A Buffer-Based Ant Colony System Approach for Dynamic Cold Chain Logistics Scheduling

Li-Jiao Wu, *Student Member, IEEE*, Lin Shi, *Student Member, IEEE*, Zhi-Hui Zhan, *Senior Member, IEEE*, Kuei-Kuei Lai, and Jun Zhang, *Fellow, IEEE*

***Abstract*—Cold chain logistics (CCL) scheduling is an emerging research problem in the logistics industry in smart cities, which mainly concerns the distribution of perishable goods. As the quality loss of goods that occurs in the distribution process should be considered, the CCL scheduling problem is very challenging. Moreover, the problem is more challenging when the dynamic characteristics (e.g., the orders are unknown beforehand) of the real scheduling environment are considered. Therefore, this paper focuses on the dynamic CCL (DCCL) scheduling problem by establishing a practical DCCL model. In this model, a working day is divided into multiple time slices so that the dynamic new orders revealed in the working day can be scheduled in time. The objective of the DCCL model is to minimize the total distribution cost in a working day, which includes the transportation cost, the cost of order rejection penalty, and the cost of quality loss of goods. To solve the DCCL model, a buffer-based ant colony system (BACS) approach is proposed. The BACS approach is characterized by a buffering strategy that is carried out at the beginning of the scheduling in every time slice except the last one to temporarily buffer some non-urgent orders, so as to concentrate on scheduling the orders that are preferred to be delivered first. Besides, to further promote the performance of BACS, a periodic learning strategy is designed to avoid local optima. Comparison experiments are conducted on test instances with different problem scales. The results show that BACS is more preferred for solving the DCCL model when compared with the other five state-of-the-art and recent well-performing scheduling approaches.**

***Index Terms*—Cold chain logistics (CCL), dynamic optimization, ant colony system (ACS), vehicle routing problem, evolutionary computation, logistics scheduling.**

Manuscript received 6 July 2021; revised 11 January 2022; accepted 26 March 2022. Date of publication 19 May 2022; date of current version 28 November 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2102102, in part by the National Natural Science Foundations of China under Grants 62176094 and 61873097, in part by the Key-Area Research and Development of Guangdong Province under Grant 2020B010166002, in part by the Guangdong Natural Science Foundation Research Team under Grant 2018B030312003, and in part by the National Research Foundation of Korea under Grant NRF-2021H1D3A2A01082705. *(Li-Jiao Wu and Lin Shi are co-first authors.) (Corresponding authors: Zhi-Hui Zhan; Jun Zhang.)*

Li-Jiao Wu, Lin Shi, and Zhi-Hui Zhan are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, Guangdong 510006, China, also with Pazhou Laboratory, Guangzhou, Guangdong 510330, China, and also with the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, Guangzhou, Guangdong 510006, China (e-mail: zhanapollo@163.com).

Kuei-Kuei Lai is with the Department of Business Administration, Chaoyang University of Technology, Taichung City 413, Taiwan.

Jun Zhang is with Zhejiang Normal University, Jinhua, Zhejiang 321004, China, and also with Hanyang University, Ansan 15588, South Korea.

This article has supplementary downloadable material available at https://doi.org/10.1109/TETCI.2022.3170520, provided by the authors.

Digital Object Identifier 10.1109/TETCI.2022.3170520

I. Introduction

The increasing demand for perishable goods of high quality due to the improving living standards has boosted the development of cold chain logistics (CCL) in recent years [1], [2]. Herein, the CCL refers to one of the emerging branches of the logistics industry, especially in smart cities, that aims to deliver perishable goods in good condition through a low-temperature transportation environment. The major difference between the CCL and the traditional logistics is that the goods transported by the CCL are always perishable, and the quality loss of goods that occurs in the distribution process cannot be neglected. Thus, the quality loss of goods should be considered in the CCL scheduling, which makes the CCL scheduling problem more challenging than the traditional logistics scheduling problem. To improve the service quality of CCL, an appropriate model should be established for the CCL scheduling problem, together with an effective scheduling approach.

As can be known from the literature, some research has attempted to study the CCL model derived from the vehicle routing problem (VRP) [3] since the VRP is one of the most significant research topics in logistics scheduling [4], and an overview of VRP is provided in [5]. For example, a VRP-based emergency CCL model aiming at minimizing the loss of vehicles, refrigeration consumption, and cargo damage was established in [6]. The scheduling problem solved in [7] was based on a multi-depot VRP for seafood delivery, which was also in the coverage of CCL scheduling. Instead of using normal reefer vehicles, the electric vehicles were employed in [8], and a scheduling model was established to minimize the total cost of fresh products distribution, including economic cost and fresh value loss cost. Besides, carbon emission minimization was considered in some CCL models [9]–[11] because refrigerated transportation was energy-consuming. Furthermore, the CCL models in [12] and [13] considered both the distribution center (i.e., depot) location problem and the VRP.

Nevertheless, to the best of our knowledge, most of the existing CCL models are to solve the static CCL scheduling problem as they are established under the static situation (i.e., all orders are known at the beginning of a working day). In other words, the dynamic situation of CCL scheduling (i.e., the dynamic CCL (DCCL) scheduling problem) has hardly been studied. The DCCL scheduling problem is more practical than the static CCL scheduling problem because it considers the real dynamic situation where only some orders are known at the beginning of a working day while the other orders are revealed gradually in the working day. Besides, the DCCL scheduling problem is different from the static CCL scheduling problem because the re-routing for reefer vehicles may occur in DCCL scheduling due to the arrival of new orders. Specifically, in the DCCL scheduling problem, the scheduling should be conducted in time when sufficient orders have been revealed. Meanwhile, the reefer vehicles should begin to distribute the orders once the scheduling result of the known orders is obtained. With the successive arrival of new orders, the scheduling should be invoked again and again to arrange the distribution sequence for the new orders and the unserved orders, meaning that the distribution routes of the reefer vehicles may be reorganized. Therefore, to solve the DCCL scheduling problem, an efficient DCCL model that considers the newly revealed orders is established in this paper.

To solve the DCCL model, we can firstly refer to the studies of solving the dynamic VRP (DVRP) since the DCCL scheduling problem is developed based on the DVRP. The DVRP is a challenging combinatorial optimization problem that schedules the newly revealed customer requests in an online manner. Systematic introductions of DVRP were provided in [14] and [15]. Generally speaking, the DVRP is usually solved following a framework that divides an entire working day into multiple time slices and schedules the revealed orders at the end of each time slice [16]. Thus, the traditional scheduling approaches can be used to solve the DVRP in a re-optimization way (i.e., the traditional scheduling approaches are performed in every time slice). In addition, research into effective scheduling approaches for DVRP has attracted increasing attention in recent decades [17]–[19]. For example, to adapt to the changing environment, a pheromone conservation procedure for ant colony optimization (ACO) was proposed in [20], which transmitted information of the previous scheduling in the form of pheromone. Besides, to maintain the diversity of a population, a population-based ant colony optimization (P-ACO) approach was proposed in [21], which used a memory that consists of ants with better performance. More dynamic scheduling approaches are introduced in [22].

