A Dynamic Logistic Dispatching System With Set-Based Particle Swarm Optimization

Ya-Hui Jia, *Student Member, IEEE*, Wei-Neng Chen, *Member, IEEE*, Tianlong Gu, Huaxiang Zhang, Huaqiang Yuan, Ying Lin, *Member, IEEE*, Wei-Jie Yu, *Member, IEEE*, and Jun Zhang, *Fellow, IEEE*

***Abstract*—With the rapid development of e-commerce, logistics industry becomes a crucial component in the e-commercial ecological chain. Impelled by both economical and environmental benefit, logistics companies demand automated tools more urgently than ever. In this paper, a dynamic logistic dispatching system is proposed. The underlying model of the dispatching system is the dynamic vehicle routing problem which allows new orders being received as the working day progress. With this feature, the system becomes more practical than the systems with traditional static vehicle routing models, but is also more challenging as the vehicles must be scheduled in a dynamic way. The core of the system is a specially designed set-based particle swarm optimization algorithm. According to the characteristic of the problem, a new encoding scheme is defined by set and possibility, and a local refinement method is designed to accelerate the convergence speed of the algorithm. In addition, two more techniques: 1) region partition and 2) archive strategy are incorporated in the dispatching system to reduce the complexity of the problem and to facilitate the optimization process, helping the dispatcher control the vehicles in real time. The proposed system is tested on various benchmarks with different scales. Experimental results show that the proposed dispatching system is effective.**

***Index Terms*—Capacitated vehicle routing problem (CVRP), dynamic vehicle routing problem (DVRP), set-based particle swarm optimization (S-PSO).**

Manuscript received December 13, 2016; accepted March 2, 2017. Date of publication April 7, 2017; date of current version August 16, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61622206, Grant 61379061, and Grant 61332002, in part by the Natural Science Foundation of Guangdong under Grant 2015A030306024, in part by the Guangdong Special Support Program under Grant 2014TQ01X550, and in part by the Guangzhou Pearl River New Star of Science and Technology under Grant 201506010002. This paper was recommended by Associate Editor J.-H. Chou. *(Corresponding authors: Wei-Neng Chen; Jun Zhang)*.

Y.-H. Jia is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with Sun Yat-sen University, Guangzhou 510006, China.

W.-N. Chen and J. Zhang are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: cwnraul634@aliyun.com; junzhang@ieee.org).

T. Gu is with the School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin 541004, China.

H. Zhang is with the School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China.

H. Yuan is with the School of Computer Science and Network Security, Dongguan University of Technology, Dongguan 523808, China.

Y. Lin and W. Yu are with Sun Yat-sen University, Guangzhou 510006, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2017.2682264

I. Introduction

In RECENT years, the rapid development of e-commerce has put forward higher requirement to conventional logistics industry. According to the data obtained from the company Alibaba, only in one day of the double-eleven shopping festival in China, the turnover reached almost 18 billion dollars, which means over 600 million parcels need to be delivered by logistics companies [1]. Facing such a big challenge to deliver so many goods to different locations within a short period, it is impossible for delivery companies to schedule every route of every vehicle manually. Meanwhile, since the concern of environment rises, logistics companies also need to pay more attention on vehicle routing to achieve the goal of environmental sustainability [2]. Thus developing automated dispatching system to help tackle the logistics distribution problem becomes more important and urgent than ever.

In essence, the logistics distribution problem can be seen as a variant of vehicle routing problem (VRP) in which a fleet of vehicles have to visit a set of customers at a minimum cost [3], [4]. Considering different constraints in real-world applications, researchers have proposed different VRP models, such as the VRP with time windows (VRPTW) [5], [6], the VRP with pickup and delivery [7], [8], the VRP with stochastic demands [9], multitrip VRP [10], multidepot VRP [11], etc. However for most VRP models, the information about customers and orders is taken as *a priori* knowledge, which is usually not the case happened in nowadays logistics scenarios. Consider two typical situations.

1) Courier service (pickup problem), empty vehicles leave the depot to collect parcels at customer locations in the working day, and new customers call for service at the meantime.
2) Consumable goods distribution (delivery problem), full-loaded vehicles leave the depot to transfer consumable goods, such as meat, cigarette, wine, and gasoline, to retailers or customers, meanwhile new customers or retailers submit their orders to suppliers.

In these cases, only a part of orders are known in advance before vehicles or workers leaving the depot. The other orders are revealed or received over time. Thus dynamism of the order should be considered in the dispatching system.

How to plan the routes of vehicles to get a minimum cost in such dynamic environment is known as the dynamic VRP (DVRP) [12] or online VRP [13]. In the dispatching system for DVRP, real-time control is highlighted as a crucial demand, because the environment of a DVRP is always

changing. Dispatchers, i.e., headquarter of the company, must react to the dynamic environment quickly as vehicles are running when they wait for commands, and new orders are also coming consecutively. If vehicles cannot be controlled in real time, the command sent from the dispatcher will be outdated. Thanks to the development of smart devices and global positioning system (GPS), nowadays a dispatcher can easily know where vehicles are and communicate with the drivers, which makes it possible to navigate vehicles in real time [12], [14]. However, how to plan the routes dynamically is still a challenging problem.

For decades, various methods have been proposed for different DVRPs [12], [13], and most of them are meta-heuristic approaches. By extending VRPTW, several dynamic VRPTW problems with different goals were proposed [15]–[17], and many algorithms were applied, such as parallel tabu search [15], genetic algorithm (GA) [17]–[20], column generation [19], adaptive large neighborhood search [16], etc. By extending capacitated VRP (CVRP), the dynamic CVRP (DCVRP) was also considered by some researchers with different approaches, such as ant colony system (ACS) [21], GA [22], adaptive particle swarm optimization (PSO) [23], variable neighborhood search [23], etc. In addition, some other dynamic scenes are also studied by some researchers, such as the dynamic pickup and delivery problem [24], the VRP with periodic changing environment [25], [26], etc. Because of the emergence of "bee box" (a free container for temporary storage), the time to collect or deliver parcels is no longer a hard constraint in many places. Even without the bee box, the time constraint is usually taken as a soft constraint in many real logistics applications where the service provider can negotiate with consumers. However, the capacity of vehicle is definitely a hard constraint that cannot be negotiated, because overload is always forbidden. Thus we build the dispatching system based on the DCVRP model. Moreover, theoretically speaking, the CVRP is the fundamental version of all VRPs, thus the CVRP with dynamism, i.e., DCVRP, can be taken as the fundamental version of all DVRPs. Therefore, building a system on DCVRP can help us explore the essence of DVRP, and further facilitate extending its usage to other situations.

Previous studies on DCVRP show that how to deal with dynamism is still an open issue. Most of the existing methods for this problem still have two common weaknesses. The first one is that the historical optimization information cannot be taken advantage effectively so that the algorithms are not efficient enough to adapt to the dynamic environment. The second weakness of the approaches is that their performance decreases rapidly with the growing of the problem scale. Considering these deficiencies, a dynamic dispatching system which applies a specially designed set-based PSO algorithm (S-PSO-D) is proposed. PSO was first proposed by Eberhart and Kennedy [27] to solve continuous optimization problems. Recently, Chen *et al.* [28] redefined the operators in PSO by set and possibility, generating a new algorithm called set-based PSO (S-PSO). S-PSO was proposed to cope with discrete combinatorial optimization problems (COPs), and its performance has been proved in several applications [6], [29], [30]. Compared with S-PSO, the encoding scheme of the original algorithm is redesigned in S-PSO-D to accommodate to the DCVRP. Furthermore, to adapt to the dynamic environment, two auxiliary techniques are proposed and incorporated in the dispatching system: 1) region partition and 2) archive strategy. Region partition is used to cut a big problem into pieces thus to reduce the difficulty of the problem. Archive strategy is used to accelerate the convergence speed of the algorithm so that the dispatcher can navigate the vehicle fleet in real time. Meanwhile, due to the introduction of archive, the velocity and position updating rules used in S-PSO are updated. In addition, a local refinement process is also designed for S-PSO-D to further improve the quality of solution. The experimental results on abundant benchmarks with different scales show that the proposed S-PSO-D dispatching system is effective.

The rest of this paper is organized as follows. Section II defines the model of the DCVRP. Section III presents a brief description of the S-PSO algorithm. In Section IV, the whole dispatching system which contains the S-PSO-D approach is demonstrated. Experimental results are shown in Section V and the conclusion is finally drawn in Section VI.

II. Dynamic Capacitated Vehicle Routing Problem

The DCVRP is defined based on the CVRP. As the elementary version of all VRPs, the CVRP is defined on a complete undirected graph $G = (C, E)$. $C = \{c_0, c_1, \ldots, c_n\}$ is the vertex set which contains n+1 elements. c_0 is the depot where all homogeneous vehicles depart from and finally return to. All vehicles have the same capacity $Q>0$, and operate at the same costs. The other n elements in C, i.e., $c_1, \ldots, c_n$, are n customers. Each customer c_i demands or has a certain quantity of goods which is called an order, and represented by a scalar $q_i > 0$. $E = \{e_{ij} | e_{ij} = (c_i, c_j), c_i, c_j \in C, i < j\}$ is the edge set. Each edge e_{ij} is associated with a weight factor w_{ij} to represent the travel time or travel distance. Supposing there are v vehicles, the CVRP can be formally stated as follows.

Define variable

$$x_{ij}^k = \begin{cases} 1, & \text{if vehicle } k \text{ travels directly from } i \text{ to } j \\ 0, & \text{otherwise} \end{cases}$$

$$y_i^k = \begin{cases} 1, & \text{if order } i \text{ is served by vehicle } k \\ 0, & \text{otherwise.} \end{cases}$$

The goal is to minimize the total travel distance

$$\min \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{v} w_{ij} \times x_{ij}^k \tag{1}$$

$$\text{s.t.} \sum_{i=0}^{n} x_{ij}^k = y_j^k \quad \forall k = 1, \ldots, v, \quad \forall j = 1, \ldots, n$$

$$\sum_{j=0}^{n} x_{ij}^k = y_i^k \quad \forall k = 1, \ldots, v, \quad \forall i = 1, \ldots, n \tag{2}$$

$$\sum_{i=0}^{n} y_i^k \times q_i \le Q \quad \forall k = 1, \ldots, v \tag{3}$$

$$\sum_{k=1}^{v} y_i^k = 1 \quad \forall i = 1, \ldots, n \tag{4}$$

$$\sum_{k=1}^{v} y_0^k = v. \tag{5}$$

A solution to the CVRP is a set of routes that satisfies: 1) the total demand of each route is no more than Q [constraint (3)]; 2) each customer is visited once and only once by only one vehicle [constraint (4)]; and 3) every route starts and ends at c_0 [constraint (5)] [31]–[34].

