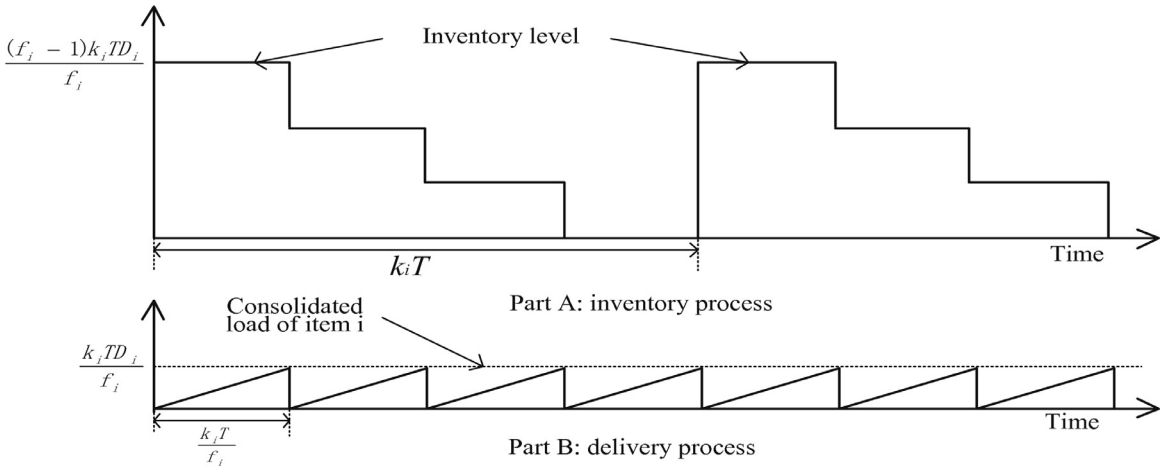

Fig. 2. Sketch map of the JRD problem [16].

total cost. According to the proof of Cha et al. [1], the total cost function TC is a convex function of T for a given set of k_is and f_is. Thus, the optimal basic cycle time T^* can be easily obtained by taking the first-order derivative of the total cost function TC as shown in Eq. (2).

$$T^* = \left[\frac{2(S + \sum_{i=1}^{n} \frac{s_i + f_i s_i^c}{k_i})}{\sum_{i=1}^{n} k_i D_i (h_i + \frac{w_i - hi}{f_i})} \right]^{1/2} \tag{2}$$

Currently, heuristic methods for JRD are mainly the recursive algorithms, such as the *SP-H* algorithm and *SP-RAND* algorithm [16]. Specifically, the essence of those heuristical algorithms is altering the values of k_is and f_is at certain times and spaces to find the best T^* and obtaining minimal TC. Given that K and F are integer vectors, the best k_i satisfies the conditions $TC(k_i) \le TC(k_i - 1)$ and $TC(k_i) \le TC(k_i + 1)$ for a given T, and the best f_i satisfies the conditions $TC(f_i) \le TC(f_i - 1)$ and $TC(f_i) \le TC(f_i + 1)$. Based on the two conditions and Eq. (1), two inequalities on k_i and f_i are deduced as follows:

$$k_i(k_i - 1) \le \frac{2(s_i + f_i s_i^c)}{T^2 D_i (h_i + \frac{w_i - hi}{f_i})} \le k_i(k_i + 1) \tag{3}$$

$$f_i(f_i - 1) \le \frac{k_i^2 T^2 D_i (w_i - h_i)}{2 s_i^c} \le f_i(f_i + 1) \tag{4}$$

2.2. New delivery strategy

2.2.1. Delivery problem description

In this part, we develop an important extension of the stationary policy, which supplements the one-to-one delivery process. In practice, jointly replenished items are usually sent to different customers who ordered them through the e-market. Single item orders may be received from some new, different customers or orders with different items are received from a single long term customer. It is apparent that the delivery cost is usually higher than when the visiting sequence is arranged situation if all the items are sent to the customers in the one-to-one mode. Thus, delivery cost can be decreased through vehicle routing techniques, such as the technique for the traveling salesman problem (*TSP*). However, the following three questions should be resolved before designing the delivery scheme:

(1) How to distinguish orders that must be synthetically dispatched from those that are not?
(2) How to link customers' orders and their locations?
(3) How to express visiting sequences and how to select the best one route?

The notations below are applied to our new designed delivery scheme:

s_i^t	outbound transportation cost of ith item per unit, per mile
p	subscript denotes location of warehouse and customers
P	the total number of customers
$d_{p_1 p_2}$	distance from p_1 to p_2
M	the unit vector consists of $P \times 1$ items
α_i	decision coefficient, and $\alpha_i = k_i / f_i$

where $p = 0, 1, 2, \ldots, P$. If $p = 0$, it represents the central warehouse, and if $1 \le p \le P$, it represents the corresponding customer who sends the order. For $d_{p_1 p_2}$, we have $p_1 \ne p_2$, and $p_1, p_2 = 0, 1, 2, \ldots, P$.

2.2.2. Delivery scheme design

(1) The answer to the first question

In our new delivery scheme, the outbound delivery frequency of item i is controlled by the decision coefficient α_i,

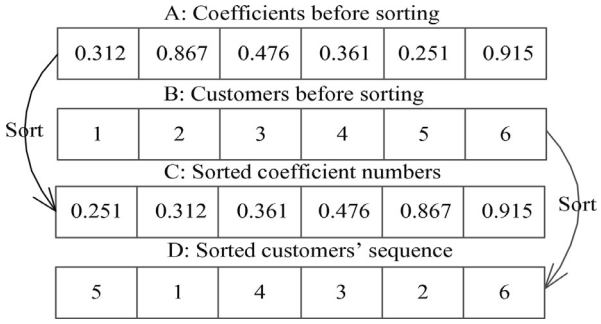

Fig. 3. Visiting sequence arrangement.

and $\alpha_i = k_i/f_i$. An observation of the stationary policy shows that the two factors k_i and f_i affect each other in the outbound interval settings and the items with different outbound intervals cannot be synthetically dispatched. Thus, a sole k_i or f_i cannot be utilized to decide the outbound delivery frequency of item i, the two factors must be taken together, see Fig. 2. The outbound interval is set as $\alpha_i T$, and $\alpha_i T = \frac{k_i}{f_i} \cdot T$. Consequently, the answer to the first question is that the outbound frequency is determined by the coefficient α_i. Specifically, the scheme is designed with following in mind: if one $\alpha_i(s)(i = 1, 2, \ldots, n.)$ is different from each other, the corresponding item is delivered to customer via the one-to-one mode. If equal α_i s $(i = 1, 2, \ldots, n.)$ exist, the corresponding items are delivered synthetically.

(2) The answer to the second question
Two types of matches should be considered here: the match of orders and customers, and the match of the customer number and the customer positions. The main processing procedure is given as follows:

Step 1: The solely dispatched orders and synthetically dispatched orders are ascertained. Specifically, if the orders are solely dispatched ones, delivery cost is computed according to the third term of Eq. (1); otherwise, turn to Step 2.
Step 2: A new binary value variable is defined as $x_{p,i} = \{0, 1\}$, $x_{p,i} = 1$ if and only if item i is ordered by customer p. Otherwise $x_{p,i} = 0$. Thus, the match of orders and customers is a $P \times n$ matrix A, and $x_{i,p} \in A$.
Step 3: The service vector is determined. Let A right multiply M, and the resulting vector is $Serv = A \times M$.
Step 4: The customers who send the orders are determined. If the element in *Serv* is not equal to 0, the corresponding customer sends the order, and the magnitude of the element represents the ordered types of items; if the element in *Serv* is equal to 0, the corresponding customer does not send the order.

(3) The answer to the third question
For the third question, many procedures are proposed to solve the touring problem. Here, we adopt a simple sequence coefficient sorting method. The specific procedure is given as follows:

Step 1: P coefficients in (0,1) are randomly generated, these numbers correspond to the customers who need to be served.
Step 2: The rank of the coefficients is determined. The coefficient numbers are sorted in ascending order.
Step 3: The visiting sequence of customers is determined according to the rearranged positions of the coefficients.

A six-customer example to illustrate this process is presented in Fig. 3. After the visiting sequence arrangement procedure is finished, the visiting sequence of six-customer is obtained as 0-5-1-4-3-2-6-0, where '0' denotes the warehouse.

After above three questions have been resolved, we learn that the total outbound delivery cost of each item is not a fixed value, but needs to be estimated according to the delivery distance. By referring to the typical uncapacitated *TSP*, the model formulation process is given as

$$min\, s_i^c = s_i^t \cdot \sum_{p_1=1}^{N_i} \sum_{p_2=0}^{N_i} d^i_{p_1,p_2} x^i_{p_1,p_2} \tag{5}$$

$$x^i_{p_1,p_2} = \begin{cases} 1 & \text{Visiting from customer} p_1 \text{ to } p_2 \\ 0 & \text{Others} \end{cases} \quad \forall p_1, p_2 \in P, p_1 \neq p_2 \tag{6}$$

$$\sum_{p_1=1}^{N_i} x_{p_1,p_2} = 1,\ \forall p_2\ and\ N_i \subset \{1, 2, \ldots P\} \tag{7}$$

$$\sum_{p_2=1}^{N_i} x_{p_1,p_2} = 1,\ \forall p_1\ and\ N_i \subset \{1, 2, \ldots P\} \tag{8}$$

$$\sum\nolimits_{p_1,p_2 \in S \times S} x_{p_1,p_2} \leq |S| - 1,\ S \subset \{1, 2, \ldots N_i\}\ and\ S \neq \Phi \tag{9}$$

where Eq. (5) is the minimum outbound delivery cost of item i; Eq. (6) denotes a binary value $x^i_{p_1,p_2}$, if and only if the visiting sequence is from Customer p_1 to Customer p_2, $x^i_{p_1,p_2} = 1$; otherwise, $x^i_{p_1,p_2} = 0$; Eqs. (7) and (8) guarantee that one customer is only visited once per tour. Eq. (9) eliminates sub-tour in an outbound delivery process. Thus, the total delivery cost $\sum_{i=1}^{n} \frac{f_i s_i^c}{k_i T}$ (the third term in Eq. (1)) is replaced by its new value $\sum_{i=1}^{n} \frac{1}{N_i} \cdot \frac{f_i s_i^c}{k_i T}$, where N_i denotes that N_i customers order item i. In the stationary policy, if N_i customers order item i, the total outbound delivery cost of item i is the summation of N_i one-to-one delivery cost. In the synthetical delivery mode, N_i customers are served by using a one-time tour.