However, although the DCCL scheduling problem has some similarities with the DVRP, it is more complicated and challenging than the DVRP because the quality loss of goods should be additionally considered as part of the objective in the DCCL scheduling problem. In this sense, the DCCL scheduling problem can be regarded as an emerging challenging research topic in the community of computational intelligence. Meanwhile, considering that the DCCL scheduling problem is a dynamic scheduling problem that new orders can be revealed at any time in the working day, the DCCL scheduling problem is also solved by dividing the working day into multiple time slices, so that the newly revealed orders in a time slice can be scheduled in time at the end of the time slice. In other words, the DCCL scheduling is carried out in an online manner to find out the optimal solution as soon as possible, meaning that the execution time allowed for the scheduling approaches is limited. Thus, an effective and efficient scheduling approach is in great need.

Given the good performances of scheduling approaches based on ant colony optimization (ACO) and its popular variant named ant colony system (ACS) [23] in solving many combinatorial/continuous/dynamic optimization problems [24], [25] and real-world application problems [26]–[29], it is promising to solve the DCCL model using an ACS-based approach. Therefore, a buffer-based ACS (BACS) approach that incorporates the ACS with a buffering strategy (BS) and a periodic learning strategy (PLS) is proposed in this paper. The BS is designed to temporarily buffer the non-urgent orders that can be delivered later and postpone their scheduling, which helps the BACS concentrate on scheduling the orders that are preferred to be delivered first. This way, the BACS can perform in a relatively smaller yet more promising search space, which is beneficial for BACS to find better solutions with less execution time. Meanwhile, the PLS is designed to help the BACS jump out of local optima, and thus solve the DCCL model more effectively.

In summary, the contributions of this paper lie in two aspects, which are the problem modeling aspect and algorithm designing aspect. Firstly, in the problem modeling aspect, a DCCL model is established to reflect the real-world CCL scheduling problem. The DCCL scheduling problem is an emerging and challenging research topic since it has to consider the quality loss of goods caused in the distribution process as well as the dynamic factors. The objective of this model is to minimize the total cost that consists of the transportation cost, the cost of order rejection penalty, and the cost of quality loss of goods. Secondly, in the algorithm designing aspect, a new scheduling approach named BACS is proposed to solve the DCCL model effectively. The BS integrated into the BACS can temporarily buffer the non-urgent orders and thus encourage the BACS to concentrate on scheduling orders that shall be delivered first. Besides, the PLS can also enhance the performance of BACS by helping the BACS avoid being trapped into the local optima.

The remainder of this paper is organized as follows. Section II introduces the description and formulation of the DCCL model. Section III presents the implementation of BACS. Section IV is the experimental design, comparison, and analysis. Finally, Section V concludes the whole paper.

II. Dynamic Cold Chain Logistics Scheduling

A. *Problem Description*

In the DCCL scheduling problem, only some orders are known at the beginning of a working day, and the other orders are revealed gradually in the working day. For the orders revealed after the end of the current working day, they will be scheduled in the next working day, which are actually the orders known at the beginning of the next working day. The orders that are known at the beginning of a working day are denoted as static orders, and the orders revealed in a working day are regarded as dynamic orders. Moreover, the major task of DCCL scheduling problem is to arrange the distribution sequence of orders by designing distribution routes for the vehicles that depart from the depot with a full load, and the objective of this problem is always determined as minimizing the distribution cost of a working day.

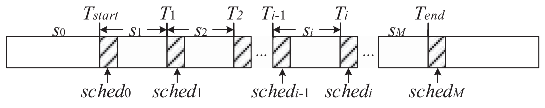

Fig. 1. Schematic diagram of DCCL scheduling.

To deal with the DCCL scheduling problem, a working day is divided into multiple time slices evenly. In this way, the DCCL scheduling problem can be divided into multiple static subproblems, and one subproblem corresponds to one time slice. Meanwhile, the whole DCCL scheduling problem can be solved by dealing with each static subproblem in sequence. Specifically, to begin with, the scheduling of all the known orders (i.e., the static orders) is carried out at the beginning of a working day. Then, the scheduling of the new orders revealed in a time slice and the orders revealed before but have not been served yet is carried out at the end of each time slice. The scheduling process is continued until the scheduling for the last time slice is completed.

For clarity, a schematic diagram of DCCL scheduling is shown in Fig. 1, where T_{start} and T_{end} represent the start time and end time of a working day, respectively. A working day is divided into M ($M \geq 1$) time slices, and the ith ($1 \leq i \leq M$) time slice is denoted as s_i. Note that the interval of s_i is [$T_{i\text{-}1}$, T_i), and T_0 equals T_{start}. As for s_0, it begins from the end of the previous working day and ends before T_{start} of the current working day. The scheduling corresponding to s_i is denoted as $sched_i$, which is carried out at the end of s_i, i.e., T_i. The execution time of $sched_i$ is a parameter that is denoted as T_{sched}, which is shown as the shadow in Fig. 1. Meanwhile, the $sched_i$ is conducted to find out the optimal distribution sequence for the orders that have been revealed but remain unserved. Herein, the orders scheduled in $sched_i$ are composed of two parts, orders revealed in time slice s_i and orders revealed before time slice s_i but still remain unserved at T_i+T_{sched}. The reason that T_i+T_{sched} rather than T_i is considered when determining orders to be scheduled in $sched_i$ is that the scheduling result of $sched_i$ can only be submitted to reefer vehicles at T_i+T_{sched} due to the time consumption of $sched_i$, i.e., T_{sched}. Hence, to avoid the reefer vehicles being idle when $sched_i$ is being conducted, the reefer vehicles should proceed to carry out distribution tasks assigned by the scheduling result of $sched_{i\text{-}1}$ until $sched_i$ is finished, i.e., T_i+T_{sched}.

Generally speaking, the reefer vehicles will carry out the new distribution tasks immediately once they receive the scheduling result of $sched_i$. However, if a reefer vehicle is on the way to the destination of an order assigned by $sched_{i\text{-}1}$, it will not carry out the new distribution tasks of $sched_i$ until the service of the current order is finished. If a reefer vehicle finishes the distribution tasks of $sched_i$ before the arrival of the new distribution tasks, it will wait at the destination of the last order that it has served. Note that if vehicle k is empty after finishing the distribution tasks of $sched_i$, it will not be used in the following scheduling in the current working day and it can return to the depot at any time. However, if vehicle k still has goods, it cannot return to the depot because it may be asked to serve the other upcoming orders, even though it cannot serve any order currently. Therefore, for uniform consideration, each vehicle in our model will return to the depot after it finishes the distribution tasks of the last scheduling $sched_M$.

The framework of DCCL scheduling can be summarized as Fig. 2, and the scheduling procedure in time slice s_i is shown in the ith dotted box. Firstly, orders revealed in s_i are collected by the order collector continuously. Then, before the beginning of $sched_i$, orders that are scheduled in $sched_{i\text{-}1}$ but cannot be served before T_i+T_{sched} are also submitted to the order collector. Later on, $sched_i$ is carried out on all orders in the order collector, and the scheduling result is submitted to reefer vehicles. Finally, the orders that cannot be served before $T_{i+1}+T_{sched}$ are returned to the order collector of time slice s_{i+1}. Note that no orders are returned from s_M since s_M is the last time slice, and each reefer vehicle should return to the depot after finishing all distribution tasks. Besides, no unserved orders except new orders revealed in s_0 are collected by the order collector of s_0.