In the CVRP, all orders are received before the working day. However, in the DCVRP, only a portion of orders are known beforehand, called "static orders." Some other orders are revealed after vehicles leaving the depot, which are called "dynamic orders." Therefore, another attribute which needs to be considered in the DCVRP is the disclosure time of the order (the time when one order is received by the dispatcher). The disclosure time of the order of c_i is denoted as t_i $(1 \leq i \leq n)$. All static orders have a disclosure time equal to 0.

Different DVRPs have different levels of dynamism. For the DCVRP studied in this paper, there are usually two ways to measure the degree of dynamism [35], [36]. One way is to consider the ratio of dynamic orders, which is calculated by

$$\delta = \frac{n_d}{n} \tag{6}$$

where n_d is the number of dynamic orders and n is the total number of orders [35]. The other considers the disclosure time [36]

$$\delta = \frac{1}{n} \sum_{i=1}^{n} \frac{t_i}{T} \tag{7}$$

where T is the length of the planning horizon. It is worth noting that in most cases, T only occupies a part of the whole working day rather than equal to it. For example, some e-commerce companies such as Amazon and JD.com, have a delivery policy that orders submitted before a certain clock should be delivered in the same day, but the orders submitted after the clock can only be served in the next day. In this situation, the planning horizon ends at the preset clock rather than at the end of the working day. In this paper, we use (6) to measure the dynamism. The objective is still measured as the total travel distance of all vehicles in DCVRP.

To better understand the meaning of dynamic, an example with only two vehicles is shown in Fig. 1. Before vehicles leave the depot (time T_0), there are seven orders known in advance and the dispatcher has already planned two routes: (Depot$\rightarrow$ $A \rightarrow B \rightarrow C \rightarrow D \rightarrow$ Depot) and (Depot$\rightarrow$ $E \rightarrow F \rightarrow G \rightarrow$ Depot). At T_1, two new orders, H and I, are submitted to the system. Then the dispatcher checks the GPS, finding that the two vehicles are on their way to B and G now. It deletes the orders A, E, and F which have been served, and reoptimizes the routes to incorporate these two new orders. Two new routes are generated and sent to the drivers as: ($B \rightarrow H \rightarrow C \rightarrow D \rightarrow$ Depot) and ($G \rightarrow I \rightarrow$ Depot). Finally at T_f, seven static orders and two dynamic orders are all served and the vehicles return to the depot. In the problem studied in this paper, a necessary assumption is that if a vehicle is on its way to serve a customer, it cannot change direction to serve another one. This assumption is made based on two practical reasons. First, calculating new route for a constantly moving vehicle is difficult and time-consuming. Second, a courier usually informs his customer in advance when he is on the way to the customer. If such informed messages change too often, it will result in bad user experience.

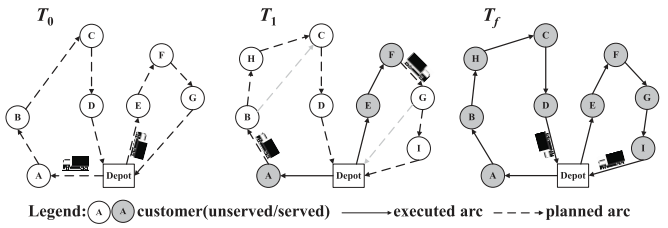

Fig. 1. Example of DVRP with two vehicles.

III. Set-Based Particle Swarm Optimization

Inspired by the behavior of bird flocking and fish schooling, Eberhart and Kennedy [27] first proposed the PSO algorithm in 1995. Since then, numbers of researches have been made to develop and improve PSO [37], [38]. Also various variants of PSO were proposed to solve different problems [39], [40].

A. Particle Swarm Optimization for Continuous Optimization

PSO maintains a population of particles. These particles are initialized randomly at different positions in the search space. Each position in the search space represents one solution. During the process of the algorithm, each particle i keeps track of a solution which is the best one it has achieved so far, called personal best, denoted as pBest_i. The whole swarm also keeps track of a solution which is the overall best, called global best, denoted as gBest. In each iteration, three steps are executed to update particles.

1) Velocity Updating: Each particle updates its velocity according to its own pBest_i and the gBest

$$V_i^d = \omega \cdot V_i^d + c_1 \cdot \text{rand}_1^d \cdot \left(\text{pBest}_i^d - X_i^d\right) + c_2 \cdot \text{rand}_2^d \cdot \left(\text{gBest}^d - X_i^d\right) \tag{8}$$

where ω is the inertia weight, c_1 and c_2 are the acceleration coefficients which are used to balance the self-cognition and the social influence; rand_1^d and rand_2^d are randomly generated within (0, 1), following the uniform distribution, and d represents the dth dimension of the problem.

2) Position Updating: Particles update their position according to their velocities

$$X_i^d = X_i^d + V_i^d. \tag{9}$$

3) Fitness Evaluation: All particles are evaluated to update their pBest values and the gBest value.

To increase the exploration ability of PSO, Liang *et al.* [41] proposed the comprehensive learning PSO (CLPSO) algorithm. In CLPSO, the velocity updating formula is modified as

$$V_i^d = \omega \cdot V_i^d + c \cdot \text{rand}^d \cdot \left(\text{pBest}_{f_i(d)}^d - X_i^d\right) \tag{10}$$

where c is the acceleration coefficient; $f_i(d) \in \{1, 2, \ldots, M\}$ (M is the swarm size) represents the particle's pBest of which i should learn from for the dth dimension and it is decided by

$$f_i(d) = \begin{cases} i, & \text{rand} < Pc_i \\ j, & \text{rand} \geq Pc_i \wedge f\left(\text{pBest}_j\right) < f\left(\text{pBest}_k\right) \\ k, & \text{rand} \geq Pc_i \wedge f\left(\text{pBest}_j\right) \geq f\left(\text{pBest}_k\right) \end{cases} \tag{11}$$

where rand is a random number within (0, 1), $f(X)$ represents the objective function value of the solution X (in this paper, all problems are assumed to be minimization problems), and Pc_i is a parameter calculated by

$$Pc_i = 0.05 + 0.45 \cdot \frac{\left(\exp\left(\frac{10(i-1)}{M-1}\right) - 1\right)}{\exp(10) - 1} \tag{12}$$

where M is the swarm size. When the random value is smaller than Pc_i, particle i will learn from its own pBest for the dth dimension. Otherwise, two candidates, j and k, will be selected randomly and particle i learns from the one with better pBest value.

B. Particle Swarm Optimization for Discrete Optimization

Originally, PSO is proposed to deal with continuous optimization problems. To extend its usage on solving COPs, Chen *et al.* [28] redefined the algorithm based on set, leading a novel algorithm called set-based PSO.

Suppose we have a COP which is defined as (S, f, Ω), where S is the search space of the problem, f is the objective function, and Ω is the set of constraints. According to the definition of the problem, we can divide S into D dimension. Each dimension is essentially a set $S_d \subseteq S$, where $S = S_1 \bigcup S_2 \bigcup \ldots \bigcup S_D$. Each solution X is a combination of elements from different dimensions, $X = X_1 \bigcup X_2 \bigcup \ldots \bigcup X_D$, where $X_d \subseteq S_d$. If X satisfies all constraints in Ω, it is considered to be feasible. Then the objective of S-PSO is to find a feasible solution X^* (a subset of S) that minimizes the objective function f.

The architecture of S-PSO is actually identical with those PSO variants used for continuous optimization problems. However in S-PSO, the particle's position is represented by set, and the velocity is represented by set with possibilities instead of numerical digits. Thus, the operators in (8)–(10) are also redefined on set and possibility. Details about how to represent the solutions of VRP by sets and the definitions of the operators used in S-PSO-D will be described in the next section. In addition, in this paper, S-PSO-D uses the learning strategy of CLPSO to prevent premature convergence.

IV. S-PSO-D-Based Dispatching System

In this paper, we build a dispatching system based on the DCVRP model. The system framework is shown in Fig. 2,

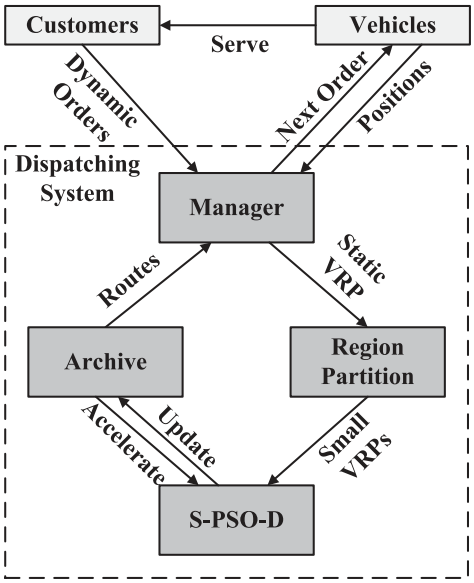

Fig. 2. Architecture of the dispatching system.

where the system runs in a centralized way and consists of four parts: 1) a manager; 2) a region partition component; 3) an optimizer which using the S-PSO-D algorithm; and 4) an archive. The manager takes charge of collecting orders and creating static CVRPs according to the information of current orders and vehicle positions. Then these static CVRPs are divided into smaller CVRPs by the component of region partition. Afterward, the optimizer solves these small CVRPs by taking advantage of the historical information stored in the archive. Also it will update the archive using the latest solutions it achieved. Finally routes will be sent from the archive back to the manager and used to navigate vehicles.

A. Manager

The fundamental idea of the dispatching system is periodic reoptimization. Before the beginning of the working day, the optimizer makes the first optimization for all static orders to get an initial solution which is a set of routes. Then, static CVRPs are periodically generated by the manager according to the status at that time, and solved by the optimizer. The process each time the optimizer solving a static CVRP is actually a reoptimization process for the whole DCVRP. Each reoptimization process can be triggered by either event such as a certain number of new orders coming which is known as decision epochs [42], or fixed time interval which is known as time slices [43]. For the manager in this paper, the method of time slice is applied, which is illustrated in Fig. 3.

First, we divide the planning horizon T into p time slices with equal length of T/p. Before T_0, a static CVPR is created for all static orders. This static problem will be solved within the period *ts*. *ts* can be long, since vehicles have not started working yet and there are enough time to find a good solution originally. Then at the end point of each slice T_i ($i = 1, 2, \ldots, p$), the information about vehicles (including their current locations, next customers, and remaining capacities) and orders (including the orders served in $[T_{i-1}, T_i]$ and

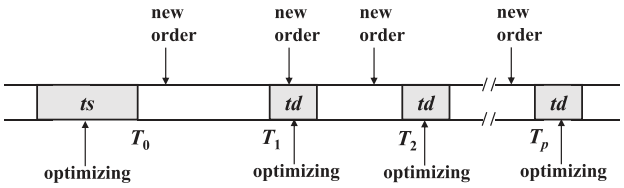

Fig. 3. Periodic reoptimization through time slice method.

the orders revealed in $[T_{i-1}, T_i]$) is collected by the manager. Accordingly, a special static VRP is created, where vehicles are with heterogeneous capacities and locations. This problem is time-dependent and needs to be solved within a relatively short period *td*. If the optimizing procedure takes too much time to obtain a result, the result would be outdated, such that the manager cannot control the vehicles in real-time.