3. Three algorithms for the new JRD

In this part, three algorithms named QEA, DE and QDE are presented for the proposed JRD. The common parameters of the three algorithms are as follows: t is the tth generation; *pop* is the population size; l is the length of the chromosome; and *fit* is the fitness function. We assume *fit* to be equal to the total cost. All the *decision variables*, namely, *K, F, T* and the visiting sequence, are expressed as a chromosome vector as follows.

$$chromosome = (k_1, k_2, \ldots, k_n, f_1, f_2, \ldots, f_n, T, p_1, p_2, \ldots, p_P) \tag{10}$$

The upper and lower bound expressions of *K, F, T*, and *P* are defined as (K^{lower}, K^{upper}), (F^{lower}, F^{upper}), (T^{lower}, T^{upper}), and (P^{lower}, P^{upper}), respectively.

3.1. QEA for the proposed JRD

QEA is developed by Han and Kim [24], and has been testified as an efficient algorithm for solving numerous optimization problems. QEA includes features of GA, but differs in chromosome initialization, reproduction, and mutation. The evolutionary stages of QEA for the proposed JRD are given as follows.

3.1.1. Initialization

In QEA, an individual chromosome usually comprises a certain number of quantum bits (Q-bits). The state of a single Q-bit is expressed as a linear superposition mode:

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{11}$$

where α and β are complex numbers, and called probability amplitudes of $|0\rangle$ and $|1\rangle$, respectively. The smallest unit of information stored in a Q-bit may be in $|0\rangle$, $|1\rangle$ or any superposition of the two states. $|\alpha|^2$and $|\beta|^2$ denote the probabilities of the Q-bit in '0' state and in '1' state, and they must satisfy the normalization condition $|\alpha|^2 + |\beta|^2 = 1$. Thus, a Q-chromosome $q_{\tilde{i}\cdot}^t$ with length l is usually encoded as:

$$q_{\tilde{i}\cdot}^t = \begin{bmatrix} \sin(\theta)_{\tilde{i},1}^t & \sin(\theta)_{\tilde{i},2}^t & \cdots & \sin(\theta)_{\tilde{i},l}^t \\ \cos(\theta)_{\tilde{i},1}^t & \cos(\theta)_{\tilde{i},2}^t & \cdots & \cos(\theta)_{\tilde{i},l}^t \end{bmatrix} \tag{12}$$

where $\theta \in [0, 2\pi]$, t is the tth iteration number of QEA. To distinguish the index of item from the index of Q-chromosome, we add a tilde ˜ over i, and $\tilde{i}\cdot = 1, 2, \ldots, pop$. In classic QEA, θ is usually initialized as $\pi/4$. The observation process is as follows:

(1) A number noted by '*rand*' in (0,1) is randomly generated. (2) A Q-bit is observed by comparing the generated number *rand* with the probability amplitude of the Q-bit. If *rand* is larger than the amplitude, '1' is obtained; otherwise, '0' is obtained. The process is continued until all Q-bits of a Q-chromosome are observed.

Instead of obtaining '0' or '1' after the observation, we adopt the approach introduced by Zheng and Yamashiro [25], in which the probability amplitudes are used directly. After the observation process, we assume that the Q-chromosomes become Q-chrom, see Eq. (13) below. In addition, to obtain integer values to denote K and F, Q-chrom also needs to be translated into a new chromosome (see Eq. (10)) to compute the fitness function.

$$Q\text{-}chrom_{\tilde{i}\cdot}^t = (\gamma_{\tilde{i},1}^t, \gamma_{\tilde{i},2}^t, \ldots, \gamma_{\tilde{i},\tilde{j}}^t, \ldots, \gamma_{\tilde{i},l}^t) \tag{13}$$

where $\gamma_{\tilde{i},\tilde{j}}^t$ is the observed value of $sin(\theta)_{\tilde{i},\tilde{j}}^t$ or $cos(\theta)_{\tilde{i},\tilde{j}}^t$, and $\tilde{j} = 1, 2, \ldots, l$. To obtain the fitness, the elements in *Q-chrom* should be processed as Eq. (14):

$$x_{\tilde{i},\tilde{j}}^{t=0} = round[abs(\gamma_{\tilde{i},\tilde{j}}^{t=0}) \times (x^{upper} - x^{lower})] + 1 \tag{14}$$

where $x_{\tilde{i},\tilde{j}}^{t=0} \in X_{\tilde{i}\cdot}^t$, $X_{\tilde{i}\cdot}^t$ represents the new real value *chromosome*, $x^{lower} = K^{lower}/F^{lower}/T^{lower}/P^{lower}$, $x^{upper} = K^{upper}/F^{upper}/T^{upper}/P^{upper}$, 'abs' is to obtain the absolute value of all $x_{\tilde{i},\tilde{j}}$, and 'round()' is to obtain the rounded value **if and only if** $\cdot\tilde{j} \le 2 \times n$.

3.1.2. Reproduction and mutation

The chromosome updating mechanism of QEA consists of two processes: reproduction and mutation. The quantum rotation gate, which includes the quantum rotation angle and direction, controls the reproduction process of QEA. In traditional, the magnitude of the quantum rotation angle and the rotation direction are mainly obtained from the lookup table. For example, a rotation gate $U(\theta_{\tilde{i},\tilde{j}})$ as an updating operator is employed to update an old Q-bit $[\alpha_{\tilde{i},\tilde{j}}, \beta_{\tilde{i},\tilde{j}}]'$, and a new *Q-bit* $[\tilde{\alpha}_{\tilde{i},\tilde{j}}, \tilde{\beta}_{\tilde{i},\tilde{j}}]'$ is reproduced by using Eq. (15):

$$\begin{bmatrix} \tilde{\alpha}_{\tilde{i},\tilde{j}} \\ \tilde{\beta}_{\tilde{i},\tilde{j}} \end{bmatrix} = U(\theta_{\tilde{i},\tilde{j}}) \begin{bmatrix} \alpha_{\tilde{i},\tilde{j}} \\ \beta_{\tilde{i},\tilde{j}} \end{bmatrix} = \begin{bmatrix} \cos(\theta_{\tilde{i},\tilde{j}}) & -\sin(\theta_{\tilde{i},\tilde{j}}) \\ \sin(\theta_{\tilde{i},\tilde{j}}) & \cos(\theta_{\tilde{i},\tilde{j}}) \end{bmatrix} \begin{bmatrix} \alpha_{\tilde{i},\tilde{j}} \\ \beta_{\tilde{i},\tilde{j}} \end{bmatrix} \tag{15}$$

where $\theta_{\tilde{i},\tilde{j}}$ is the rotation angle obtained by the equation: $\theta_{\tilde{i},\tilde{j}} = s(\alpha_{\tilde{i},\tilde{j}}, \beta_{\tilde{i},\tilde{j}})\Delta\theta_{\tilde{i},\tilde{j}}$. $s(\alpha_i, \beta_i)$ is the sign of the rotation angle to control the positive and negative rotation of $\theta_{\tilde{i},\tilde{j}}$, $\Delta\theta_{\tilde{i},\tilde{j}}$ is the magnitude of the rotation angle, all of which are the empirical values from the lookup table [24], $\tilde{i}\cdot = 1, 2, \ldots, pop$, $\cdot\tilde{j} = 1, 2, \ldots, l$, and the single quotation mark $'$ means transposition.

In the mutation operation, a quantum *NOT* gate is commonly used. The *NOT* gate mutation is to swap the two positions of a Q-bit. A *NOT* gate U is expressed as follows:

$$U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{16}$$

The mutation process usually adopts the single-point crossover method. The mutation procedure is provided as follows. (1) An integer number (noted by *pos*) in (1, l) to represent the mutation position of a chromosome is randomly generated. (2) Using the *Not* gate U right multiplies the *pos*-th Q-bit $[\alpha_{\tilde{i},\tilde{j}}, \beta_{\tilde{i},\tilde{j}}]'$. Then, the Q-chromosome is mutated. According to our experience, the single point mutation is not sufficient to generate a new Q-chromosome with high fitness. Thus, in our settings for JRD, the mutation operation is run 10 times.

3.1.3. Selection

The fitness function guides the selection process of QEA. The fitness function updating is similar to that of other EA algorithms using the greedy algorithm. As noted above, the total cost function is adopted as the fitness function (noted by *fit*). The selection procedure is as follows: (1) The current best *fit* = the best *fit* of the initialized population is set. (2) Children's fitness is computed. (3) The fitness of the children's and the current best *fit* is compared: if the new *fit* is lower than the current best *fit*, the current best *fit* and its corresponding chromosome are updated. Otherwise, the current best *fit* is retained and the chromosome is unchanged.