B. Model Formulation

Since the DCCL scheduling problem is decomposed into $M+1$ static subproblems (including the subproblem corresponding to the time slice s_0), this part firstly introduces the formulation for the ith ($0 \leq i \leq M$) static subproblem, and then introduces the formulation of the complete DCCL model.

The set of the indexes of orders to be scheduled in $sched_i$ is denoted as C_i. The set of the indexes of the departure places of vehicles in $sched_i$ is denoted as dp, and dp_k is the departure place of vehicle k. In the case that $i = 0$, dp_k is the depot, which is denoted as 0. In the case that $i > 0$, if vehicle k has left the depot, dp_k is modified as the index of the last order that vehicle k has delivered before it can carry out the new distribution tasks of $sched_i$. Otherwise, dp_k is not modified (i.e., dp_k is 0). The union set of C_i and dp is denoted as V_i. The first decision variable required in $sched_i$ is denoted as $x^i{}_{owk}$, which is defined as

$$x^i_{owk} = \begin{cases} 1, & \text{if vehicle } k \text{ travels from } o \text{ to } w \text{ in } sched_i \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where o and w refer to two different vertexes. The second decision variable in $sched_i$ is denoted as $y^i{}_k$, which is defined as

$$y^i_k = \begin{cases} 1, & \text{if } dp_k \neq 0 \text{ or } k \text{ is used in } sched_i \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

which means that $y^i{}_k$ is 1 only if vehicle k has left the depot or k is used in $sched_i$. Note that vehicle k is filled with goods when it departs from the depot, and it cannot return to the depot until it finishes the distribution tasks of $sched_M$, even though it becomes empty in the distribution process.

The objective of the ith static subproblem is to minimize the total distribution cost caused by serving orders in C_i, which consists of the transportation cost, the cost of order rejection penalty, and the cost of quality loss of goods. The specific calculation for the objective of the ith static subproblem is described as follows.

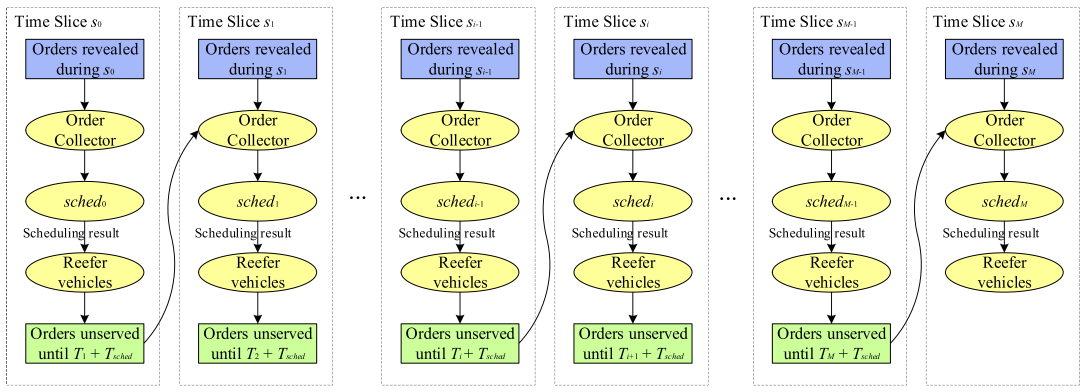
Time Slice s_0
Orders revealed during s_0
Order Collector
$sched_0$
Scheduling result
Reefer vehicles
Orders unserved until $T_1 + T_{sched}$
Time Slice s_1
Orders revealed during s_1
$sched_1$
Orders unserved until $T_2 + T_{sched}$
Time Slice s_{i-1}
Orders revealed during s_{i-1}
$sched_{i-1}$
Orders unserved until $T_i + T_{sched}$
Time Slice s_i
Orders revealed during s_i
$sched_i$
Orders unserved until $T_{i+1} + T_{sched}$
Time Slice s_{M-1}
Orders revealed during s_{M-1}
$sched_{M-1}$
Orders unserved until $T_M + T_{sched}$
Time Slice s_M
Orders revealed during s_M
$sched_M$
...

Fig. 2. Framework of DCCL scheduling.

The transportation cost F_1 is measured by the cost of vehicle usage and driving distance, which is formulated as

$$F_1 = \sum_{k=1}^{K} y_k^i \times p_0 + \sum_{k=1}^{K} y_k^i \times \left(\sum_{o\in V_i} \sum_{w\in V_i} x_{owk}^i \times dis_{ow} \right) \times p_1 \tag{3}$$

where K is the maximum number of available reefer vehicles, V_i is the set of vertexes, including the indexes of orders to be scheduled and the departure places of vehicles. The variable dis_{ow} is the driving distance from vertex o to vertex w. Besides, p_0 is the fixed cost of vehicle usage, which is determined as 100, and p_1 is the cost of unit driving distance, which is 1.

The order rejection penalty is considered since an order should be rejected if it cannot be served by any vehicle without violating constraints. However, it may decrease customer satisfaction if too many orders are rejected. Hence, the number of rejected orders should be minimized, and the cost of order rejection penalty F_2 is computed as

$$F_2 = \sum_{o\in C_i} \left(1 - \sum_{k=1}^{K} \sum_{w\in V_i} x_{wok}^i \right) \times p_2 \tag{4}$$

where p_2 is the penalty factor, which is determined as 1000. Note that the value of the expression in brackets is either 1 or 0, meaning that order o is rejected or accepted.

The cost of quality loss of goods is one of the major focuses of CCL scheduling since goods being distributed are often sensitive to time and temperature. Herein, the cost of quality loss F_3 is calculated as

$$F_3 = \sum_{k=1}^{K} y_k^i \times \left(\left(\sum_{w\in V_i} \sum_{o\in C_i} x_{wok}^i \times ql_o \times dmd_o \right) + ql_{rq(i,k)} \times rq(i,k) \right) \times p_3 \tag{5}$$

where dmd_o is the demand of order o, p_3 is the cost of unit quality loss for unit goods, which is set as 1. The variable ql_o is used to measure the quality loss of goods when the goods of order o are delivered. The variable $rq(i, k)$ is the remaining quantity of goods in vehicle k when k finishes the distribution tasks of $sched_i$, and $ql_{rq(i,\ k)}$ is the quality loss of remaining goods at that time. As the quality of goods at time t is always calculated as $Q(t) = Q(0) \times e^{-\theta \times t}$ (θ is the spoilage rate), ql_o can be calculated as