After the manager receives the result from the optimizer, there are two methods to control the vehicles in real-time. The first one is a passive method that each time a vehicle finishes an order, it asks the manager for the location of the next order. The second one is an active method that the manager broadcasts the routes scheduled for all vehicles and each vehicle serves the customers according to the latest routes assigned to it. In our system, the passive method is recommended, because whenever a vehicle calls for a new order, the system can know that it has finished the last order without checking its position. In addition, if a vehicle has served all the orders assigned to it, it will stay at the location of the last order rather than returning back. Only if the last order is the final order of the result of the final optimization, it is allowed to return to the depot.

B. Region Partition

Region partition is used to divide a big static CVRP into several small static CVRPs which are independent of each other. This process is based on the map information and customers' locations. As an NP-hard problem, the CVRP's complexity grows exponentially with its scale. Thus, dividing a big CVRP into small ones can decrease the difficulty of the problem effectively, and the small CVRPs can be optimized in parallel so that the computing time can be decreased. Besides, in real applications, a big city or a big area is usually divided and managed in different regions. Thus region partition is helpful and applicable in both theory and practice.

Once the region partition component receives the static CVRP with all static orders, the K-means algorithm [44] is applied to group orders into different clusters and all centers are recorded to group the following orders. A cluster represents a small CVRP. As for the number of clusters, if it is hard to see how many clusters in the map clearly, it is suggested to divide the orders into n_c clusters according to load balance that each cluster owns roughly three vehicles

$$n_c = \left(\sum_{i=0}^{n_s} q_i\right)/Q/3 + 1 \tag{13}$$

where n_s is the number of static orders.

Except the first static CVRP, the other CVRPs contain not only different kinds of orders but also vehicles with different locations and capacities. Thus for the static problems generated during the working day, region partition acts on both orders (grouping the orders according to the known centers) and outside vehicles (assigning the outside vehicles to different small problems).

C. S-PSO-D With Archive

The core of the system is the S-PSO-D algorithm which is a variant of S-PSO designed for DCVRP. Meanwhile, in order to gain a fast convergence speed to obtain a relatively good solution within the limited time, an archive strategy and a local refinement algorithm are incorporated. Archive strategy is used in the process of velocity updating to make full use of the evolution experience of the previous optimizations. Local refinement is used to modify solutions slightly in order to make further improvement. Since these two techniques directly affects the execution process of S-PSO-D, they will be introduced along with the description of the S-PSO-D algorithm.

Besides the basic four steps: 1) initialization; 2) velocity updating; 3) position updating; and 4) fitness evaluation [28], two more steps: 1) local refinement and 2) archive updating, are adopted in S-PSO-D. Following, we will demonstrate the S-PSO-D from four aspects: 1) particle representation; 2) velocity updating; 3) position updating; and 4) local refinement.

1) Particle Representation: The solution of a static CVRP is a set of routes. For the first static CVRP which consists of all static orders, each route starts and ends with the depot c_0. For the other CVPRs generated during the working day, routes may start at different locations but still end with c_0. Thus the first CVRP can be seen as a special case of the CVRPs generated during the day. We take a solution of a static CVRP which is generated during the working day as an example to show the encoding scheme in S-PSO-D. Suppose there are ten orders $\{c_1, c_2, \ldots, c_{10}\}$ unserved and two outside vehicles which are now at the locations of c_{11} and c_{12} (or on their ways to c_{11} and c_{12}), respectively. One hypothetical solution is shown in Fig. 4. Besides the two outside vehicles, two more vehicles will depart from the depot.

In S-PSO-D, the position of a particle is represented by

$$X_i = \left[X_i^0, X_i^1, \ldots, X_i^n\right] \tag{14}$$

$$\begin{aligned} X_i^d &= \{\langle d, nb_1\rangle, \langle d, nb_2\rangle, \ldots, \langle d, nb_x\rangle\} \\ &nb_1, \ldots, nb_x \in \{0, 1, \ldots, n\} \\ \forall j \quad &\forall k \in \{1, \ldots, x\}, \text{ if } j \neq k, nb_j \neq nb_k \\ &\forall l \in \{1, \ldots, x\}, nb_l \neq d \end{aligned} \tag{15}$$

where X_i is an arc set. Each dimension X_i^d of X_i represents the set of arcs associated with the vertex c_d, and x is the number of arcs. n is the number of customers involved in the problem which is equal to 12 in the example. For an unserved order like c_1, it has two neighbors, c_{12} and c_8, thus $x = 2, X_i^1 = \{\langle 1, 12\rangle, \langle 1, 8\rangle\}$. For c_0, usually it has more than two neighbors, unless there is only one vehicle, in the example

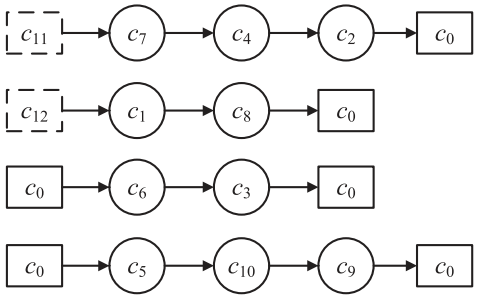

Fig. 4. Example of solution.

$x = 6, X_i^0 = \{\langle 0, 2\rangle, \langle 0, 8\rangle, \langle 0, 6\rangle, \langle 0, 3\rangle, \langle 0, 5\rangle, \langle 0, 9\rangle\}$. For an outside vehicle's location like c_{11}, it has only one neighbor c_7, thus $x = 1, X_i^{11} = \{\langle 11, 7\rangle\}$. Through this encoding scheme, any route can be built from the starting point step by step, whether or not it begins with c_0.

Accordingly, the velocity of a particle is represented by

$$V_i = \left[V_i^0, V_i^1, \ldots, V_i^n\right] \tag{16}$$

$$V_i^d = \{\langle d, u\rangle \backslash p(d, u) | u = 0, 1, \ldots, d-1, d+1, \ldots, n\} \tag{17}$$

where V_i is an arc set with possibilities. The number of elements in each dimension V_i^d will not excess n. For each arc $\langle d, u\rangle$, $p(d, u)$ is the corresponding possibility which will be used for element selection in the process of position updating. Details about how it affects the selection operation will be introduced in the following sections. In implementation, to reduce the number of elements in velocity and improve execution efficiency, if $p(d, u) < \varepsilon$ (ε is a very small value which is set artificially), $\langle d, u\rangle$ will be omitted from V_i^d.

2) Archive-Based Velocity Updating: The solution archive plays an important role in the process of velocity updating. Since each static CVRP generated during the working day can be seen as an extension of the previous one, the pBest solutions of the last static CVRP are valuable to be learned from. Thus all pBest solutions and their fitness generated during the last optimization are stored in the archive, e.g., the archive consists of solutions generated at T_{i-1} when the optimizer is currently working on the problem generated at T_i. We call this learning strategy "inheritance" in this paper. Although new orders received during $[T_{i-1}, T_i]$ do not appear in the solutions generated at T_{i-1}, learning from the archive can still rapidly determine the sequence of previous orders, thus accelerating the convergence speed of the algorithm.

Suppose there are M particles in the swarm for every optimization and the solutions in archive are denoted as $\{\text{aBest}_1, \text{aBest}_2, \ldots, \text{aBest}_M\}$. As we have mentioned earlier, CLPSO uses a parameter Pc_i and the tournament selection strategy to decide the particle to be learned from. In S-PSO-D, we employ another parameter Pca to decide whether to learn from the archive or not. The velocity updating rule is modified as

$$V_i^d = \omega \cdot V_i^d + c \cdot \text{rand}^d \cdot \left(\text{LF}_{fi(d)}^d - X_i^d\right) \tag{18}$$

where LF means "learn from" and it is decided by both Pc_i and Pca

$$\text{LF}_{fi(d)}^d = \begin{cases} \text{pBest}_i^d, & \text{rand}_1 < Pc_i \\ \text{pBest}_j^d, & \text{rand}_1 \geq Pc_i \wedge \text{rand}_2 \geq Pca \wedge f(\text{pBest}_j) < f(\text{pBest}_k) \\ \text{pBest}_k^d, & \text{rand}_1 \geq Pc_i \wedge \text{rand}_2 \geq Pca \wedge f(\text{pBest}_j) \geq f(\text{pBest}_k) \\ \text{aBest}_j^d, & \text{rand}_1 \geq Pc_i \wedge \text{rand}_2 < Pca \wedge f(\text{aBest}_j) < f(\text{aBest}_k) \\ \text{aBest}_k^d, & \text{rand}_1 \geq Pc_i \wedge \text{rand}_2 < Pca \wedge f(\text{aBest}_j) \geq f(\text{aBest}_k) \end{cases} \tag{19}$$

where rand_1 and rand_2 are both random numbers within (0,1); j and k are two candidates in the tournament selection.

Following, the operators used in (18) are defined in (20)–(23), respectively. *coefficient* $\cdot$ *velocity* and *velocity* + *velocity* are defined as the changing of the possibilities of arcs, shown as (20) and (21). *position* − *position* is defined as the subtraction of two arc sets (22). *coefficient*·(*position*−*position*) is defined as turning an arc set into a set with possibilities

$$c \cdot V_i^d = \left\{\langle d, u\rangle \backslash p'(d, u) \middle| u = 0, 1, \ldots, d-1, d+1, \ldots, n\right\} \tag{20}$$

where $p'(d, u) = c \cdot p(d, u)$

$$V_i^d + V_j^d = \left\{\langle d, u\rangle \backslash p'(d, u) | u = 0, 1, \ldots, d-1, d+1, \ldots, n\right\} \tag{21}$$

where $p'(d, u) = \max(p_i(d, u), p_j(d, u))$

$$X_i^d - X_j^d = U^d = \left\{\langle d, u\rangle | \langle d, u\rangle \in X_i^d \wedge \langle d, u\rangle \notin X_j^d\right\} \tag{22}$$

$$c \cdot U^d = \left\{\langle d, u\rangle \backslash p'(d, u) \middle| \langle d, u\rangle \in U^d\right\} \tag{23}$$

where $p'(d, u) = c$.