The superiorities of QEA have been testified in solving *TSP* and some binary value optimization, but it still has some shortcomings. First, the encoding scheme of QEA makes it efficient in solving some binary value problems though [24], few studies discussed its performance on real value encoding problems. Second, the updating mechanism is too complicated and stylized. Third, the Q-bit easily leads to fine searches, but small rotation angle makes a sudden exit from the local optimal difficult.

3.2. DE for JRD

Storn and Price [26] first presented DE, particularly in terms of solving multi-modal functions. Recently, Wang et al. [9] found that DE is an efficient and effective algorithm for solving JRP. Thus, DE is adopted to solve the proposed JRD. The general operations of DE for the proposed JRD are given as follows.

3.2.1. Initialization

DE initializes the population randomly through the uniform distribution over the search spaces [18]. Specifically, if we define the tth generation population as D_{pop}^t, one of its chromosomes as $X_{\tilde{i}\cdot}^t$, then we have $D_{pop}^t = \{X_{\tilde{i}\cdot}^t | x_{\tilde{i},\tilde{j}}^{lower} \le x_{\tilde{i},\tilde{j}}^t \le x_{\tilde{i},\tilde{j}}^{upper}, \tilde{i} = 1, 2, \ldots, pop; \tilde{j} = 1, 2, \cdots, l; x_{\tilde{i},\tilde{j}}(t) \in X_{\tilde{i}\cdot}^t\}$, where $x_{\tilde{i},\tilde{j}}^{lower} = K^{lower}/F^{lower}/T^{lower}/P^{lower}$, and $x_{\tilde{i},\tilde{j}}^{upper} = K^{upper}/F^{upper}/T^{upper}/P^{upper}$, $x_{\tilde{i},\tilde{j}}^t$ is generated according to Eq. (17).

$$x_{\tilde{i},\tilde{j}}^{t=0} = round(x_{\tilde{i},\tilde{j}}^{lower} + rand(0, 1) \times (x_{\tilde{i},\tilde{j}}^{upper} - x_{\tilde{i},\tilde{j}}^{lower})) \tag{17}$$

where $x_{\tilde{i},\tilde{j}}^{t=0} \in X_{\tilde{i}\cdot}^t$, and 'round()' obtains the rounding value **if and only if** $\cdot\tilde{j} \le 2 \times n$.

3.2.2. Reproduction and crossover

The reproduction and crossover processes of DE are mainly controlled by the scaling factor F_c and crossover probability*CR*. For each parent $X_{\tilde{i}}^{t=G}$, a new reproduced chromosome $V_{\tilde{i}}$ at the *G+1*th generation is generated by Eq. (18):

$$V_{\tilde{i}}^{t=G+1} = X_{r_1}^{t=G} + F_c \times (X_{r_2}^{t=G} - X_{r_3}^{t=G}) \tag{18}$$

where F_c is the scaling factor and $F_c \in [0, 1]$, r_1, r_2, r_3 are three randomly generated distinct numbers. None of them coincide with the current target parent $\tilde{i}$, also $r_1 \ne r_2 \ne r_3 \ne \tilde{i}$. To guarantee the success of the mutation operation, the population size must be larger than 4.

Through the reproduction process, possibilities are available that generating illegal offsprings, such as the negative values and non-integer values. To eliminate the illegal genes, we first generate a temporary population as the substitution population. Then, if there has

Temporary V:	***3***	***4***	***6***	***8***	***0.45***	***0.23***	***0.19***
X_{r1}:	1	4	2	8	0.95	0.25	0.54
X_{r2}:	2	3	2	6	0.76	0.02	0.62
X_{r3}:	5	1	8	1	0.32	0.87	0.41
V:	-0.8	5.2	-1.6	11	1.21	-0.26	0.67
Adjusted V:	***3***	***4***	***6***	***8***	***0.45***	***0.23***	0.67

$k_i \in [1,5]$, $f_i \in [1,10]$, $\mathcal{T} \in (0,1)$ and $p_i \in (0,1)$

Fig. 4. Exceptions processing of DE.

an illegal gene, a temporary gene in the corresponding position of the temporary population is utilized to replace it. To illustrate the processing procedure, a population with 2 items and 2 customers is applied as an illustration example, and the scaling factor is set as $F_c = 0.6$. Results obtained by using above rules are shown in Fig. 4 below.

Then, the crossover operation is performed to generate the children chromosomes. We assume a temporary population U to store the new crossed gene $u_{\tilde{i},\tilde{j}}$. Then the crossover procedure is given as follows:

$$u_{\tilde{i},\tilde{j}}^{G+1} = \begin{cases} v_{\tilde{i},\tilde{j}}^{G+1} & \text{if } \mathrm{rand}(\tilde{j}) \le CR \text{ or } \tilde{j} = \mathrm{randn}(\tilde{j}) \\ x_{\tilde{i},\tilde{j}}^{G} & \text{otherwise} \end{cases} \tag{19}$$

where $u_{\tilde{i},\tilde{j}} \in U_{\tilde{i}}$, $v_{\tilde{i},\tilde{j}} \in V_{\tilde{i}}$, $rand(\tilde{j}) \in (0,1)$, and randn$(\tilde{j})$ used to generate $\tilde{j}$-th integer number and $randn(\tilde{j}) \in [1,2,\cdots,l]$, CR is the crossover probability.

3.2.3. Selection

In selection operation, the $(G+1)$-th generation $X_{\tilde{i}}^{G+1}$ is constructed, which is guided by the fitness function using the greedy strategy. The selection rule is:

$$X_{\tilde{i}\cdot}^{G+1} = \begin{cases} U_{\tilde{i}\cdot}^{G+1} & fit(U_{\tilde{i}\cdot}^{G}) < fit(X_{\tilde{i}\cdot}^{G}) \\ X_{\tilde{i}\cdot}^{G} & \text{otherwise} \end{cases} \tag{20}$$

where the $fit(\cdot)$ is the fitness function of DE. The updating takes places when the fitness of mutated vector $U_{\tilde{i}\cdot}^{G}$ is better than that of the father chromosome $X_{\tilde{i}\cdot}^{G}$.

The superiorities of DE are mainly reflected in its coarse search abilities and fewer parameters [9]. However, some uncertainties remain exist with regard to its two parameter settings, namely, F_c and CR. Finding an effective and efficient combination of F_c and CR to strengthen the fine searching abilities of DE also requires extensive effort.

3.3. QDE for the proposed JRD

We propose a new QDE for the JRD by combing the superiorities of QEA and DE. Some researchers have already focused on the superiorities of QDE. For example, Zheng and Yamashiro [25] used QDE to solve flow shop scheduling problems, in which the quantum encoding and DE based rotation angle updating are adopted. Su et al. [27] proposed the DE/QDE algorithm to discover classification rules, in which quantum initialization and DE mutation and crossover were adopted. In another paper, a DE/QDE algorithm was proposed to explore the Takagi-Sugeno fuzzy models [28]. The two papers have demonstrated the superiorities of QDE in searching the binary-valued spaces. However, the performance of QDE in searching the discrete spaces is still need to investigate. Thus, a QDE algorithm is presented below.

3.3.1. Initialization

The initialization of QDE is shared with the initialization process of QEA.

3.3.2. Reproduction and Crossover

Unlike Su et al. [27] who conducted the reproduction and crossover operations to the quantum rotation angles, we design a new strategy that are directly performed on Q-bits. The procedures for reproducing of QDE are as follows:

(1) The notations adopted are the same as those in Section 3.3.2. We assume that the quantum population has been obtained, and for each of its parent Q-chromosome, $Q_{\tilde{i}\cdot}^{t=G}$ the reproduction is performed. A single $Q_{\tilde{i}\cdot}^{t=G}$ has two chains including $\alpha_{\tilde{i}}$s and $\beta_{\tilde{i}}$s. Then, we rewrite the $Q_{\tilde{i}\cdot}^{t=G}$ as $Q_{\tilde{i}\cdot}^{t=G}(1)$ and $Q_{\tilde{i}\cdot}^{t=G}(2)$, where $\alpha_{\tilde{i}}^{t} \in Q_{\tilde{i}\cdot}^{t=G}(1)$, and $\beta_{\tilde{i}}^{t} \in Q_{\tilde{i}\cdot}^{t=G}(2)$.

(2) A new reproduced chromosome $V_{\tilde{i}}$ at $G+1$-th generation is generated using Eq. (21), where the settings of F_c, r_1, r_2, and r_3 are shared those from Section 3.3.2.

$$V_{\tilde{i}}^{t=G+1} = Q_{r_1}^{t=G}(1) + F_c \times (Q_{r_2}^{t=G}(1) - Q_{r_3}^{t=G}(1)) \tag{21}$$

(3) Exceptions for $v_{\tilde{i},\tilde{j}} \in V_{\tilde{i}}$ are processed as that: if $v_{\tilde{i},\tilde{j}} > 1$, $v_{\tilde{i},\tilde{j}} = rand(0,1)$; if $v_{\tilde{i},\tilde{j}} < 0$, $v_{\tilde{i},\tilde{j}} = rand(0,1)$.

For the crossover operations, we also assume a temporary population U to hold the new crossed gene $u_{\tilde{i},\tilde{j}}$. The crossover procedure is given below:

$$u_{\tilde{i},\tilde{j}}^{G+1} = \begin{cases} v_{\tilde{i},\tilde{j}}^{G+1} & \text{if } \mathrm{rand}(\tilde{j}) \le \text{ CR or } \tilde{j} = \mathrm{randn}(\tilde{j}) \\ q_{\tilde{i},\tilde{j}}^{G}(1) & \text{otherwise} \end{cases} \tag{22}$$

where $u_{\tilde{i},\tilde{j}} \in U_{\tilde{i}}$, $v_{\tilde{i},\tilde{j}} \in V_{\tilde{i}}$, $q_{\tilde{i},\tilde{j}}^{G}(1) \in Q_{\tilde{i}\cdot}^{G}(1)$, $rand(\tilde{j}) \in (0,1)$, and randn$(\tilde{j})$ is used to generate $\tilde{j}$-th integer number and $randn(\tilde{j}) \in [1,2,\cdots,l]$, and CR is the crossover probability. Being observed that after crossover, the two probability amplitudes of a Q-bit are applied, that is to say more carrying information of a quantum gene is contained in QDE, which strengthens its searching abilities.