$$ql_o = 1 - e^{-\theta_1 \times t_1(o) - \theta_2 \times t_2(o)} \tag{6}$$

where two states of reefer vehicle k's carriage that can result in different spoilage rates are considered in (6). The first state is the opened state that appears when k is delivering (i.e., unloading) goods for other orders. In this case, the temperature of the carriage may rise due to air circulation, leading to a higher spoilage rate. The second state is the closed state that appears when k is driving or waiting. The spoilage rates corresponding to the first state and the second state are denoted as θ_1 and θ_2, which are determined as 0.003 and 0.002, respectively. The duration of the first state that goods of order o go through, $t_1(o)$, is computed as

$$t_1(o) = t_1(dp_k) + \sum_{j=0}^{j'-1} st_{R_i(k,j)}, \text{ if } R_i(k, j') = o \tag{7}$$

where st_o refers to the service time (i.e., the time required for unloading goods) required by order o. R_i is the set of distribution routes for reefer vehicles in $sched_i$. $R_i(k, j)$ is the jth destination of vehicle k in $sched_i$, and $R_i(k, 0)$ represents the departure place of vehicle k, i.e., dp_k. Besides, $t_1(dp_k)$ is 0 when dp_k is the depot (i.e., $t_1(0) = 0$), and st for the depot is 0 (i.e., $st_0 = 0$). As for $t_1(rq(i, k))$, it is calculated as

$$t_1(rq(i,k)) = t_1(dp_k) + \sum_{j=0}^{|R_i(k)|-1} st_{R_i(k,j)} \tag{8}$$

where $R_i(k)$ is the distribution route of vehicle k in $sched_i$, and $|R_i(k)|$ refers to the number of elements in $R_i(k)$. That is, the duration of the first state that the remaining goods experience is the sum of the service time of the orders that have been delivered by vehicle k. As for the duration of the second state that goods of order o experience, $t_2(o)$, it can be calculated by

$$t_2(o) = pt_o - t_1(o) - dt_k(0) \tag{9}$$

where pt_o is the time that vehicle *k* starts to deliver goods for order *o*, $dt_k(0)$ is the time vehicle *k* departs from the depot. As for $t_2(rq(i,k))$, it is computed as

$$t_2\left(rq\left(i,k\right)\right) = \begin{cases} t_2\left(R_i\left(k,\left|R_i(k)\right|-1\right)\right), \text{ if } \left|R_i(k)\right|>1 \\ dt_k(dp_k) - t_1\left(rq\left(i,k\right)\right) - dt_k(0), \text{ otherwise} \end{cases} \tag{10}$$

If vehicle *k* is used in $sched_i$, i.e., $|R_i(k)| > 1$, $t_2(rq(i,k))$ equals the duration of the second state that goods of the last order delivered by vehicle *k* experience. Otherwise, $t_2(rq(i,k))$ is calculated according to the time that vehicle *k* is going to leave its departure place. Furthermore, the variable pt_o is computed as

$$pt_o = \max\left\{dt_k(w) + dur_{wo}, a_o\right\} \tag{11}$$

where $dt_k(w)$ is the time that vehicle *k* leaves *w*, dur_{wo} is the driving duration from *w* to *o*. The variable a_o refers to the lower bound of *o*'s service time window. Note that vehicle *k* will leave the destination of an order once it finishes the service of the order if there are other orders to be served. Otherwise, vehicle *k* will wait at the current position until the arrival of the new distribution tasks (i.e., the time that the next scheduling is finished) or it is allowed to return to the depot.

Therefore, the formulation of the *i*th ($0 \le i \le M$) static subproblem of the DCCL model is given as

$$\min F_i = F_1 + F_2 + F_3 \tag{12}$$

$$\sum_{k=1}^{K}\sum_{w\in V_i} x_{wok}^i \le 1, \forall o \in C_i \tag{13}$$

$$\sum_{k=1}^{K} y_k^i - K \le 0 \tag{14}$$

$$\sum_{w\in V_i}\sum_{o\in C_i} x_{wok}^i \times dmd_o - Cap_k \le 0, 1\le k\le K \tag{15}$$

$$pt_o + st_o - dt_k(o) \le 0, \forall o \in C_i \tag{16}$$

$$a_o - pt_o \le 0, \forall o \in C_i \tag{17}$$

$$pt_o - b_o \le 0, \forall o \in C_i \tag{18}$$

$$x_{owk}^i, y_k^i \in \{0,1\}, \forall o, w \in V_i, 1 \le k \le K \tag{19}$$

where (12) is the objective function, i.e., minimization of the total distribution cost. Constraint (13) means that an order can be served no more than once. If an order is not served by any vehicle, it means that the order is rejected. Constraint (14) limits the maximum number of vehicles that can be used. Constraint (15) refers to the capacity constraint, meaning that the total demands of orders assigned to vehicle *k* in $sched_i$ cannot exceed Cap_k, and Cap_k is the remaining capacity of vehicle *k* when *k* begins to carry out distribution tasks of $sched_i$. Constraint (16) specifies that vehicle *k* cannot leave *o* before it finishes the service of *o*. Constraints (17) and (18) refer to the hard time window constraint, and $[a_o, b_o]$ is the service time window of *o*. Constraint (19) indicates the range of decision variables.

Finally, the complete formulation of the whole DCCL model is given as

$$\begin{aligned} F = {} & \sum_{k=1}^{K} y_k^M \times p_0 + \sum_{i=0}^{M}\sum_{k=1}^{K} y_k^i \\ & \times \left(\sum_{o\in V}\sum_{w\in V} x_{owk}^i \times dis_{ow}\right) \times p_1 \\ & + \sum_{o\in V\backslash 0}\left(1 - \sum_{i=0}^{M}\sum_{k=1}^{K}\sum_{w\in V} x_{wok}^i\right) \times p_2 \\ & + \sum_{k=1}^{K} y_k^M \times \left(\left(\sum_{i=0}^{M}\sum_{w\in V}\sum_{o\in V\backslash 0} x_{wok}^i \times ql_o \times dmd_o\right)\right. \\ & \left. + ql'_{rq(M,k)} \times rq(M,k)\right) \times p_3 \end{aligned} \tag{20}$$

$$\sum_{i=0}^{M}\sum_{k=1}^{K}\sum_{w\in V} x_{wok}^i \le 1, \forall o \in V\backslash 0 \tag{21}$$

$$\sum_{k=1}^{K} y_k^i - K \le 0, 0 \le i \le M \tag{22}$$

$$\sum_{i=0}^{M}\sum_{w\in V}\sum_{o\in V\backslash 0} x_{wok}^i \times dmd_o - maxCap \le 0, 1 \le k \le K \tag{23}$$

$$pt_o + st_o - dt_k(o) \le 0, \forall o \in V\backslash 0 \tag{24}$$

$$a_o - pt_o \le 0, \forall o \in V\backslash 0 \tag{25}$$

$$pt_o - b_o \le 0, \forall o \in V\backslash 0 \tag{26}$$

$$x_{owk}^i, y_k^i \in \{0,1\}, \forall o, w \in V, 1 \le k \le K, 0 \le i \le M \tag{27}$$