Reusing the example in Fig. 4, we already know $X_i^1 = \{\langle 1, 12\rangle, \langle 1, 8\rangle\}$. Suppose that $V_i^1 = \{\langle 1, 3\rangle \backslash 0.5, \langle 1, 4\rangle \backslash 0.3\}$, $\text{LF}_{f_i\ (1)}^1 = \{\langle 1, 12\rangle, \langle 1, 7\rangle\}$, $\omega = 0.9$, $c = 2.0$, and $\text{rand}^1 = 0.4$. Then we have $\omega \cdot V_i^1 = \{\langle 1, 3\rangle \backslash 0.45, \langle 1, 4\rangle \backslash 0.27\}$, $\text{LF}_{f_i\ (1)}^1 - X_i^1 = \{\langle 1, 7\rangle\}$, and $c \cdot \text{rand}^1 \cdot (\text{LF}_{fi(1)}^1 - X_i^1) = 2.0 \cdot 0.4 \cdot \{\langle 1, 7\rangle\} = \{\langle 1, 7\rangle \backslash 0.8\}$. Finally we get $V_i^1 = \{\langle 1, 3\rangle \backslash 0.45, \langle 1, 4\rangle \backslash 0.27, \langle 1, 7\rangle \backslash 0.8\}$. Through learning from $\text{LF}_{f_i\ (1)}^1$, a new arc $\langle 1, 7\rangle$ is added to V_i^1 and the possibilities of the original two arcs are decreased due to the inertia weight ω.

3) Position Updating: Positions are still updated according to (9), yet the related operations are redefined. First, velocity V_i is converted into a crisp set by

$$\text{Cut}\left(V_i^d\right) = \left\{\langle d, u\rangle | \langle d, u\rangle \backslash p(d, u) \in V_i^d \wedge p(d, u) \geq \text{rand}\right\} \tag{24}$$

where rand is a random number within (0,1). In this way, arcs with smaller possibilities are eliminated and those with larger possibilities are more likely to be chosen.

Then, we build new particles in a constructive way where the capacity constraint is taken into account, so that every particle is built to be legal. The flow chart of the position updating process is shown in Fig. 5. To begin with, a new solution (X_i)' is initialized with k routes, where k is the number of the outside vehicles. Each route begins with a vertex where an outside vehicle is. Considering the example in Fig. 4, $(X_i)'$ is initialized with two routes, starting with c_{11} and c_{12}, respectively. Afterward, each route will be built by finding the next vertex

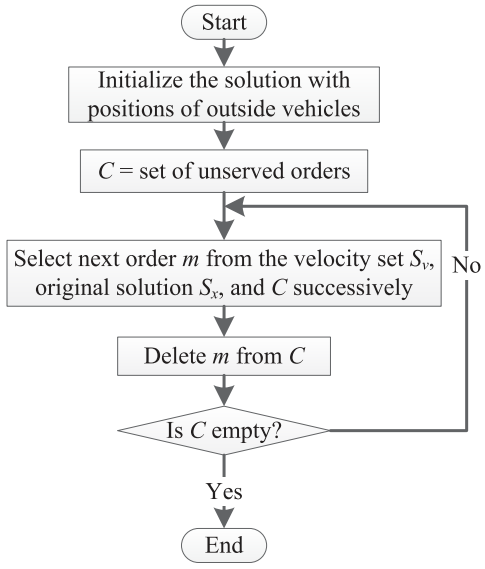

Fig. 5. Flow chart of position updating.

step by step. Suppose the algorithm is finding the next vertex for c_d. According to the construction rule of S-PSO, the next vertex m may come from one of the following three crisp set.

1) $S_V^d = \{m|\langle d, m\rangle \in \text{Cut } (V_i^d)\}$, the subset of the velocity.
2) $S_X^d = \{m|\langle d, m\rangle \in X_i^d\}$, the set of original X_i.
3) C, the set of all vertices.

If there are vertices which satisfy the capacity constraint available in S_V, m is selected from S_V. If we cannot find feasible vertex in S_V, m is selected from S_X. If S_X does not contain any feasible vertex either, m is finally selected in C. If the selected vertex is not the depot, it will be inserted into the current route. Otherwise, we start to build the next route. In the example, after constructing the first route (11, 7, 4, 2, 0), the second route is built form the starting vertex c_{12}. If all routes associated with outside vehicles are constructed successfully and there are still unplanned orders, new route will be built from the depot c_0, as the third and the fourth route shown in the example. If all orders are planned, the whole solution is built successfully.

In canonical S-PSO, if several vertices are available, the algorithm will randomly choose one. However, in S-PSO-D, to accelerate the convergence speed, a pseudorandom selection method is applied. When facing multiple choices $\{m_1, m_2, \ldots, m_H\}$, the algorithm with a certain probability will directly choose the nearest customer

$$m = \begin{cases} m_a|a\forall i \in \{1, \ldots, H\} \text{ s.t. } \text{dis}(d, m_a) \le \text{dis}(d, m_i), \\ \text{rand}_1 < Pcg \\ m_b|b = \lceil \text{rand}_2 \cdot H \rceil, \ \text{rand}_1 \ge Pcg \end{cases} \tag{25}$$

where Pcg is the certain probability and rand_1 and rand_2 are two random numbers within (0,1).

4) Local Refinement: A newborn solution before evaluated will always be refined to make a further improvement. In S-PSO-D, the local refinement process consists of two steps: 1) refinement among routes and 2) refinement among orders. Refinement among routes is used to merge routes in order to decrease the number of vehicles. Refinement among orders is used to adjust the sequence of customers within a route in order to decrease the total travel distance of the route.

Algorithm 1 Local Refinement Among Routes

Input: original solution X_i, load of each route $\{l_1, l_2, \ldots, l_n\}$, number of outside vehicle k, capacity Q
Output: refined solution X_i

```
1   fr = 1, sr = k + 1, changed = true;
2   while (changed) do
3     for i = 2 to k do
4       if l_i < l_fr then fr = i;
5     for i = k + 2 to n do
6       if l_i < l_sr then sr = i;
7   if (l_fr + l_sr < Q)
8         combine the fr route with the sr route;
9         n = n − 1; l_fr = l_fr + l_sr;
10    else changed = false;
11  fr = k + 1, sr = k + 2, changed = true;
12  if (l_fr > l_sr) then swap fr with sr;
13  while (changed) do
14      for i = k + 3 to n do
15        if (l_i < l_sr)
16          sr = i;
17          if (l_fr > l_sr) then swap fr with sr;
18  if (l_fr + l_sr < Q)
19    combine the fr route with the sr route;
20    n = n − 1; l_fr = l_fr + l_sr;
21  else changed = false;
```

Pseudocode of the refinement among routes is shown in Algorithm 1. First, we check whether there is a route starting with c_0 that can be merged into a route of an outside vehicle while satisfying the capacity constraint. If there is a pair of such routes, we merge them into a single route which still starts with the outside vehicle's position, by simply splicing in line. This process is shown in lines 1–10. The sequence of vertices in the new route is unconcerned since we have the process of the refinement among orders which is responsible for reordering the vertices. In Fig. 4, assume the third route can be merged into the second route. We merge them into one route which still starts with c_{12}, and Fig. 6 shows how they merge into one route. Second, we check whether there are two routes both starting with c_0 that can be merged. If so, merge them using the same method in lines 11–21 and the sequence of these two routes is also unconcerned.

For the refinement among orders, the two-opt method is employed [45]. According to [46], for a route with 100 or fewer vertices, the two-opt method has basically the same performance with the three-opt method. Moreover, two-opt is much faster than three-opt. Thus it is chosen as the refinement tech among orders.

V. Experiments

In this section, the proposed dispatching system is sufficiently tested on the datasets with different scales. First, the experimental data is generated and experimental settings are introduced in detail. Then, effects of the region partition and

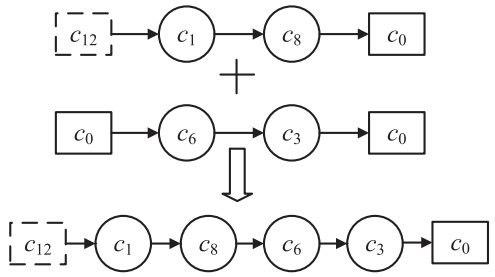

Fig. 6. Merging two routes into one.

the archive strategy are discussed separately. Afterward, the S-PSO-D approach is compared with another two approaches to show its effectiveness. In the end, some specially designed experiments are conducted to explore the essence of the DCVRP and the innate character of the reoptimization framework.

TABLE I
DCVRP INSTANCES

Fisher		
F-72-H	F-72-N	F-72-U
F-135-H	F-135-N	F-135-U
Taillard		
T-75-H	T-75-N	T-75-U
T-100-H	T-100-N	T-100-U
T-150-H	T-150-N	T-150-U
T-385-H	T-385-N	T-385-U
Clustered		
C-200-H	C-200-N	C-200-U
C-250-H	C-250-N	C-250-U
C-300-H	C-300-N	C-300-U
C-350-H	C-350-N	C-350-U
C-400-H	C-400-N	C-400-U
Random		
R-200-H	R-200-N	R-200-U
R-250-H	R-250-N	R-250-U
R-300-H	R-300-N	R-300-U
R-350-H	R-350-N	R-350-U
R-400-H	R-400-N	R-400-U

A. Benchmark Description and Parameter Settings

1) Benchmark Description: The experimental data consists of two parts. The first part is generated based on two well-known static CVRP benchmarks which were first used by Taillard [47] and Fisher and Jaikumar [48]. As there is a big gap in the scale of these two benchmarks, some other DCVRP instances are generated with scales from 200 to 400, and both randomly distributed locations and clustered locations are considered. To obtain dynamic problems, the following features are added to the static benchmarks.

1) *Level of Dynamism:* Identical to [21], in this paper, the level of dynamism considered is also 0.5, which means for each static instance, half of the orders are randomly chosen as dynamic orders. It should be noted that different scenarios have different levels of dynamism. Setting this value to 0.5 is just for the following experiments.
2) *Length of the Working Day:* As the level of dynamism is set to 0.5, the length of the working day is set as two times of the planning horizon, which means that all dynamic orders planned in the same day are submitted in the first half of the working day. In this paper, the planning horizon T is set to 100, and the working day is 200 (these two values are only used for calculation which have no realistic meaning).
3) *Disclosure Time of Orders:* In real applications, orders are not always coming uniformly. The case that lots of orders are submitted within a very short period occurs frequently. Thus, three different distributions of disclosure time are considered in this paper, uniform distribution, normal distribution, and half normal distribution. For the uniform distribution, each dynamic order's disclosure time t_i is randomly generated within (0, 100); for the normal distribution, t_i is generated following the normal distribution $N(50, 17.5)$ which is also within (0, 100); for the half normal distribution, t_i is first generated according to $N(100, 34)$. If it is bigger than 100, $t_i = 200 - t_i$.
4) *Vehicle Speed:* Before setting the vehicle speed, a greedy method is applied on each static instance to get a solution. Then we calculate the vehicle speed according to the length of the working day and the longest route in the solution.

With the four new features, a DCVRP dataset with 48 instances are generated, which are shown in Table I. Based on the aforementioned features, a nomenclature "name-size-distribution" is used to identify each dynamic instance. For example, F-72-U means the instance which is generated from the Fisher's dataset with 72 orders and with disclosure time following the uniform distribution. Among the four datasets, the clustered dataset and the random dataset are newly generated. The location distribution of customers of each instance is shown in Fig. 7. As the instances in two new datasets are generated in the same pattern, respectively, only one instance for each dataset is illustrated. From Fig. 7, we can see that the benchmark dataset has covered multiple types of location distributions, including clustered distribution, random distribution, clustered-random-mixed distributions, and distributions with strange shapes.