3.3.3. Selection

In the selection operation, the $(G+1)$th generation $Q_{\tilde{i}\cdot}^{G+1}$ is generated, which is guided by the fitness function using the greedy strategy. The selection rule is given below:

$$Q_{\tilde{i}\cdot}^{G+1} = \begin{cases} U_{\tilde{i}\cdot}^{G+1} & fit(U_{\tilde{i}\cdot}^{G}) < fit(Q_{\tilde{i}\cdot}^{G}) \\ Q_{\tilde{i}\cdot}^{G} & \text{otherwise} \end{cases} \tag{23}$$

where the $fit(\cdot)$ is the fitness function of QDE. The updating takes place when the fitness of mutated vector $U_{\tilde{i}\cdot}^{G}$ is better than that of the father $Q_{\tilde{i}\cdot}^{G}$.

As the three algorithms, QEA, DE and QDE, share similar evolutionary processes, the pseudocodes of the three algorithms provide as Algorithm 1. The flow chart of three algorithms is given in Fig. 5.

4. Numerical experiment and discussion

In this section, three algorithms are utilized to solve the new proposed JRD. All the algorithms are coded by using Matlab 7.1 and the experiments are conducted on a personal computer with a 2.7 GHz CPU and 2 GB RAM. First, the parameter sensitivity of the three algorithms is extensively analyzed. The case in Moon et al. [16] is used to find the proper parameter settings. Then, the three algorithms are

Algorithm 1 Pseudocodes for the improved JRD.

Initialize the parameters of JRD and set the total cost $fit(T, K)$ of JRD as the fitness function.
Initialize each algorithm, including the iteration step $n = 0$ and the maximum iteration *Max*, population size P, the initialized chromosome, and so on. Compute $fit_n(T, K)$ based on Eq. (5) and output current best chromosome and current test fitness as Bestfitness.
while $n \leq Max$ **do**
  Perform the reproduction operation according to Eq. (15) (or Eq. (18)) and output the reproduced chromosome.
  Perform the crossover operation on the reproduced chromosome according to Eq. (16) (or Eq. (19)) and output the crossbred chromosome.
  if a gene in a chromosome is illegal **then**
    Regenerate a new gene to replace the illegal one.
  end if// Below procedure is utilized to perform the selection operation
  for $p = 1$ to P **do**
    Compute $fit_n(T, K)$ based on population $P(n)$ and
  end for
  Output the smallest $fit_n(T, K)$ as the *CurBesfit(n)* and its corresponding position *pos* in the population $P(n)$.
  if $CurBesfit(n) < CurBesfit(n-1)$ **then**
    CurrentBestChromosome = chromosome(n, *pos*). // *chromosome* is updated
    Bestfitness =$CurBesfit(n)$.
  else
    CurrentBestChromosome = chromosome($n-1$, *pos*); // *chromosome* is not updated
    Bestfitness =$CurBesfit(n-1)$.
  end if
  n=n+1.
end while
Output the Bestfitness and CurrentBestChromosome. Translate the outputs into the decision variables.

Table 1
Basic data for the test example ($S = 200$).

Item i	1	2	3	4	5	6
D_i	10000	5000	3000	1000	600	200
s_i	45	46	47	44	45	47
h_i	1	1	1	1	1	1
s_i^c	5	5	5	5	5	5
w_i	1.5	1.5	1.5	1.5	1.5	1.5

experimented on 10 benmark functions to test their searching abilities and stableness. After that the proposed algorithms are utilized to solve the proposed JRD. Finally, discussions on the findings and two further explorations are provided.

4.1. Parameter sensitivity analysis

4.1.1. Overall investigation

No mature general benchmark on the stationary JRD problem exists. Thus, a numerical example with six items is adopted, which was given in Cha et al. [1] to investigate the performance of GA. GA is a stochastic search algorithm based on the mechanism of natural selection and natural genetics [16]. GA has been testified as an effective algorithm in solving continuous and discrete optimization problems, which makes GA very suitable for processing the integer decision variables of JRP [14,16]. Thus, it is reasonable to adopt GA as a comparing algorithm in contrasting the performance of three algorithms. Basic data for overall tests are provided in Table 1.

Table 2
Overall comparison of different algorithms.

Algorithm	k_i	f_i	T	TC	Error rate(%)	CPU time(s)
SP-CC[a]	1,1,1,1,1,1	5,4,3,2,1,1	0.2215	5001.31	3.57	–
SP-H[a]	1,1,1,1,2,3	4,3,2,1,2,2	0.1973	4850.39	0.45	–
SP-RAND[a]	1,1,1,2,2,4	4,3,2,3,2,2	0.1881	4828.89	0.00	–
GA[b]	1,1,1,2,3,4	4,3,2,3,2,3	0.1881	4828.89	0.00	–
GA	1,1,1,2,2,4	4,3,3,2,2,4	0.1872	4856.95	0.58	1.662
QEA	1,1,1,2,2,4	4,3,3,3,3,2	0.2047	4846.76	0.37	1.628
DE	1,1,1,2,2,5	4,3,2,3,2,3	0.1881	4828.89	0.00	1.514
QDE	1,1,1,2,2,5	4,3,2,3,2,3	0.1881	4828.89	0.00	1.507

[a] is from Moon et al. [16],
[b] is from Cha et al. [1].

The comparing results of different algorithms are provided in Table 2. The optimal value of the case is 4828.89. In Table 2, three algorithms, SP-CC, SP-H and SP-RAND, are provided by [16], GA given by [1] is also recoded and added in our comparisons. All algorithms are all run 10 times and their median values are adopted, see Fig. 6. The initial quantum rotation angle of QEA and QDE is set as $\pi/4$, the scaling factor as $F_c = 0.6$ and crossover probability as CR=0.3, and the maximum generation is 150.

Error rate $= (TC - optimal)/optimal \times 100\%$. During the overall test, DE and QDE can always obtain their optimal value, whereas GA and QEA cannot find a better optimum. The data in Table 2 show that the performance of the beta-heuristic algorithms is close to or better than that of some heuristic algorithms.

4.1.2. Parameter analysis of the three algorithms

The parameters usually have influential effects on the performance of a beta-heuristic algorithm. As noted above, three main parameters, namely, the initial angle of the quantum chromosome, the scaling factor F_c and the mutation probability CR, may affect the performance of corresponding algorithm. Thus, we investigate their different settings to find their best performance one in solving JRD.

(1) Parameters tuning test 1 for QEA
The effects of different initial angles of QEA are first investigated. The reasons for the initial angles selected in $[0.01\pi, 0.5\pi] \times rand(0, 1)$ are as follows. (1) If the initial angle is too small (less than 0.01π), the evolutionary process is too slow and the results are not very good. (2) If the initial angle is larger than 0.5π, then the tests on angles that larger than 0.5π are redundant because the angles are centrosymmetric and only positive values are supported. The role of $rand(0, 1)$ is applied to increase angle randomness.
Each algorithm is run 10 times, their results are listed in Table 3. Note that TC and the computation time are adopted as two main factors for evaluating QEA.
Table 3 shows that 0.25π is the most proper initialization angle, the average TC is 4832.05, and the computation time is 1.46 s.

(2) Parameter tuning test 2 for DE
Scaling factor F_c and mutation probability CR can solely affect the efficiency of DE. Furthermore, different combinations of F_c and CR can also significantly influence DE. Thus, the two parameters are investigated together in the following experiments. Each combination is run 10 times so that the test in Table 4 is run 1000 times in total. The average value of ten run results is adopted and the maximum generation is set to 100. As the CPU time in each run is quite randomly distributed and close to each other in different settings, Thus, TC is selected as the unique evaluation factor.
Results of Table 4 show that the scaling factor F_c plays a more important role than CR in some extent. Taking the grey colored positions for example, CR lies in a relatively wide range from