The final fitness of the complete scheduling result is calculated as (20), where *V* refers to the vertexes set of the depot and all orders to be scheduled in a working day. The first two rows of (20) refer to the calculation of the transportation cost, which also consider the transportation cost of going back to the depot. The third row of (20) refers to the calculation of the cost of the order rejection penalty. The last two rows of (20) are to calculate the cost of the quality loss of goods. Note that the quality loss that occurs during the return trip to the depot is also considered in $ql'_{rq(M,k)}$. In detail, the driving duration of going back to the depot is also included when calculating $ql'_{rq(M,k)}$. The constraint (21) means that each order can only be served at most once in a working day. The constraint (22) is the limitation of the vehicles that can be used. The constraint (23) shows that the total demand of orders that are served by vehicle *k* in a working day cannot exceed the maximum capacity (i.e., *maxCap*) of vehicle *k*. Similar to the static CCL model, the constraint (24) means that the departure time of vehicle *k* at the destination of order *o* cannot be earlier than the time that vehicle *k* finishes the service

Algorithm 1: Procedure of BACS.

```
   Input: $C_i$
   Output: $gb$: the historical best solution, $B$: the set of
           buffered orders
 1 Begin
   /* Adoption of BS                                        */
 2 $B \leftarrow getBufferedOrder(C_i)$ ;             // Algorithm 2
 3 $C_i \leftarrow C_i \setminus B$ ;
 4 Solution $initSol \leftarrow getInitSol(C_i)$ ;    // Algorithm S.1
 5 Pheromone Initialization using $initSol$;
 6 Solution $gb \leftarrow initSol$;
 7 $g \leftarrow 0$;
 8 while $cpuTime < T_{sched}$ do
 9     $g \leftarrow g + 1$;
10     for $j = 1 : NP$ do
11         Solution Construction for $ant_j$;
12         if $f_{gb} > f_{ant_j}$ then
13             $gb \leftarrow ant_j$;
14         Local update of pheromone;
       /* Adoption of PLS                                       */
15     if $g$ $mod$ $period == 0$ then
16         Improvement of $gb$ using BCRC ;       // Algorithm 3
17     Global update of pheromone;
18 End
```

Algorithm 2: Procedure of Buffering Strategy.

```
   Input: $C_i$
   Output: Set of buffered orders, $B$
 1 Begin
 2 Solution $sol \leftarrow getInitSol(C_i)$ ;        // Algorithm S.1
 3 $B \leftarrow \varnothing$ ;
 4 $R \leftarrow R$ of $sol$ ;
 5 $T_{buffer} \leftarrow T_i + delta * lenSlice$;
 6 for $k = 1 : K$ do
 7     for $j = 1 : |R_k| - 1$ do
 8         $r \leftarrow$ the $j^{th}$ order served in $R_k$;
 9         if $pt_r \geq T_{buffer}$ then
10             $B \leftarrow B \vee r$;
11 End
```

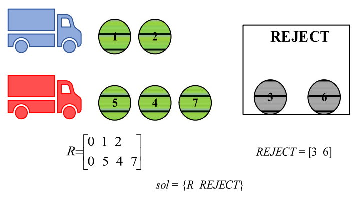

Fig. 3. Schematic diagram of solution encoding.

of order o. The constraints (25) and (26) indicate the satisfaction of the hard time window constraint. The constraint (27) shows that the two decision variables are binary.

III. Buffer-Based Ant Colony System

To effectively solve the DCCL model, the BACS approach is proposed in this paper. In BACS, the BS is to buffer orders that can be delivered later. Scheduling of buffered orders is then postponed, and the following steps of BACS are to determine the distribution sequence for orders that are not buffered. Moreover, to make full use of the solutions obtained in the search process, the PLS is proposed and integrated into BACS.

The procedure of BACS for s_i is shown as **Algorithm 1**. When $sched_i$ begins, the set of orders to be scheduled, i.e., C_i, is adopted as the input of BACS. The historical best solution gb and the set of buffered orders, B, are returned by BACS after T_{sched} minutes (i.e., the executing CPU time of BACS). The procedure of BACS can be decomposed into the following four components.

1) Buffering Strategy: As line 2 shows, the BS is applied to C_i, and orders being buffered are stored in B. The pseudo-code of BS is shown as **Algorithm 2** in Section III-B. Later on, the buffered orders are eliminated from C_i, and the following operators are conducted on orders retained in C_i.

2) Pheromone Setting: The pheromone setting consists of two parts, pheromone initialization and pheromone update. The pheromone initialization is conducted in line 5, and the initial solution *initSol* is constructed using **Algorithm S.1** in the supplementary material. The pheromone update includes the local update and the global update, which are shown in lines 14 and 17, respectively. More detailed descriptions of the pheromone setting are given in Section III-C.

3) Solution Construction: This is the main component of the ACS approach. As shown in lines 10 to 13, the solution of each ant is reconstructed in every generation, which will be further described in Section III-D. Besides, the historical best solution *gb* can be replaced by the solution of an ant if the latter has a better fitness.

4) Periodic Learning Strategy: Lines 15 and 16 show that the PLS is conducted at the end of every *period* generations. Besides, the application of best cost route crossover (BCRC) [30] operator is described as **Algorithm 3** in Section III-E.

The remaining part of this section firstly introduces the solution encoding format of BACS and then illustrates the major components of BACS in detail. Finally, the complete procedure for using BACS in DCCL scheduling and the time complexity analysis of BACS are given.

A. Solution Encoding

Considering that orders to be scheduled in $sched_i$ are either rejected or served by a vehicle, the solution encoding should satisfy two requirements. The first requirement is to show that whether an order is rejected, the second one is to record the distribution sequence for orders that are not rejected. Therefore, a solution is encoded as a special structure with two components. The first component is denoted as R, which records the distribution routes for vehicles. The second component is denoted as *REJECT*, which is the set of orders that are rejected.

For clarity, a schematic diagram of solution encoding is shown in Fig. 3. Given seven orders to be scheduled, i.e., $C_i = \{1, 2, 3, 4, 5, 6, 7\}$, and two reefer vehicles departing from the depot, a feasible solution can be shown as the *sol* in Fig. 3. It is presented that R is a two-dimensional array with K rows, and K is the maximum number of available vehicles. Herein, $K = 2$. The first row of R is the distribution route of vehicle 1, meaning that vehicle 1 distributes order 1 and order 2 in sequence after it leaves the depot. Similarly, the second row refers to the distribution route of vehicle 2, which means that order 5, order 4, and order 7 are distributed by vehicle 2 successively. Note that the first element of a row is the departure place of the corresponding vehicle. Besides, as shown in Fig. 3, both order 3 and order 6 are rejected. Thus, *REJECT* is determined as a one-dimensional array, and elements of *REJECT* are 3 and 6.

B. Buffering Strategy

Owing to the uncertain arrival frequency of orders, there may be a lot of orders to be scheduled in $sched_i$. Even though the optimal distribution sequence of these orders can be determined if sufficient scheduling time is allowed, these orders may not be served in this sequence. This is because that there are so many orders that the distribution duration would be long and therefore the next scheduling, i.e., $sched_{i+1}$, is likely to be invoked before the vehicles can finish the distribution tasks of $sched_i$. Hence, the orders that are not served will be scheduled again, and the previous scheduling result of these orders may become invalid. In this sense, the orders that are scheduled but cannot be served before the next scheduling is finished are regarded as non-urgent, and the consideration of these orders in the current scheduling is redundant to some extent. Furthermore, the redundant scheduling of these non-urgent orders may reduce the efficiency of a scheduling approach, and then is not conducive to solving the DCCL scheduling problem in an online manner that only allows limited scheduling time.