2) Parameter Settings: For all instances, the planning horizon is divided into ten pieces. Here, we do not discuss how much pieces is appropriate, because such a problem is instance-dependent and several works have studied on it [21]. The number of the generation is set as three times of total order quantity for the first optimization of all static orders and set equal to the order amount for the other optimizations of dynamic orders.

As mentioned before, the framework of S-PSO-D is CLPSO. In the experiments, the inertia weight ω is initialized as 0.9 and linearly decreased to 0.4 at the end of the optimization. c in (10) is set to 2 and the refreshing gap in CLPSO is set to 4. The population size is set to 20. ε is set to 0.001. These parameters are set according to [28]. Then Pca is set to 0.5 initially and linearly decreased to 0 at the end of the optimization, and Pcg is set to 0.9. Since some parts of the

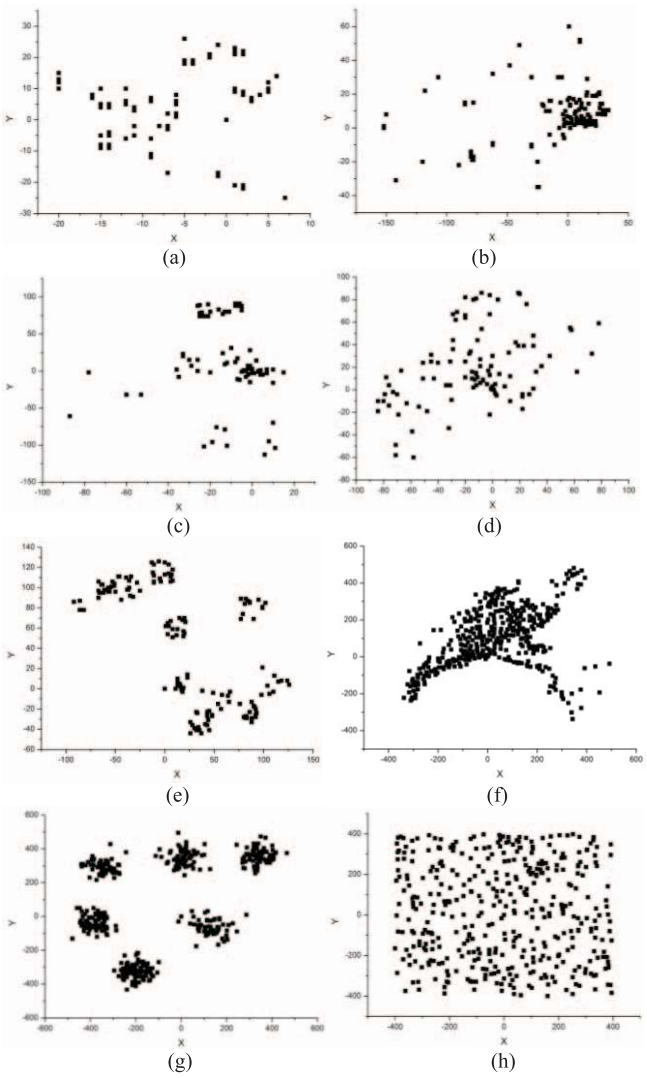

Fig. 7. Distribution of customers' locations. (a) F-72. (b) F-135. (c) T-75. (d) T-100. (e) T-150. (f) T-385. (g) C-400. (h) R-400.

following experiment are specially designed, several parameters may change. The changed parameters will be shown in the corresponding sections.

To verify the effectiveness of the proposed S-PSO-D approach completely, it is compared with the ACS approach [21] and an insertion approach [14], [16] in Section V-D. Like the S-PSO-D approach proposed in this paper, the ACS approach also utilizes the periodic reoptimization framework and metaheuristic algorithm. Moreover, the ACS algorithm shares a nearly equal time complexity with the S-PSO algorithm [28]. Thus it is chosen as the comparing approach. To make a fair comparison, the local refinement process is also added into the ACS approach. Parameters in the ACS approaches are set according to [21]: $q_0 = 0.9$, $\beta = 1$, $\rho = 0.1$, $\gamma_r = 0.3$, and $\tau_0 = 1/(n \times \text{Cost(PI)})$ is calculated based on the greedy method. The colony size is also set to 20. In the insertion approach, the first static problem which contains all static orders is solved by the ACS algorithm to get an initial solution. Then, whenever a new order comes, it will be inserted into the solution based on the nearest

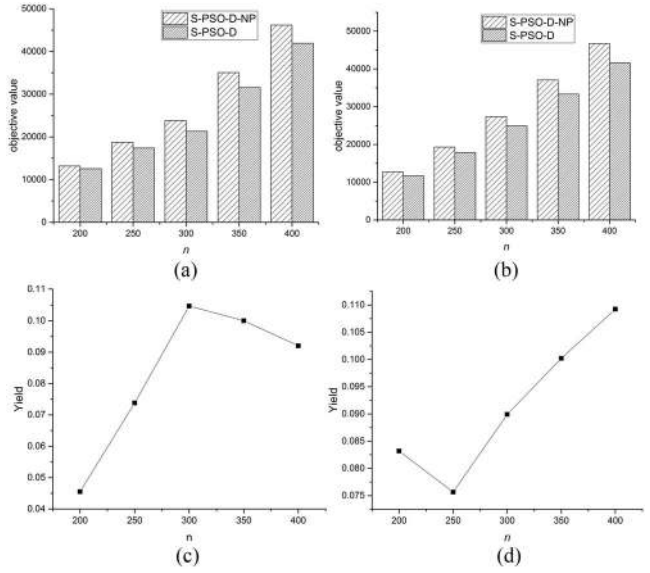

Fig. 8. Results of region partition. n represents the order amount. (a) and (c) C-n-U. (b) and (d) R-n-U.

insertion heuristic [45]. As a deterministic approach, its time complexity is lower than the other two approaches. However, its performance is much worse at the same time. Thus, it can be taken as the baseline.

Thirty independent runs are executed for every tested approaches to get statistic results such as mean value, median value and value of Wilcoxon rank sum test.

B. Region Partition

To show that region partition is useful for a dispatching system in handling large-scale problems, a special dispatching system without region partition technique is created, which is denoted as S-SPO-D-NP. The original dispatching system proposed in this paper is still denoted as S-PSO-D. Ten instances with medium or large scale are selected as the test cases, i.e., C-200-U, C-250-U, C-300-U, C-350-U, C-400-U, R-200-U, R-250-U, R-300-U, R-350-U, and R-400-U.

Results are shown in Fig. 8. In addition, the yield rate brought by the region partition technique is also shown in Fig. 8, which is calculated by

$$\text{yield} = \frac{f_{S-\text{PSO}-D-\text{NP}} - f_{S-\text{PSO}-D}}{f_{S-\text{PSO}-D-\text{NP}}} \tag{26}$$

where $f_{S-\text{PSO}-D-\text{NP}}$ represents the mean value of the S-PSO-D-NP approach and $f_{S-\text{PSO}-D}$ represents the mean value of the S-PSO-D approach.

Seeing from Fig. 8(a) and (b), we can find that S-PSO-D yielded better objective values than S-PSO-D-NP on all of the tested instances, which implies that the region partition technique used in the proposed dispatching system is really effective. According to Fig. 8(c), it cuts down the total distance at most 10.5% for the C-300-U instance, and at least 4.5% for the C-200-U instance. According to Fig. 8(d), it cuts down the total distance at most 11% for the R-400-U, and at least 7.5% for the R-250-U. The numeric results show

TABLE II
COMPARISON OF THE EFFECT OF THE ARCHIVE STRATEGY

Instance	Approach	n	$2{\cdot}n$	$3{\cdot}n$
F-72-U	S-PSO-D	19	18	16
	S-PSO-D-NA	11	12	14
F-135-U	S-PSO-D	20	15	12
	S-PSO-D-NA	10	15	18
T-75-U	S-PSO-D	18	15	10
	S-PSO-D-NA	12	15	20
T-100-U	S-PSO-D	15	15	10
	S-PSO-D-NA	15	15	20

n is the number of orders.

that the region partition technique used in the proposed dispatching system is comprehensively useful for both randomly distributed instances and clustered instances. But comparing the curves shown in Fig. 8(c) and (d), we can see that its effect still can be influenced by the location distribution. For instances with randomly distributed locations, except the R-200-U, the yield rate increases with the growing of the problem scale. For instances with clustered locations, the yield rate first increases along with the scale, but after the scale reaching 300, it stops to decline. Since the specific relationship between the location distribution and the region partition technique is complicated to figure out, we take it as an open issue and will study it in future works. Overall, experimental results show that the region partition is useful in handling large-scale DCVRPs.

C. Archive Strategy

To show the effect of the archive strategy, a special S-PSO-D without archive-based velocity updating is designed to make a comparison, which is denoted as S-PSO-D-NA. To avoid the interference of the local refine and the region partition techniques, they are removed from the dispatching system for both S-PSO-D and S-PSO-D-NA. However without region partition, the approaches' performance varies so much each time on large-scale problems that we cannot obtain stable results. Thus four small instances which do not need to be partitioned are taken for the test: 1) F-72-U; 2) F-135-U; 3) T-75-U; and 4) T-100-U.

First of all, the static optimization generated before T_0 is executed. Then, based on the same result of the first static optimization, two algorithms are, respectively, applied to the first dynamic problem, and their convergence rates are collected to show the effect of the archive. Due to the way we design the experiment, only instances with uniform-distributed disclosure time are chosen. For the other two distributions, few dynamic orders appear in $[T_0, T_1]$, so the problem generated at T_1 is too similar to the static one generated before T_0.

For both first optimization of all static orders and first optimization of dynamic orders, the number of generations is set as three times of total order quantity. *Pcg* is set to 0. The other parameters are kept unchanged. The results about how many times one approach beats the other among 30 runs are shown in Table II.

According to the data form Table II, in most cases, S-PSO-D-NA is better than S-PSO-D when they both executes

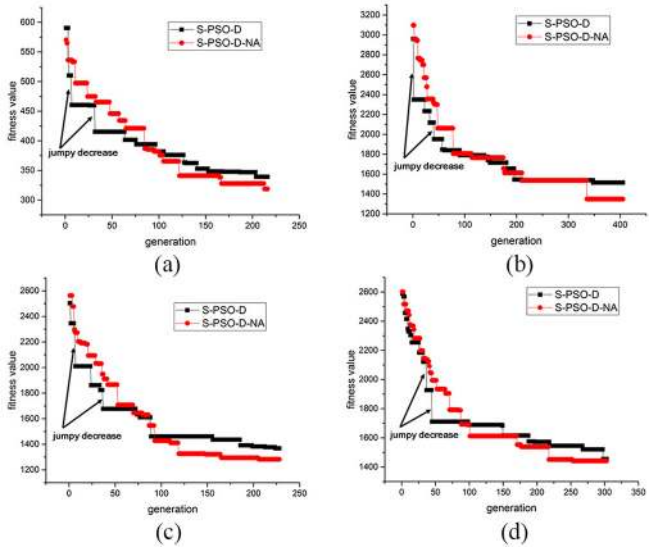

Fig. 9. Convergence rates of (red) S-PSO-D and (black) S-PSO-D-NA. (a) F-72-U. (b) F-135-U. (c) T-75-U. (d) T-100-U.