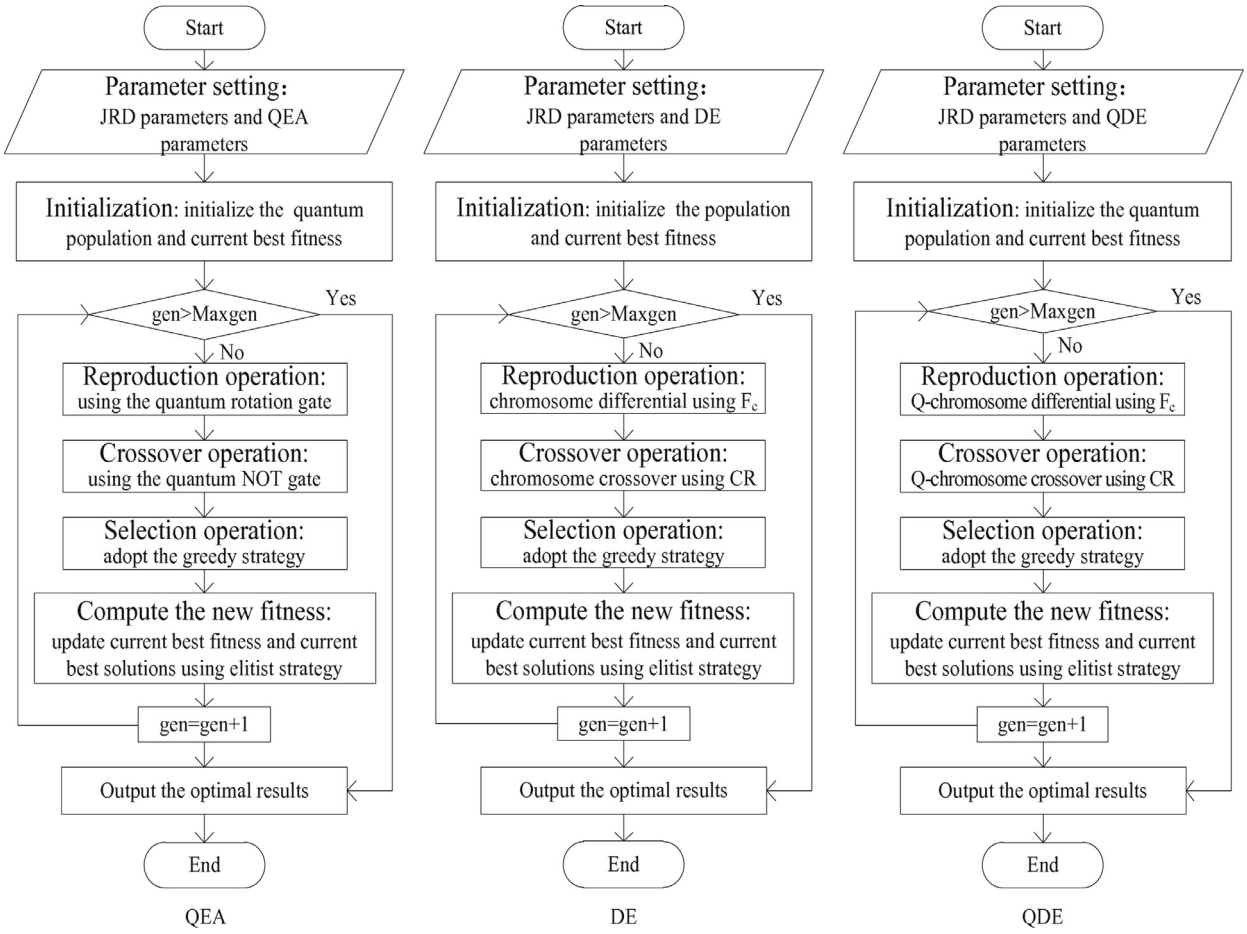

Fig. 5. Flow chart of three algorithms for the proposed JRD.

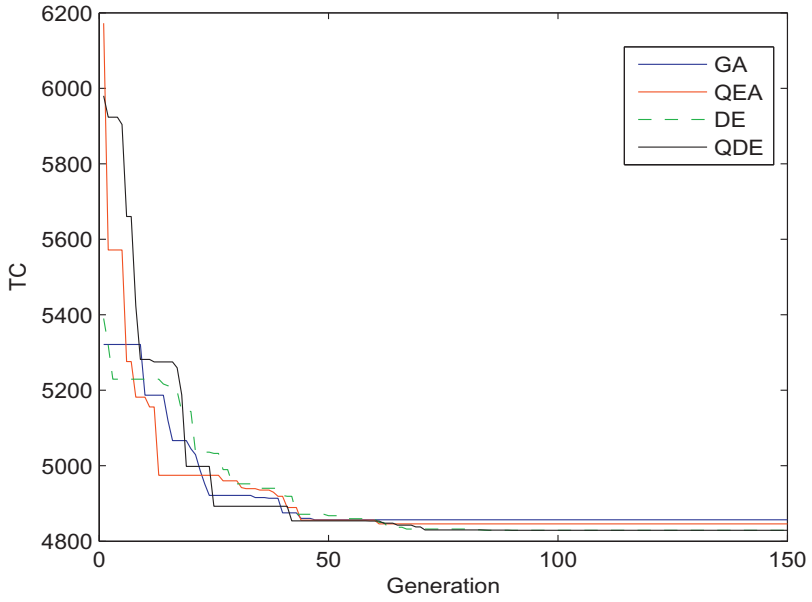

Fig. 6. Overall performance of four algorithms.

0.2 to 0.8 for $F_c = 0.5$. Thus, the optimal total cost can always be achieved given that the proposed DE rapidly converges to the optimal, many combinations of F_c and CR can be used as ideal candidates for the JRD under the stationary policy. Thus, further tests are needed to find the most proper or best combinations of three different parameters.

(3) Parameters tuning test 3 for QDE

The third test are on the three parameters, namely, quantum initial angles, F_c and CR are conducted for QDE. Each test is run 10 times and their mean value is selected. Other settings are kept the same as those in test 2, and the total cost is selected as an evaluation factor. The results are listed in Table 5.

Table 3
Parameter test 1 – TCs and CPU time under different QEA's initial angles.

	0.01π		0.05π		0.1π		0.25π		0.5π	
	TC	Time(s)	TC	Time(s)	TC	Time(s)	TC	Time(s)	TC	Time(s)
1	4834.84	1.51	4839.92	1.80	4833.01	1.59	4828.89	1.56	4832.21	1.25
2	4832.77	1.78	4836.68	1.22	4843.47	1.62	4837.77	1.25	4838.29	1.85
3	4855.01	1.63	4831.36	1.36	4843.34	1.33	4831.90	1.56	4837.06	1.86
4	4838.15	1.34	4842.65	1.43	4832.07	1.48	4828.89	1.00	4829.55	1.45
5	4832.06	1.18	4835.38	1.07	4830.45	1.97	4832.02	1.91	4840.64	1.48
6	4834.20	1.81	4854.03	1.60	4833.62	1.26	4828.89	1.17	4833.67	1.11
7	4832.23	1.06	4828.89	1.20	4846.65	1.41	4836.07	1.50	4832.82	1.21
8	4833.46	1.57	4834.18	1.76	4829.39	1.92	4834.43	1.62	4831.74	1.48
9	4838.90	1.55	4831.32	1.14	4837.43	1.80	4832.78	1.94	4835.29	1.39
10	4828.89	1.17	4836.88	1.11	4829.73	1.57	4828.89	1.06	4832.87	1.67
AVG	4836.05	1.46	4837.13	1.37	4835.92	1.60	4832.05	1.46	4834.41	1.47

Note: AVG is the average value, Time is the CPU's running time.

Table 4
Parameter test 2 – TCs under differentF_c and *CR* combinations of DE.

F_c\CR	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	4831.13	4831.71	4831.01	4836.09	4839.45	4862.50	4889.38	4904.08	5098.61	5632.56
0.2	4830.69	4829.21	4828.96	4830.13	4844.87	4869.80	4855.40	4920.47	5023.06	5610.50
0.3	4829.77	4828.90	4829.73	4830.56	4831.89	4847.95	4834.62	4849.61	4928.97	5509.38
0.4	4830.10	4828.89	4828.89	4828.89	4828.89	4830.45	4831.17	4833.02	4851.31	5157.62
0.5	4829.61	4828.89	4828.89	4828.89	4828.89	4828.89	4828.89	4828.89	4830.61	4944.61
0.6	4831.35	4828.92	4828.89	4828.89	4828.89	4829.21	4829.42	4831.54	4839.45	4899.12
0.7	4832.64	4829.24	4828.96	4828.93	4829.00	4829.71	4831.97	4838.50	4855.70	4940.91
0.8	4832.11	4829.24	4829.30	4829.01	4829.30	4831.89	4841.40	4845.91	4883.68	4946.22
0.9	4836.00	4830.67	4829.55	4829.53	4834.62	4839.68	4848.88	4863.12	4888.19	4979.56
1.0	4348.35	4346.59	4346.45	4347.13	4349.63	4357.07	4359.18	4371.59	4420.05	4662.15

Note: The row value are CRs, and the column values are F_cs.

Table 5
Parameter test 3 — QDE's TCs under different combinations.

$(F_c, \mathrm{CR})\backslash\theta$	0.01π	0.05π	0.1π	0.25π	0.5π
(0.4,0.2)	4833.01	4831.73	4828.90	4832.76	4837.52
(0.4,0.3)	4831.69	4907.00	4831.79	4835.45	4837.01
(0.4,0.4)	4831.61	4897.61	4830.76	4828.93	4831.92
(0.5,0.2)	4828.89	4835.44	4838.07	4859.92	4885.81
(0.5,0.3)	4833.01	4828.94	4832.72	4829.89	4835.44
(0.5,0.4)	4828.90	4829.89	4829.24	4829.24	4843.68
(0.5,0.5)	4831.92	4828.99	4845.12	4833.01	4838.88
(0.5,0.6)	4832.42	4829.29	4838.89	4831.69	4868.27
(0.5,0.7)	4834.65	4832.04	4829.57	4831.70	4850.90
(0.5,0.8)	4830.48	4829.32	4831.70	4850.90	4907.00
(0.6,0.3)	4828.89	4828.89	4839.01	4840.27	4849.69
(0.6,0.4)	4829.68	4829.53	4849.44	4838.02	4854.76
(0.6,0.5)	4839.26	4829.29	4839.29	4856.65	4872.59

Table 6
Benchmark functions.