Hence, to improve the efficiency of the scheduling approach, the BS is designed to buffer the non-urgent orders that ought to be delivered later. In this way, it is more likely for a scheduling approach to find the optimal solution within the same execution time (i.e., T_{sched}) because the number of orders to be scheduled is reduced. However, the scheduling of orders that should be delivered later is not absolutely meaningless because the global information provided by these orders is beneficial for a scheduling approach to avoid being trapped into the local optima. Therefore, the tradeoff between global information and problem scale should be considered. In other words, it is critical for the BS to determine the number of orders to be buffered in $sched_i$. If too many orders are buffered, a scheduling approach is very likely to be trapped into local optima. On the contrary, if few orders are buffered, it cannot improve the efficiency of the scheduling approach.

The procedure of BS is shown as **Algorithm 2**, which can be concluded as two steps. The first step is to approximate the delivery time of the orders by scheduling all orders using a heuristic algorithm named ranking time windows based nearest neighbor algorithm proposed in [31], which is shown as line 2 in **Algorithm 2**. The detail of this heuristic algorithm is given in **Algorithm S.1** in the supplementary material. The second step is to determine whether an order can be buffered by comparing its delivery time with T_{buffer}. Herein, T_{buffer} is a threshold, and its value depends on a parameter *delta* and the length of a time slice, *lenSlice* (i.e., (T_{end} – T_{start})/*M*). If an order cannot be delivered before T_{buffer}, it is buffered. Otherwise, the order is retained. The second step is illustrated in lines 5 to 10.

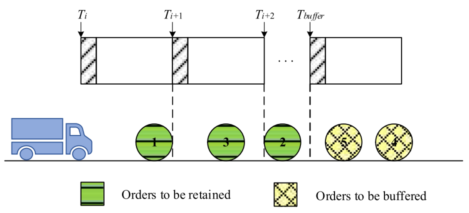

Fig. 4. Schematic diagram of BS.

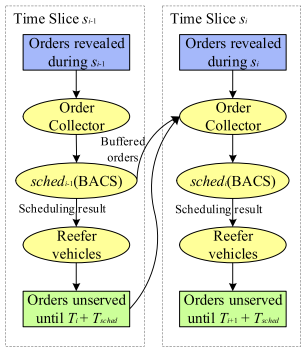

Fig. 5. Framework of DCCL scheduling using BACS.

For clarity, an example of BS is shown in Fig. 4. Specifically, given $C_i = \{1, 2, 3, 4, 5\}$, the solution shown in Fig. 4 presents that order 1, order 3, and order 2 can be delivered before T_{buffer}, while order 5 and order 4 are delivered after T_{buffer}. Hence, order 5 and order 4 are buffered, while order 1, order 3, and order 2 are retained and scheduled in $sched_i$.

Besides, it should be highlighted that the buffered orders are not rejected but just temporarily neglected in $sched_i$, and these orders will be scheduled in the next scheduling. Therefore, the framework of DCCL scheduling with BACS as the optimizer is specifically shown in Fig. 5, which is derived from Fig. 2 but is different from Fig. 2. Note that as $sched_M$ is the last scheduling within a working day, there is no scheduling that can accept the buffered orders from $sched_M$, so the BS is not used in $sched_M$.

C. Pheromone Setting

1) Pheromone Initialization: To start with, an initial solution is constructed by using the ranking time windows based nearest neighbor algorithm (i.e., **Algorithm S.1** in the supplementary material).

In this algorithm, all orders are firstly sorted in the increasing order of the lower bound of their service time windows. For orders that have the same lower bound, they are sorted ascendingly according to the upper bound of their time windows. Besides, the route of a vehicle is initialized with the departure place of the vehicle. Subsequently, the orders are inserted to the end of a route one by one following the sorted order. Note that each order can only be appended to the route of a vehicle that can serve the order without constraint violation. If more than one vehicle can satisfy this condition, the vehicle that can serve the order with the least additional distance is selected. Nevertheless, if no feasible vehicle can be found, the order is rejected. Finally, with the initial solution, *initSol*, the initial value of pheromone can be determined as

$$\tau_0 = (|C_i| \times f_{initSol})^{-1} \tag{28}$$

where $f_{initSol}$ is the fitness of *initSol*.

2) Pheromone Update: There are two kinds of pheromone update behaviors in BACS. The first one is the local update, which is designed to maintain the diversity of the colony. When an ant finishes the solution construction, it will conduct the local update of pheromone on edges it passed by. The local update rule is shown as

$$\tau(o, w) = (1-\rho) \times \tau(o, w) + \rho \times \tau_0 \tag{29}$$

where ρ is a parameter, *o* and *w* are two vertexes visited successively by the same vehicle. The variable $\tau(o, w)$ refers to the pheromone deposited on the edge between *o* and *w*. The global update is conducted on edges traversed by the historical best solution, named *gb*, at the end of every generation, and is represented as

$$\tau(o, w) = (1-\varepsilon) \times \tau(o, w) + \varepsilon \times (f_{gb})^{-1} \tag{30}$$

where ε is a parameter, f_{gb} is the fitness of *gb*.

D. Solution Construction

To construct a solution for an ant, the *R* of the solution is first initialized as a two-dimensional array, and the kth row of *R* is initialized as an array with the departure place of vehicle *k* as its first element. The *REJECT* of the solution is initialized as an empty set. Besides, a vehicle sequence *Vseq* is generated by first sorting the indexes of vehicles in use randomly and then appending the indexes of the other vehicles that are not in use sequentially.

Assume that *o* is the last vertex visited by vehicle *k* (*k* is the first element in *Vseq*), the next order, *w*, to be served after *o* can be determined through the state transition rule that is represented as

$$w = \begin{cases} \arg\max\limits_{j \in S} \left\{\tau(o, j) \times \eta(o, j)^{\beta}\right\}, \text{ if } q \leq q_0 \\ \text{roulette wheel selection, otherwise} \end{cases} \tag{31}$$

where β is a parameter. *S* is the set of orders that have not been assigned to any vehicles yet and can be served by vehicle *k* without violating any constraints. If *S* is empty, the next vehicle of *k* in *Vseq*, denoted as *k*', is selected. Then, the next order is determined according to the last vertex in the route of *k*' and the state transition rule, and this order will be served by vehicle *k*'. Note that if no orders can be served by any vehicle, the remaining orders that are unassigned should be inserted to the *REJECT* of the solution, and the solution construction process is terminated. The q $(0 \leq q \leq 1)$ in (31) is a random number and q_0 is a parameter. If q does not exceed q_0, *w* is determined as the order with the maximum product of $\tau(o, j)$ (i.e., pheromone) and $\eta(o, j)^{\beta}$ (i.e., heuristic information). Otherwise, *w* is determined through the roulette wheel selection, and the probability for order *j* to be selected is

$$p(o, j) = \begin{cases} \frac{\tau(o,j) \times \eta(o,j)^{\beta}}{\sum_{u \in S} \tau(o,u) \times \eta(o,u)^{\beta}}, \text{ if } j \in S \\ 0, \text{ otherwise} \end{cases} \tag{32}$$