$3 \cdot n$ generation, except for F-72-U. However, as we have mentioned earlier, in the dispatching system for DVRP, real-time control is a crucial demand which means we do not have much time to wait for a perfect solution in dynamic environment. Decisions should be made in short time. In such situation, the effect of the archive strategy shows up. We can find that before n generations, the number of times where S-PSO-D obtains better results is greater than or equal to the number of times where S-PSO-D-NA obtains better results. The data about their convergence rates displayed in Fig. 9 also show us the similar phenomenon. Due to the archive strategy, S-PSO-D gains a faster convergence speed at the early stage of the optimization. Every time it learns something helpful from the archive, we can see a "jumpy decrease" on its fitness value. By contrast, the descent curve of S-PSO-D-NA appears smoother than S-PSO-D. These results imply that though the archive strategy may have negative effect during a long-term optimization, it truly has accelerated the algorithm's convergence speed in the early stage. Considering the demand of real applications, we think that the archive is effective and should be employed as a component of the dispatching system.

D. S-PSO-D Performance

S-PSO-D is compared with ACS and the insertion method on all instances. Besides the best value and the mean value, a Wilcoxon rank sum test is also performed to check whether the S-PSO-D approach is significantly better or worse than the other two approaches. Numerical results are shown in Table III.

First, we check the experimental results generally. Seeing from the p-value of the Wilcoxon rank sum test, S-PSO-D is significantly better than ACS on 29 instances, and ACS performs significantly better than S-PSO-D on four instances. On the rest of the instances these two approaches have similar performances. Observing the instances on which the ACS approach surpasses the other two approaches, we find that their

TABLE III
COMPARISON OF THE TOTAL DISTANCE ON THE 48 INSTANCES

	Mean	Best	Wilcoxon	Mean	Best	Wilcoxon	Mean	Best	Wilcoxon
instance	F-72-H			F-72-N			F-72-U		
S-PSO-D	305.49	295.03		283.65	277.99		300.86	275.15	
ACS	303.15	296.05	0.0501	278.17	263.83	0.0212#	301.43	271.56	0.8935
Insertion	452	452	3.54e-13*	462.93	462.93	3.46e-13*	487	487	3.53e-13*
instance	F-135-H			F-135-N			F-135-U		
S-PSO-D	1660.15	1552.51		1629.22	1498.14		1691.57	1590.41	
ACS	1640.19	1526.71	0.113	1594.43	1462.76	0.0145#	1678.34	1565.34	0.3346
Insertion	2322.21	2268.39	1.14e-11*	2312.11	2246.86	9.24e-12*	2297.69	2268.39	4.1e-12*
instance	T-75-H			T-75-N			T-75-U		
S-PSO-D	1630.41	1544.18		1583.59	1498.41		1615.97	1542.54	
ACS	1635.29	1558.65	0.927	1564.41	1484.59	0.161	1585.37	1527.62	0.006#
Insertion	1934.4	1829.94	1.41e-11*	1979.73	1826.4	1.43e-11*	1948.77	1803.63	1.42e-11
instance	T-100-H			T-100-N			T-100-U		
S-PSO-D	2225.71	2030.23		2085.1	1917.39		2126.77	1990.62	
ACS	2169.23	2050.52	0.0097#	2053.03	1842.25	0.0951	2103.43	1929.16	0.2628
Insertion	2806.64	2586.68	1.43e-11*	2778.53	2578.84	1.43e-11*	2883.96	2656.19	1.43e-11*
instance	T-150-H			T-150-N			T-150-U		
S-PSO-D	3749.66	3604.82		3625.15	3467.32		3693.09	3508.15	
ACS	3769.65	3613.8	0.5683	3645.68	3469.81	0.5125	3781.62	3594.12	0.0052*
Insertion	5010.63	4681.01	1.43e-11*	4918.99	4549.65	1.44e-11*	4948.17	4522.63	1.44e-11*
instance	T-385-H			T-385-N			T-385-U		
S-PSO-D	31098.7	30378		30148.4	29253.2		30813.5	30100.1	
ACS	32722.1	31494.3	4.54e-11*	32673.7	31316.1	1.43e-11*	33093	31976.4	1.44e-11*
Insertion	51745.5	48533.3	1.44e-11*	50635.5	46482.4	1.44e-11*	50823.5	47552.7	1.44e-11*
instance	C-200-H			C-200-N			C-200-U		
S-PSO-D	12227.5	11901.8		11905.3	11538.6		12246.6	11636	
ACS	12292.4	11922.7	0.6993	11812.1	11267.3	0.3007	12178.7	11861.7	0.1537
Insertion	19638.6	19139	1.63e-14*	19477.3	18382.7	3.02e-11*	19480.7	18653.2	3.01e-11*
instance	C-250-H			C-250-N			C-250-U		
S-PSO-D	17001.1	16253.5		16591.7	15904.9		17214	16172.1	
ACS	17264	16256.3	0.0122*	16962.7	16348.1	1.04e-4*	17516.7	16654.2	0.0089*
Insertion	30175.2	28813.7	3.02e-11*	29900.8	28756.2	3.02e-11*	30027.5	28453	3.02e-11*
instance	C-300-H			C-300-N			C-300-U		
S-PSO-D	21215.6	20603.8		20655.6	20102.5		21257.4	20504.6	
ACS	21496.3	20750.8	0.0091*	21394.1	20576.2	8.2e-7*	21929	21283.3	2.43e-5*
Insertion	34281.2	32744.2	3.02e-11*	34206.2	32640.9	3.02e-11*	34220.9	31787.2	3.02e-11*
instance	C-350-H			C-350-N			C-350-U		
S-PSO-D	31595.4	30723.7		31020.6	30291		31181.4	30238.4	
ACS	32104.7	31281	0.0026*	32048.7	30655.5	4.11e-7*	32406.3	31064	4.57e-9*
Insertion	56596.5	54899.7	3.02e-11*	56180.1	54334.7	3.02e-11*	56156.1	54651.2	3.02e-11*
instance	C-400-H			C-400-N			C-400-U		
S-PSO-D	41128.3	40258.9		40721.7	40118.2		41511.4	39946.3	
ACS	42845.2	41577.6	5.07e-10*	42667.3	41078.7	6.06e-11*	43527.8	42143.6	6.7e-11*
Insertion	72105	70366.2	3.02e-11*	72149.5	68520.7	3.02e-11*	72023.2	70314.3	3.02e-11*
instance	R-200-H			R-200-N			R-200-U		
S-PSO-D	11106.5	10864.4		10860.7	10098.4		11276.3	10816.9	
ACS	11117.2	10720.4	0.8418	11073.1	10571.9	0.0023*	11380.1	11026.6	0.1103
Insertion	16049.7	15305.2	3.02e-11*	16116.8	15433	3.02e-11*	16101.6	15128.8	3.02e-11*
instance	R-250-H			R-250-N			R-250-U		
S-PSO-D	17288.5	16418.5		16708.8	16217.8		17391	16812.7	
ACS	17877.2	17131.8	1.33e-6*	17443.5	16870.7	7.38e-10*	18018.4	17270.2	1.29e-6*
Insertion	28888.4	27678.1	3.02e-11*	28675.7	27526.1	3.02e-11*	28521.5	27514.2	3.02e-11*
instance	R-300-H			R-300-N			R-300-U		
S-PSO-D	24904.8	23834.3		23963.2	23062.1		24809.3	23857.3	
ACS	25375	24653.6	8.66e-5*	24873.5	24067.1	1.87e-7*	25684.9	24693.2	2.38e-7*
Insertion	40211.7	39050	3.02e-11*	39771.3	38119.5	3.02e-11*	39702.2	37912.7	3.02e-11*
instance	R-350-H			R-350-N			R-350-U		
S-PSO-D	32522.3	31350		31489.9	30540.5		32821.6	32204.9	
ACS	34585.5	33194.4	4.08e-11*	34442.5	33154.9	3.02e-11*	34930.4	33545.9	3.02e-11*
Insertion	53514.7	51127	3.02e-11*	53386.6	51063.3	3.02e-11*	53289.6	52109.7	3.02e-11*
instance	R-400-H			R-400-N			R-400-U		
S-PSO-D	40757.9	39798.2		39856.2	38016.2		41296.8	40152.4	
ACS	42735.2	41124.5	1.61e-10*	43183.1	41753.7	3.02e-11*	43570.2	41902.7	4.98e-11*
Insertion	66123.2	63833.9	3.02e-11*	66008.2	63863.8	3.02e-11*	65770.3	63376.1	3.02e-11*

#The results of S-PSO-D are significantly worse according to a Wilcoxon rank sum test at level 0.05;
*The results of S-PSO-D are significantly better according to a Wilcoxon rank sum test at level 0.05.

scales are relatively small, such as 72, 75, 100, 135, and 200. In contrast, the S-PSO-D approach seems more capable than ACS on large instances. However, an interesting phenomenon is that on three T-150 instances, S-PSO-D yields better results. Especially for T-150-U, S-PSO-D is significantly better than ACS based on the *p*-value of the Wilcoxon rank sum test.

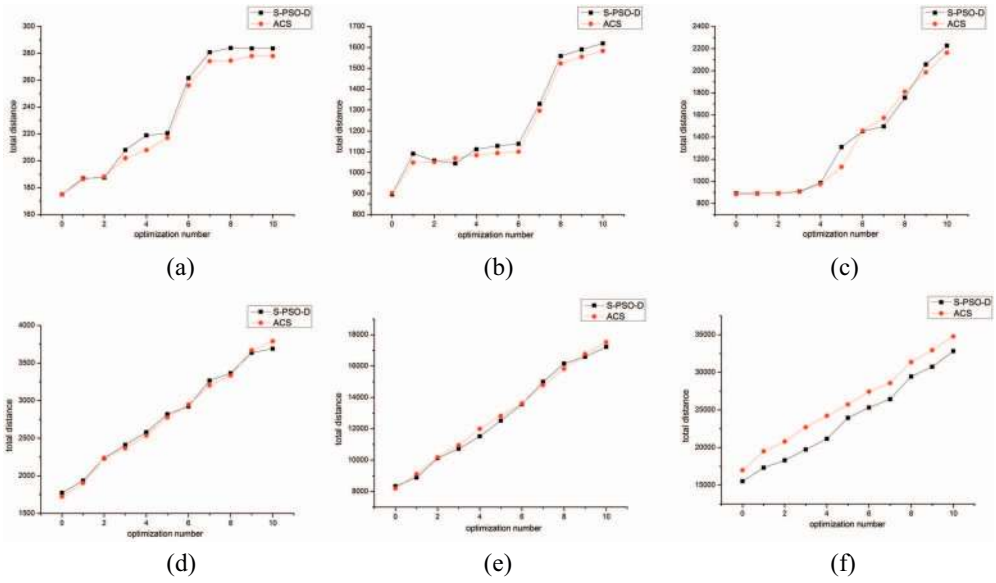

Fig. 10. Optimizing processes of (black) S-PSO-D and (red) ACS. (a) F-72-N. (b) T-75-U. (c) T-100-H. (d) T-150-U. (e) C-250-U. (f) R-350-U.