Function $f(x)$	Domain	n	x^*	$f(x^*)$
$f_1(x) = \sum_{i=1}^{n} x_i^2$	[−100,100]	30	0	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	[−10,10]	30	0	0
$f_3(x) = \sum_{i=1}^{n-1} (\sum_{j=1}^{n} x_j)^2$	[−100,100]	30	0	0
$f_4(x) = \sum_{i=1}^{n} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[−10,10]	30	0	0
$f_5(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5\rfloor)^2$	[−100,100]	30	0	0
$f_6(x) = 10n + \sum_{i=1}^{n} [x_i^2 - 10cos(2\pi x_i)]$	[−5.12,5.12]	30	0	0
$f_7(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$	[−600,600]	30	0	0
$f_8(x) = 1 - cos(2\pi \sqrt{\sum_{i=1}^{n} x_i^2}) + 0.1\sqrt{\sum_{i=1}^{n} x_i^2}$	[−100,100]	30	0	0
$f_9(x) = \sum_{i=1}^{n} \lvert x_i sin(x_i) + 0.1x_i\rvert$	[−10,10]	30	0	0
$f_{10}(x) = -20exp(0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})$ $-exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)) + 20 + exp(1)$	[−32,32]	30	0	0

The results of the grey-colored positions in Table 5 show that the most effective combination is the angle $= 0.01\pi \times rand(0,1)$, $F_c = 0.5$ and $CR = 0.2$. Thus, similar settings can also be applied to solve the new JRD.

4.2. Experiments on benchmark functions

The parameter tests for QEA, DE have been conducted to reveal their best settings, but the efficiency and effectiveness of QEA, DE and the improved QDE are still need to be verified through further experiments. Therefore, 10 widely applied benchmark functions [23,29,30], including 5 unimodal functions and 5 multimodal functions, are adopted to test the performance of three algorithms. The information on the 10 benchmark functions is provided in Table 6, and the computed results are listed in Table 7 after ten independent runs of each algorithm. The maximum iteration of each algorithm is set as 500 and the other parameter settings are inherited from above settings.

The results in Table 7 reveal us that the searching effectiveness of three algorithms, DEA, DE and QDE shows disparities. Specifically, QEA shows the worst results of all experiments, while the superiorities of DE and QDE reflected in stableness and effectiveness differ from each other when searching the best solution of different bench mark functions. Thus, the performance of three algorithms in solving JRD should be strictly compared and analyzed in the following contents.

4.3. Experiments on the new proposed JRD

The new proposed JRD links the replenishment and delivery processes together. As researchers have not presented JRD benchmarks, customer data from Wang et al. [9] with three suppliers $1 \le p \le 3$ are adopted. The distance matrix of the warehouse and customers is provided in Table 8. As the total delivery cost (not the distance) is one of our concerns, the delivery cost per mile is defined as $cpm = 0.1$. We also adopt the item data in Moon et al. [16] with 6 items, $1 \le i \le 6$. Subsequently, the newly designed matching matrix of orders and

Table 7
Computational results of three algorithms.

$f(x)$	QEA				DE				QDE			
	Min	Max	Mean	Std.	Min	Max	Mean	Std.	Min	Max	Mean	Std.
$f_1(x)$	3.62E−04	4.56E−04	4.20E−04	6.31E−05	2.75E−06	7.32E−06	3.29E−06	3.03E−06	4.01E−06	8.23E−07	1.94E−06	2.62E−06
$f_2(x)$	2.22E−02	9.34E−02	8.93E−02	5.14E−02	8.29E−04	5.93E−03	3.03E−03	3.02E−03	4.45E−04	1.74E−03	7.69E−04	9.72E−04
$f_3(x)$	6.23E−02	8.04E−02	6.97E−02	1.16E−02	6.20E−05	5.13E−03	1.39E−04	3.42E−03	1.01E−06	5.34E−04	9.87E−05	3.19E−04
$f_4(x)$	9.02E−04	4.24E−03	1.23E−03	2.26E−03	4.52E−07	2.39E−06	8.63E−07	1.21E−06	6.94E−06	1.99E−07	3.33E−07	4.84E−06
$f_5(x)$	15	45	30.27	21.23	0	7.45	3.76	4.99	0	5.21	2.2	3.04
$f_6(x)$	7.23E−02	0.21	9.90E−02	8.60E−02	5.30E−02	0.44	9.20E−02	0.25	1.19E−02	0.11	3.87E−02	6.72E−02
$f_7(x)$	0.02	12.79	7.91	5.08	0	3.42	0.88	2.35	0	1.29	0.43	0.90
$f_8(x)$	5.35E−04	3.46E−03	2.07E−03	2.07E−03	7.24E−06	9.34E−05	5.01E−05	5.99E−05	3.43E−07	3.62E−06	9.26E−07	2.28E−06
$f_9(x)$	1.63E−05	4.65E−04	2.45E−04	2.97E−04	5.50E−08	9.10E−06	6.72E−07	6.31E−06	4.30E−09	6.65E−06	2.76E−06	4.68E−06
$f_{10}(x)$	3.54	0.03	2.61	2.63	5.81E-04	7.40E−02	6.98E−03	5.51E−02	6.69E−06	9.26E−04	5.51E−05	6.69E−04

Table 8
Geographical layout (distances in miles) [9].

	Warehouse	Customer 1	Customer 2	Customer 3
Warehouse	–	11	9	7
Customer 1	11	–	5	8
Customer 2	9	5	–	10
Customer 3	7	8	10	–

Table 9
Matching matrix of customers and orders.

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
Customer 1	1	0	0	0	0	1
Customer 2	0	1	0	0	1	0
Customer 3	0	0	1	1	0	1

customers is provided in Table 9, where '0' means no order placed, and '1' means that an order is placed.

4.3.1. Computational results

The experiments are performed with the following parameter settings: the max generation is 500, each algorithm is run 30 times, the initial rotation angle is $0.01\pi \times rand(0, 1)$, $F_c = 0.5$ and $CR = 0.2$. The computational results are listed in Table 10. The results in Table 10 show the following conclusions:

(1) In terms of searching efficiency, QDE is the most efficient algorithm (15.3911s), and QEA is the slowest algorithm (16.7463s). These findings prove that our improvements to the classic QEA enhanced the computational speed. Both GA and DE exhibit moderate searching efficiencies, but GA (15.6896s) is a slightly better than DE (16.5452s).
(2) In terms of searching for the best total cost, DE and QDE can find their best and the lowest total cost (4448.63). QEA is the worst performing algorithm in this aspect, the lowest total cost that QEA found is only 4718.43.
(3) In terms of the stabilities of the evolutionary process, QDE is the best algorithm and has the lowest average total cost (4456.31), which proves that our improvements enhanced the searching ability of the classic QEA in another aspect. DE and GA obtained the moderate results and QEA remains the worst algorithm (4818.83).
(4) In terms of searching for the full-customer visiting sequence, both DE and QDE can find the shortest route, whereas GA and QEA cannot.

The evolutionary processes of the four algorithms are provided in Fig. 7.

Fig. 7 shows that GA and QEA are the fastest algorithms that converge to their (local) optimal values, while DE is the third fastest algorithm that converges to a stable value. QDE is the last algorithm that converges to a stable value. Comparison GA, QEA and DE, Fig. 7 provides us some basic understanding of the convergence process of QDE.

Overall, the improvement to the classic JRD is reduced the total cost from 4829.29 to 4448.63. Moreover, the improvement to QEA is also considered successful with the introduction of the differential operation of DE.

4.3.2. Parameter sensitivity analysis of the new proposed JRD

The disturbance of problem related parameters may affect the total cost of JRD, as well as the searching effectiveness of these algorithms. Thus, these parameters need to be tested further.

Six parameters of the improved JRD problem, namely, D_i, h_i, S, s_i, cpm, and w_i, have close relationships with the total cost. Thus, these parameters are analyzed under different settings to observe their effect on the efficiency of different algorithms. Given that only DE and QDE are superior in solving the new JRD, the next analysis is performed on the two algorithms with the same settings as in Section 4.3. The amount of item is still set as three. Each algorithm is run 20 times and the average total cost is utilized. The computational results are reported in Table 11.

The results in Table 11 show the fluctuation of different parameters to the total cost of the improved JRD. For example, the total cost is less than 4,000 if the demand decreases by 30%, and the total cost is more than 5,000 if the annual demand increases by 30%. Generally speaking, annual demand holding cost of per item and two types of ordering cost have the most significant influence. For example, the change rate of total cost is 13.3% = (4448.6–3857.6)/4448.6 × 100% (solved by DE) when the holding cost decreases by 30%. Whereas per-mile delivery cost and customer waiting cost have a slight effect

Table 10
Computational results of GA, QEA, DE, and QDE.

	Best k_i	Best f_i	Sequence*	Best T	Best TC	AVE.TC	AVE.CPU time(s)
GA	1,1,1,1,2,3	4, 4, 4, 4, 4, 2	[0-1-2-3-0]	0.1942	4528.08	4620.56	15.6896
QEA	1,1,1,1,2,4	6, 3, 9, 1, 1, 1	[0-1-2-3-0]	0.1695	4718.43	4818.83	16.7463
DE	1,1,1,2,2,4	5, 5, 5,10,10,3	[0-2-1-3-0]	0.1848	4448.63	4468.78	16.5452
QDE	1,1,1,2,2,4	5, 5, 5,10,10,3	[0-2-1-3-0]	0.1848	4448.63	4456.31	15.3911

Note: The * denotes the warehouse, and AVE is the average value.