The heuristic information between *o* and *w* is designed as

$$\eta(o, w) = \frac{1}{pt_w - dt_k(o)} \times \left(1 + e^{-(b_w - dt_k(o) - dur_{ow})}\right) \times \frac{dmd_w}{pt_w - dt_k(0)} \tag{33}$$

The first term of (33) is used to measure the time required for waiting and driving from *o* to *w*, where pt_w is the time that goods of *w* start to be delivered, and $dt_k(o)$ is the time that vehicle *k* leaves *o*. The second term is to measure the urgency of *w*, where b_w is the upper bound of *w*'s service time window. The third term tends to make vehicle *k* serve orders with greater demand first under the consideration of both pt_w and the time that *k* leaves the depot, where dmd_w is the demand of *w*. In this way, vehicle *k* may prefer to serve orders with greater demand, especially at the early stage after it leaves the depot, which is beneficial to reduce the total quality loss of goods in the long distribution duration.

E. Periodic Learning Strategy

Since the global update of pheromone is only conducted on edges traversed by the historical best solution, *gb*, it is still likely for BACS to fall into the local optima when *gb* does not change for a long time. To help BACS jump out of the local optima, the PLS is proposed.

The main idea of PLS is to divide the iterative search process of BACS into several periods with equal length, and the length of a period is determined by a parameter, *period*. In the last generation of a period, the best solution found in the current period, i.e., the periodic best solution, *pgb*, and the historical best solution, *gb*, are selected, and the BCRC operator is applied to these two solutions to obtain two offspring. The *gb* can be replaced by any offspring that has better performance than *gb*.

By referring to [30], the procedure of BCRC operator is shown as **Algorithm 3**. As shown in lines 2 to 5, two routes named rt_1 and rt_2 are randomly selected from *gb* and *pgb*, respectively, then orders served in rt_1 of *gb* are deleted from the routes of *pgb*, and orders served in rt_2 of *pgb* are deleted from the routes of *gb*. Later on, two new solutions are generated by reinserting the orders that are deleted or rejected to the most feasible position of their parent solution, which is illustrated in lines 17 to 22. Note that the most feasible position for an order refers to the feasible position that can serve the order with the minimal total distribution cost. Finally, *gb* is updated if any new solution outperforms *gb*.

F. Complete Procedure of BACS for DCCL Scheduling

This part describes how to solve the whole DCCL scheduling problem using BACS and combine the scheduling result of each time slice into the complete scheduling result of a whole working day. The complete procedure of the BACS approach for DCCL scheduling is shown as **Algorithm 4**, including the scheduling

Algorithm 3: Procedure of BCRC.

```
Input: historical best solution $gb$, periodic best solution $pgb$
Output: $gb$
Begin
/* $rt_1$ and $rt_2$ are two routes randomly selected from
   $gb$ and $pgb$, respectively */
$isr_1 \leftarrow$ orders served in $rt_2$ of $pgb$;
$isr_2 \leftarrow$ orders served in $rt_1$ of $gb$ ;
$gbR \leftarrow R$ of $gb$ without orders in $isr_1$ ;
$pgbR \leftarrow R$ of $pgb$ without orders in $isr_2$ ;
$isr_1 \leftarrow isr_1 \cup REJECT$ of $gb$;
$isr_2 \leftarrow isr_2 \cup REJECT$ of $pgb$;
Solution $sol_1, sol_2$;
$R$ of $sol_1 \leftarrow gbR$, $REJECT$ of $sol_1 \leftarrow isr_1$;
$R$ of $sol_2 \leftarrow pgbR$, $REJECT$ of $sol_2 \leftarrow isr_2$;
for $ind = 1 : 2$ do
    Solution $sol$, $isr \leftarrow \varnothing$;
    if $ind == 1$ then
        $sol \leftarrow sol_1$, $isr \leftarrow isr_1$;
    else
        $sol \leftarrow sol_2$, $isr \leftarrow isr_2$;
    foreach order $o \in isr$ do
        if $o$ cannot be served by any vehicle then
            continue;
        Find out the most feasible position that can serve $o$
          from $R$ of $sol$;
        Insert $o$ to the most feasible position;
        $REJECT$ of $sol \leftarrow REJECT$ of $sol \setminus o$;
    if $f_{sol} < f_{gb}$ then
        $gb \leftarrow sol$;
End
```

for each time slice (lines 7 to 17) and the update of both the vehicle state and the complete scheduling result (lines 18 to 34).

At first, the scheduling result of a working day, *SR*, is initialized as {*R*, *REJECT*}. Note that *R* of *SR* is the set of the *K* vehicle routes that start from the depot, and *REJECT* of *SR* is an empty set that is used to record the rejected orders. Line 6 means that the scheduling is carried out $M+1$ times in the whole DCCL scheduling procedure. The scheduling process of i^{th} time slice is displayed as lines 7 to 17. Line 8 means that when *i* is 0, C_i consists of orders revealed in s_0. Lines 9 to 12 show that the remaining capacity of vehicle *k*, denoted as Cap_k, is initialized as its maximum capacity, the departure place of vehicle *k*, dp_k, is set as the depot, and the time that *k* departs from dp_k, named $dt_k(dp_k)$, is set as T_0+T_{sched}. Lines 13 to 16 mean that when *i* exceeds 0, C_i consists of three parts. The first one is the orders revealed in s_i, the second one is the orders that are scheduled in $sched_{i\text{-}1}$ but remain unserved until T_i+T_{sched}, and the third one is the orders being buffered in $sched_{i\text{-}1}$. As for Cap_k, dp_k, and $dt_k(dp_k)$ in the case that $i > 0$, they are determined in the upcoming steps. In line 17, the BACS is used, and the historical best solution of BACS and the buffered orders are returned to *sol* and *B*, respectively.

The update of both the vehicle state (including Cap_k, dp_k, and $dt_k(dp_k)$) and the scheduling result for the whole working day, *SR* is shown in lines 18 to 34. Lines 20 to 21 mean that the route of vehicle *k* in *SR* and the state of vehicle *k* are not updated if the vehicle *k* cannot leave for the destination of the first order assigned by *sol* of $sched_i$ before $sched_{i+1}$ terminates.