Reviewing the location distribution map in Fig. 7(e), we can see that customers in T-150 are somewhat clustered. Thus a rational conjecture is that such location distribution may be in favor of the region partition process in the dispatching system.

Second, seeing the instances where ACS surpasses S-PSO-D, we find that only on four instances, i.e., F-72-N, F-135-N, T-75-U, and T-100-H, ACS is significantly better than S-PSO-D. For the other small instances whose scales are equal to or smaller than 200, in fact, the S-PSO-D approach is competitive. Then, observing the results on the instances which have more than 200 customers, we find that S-PSO-D is significantly better than ACS on all of these instances. Thus we can take the 200 scale as a watershed. Before the watershed, ACS is a little bit better. After the watershed, clearly the S-PSO-D approach is more promising.

Third, we examine whether the closure time distribution has influence on algorithm's performance. Seeing the results on the six Fisher instances, we can find that for the half normal distribution and the uniform distribution, S-PSO-D can match ACS. However, for the normal distribution, S-PSO-D is significantly worse than ACS. Similar phenomenon also occurs on T-75, T-100, T-150, and R-200 instances. For the three T-75 instances, S-PSO-D lost on the uniform distribution; for T-100, S-PSO-D lost on the half normal distribution; for T-150, S-PSO-D is significantly better only for the uniform distribution; for R-200, S-PSO-D is significantly better only for the normal distribution. Thus we think that the closure time distribution really can influence the algorithm's performance, but how it affects the algorithm's performance is still a problem to be studied.

E. Performance Analysis

In this section, we try to find the reason why S-PSO-D outperforms ACS on some specific instances, while ACS surpasses S-PSO-D on some other instances. The optimizing processes of S-PSO-D and ACS on six instances are shown in Fig. 10. Among 30 independent runs, the once in which the final objective value is close to the mean value is chosen, and these six instances are F-72-N, T-75-U, T-100-H, T-150-U, C-250-U, and R-350-U. On F-72-N, T-75-U, and T-100-N, the ACS approach is significantly better than S-PSO-D; on T-150-U, C-250-U, and R-350-U, the contrary is the case.

Observing Fig. 10, we can find that there three patterns for one method to beat another method. The first one is shown in Fig. 10(a) and (b), where both approaches perform similarly before the first four optimizations. Starting from the 4th optimization, ACS shows its advantage and keeps that to the end. The second pattern is displayed in Fig. 10(c)–(e). Before the eight optimization, two approaches tie with each other. Finally during the last two or three optimizations, ACS surpasses S-PSO-D on T-100-H, while S-PSO-D beats ACS on T-150-U and C-250-U. Fig. 10(f) shows the third pattern where S-PSO-D builds the lead from the first optimization and keeps the advantage to the end. In the experiment, the third pattern is the most common one. Whenever the p-value of the Wilcoxon rank sum test is smaller than 1e-4, S-PSO-D beats the other two approaches by the third pattern.

Among these three patterns, the first pattern and the third pattern are easy to understand. According to the greedy strategy and the inheritance strategy, once an algorithm obtains good results during one optimization, this goodness can be passed to the following optimizations. That is the situation appeared in Fig. 10(a), (b), and (f). However, the second pattern is really confusing. Considering the essence of the reoptimization scheme and the experimental settings in the experiment, we have come up with a bold conjecture that the last two or three optimizations may play a decisive role among all of the optimizations.

F. Discussion of the Periodic Reoptimization Framework

First, we give an explanation of the conjecture proposed in the last section, then a specially designed experiment is conducted to verify the conjecture.

TABLE IV
COMPARISON OF THE TOTAL DISTANCE IN THE NEW EXPERIMENT

Instance	Approach	Mean	Best
F-72-N	O-S-PSO-D	283.65	277.99
	N-S-PSO-D	284.94	275.56
	O-ACS	278.17	263.83
	N-ACS	282.293	258.503
T-75-U	O-S-PSO-D	1615.97	1542.54
	N-S-PSO-D	1616.06	1547.04
	O-ACS	1585.37	1527.62
	N-ACS	1594.49	1495.95
T-100-H	O-S-PSO-D	2225.71	2030.23
	N-S-PSO-D	2203.04	1995.98
	O-ACS	2169.23	2050.52
	N-ACS	2131.98	1922.15
T-150-U	O-S-PSO-D	3693.09	3508.15
	N-S-PSO-D	3675.57	3515.23
	O-ACS	3781.62	3594.12
	N-ACS	3748.99	3474.02
C-250-U	O-S-PSO-D	17214	16172.1
	N-S-PSO-D	17193.8	16415.9
	O-ACS	17516.7	16654.2
	N-ACS	17341.2	16527.8
R-350-U	O-S-PSO-D	32821.6	32204.9
	N-S-PSO-D	32608.5	31827.8
	O-ACS	34930.4	33545.9
	N-ACS	34808.1	33865.6

If we set the inheritance strategy aside, actually except for the last optimization, the period of validity of each optimization is only one time slice, which means that the ith ($i < p$) optimization is only useful for the period $[T_i, T_{i+1}]$. Meanwhile, in the experiment, we have set the level of dynamism to 0.5 which implies that the last optimization has decided the routes for the last half day. Certainly, this is just the extreme case, but the fact shown in Fig. 10(c)–(e) is closely related to the explanation. With the inheritance strategy, an optimization may has effect on more than one slice. Definite number depends on the inheritance technique.

To testify the conjecture, we retest the aforementioned six instances. In the new experiment, the number of iteration for the first optimization is set equal to the scale of the problem, and the iteration number for the last two optimizations is set to two times of the customer number. Thus the total number of iterations is unchanged. Other parameters are kept the same. Numeric results are shown in Table IV where the original results of the two algorithms are shown as O-S-PSO-D and O-ACS. The new results are shown as N-S-PSO-D and N-ACS.

Focusing on the mean objective value, we can find that for the two instances F-72-N and T-75-U which belong to the first pattern, the results of the new experiment are worse than the original results. However, observing the results on the next four instances, we find that the new experiment indeed increases the level of the performance of the two algorithms. Such phenomenon proves that our conjecture really works for some instances but not all of them. In the previous subsection, based on the results of the second pattern, we made the conjecture that the last two or three optimizations play a decisive role among all optimization procedures. Here, the results in Table IV proves its correctness. However, this conjecture contradicts the truth revealed in the first pattern that the early couple rounds may be more important for some other instances. Thus, on the whole, we conclude that for different kinds of instances, different rounds of optimizations have different importance.

VI. CONCLUSION

In this paper, we have proposed a novel dispatching system to handle the DCVRP by assembling the S-PSO-D algorithm with the framework of periodic reoptimization. According to the structure of the DCVRP solution, a new particle representation scheme is defined and new solution construction method is designed in S-PSO-D. In addition, considering the essence of the dynamic environment, two techniques are creatively proposed in the dispatching system: region partition and archive strategy. Moreover, a local refinement process is combined with the S-PSO-D algorithm. These three techniques, make the proposed dispatching system more effective and efficient. In the experiment, not only the effectiveness of the system is testified, but the intrinsic features of the periodic reoptimization method are discussed. Experimental results show that our approach performs obviously better on large-scale instances, and is competitive on small-scale instances as well.

In future researches, besides designing more effective algorithms, there are also some questions worth studying.

1) In the perspective of methodology, it is meaningful to study the difference of the decision epoch method and the time slice method, thus to explore which one is more suitable for a specific algorithm under certain circumstances.
2) In the experiments, we have mentioned that there are different kinds of disclosure time distributions, and also we found that these distributions can influence the algorithm's performance. However, the specific cause-effect influence needs to be further studied to figure out what the result would be if we combine them with different methods.
3) The third question has been mentioned in the discussion of the periodic reoptimization framework: what the relationship between the instance and the number of optimization is, and how to find which round of optimization is more important.

Aiming at these three problems, the DVRP can be further studied and more automated assistive tools can be made for modern logistics industry.

REFERENCES

[1] Z. Wenqian, *Traditional Shops Must Boost Their Online Links*, China Daily USA, Beijing, China, Nov. 2016. [Online]. Available: http://usa.chinadaily.com.cn/epaper/2016-11/15/content_27397828.htm
[2] A. C. Mckinnon, M. Browne, A. E. Whiteing, and M. Piecyk, *Green Logistics: Improving the Environmental Sustainability of Logistics*. London, U.K.: Kogan Page, 2015.
[3] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Manag. Sci.*, vol. 6, no. 1, pp. 80–91, 1959.
[4] W. Y. Szeto, Y. Wu, and S. C. Ho, "An artificial bee colony algorithm for the capacitated vehicle routing problem," *Eur. J. Oper. Res.*, vol. 215, no. 1, pp. 126–135, 2011.
[5] L. H. Lee, K. C. Tan, K. Ou, and Y. H. Chew, "Vehicle capacity planning system: A case study on vehicle routing problem with time windows," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 33, no. 2, pp. 169–178, Mar. 2003.