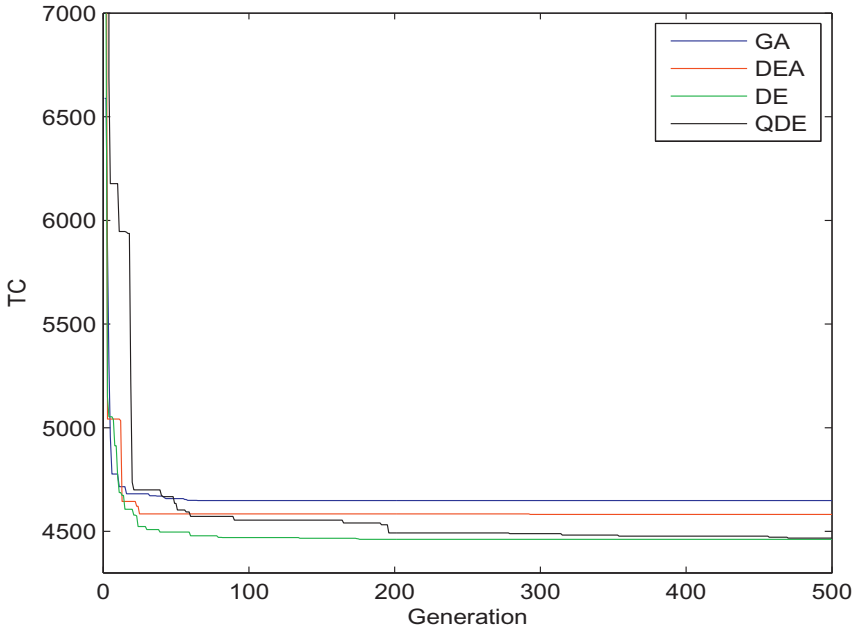

Fig. 7. Performance of four algorithms for the new JRD.

Table 11
Parameter sensitivity analysis under DE and QDE.

Δ	−30%		−20%		−10%		10%		20%		30%	
	DE	QDE	DE	QDE	DE	QDE	DE	QDE	DE	QDE	DE	QDE
D_i	3729.0	3728.3	3984.1	3984.2	4224.4	4224.4	4671.8	4673.4	4873.2	4873.6	5078.6	5078.6
h_i	3857.6	3863.0	4073.9	4074.7	4273.9	4276.6	4616.1	4616.8	4765.9	4765.9	4913.8	4913.8
S	4114.3	4114.2	4229.0	4233.1	4341.0	4346.1	4557.5	4558.1	4660.9	4656.0	4749.1	4749.5
s_i	4114.6	4114.9	4233.3	4234.3	4343.6	4346.1	4558.7	4558.7	4657.8	4661.4	4755.6	4759.3
cpm	4242.4	4242.4	4431.9	4433.7	4442.9	4442.8	4463.4	4497.0	4567.1	4518.9	4475.5	4527.2
w_i	4417.3	4417.6	4434.7	4433.2	4444.3	4444.4	4459.9	4461.8	4467.7	4466.7	4479.3	4479.2

Table 12
Input parameters of the large JRD.

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8	Item 9	Item 10
D_i	700	8000	600	500	1400	900	15000	1200	800	700
s_i	26	30	38	18	42	36	29	16	27	30
h_i	9.6	7	8	24	30	28	36	40	26	15
w_i	1	1	1	1	1	1	1	1	1	1
cpm	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

on the total cost. For example, the change rate of total cost is 0.7% = (4448.6–4417.3)/4448.6 × 100% (solved by DE) if delivery cost decreases by 30%. Notably, the major (S) and minor (s_i) ordering cost show a nearly identical influence on the total cost, which is due to the linear relationships between the two parameters.

A comparison between the performance of the two algorithms shows that DE obtained slightly better results than QDE in a few cases. In most cases, QDE is as capable as DE in finding the optimal value and it performs better in some cases. For example, the total cost solved by QDE is lower than that of DE when w_i decreases by 20%, which shows that QDE is a promising algorithm for solving the new JRD. In terms of searching efficiency (which is not shown in our computational results), the searching time of QDE (approximately 13 s) is shorter than that of DE (approximately 14s) by more than 1s, which reveals us that QDE solves JRD more quickly.

4.4. Two further explorations

Two remaining problems need to be addressed: the performance of DE and QDE in solving large scale JRD, and investigating the performance of the two algorithms in solving JRD with resource limitations. Further investigations would provide a deep understanding of the new improved JRD and the most suitable algorithm.

4.4.1. Test for a large-scale JRD case

In this part, a revised large-scale JRD case with 10 items and six customers is introduced by referring to the large-scale JRD case in Wang et al. [9] and Cui et al. [18]. Related parameters are listed in Tables 12–14. The two algorithms are all run 5 times and the computational results are listed in Table 15.

From the results in Table 15, we can conclude that: (1) DE and QDE show disparities in solving large-scale JRD problem. Generally speaking, DE can find the lowest total cost, while QDE searches more quickly than DE, but both algorithms can obtain similar basic cycle times (approximately 0.15). (2) In finding the minimum total cost, DE shows better performance (5266.82), also in finding the maximum of the lowest total cost, DE obtains the largest value 5241.41. (3) The proposed QDE shows the highest stability in solving the large case JRD (the average total cost of JRD is 5234.18 through QDE). Moreover, both of the two algorithms cannot find a stable visiting sequence,

Table 13
Matching matrix of customers and orders.

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8	Item 9	Item 10
Customer 1	1	0	0	0	0	0	0	0	0	0
Customer 2	0	1	0	0	0	0	0	0	0	0
Customer 3	0	0	1	1	0	0	0	0	0	0
Customer 4	0	0	0	0	1	1	1	0	0	0
Customer 5	0	0	0	0	0	0	1	1	0	0
Customer 6	0	0	0	0	0	0	0	0	1	1

Table 14
Coordinates of warehouse and customers.

	Warehouse	Suppliers					
		1	2	3	4	5	6
x	265	295	301	309	217	218	282
y	257	272	258	260	274	278	267

which leaves some spaces for improvement in future study. Therefore, QDE can be considered as the most efficient and effective algorithm in solving large-scale JRDs.

4.4.2. Resource limitation consideration

The delivery process of JRD is assumed to have no limitations. In practice, managers are always faced with limited resources such as storage, transport equipment capacity, and budget [1]. Thus, the performance of the two algorithms should be further verified on JRD with resource limitations.

The limitations are mainly two types: the joint replenishment restriction and the outbound delivery restriction, which were introduced by Cha et al. [1]. The upper bounds of these restrictions are $\sum_{i=1}^{n} D_i K_i T b_i \le B^W$ and $\alpha_i D_i T b_i \le B_i^R$, respectively, where b_i is the unit weight of item i, B^W is the maximum weight capacity of a full-ship load of jointly replenished item(s), and B_i^R is the maximum weight capacity of a full truckload of the outbound delivered item(s). According to the searching strategy of T in Cha et al. [1], we have $T^\star = min(T_0, T_1)$, where T_0 is the original T computed by using Eq. (2), and $T_1 = min(\frac{B^W}{\sum_{i=1}^{n} D_i K_i T b_i}, \sum_{i=1}^{n} \frac{B_i^R}{\alpha_i b_i D_i})$.

We also assume $B^W = 25,000$, $B_i^R = 2,000$ and $b_i = 6.25$. Table 16 lists the computed results that were obtained after adding a resource restriction to the previous JRD example. The evolutionary processes of DE and QDE are illustrated in Fig. 8. Each algorithm is run 30 times, and the maximum generation is 500.

Results shown in Table 16 and Fig. 8 indicate the following conclusions: (1) QDE and DE have a similar ability in finding the replenishment frequencies, outbound frequencies, and the optimal total cost (see the Best k_i, Best f_i and 'Best TC (4449.15)'). (2) DE and QDE have almost similar stability when solving the resource-limited JRD (see the average value of TC 'AVG.TC'). (3) QDE is slightly faster than DE in finding the optimal value (see the average CPU time). (4) DE converges faster than QDE, but the converged results are opposite (see Fig. 8).

Furthermore, we explained why DE and QDE obtained an optimal value in the resource-limited JRD than in JRD without resource constraints. The main differences are that in the resource-limited JRD, the best T (equals to 0.1818) of JRD is lower than that in the situation without resource limitations, which indicates that resource limitations affect searching abilities of JRD for the optimal total cost. While in case of resource constraints are considered in JRD, the basic cycle time is shortened when limitations are considered in JRD.

5. Discussions and conclusions

In this study, based on the work of previous researchers, a new outbound delivery strategy is designed to deal with three problems, namely, synthetical dispatched multi-items grouping, the design of the matching matrix of orders and corresponding customers, and the expression design of the visiting sequence of different customers. The outbound delivery procedure is outlined by using this design. Then, the joint replenishment of multi-item and the improved outbound delivery strategy are considered simultaneously to formulate the JRD model.

To solve the improved JRD effectively, three algorithms, namely, QEA, DE and QDE, are adopted and revised in the initialization, the reproduction and crossover, and the selection. Their performance is

Table 15
Computational results of DE and QDE.

	DE						QDE					
	k_i	f_i	Sequence	T	TC	CPU time(s)	k_i	f_i	Sequence	T	TC	CPU time(s)
1	21212 21122	11111 11111	[0-3-1-2-6-4-5-0]	0.1514	5241.41	25.52	21212 21122	11111 11111	[0-1-2-6-3-4-5-0]	0.1495	5234.17	21.09
2	21212 21122	11111 11111	[0-6-1-2-5-5-4-0]	0.1500	5234.17	25.08	21212 21122	11111 11111	[0-1-2-6-5-4-3-0]	0.1505	5234.17	21.33
3	21212 21122	11111 11111	[0-2-3-4-6-1-5-0]	0.1499	5234.17	25.10	21212 21122	11111 11111	[0-6-1-2-4-5-3-0]	0.1503	5234.17	21.80
4	21222 21112	11111 11111	[0-2-1-6-4-3-5-0]	0.1499	5234.17	25.02	21212 21122	11111 11111	[0-1-5-2-3-4-6-0]	0.1500	5234.20	21.33
5	21222 21112	11111 11111	[0-1-4-6-2-3-5-0]	0.1488	5266.82	25.82	21212 21122	11111 11111	[0-1-3-2-4-6-5-0]	0.1498	5234.20	20.89
Max	–	–	–	–	5241.41	25.82	–	–	–	–	5234.20	21.80
Min	–	–	–	–	5266.82	25.02	–	–	–	–	5234.17	21.33
AVE	–	–	–	–	5242.15	25.31	–	–	–	–	5234.18	21.29

Table 16
Computational results of DE and QDE.

	Best k_i	Best f_i	Sequence	Best T	Best TC	AVE.TC	AVE.CPU time (s)
DE	1,1,1,2,2,4	5, 5, 5, 10, 10, 3	[0-2-1-3-0]	0.1818	4449.15	4452.86	13.68
QDE	1,1,1,2,2,4	5, 5, 5, 10, 10, 3	[0-2-1-3-0]	0.1818	4449.15	4452.39	13.49

Note: The AVE.TC is the average value of TC.

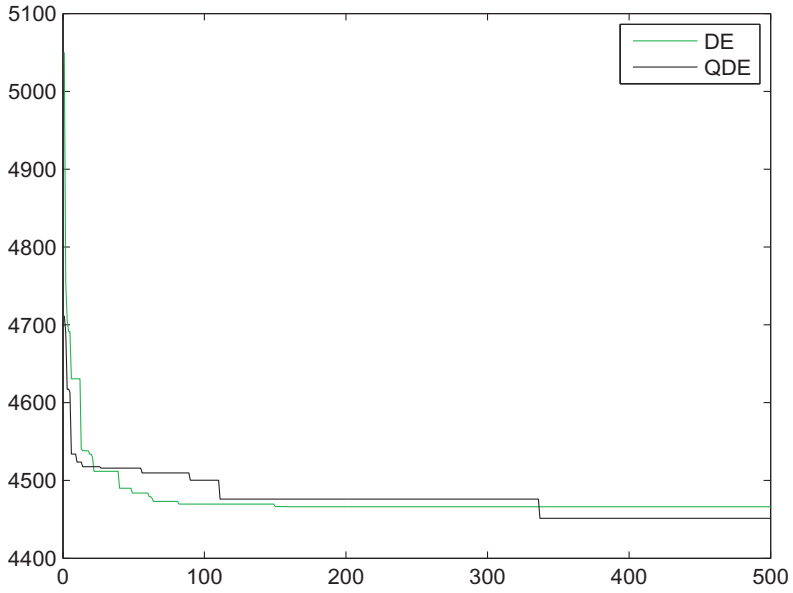

Fig. 8. Evolutionary processes of DE and QDE.

first verified by a JRD case under the stationary policy, in which the parameter analyses are conducted to find the best combinations of three important parameters. Then the proposed JRD is solved by using the above three algorithms. The main findings are summarized as follows:

(1) To our knowledge, this study is the first to propose a new joint replenishment and synthetical delivery model in a warehouse centralized supply chain.
(2) Results of numerical experiments show that QEA, DE and QDE are capable to solve the improved JRD. Particularly, DE and QDE perform better than some heuristic algorithms in some aspects. Notably, different parameter settings have different effects on obtaining the total cost of JRD.
(3) DE and QDE show disparities in three aspects, searching speed, searching result and searching stableness, for the small-scale JRD, large-scale JRD and JRD problem with resource limitations. Compared with DE, QDE solves the new JRD more quickly, has a more stable evolutionary process, and has an identical ability in finding optimal values.

Other intelligent algorithms, such as the genetic-simulated annealing algorithm and the fruit fly optimization algorithm, also show good performances in solving complex optimization problems. In the future, we will design hybrid algorithms by using the advantages of the above algorithms to handle more complex variations of JRDs.

Acknowledgments

This research is partially supported by National Natural Science Foundation of China (grant nos. 71371080; 71531009; 71131004; 71471024), Humanities and Social Sciences Foundation of Chinese Ministry of Education (No. 15YJA630095), and Scientific and Technological Research Program of Chongqing Municipal Education Commission (No. KJ1500523).

References

[1] B. Cha, I. Moon, J. Park, The joint replenishment and delivery scheduling of the one-warehouse, *n*-retailer system, Transp. Res. Part E: Logist. Transp. Rev. 44 (5) (2008) 720–730.
[2] H. Qu, L. Wang, Y.R. Zeng, Modeling and optimization for the joint replenishment and delivery problem with heterogeneous items, Knowl.-Based Syst. 54 (2013) 207–215.
[3] S. Goyal, Determination of optimum packaging frequency of items jointly replenished, Manag. Sci. 21 (4) (1974) 436–443.
[4] C. Chan, L. Li, T. Chi, et al., Scheduling of multi-buyer joint replenishments, Int. J. Product. Econ. 102 (1) (2006) 132–142.
[5] P. Robinson, A. Narayanan, F. Sahin, Coordinated deterministic dynamic demand lot-sizing problem: a review of models and algorithms, Omega 37 (1) (2009) 3–15.
[6] M. Khouja, S. Goyal, A review of the joint replenishment problem literature: 1989–2005, Eur. J. Oper. Res. 186 (1) (2008) 1–16.
[7] Q. Wang, S. Axsäter, Fixed-interval joint-replenishment policies for distribution systems with multiple retailers and stochastic demand, Naval Res. Logist. 60 (8) (2013) 637–651.
[8] E. Porras, R. Dekker, A solution method for the joint replenishment problem with correction factor, Int. J. Product. Econ. 113 (2) (2008) 834–851.
[9] L. Wang, J. He, D. Wu, et al., A novel differential evolution algorithm for joint replenishment problem under interdependence and its application, Int. J. Product. Econ. 135 (1) (2012) 190–198.
[10] W.W. Qu, J.H. Bookbinder, P. Iyogun, An integrated inventory – transportation system with modified periodic policy for multiple products, Eur. J. Oper. Res. 115 (2) (1999) 254–269.
[11] S. Çetinkaya, C.Y. Lee, Stock replenishment and shipment scheduling for vendor-managed inventory systems, Manag. Sci. 46 (2) (2000) 217–232.
[12] S. Sindhuchao, H. Romeijn, E. Akçali, et al., An integrated inventory-routing system for multi-item joint replenishment with limited vehicle capacity, J. Glob. Optim. 32 (1) (2005) 93–118.
[13] S. Hsu, Optimal joint replenishment decisions for a central factory with multiple satellite factories, Expert Syst. Appl. 36 (2) (2009) 2494–2502.
[14] C.K. Chan, B. Cheung, A. Langevin, Solving the multi-buyer joint replenishment problem with a modified genetic algorithm, Transp. Res. Part B: Methodol. 37 (3) (2003) 291–299.
[15] T. Kim, Y. Hong, S. Chang, Joint economic procurement–production–delivery policy for multiple items in a single-manufacturer, multiple-retailer system, Int. J. Product. Econ. 103 (1) (2006) 199–208.
[16] I. Moon, B. Cha, C. Lee, The joint replenishment and freight consolidation of a warehouse in a supply chain, International Journal of Production Economics 133 (1) (2011) 344–350.
[17] S. Çetinkaya, F. Mutlu, C.Y. Lee, A comparison of outbound dispatch policies for integrated inventory and transportation decisions, Eur. J. Oper. Res. 171 (3) (2006) 1094–1112.
[18] L. Cui, L. Wang, J. Deng, RFID technology investment evaluation model for the stochastic joint replenishment and delivery problem, Expert Syst. Appl. 41 (4) (2014) 1792–1805.
[19] M. Kaspi, M. Rosenblatt, An improvement of silver's algorithm for the joint replenishment problem, IIE Trans. 15 (3) (1983) 264–267.
[20] F.C. Lee, M.J. Yao, A global optimum search algorithm for the joint replenishment problem under power-of-two policy, Comput. Oper. Res. 30 (9) (2003) 1319–1333.
[21] L. Wang, J. He, Y.R. Zeng, A differential evolution algorithm for joint replenishment problem using direct grouping and its application, Expert Syst. 29 (5) (2012) 429–441.
[22] B. Cha, I. Moon, The joint replenishment problem with quantity discounts under constant demand, OR Spectr. 27 (4) (2005) 569–581.
[23] L. Wang, C. Dun, W. Bi, et al., An effective and efficient differential evolution algorithm for the integrated stochastic joint replenishment and delivery model, Knowl.-Based Syst. 36 (2012) 104–114.

[24] K. Han, J. Kim, Genetic quantum algorithm and its application to combinatorial optimization problem, in: Proceedings of the 2000 Congress on Evolutionary Computation, 2, IEEE, 2000, pp. 1354–1360.
[25] T. Zheng, M. Yamashiro, Solving flow shop scheduling problems by quantum differential evolutionary algorithm, Int. J. Adv. Manuf. Technol. 49 (5-8) (2010) 643–662.
[26] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[27] H. Su, Y. Yang, L. Zhao, Classification rule discovery with DE/QDE algorithm, Expert Syst. Appl. 37 (2) (2010) 1216–1222.
[28] H. Su, Y. Yang, Differential evolution and quantum-inquired differential evolution for evolving takagi–sugeno fuzzy models, Expert Syst. Appl. 38 (6) (2011) 6447–6451.
[29] Q.-K. Pan, H.-Y. Sang, J.-H. Duan, L. Gao, An improved fruit fly optimization algorithm for continuous function optimization problems, Knowl.-Based Syst. 62 (2014) 69–83.
[30] L. Wang, Y. Shi, S. Liu, An improved fruit fly optimization algorithm and its application to joint replenishment problems, Expert Syst. Appl. 42 (9) (2015) 4310–4323.