[6] Y.-J. Gong *et al.*, "Optimizing the vehicle routing problem with time windows: A discrete particle swarm optimization approach," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 2, pp. 254–267, Mar. 2012.
[7] S. N. Parragh, K. F. Doerner, and R. F. Hartl, "A survey on pickup and delivery problems," *J. für Betriebswirtschaft*, vol. 58, no. 1, pp. 21–51, 2008.
[8] T. J. Ai and V. Kachitvichyanukul, "A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery," *Comput. Oper. Res.*, vol. 36, no. 5, pp. 1693–1702, 2009.
[9] Y. Marinakis, G.-R. Iordanidou, and M. Marinaki, "Particle swarm optimization for the vehicle routing problem with stochastic demands," *Appl. Soft Comput.*, vol. 13, no. 4, pp. 1693–1704, 2013.
[10] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 1, pp. 70–85, Jan. 2016.
[11] X. Wang, T.-M. Choi, H. Liu, and X. Yue, "A novel hybrid ant colony optimization algorithm for emergency transportation problems during post-disaster scenarios," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published.
[12] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *Eur. J. Oper. Res.*, vol. 225, no. 1, pp. 1–11, 2013.
[13] S. N. Kumar and R. Panneerselvam, "A survey on the vehicle routing problem and its variants," *Intell. Inf. Manag.*, vol. 4, no. 3, pp. 66–74, 2012.
[14] B. K.-S. Cheung, K. L. Choy, C.-L. Li, W. Shi, and J. Tang, "Dynamic routing model and solution methods for fleet management with mobile technologies," *Int. J. Prod. Econ.*, vol. 113, no. 2, pp. 694–705, 2008.
[15] M. Gendreau, F. Guertin, J.-Y. Potvin, and E. Taillard, "Parallel tabu search for real-time vehicle routing and dispatching," *Transp. Sci.*, vol. 33, no. 4, pp. 381–390, 1999.
[16] N. Azi, M. Gendreau, and J.-Y. Potvin, "A dynamic vehicle routing problem with multiple delivery routes," *Ann. Oper. Res.*, vol. 199, no. 1, pp. 103–112, 2012.
[17] S. F. Ghannadpour, S. Noori, and R. Tavakkoli-Moghaddam, "Multiobjective dynamic vehicle routing problem with fuzzy travel times and customers' satisfaction in supply chain management," *IEEE Trans. Eng. Manag.*, vol. 60, no. 4, pp. 777–790, Nov. 2013.
[18] H. Housroum, T. Hsu, R. Dupas, and G. Goncalves, "A hybrid ga approach for solving the dynamic vehicle routing problem with time windows," in *Proc. IEEE 2nd Int. Conf. Inf. Commun. Technol.*, vol. 1. Damascus, Syria, 2006, pp. 787–792.
[19] G. B. Alvarenga, R. M. de Abreu Silva, and G. R. Mateus, "A hybrid approach for the dynamic vehicle routing problem with time windows," in *Proc. 5th Int. Conf. Hybrid Intell. Syst.*, Rio de Janeiro, Brazil, 2005, p. 7.
[20] G. B. Alvarenga, G. R. Mateus, and G. De Tomi, "A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows," *Comput. Oper. Res.*, vol. 34, no. 6, pp. 1561–1584, 2007.
[21] R. Montemanni, L. M. Gambardella, A. E. Rizzoli, and A. V. Donati, "Ant colony system for a dynamic vehicle routing problem," *J. Comb. Optim.*, vol. 10, no. 4, pp. 327–343, 2005.
[22] F. T. Hanshar and B. M. Ombuki-Berman, "Dynamic vehicle routing using genetic algorithms," *Appl. Intell.*, vol. 27, no. 1, pp. 89–99, 2007.
[23] M. R. Khouadjia, B. Sarasola, E. Alba, L. Jourdan, and E.-G. Talbi, "A comparative study between dynamic adapted PSO and VNS for the vehicle routing problem with dynamic requests," *Appl. Soft Comput.*, vol. 12, no. 4, pp. 1426–1439, 2012.
[24] A. Núñez, C. E. Cortés, D. Sáez, B. De Schutter, and M. Gendreau, "Multiobjective model predictive control for dynamic pickup and delivery problems," *Control Eng. Pract.*, vol. 32, pp. 73–86, Nov. 2014.
[25] M. Mavrovouniotis and S. Yang, "Ant colony optimization with memory-based immigrants for the dynamic vehicle routing problem," in *Proc. IEEE CEC*, Brisbane, QLD, Australia, 2012, pp. 1–8.
[26] L. M. Hvattum, A. Løkketangen, and G. Laporte, "Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic," *Transp. Sci.*, vol. 40, no. 4, pp. 421–438, 2006.
[27] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Human Sci.*, vol. 1. Nagoya, Japan, 1995, no. 39–43.
[28] W.-N. Chen *et al.*, "A novel set-based particle swarm optimization method for discrete optimization problems," *IEEE Trans. Evol. Comput.*, vol. 14, no. 2, pp. 278–300, Apr. 2010.
[29] H. Wu, C. Nie, F.-C. Kuo, H. Leung, and C. J. Colbourn, "A discrete particle swarm optimization for covering array generation," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 575–591, Aug. 2015.
[30] Y.-H. Jia, W.-N. Chen, and X.-M. Hu, "A PSO approach for software project planning," in *Proc. ACM GECCO*, Vancouver, BC, Canada, 2014, pp. 7–8.
[31] J. E. Bell and P. R. McMullen, "Ant colony optimization techniques for the vehicle routing problem," *Adv. Eng. Inf.*, vol. 18, no. 1, pp. 41–48, 2004.
[32] Y. Marinakis, M. Marinaki, and G. Dounias, "A hybrid particle swarm optimization algorithm for the vehicle routing problem," *Eng. Appl. Artif. Intell.*, vol. 23, no. 4, pp. 463–472, 2010.
[33] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Comput. Oper. Res.*, vol. 31, no. 12, pp. 1985–2002, 2004.
[34] P. Toth and D. Vigo, *Vehicle Routing: Problems, Methods, and Applications.* Philadelphia, PA, USA: SIAM, 2014.
[35] K. Lund, O. B. G. Madsen, and J. M. Rygaard, "Vehicle routing problems with varying degrees of dynamism," IMM Inst. Math. Model., Tech. Univ. Denmark, Lyngby, Denmark, Tech. Rep. IMM-REP-1996-1, 1996.
[36] A. Larsen, "The dynamic vehicle routing problem," Ph.D. dissertation, Dept. Comput. Inf. Sci., Tech. Univ. Denmark, Kongens Lyngby, Denmark, 2001.
[37] Y. V. Pehlivanoglu, "A new particle swarm optimization method enhanced with a periodic mutation strategy and neural networks," *IEEE Trans. Evol. Comput.*, vol. 17, no. 3, pp. 436–452, Jun. 2013.
[38] C. Li, S. Yang, and T. T. Nguyen, "A self-learning particle swarm optimizer for global optimization problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 3, pp. 627–646, Jun. 2012.
[39] Y. Wang and L. Li, "Heterogeneous redundancy allocation for series-parallel multi-state systems using hybrid particle swarm optimization and local search," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 42, no. 2, pp. 464–474, Mar. 2012.
[40] Y. Fu, M. Ding, C. Zhou, and H. Hu, "Route planning for unmanned aerial vehicle (UAV) on the sea using hybrid differential evolution and quantum-behaved particle swarm optimization," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 6, pp. 1451–1465, Nov. 2013.
[41] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.
[42] Z.-L. Chen and H. Xu, "Dynamic column generation for dynamic vehicle routing with time windows," *Transp. Sci.*, vol. 40, no. 1, pp. 74–88, 2006.
[43] P. Kilby, P. Prosser, and P. Shaw, "Dynamic VRPs: A study of scenarios," Dept. Comput. Inf. Sci., Univ. Strathclyde, Glasgow, U.K., Tech. Rep. APES-06-1998, 1998.
[44] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.
[45] C. Nilsson, "Heuristics for the traveling salesman problem," Dept. Cybern. Artif. Intell., Linköping Univ., Linköping, Sweden, Tech. Rep., 2003.
[46] B. I. Kim, J. I. Shim, and M. Zhang, "Comparison of TSP algorithms," Project for Models in Facilities Planning and Materials Handling, 1998.
[47] É. Taillard, "Parallel iterative search methods for vehicle routing problems," *Netw.*, vol. 23, no. 8, pp. 661–673, 1993.
[48] M. L. Fisher and R. Jaikumar, "A generalized assignment heuristic for vehicle routing," *Netw.*, vol. 11, no. 2, pp. 109–124, 1981.

Ya-Hui Jia (S'14) received the bachelor's degree from Sun Yat-sen University, Guangzhou, China, in 2013, where he is currently pursuing the Ph.D. degree.

He is a Research Assistant with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. His current research interests include evolutionary computation algorithms and their applications on software engineering, cloud computing, and intelligent transportation.

Wei-Neng Chen (S'07–M'12) received the bachelor's and Ph.D. degrees from Sun Yat-sen University, Guangzhou, China, in 2006 and 2012, respectively.

He is currently a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. His current research interests include swarm intelligence algorithms and their applications on cloud computing, operations research, and software engineering. He has published over 70 papers in international journals and conferences.

Dr. Chen was a recipient of the IEEE Computational Intelligence Society Outstanding Dissertation Award for his doctoral thesis in 2016, and the National Science Fund for Excellent Young Scholars in 2016.

Tianlong Gu received the M.Eng. degree from Xidian University, Xi'an, China, in 1987, and the Ph.D. degree from Zhejiang University, Hangzhou, China, in 1996.

From 1998 to 2002, he was a Research Fellow with the School of Electrical and Computer Engineering, Curtin University of Technology, Bentley, WA, Australia, and a Post-Doctoral Fellow with the School of Engineering, Murdoch University, Perth, WA, Australia. He is currently a Professor with the School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin, China. His current research interests include formal methods, data and knowledge engineering, software engineering, and information security protocol.

Huaxiang Zhang received the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China, in 2004.

He is currently a Professor with the School of Information Science and Engineering, Shandong Normal University, Jinan, China, where he was an Associate Professor with the Department of Computer Science, from 2004 to 2005. He has authored over 100 journal and conference papers and holds eight invention patents. His current research interests include machine learning, pattern recognition, evolutionary computation, and Web information processing.

Huaqiang Yuan received the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China, in 1996.

He is currently a Professor with the School of Computer Science and Network Security, Dongguan University of Technology, Dongguan, China. His current research interests include computational intelligence and cyberspace security.

Ying Lin (M'12) received the Ph.D. degree in computer applied technology from Sun Yat-sen University, Guangzhou, China, in 2012.

She is currently an Assistant Professor with the Department of Psychology, Sun Yat-sen University. Her current research interests include computational intelligence and its applications in network analysis and cognitive diagnosis.

Wei-Jie Yu (S'10–M'14) received the bachelor's degree in network engineering and the Ph.D. degree in computer application technology from Sun Yat-sen University, Guangzhou, China, in 2009 and 2014, respectively.

He is currently a Lecturer with the School of Information Management, Sun Yat-sen University. His current research interests include differential evolution, artificial bee colony optimization, ant colony optimization, particle swarm optimization, and their applications in real-world problems.

Jun Zhang (M'02–SM'08–F'16) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Professor with the South China University of Technology, Guangzhou, China. His current research interests include computational intelligence, cloud computing, wireless sensor networks, operations research, and power electronic circuits. He has authored seven research books and book chapters, and over 50 IEEE TRANSACTIONS papers in the above areas.

Prof. Zhang was a recipient of the National Science Fund for Distinguished Young Scholars in 2011, and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He was also appointed as the Changjiang Chair Professor in 2013. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, and the IEEE TRANSACTIONS ON CYBERNETICS. He is the Founding and Current Chair of the IEEE Guangzhou Subsection, the IEEE Beijing (Guangzhou) Section Computational Intelligence Society Chapters, and the ACM Guangzhou Chapter.