Algorithm 4: Complete Procedure of BACS for DCCL Scheduling.

```
Input: Orders
Output: Scheduling Result, $SR$
1  Begin
2  $SR \leftarrow \{R, REJECT\}$;
3  $R$ of $SR \leftarrow K$ routes that depart from the depot;
4  $REJECT$ of $SR \leftarrow \varnothing$;
5  $B \leftarrow \varnothing$;
   /* Scheduling for $i^{th}$ time slice */
6  for $i = 0 : M$ do
7      if $i == 0$ then
8          $C_i \leftarrow$ orders revealed in $s_0$;
9          for $k = 1 : K$ do
10             $Cap_k \leftarrow maxCap$;
11             $dp_k \leftarrow$ depot;
12             $dt_k(dp_k) \leftarrow T_0 + T_{sched}$;
13     if $i > 0$ then
14         $U_i \leftarrow$orders scheduled in $sched_{i-1}$ but remain
             unserved until $T_i + T_{sched}$;
15         $C_i \leftarrow$orders revealed in $s_i$;
16         $C_i \leftarrow C_i \cup U_i \cup B$;
       /* Optimization of orders in $C_i$ */
17     $sol, B \leftarrow$ BACS($C_i$);
       /* Update $SR$ and the state of vehicles */
18     $R^* \leftarrow R$ of $sol$, $REJECT^* \leftarrow REJECT$ of $sol$;
19     for $k = 1 : K$ do
20         if $dt_k(dp_k) \geq T_{i+1} + T_{sched}$ and $i < M$ then
21             continue;
22         $dp_k \leftarrow$ last position in $R_k$ of $SR$;
23         $dt_k(dp_k) \leftarrow T_{i+1} + T_{sched}$;
24         for $j = 1 : |R^*_k| - 1$ do
25             $r \leftarrow$ the $j^{th}$ position of $R^*_k$;
26             $R_k$ of $SR \leftarrow R_k$ of $SR \vee r$;
27             $Cap_k \leftarrow Cap_k - dmd_r$;
28             if $(pt_r + st_r \geq T_{i+1} + T_{sched} \wedge i < M)$ or
                 $j == |R^*_k| - 1$ then
29                 $dp_k \leftarrow r$;
30                 $dt_k(dp_k) \leftarrow \max\{pt_r + st_r, T_{i+1} + T_{sched}\}$;
31                 break;
32         if $i == M$ then
33             $R_k$ of $SR \leftarrow R_k$ of $SR \vee$ depot;
34     $REJECT$ of $SR \leftarrow REJECT$ of $SR \vee REJECT^*$;
35 End
```

Otherwise, as shown in lines 24 to 26, orders that can be served before $sched_{i+1}$ is finished should be appended to the *R* of *SR*. In addition, the Cap_k, dp_k, and $dt_k(dp_k)$ for $sched_{i+1}$ are updated accordingly in lines 27 to 31. Besides, as shown in lines 32 to 33, in the case that *i* equals *M* (i.e., the last scheduling), the depot vertex is appended to the route of each vehicle in *SR*, meaning that each vehicle should return to the depot after finishing all the distribution tasks. Thus, the transportation cost and quality loss of remaining goods that occur during the return trip to the depot are included when the fitness of *SR* is calculated. The orders rejected in $sched_i$ are merged with the *REJECT* of *SR* in line 34. Finally, *SR* is returned.

G. Time Complexity Analysis of BACS

The time complexity of BACS is analyzed as follows. To begin with, as shown in **Algorithm 2**, the BS includes two steps with time complexity of $O(|C_i|\times K)$. Hence, the time complexity of the BS is $O(2|C_i|\times K)$. Due to the BS, the number of orders that should be scheduled in the following steps is reduced to $|C_i'|$. Note that $|C_i'|$ is the difference between $|C_i|$ and the number of buffered orders. Thus, the time complexity of initial solution construction is $O(|C_i'|\times K)$. Besides, as the pheromone can be deposited on all edges among vertexes in the union set of C_i' and *dp* (i.e., the departure places of K vehicles), the time complexity of pheromone initialization is $O((|C_i'|+K)^2)$. Considering that the distribution sequence of each order is determined through the state transition rule with time complexity of $O(|C_i'|)$, the time complexity of solution construction (i.e., determining the distribution sequence for all orders) is $O(|C_i'|^2)$. Hence, for the solution construction of the whole colony with the size of NP in NG generations, the time complexity is $O(NG\times NP\times|C_i'|^2)$. As for the PLS, it includes the BCRC local search operator with time complexity of $O(2|C_i'|^2)$, and it is carried out every *period* generations. Thus, the time complexity of the PLS is $O(NG/period\times 2|C_i'|^2)$. Besides, the time complexity of pheromone update (including local update and global update) is $O(NG\times NP\times|C_i'|\times K)$. Therefore, the time complexity of the whole BACS is $O(2|C_i|\times K+|C_i'|\times K+(|C_i'|+K)^2+NG\times NP\times|C_i'|^2+NG/period\times 2|C_i'|^2+NG\times NP\times|C_i'|\times K)$, i.e., $O(|C_i|\times K+K^2+NG\times NP\times|C_i'|^2+NG/period\times|C_i'|^2+NG\times NP\times|C_i'|\times K)$.

IV. Experimental Studies

A. Test Instances

The test instances used in experiments are based on the benchmark instances provided by Solomon [32] and Homberger [33] and are further developed by considering the DCCL issues. The benchmark instances in [32] and [33] can be divided into 6 categories, which are denoted as C1, C2, R1, R2, RC1, and RC2. The uppercase letters C, R, RC are used to indicate the vertexes distribution characteristic of instances. In detail, C means clustered distribution, R means random distribution, and RC means the mixture of clustered distribution and random distribution. Besides, only a few orders can be served by a vehicle in instances of C1, R1, and RC1 due to the capacity limitation, while the case is opposite (i.e., the capacity is large) in instances of C2, R2, and RC2.

To use the benchmark instances in the DCCL model, dynamism should be introduced to the benchmark instances. Hence, a new attribute named ACCEPT TIME is added to each vertex, which is used to record the revealed time of order. For order o in an instance, the ACCEPT TIME of o is a random number ranging from 0 to a_o. Besides, the first 50% of vertexes in an instance are set as the static orders (i.e., orders revealed in time slice s_0), and the ACCEPT TIME of these orders are set as –1. In this way, we obtain the DCCL benchmark instances with dynamic characteristics.

Moreover, for thorough comparisons, instances with various scales and categories are contained in the test instances, which are shown as the 48 DCCL instances in Table S.I in the supplementary material. Herein, the name of an instance is explained by taking S50–C101 as an example. S50 means that the scale of the instance is 50 (i.e., with 50 orders), C1 refers to the category of the instance, and 01 indicates that the instance is the first instance of category C1.

B. Experimental Settings

1) Compared Approaches: To evaluate the performance of BACS on DCCL optimization, four ACO-based dynamic approaches are adopted as compared approaches. Moreover, BACS is also compared with a heuristic approach that is proposed to solve DVRP. Brief introductions of the compared approaches are shown as follows.

a) ACS [23]: To solve the DCCL model, the original ACS is invoked at the end of each time slice. The distribution sequence of orders to be scheduled in $sched_i$ is determined by ACS through three major steps, i.e., pheromone initialization, solution construction, and pheromone update.

b) ACS-DVRP [20]: ACS-DVRP is an approach developed by introducing a pheromone conservation procedure to the ACS approach. With the pheromone conservation procedure, the experience of previous scheduling can be exploited in the form of pheromone by the following scheduling. Specifically, if order o and order w are scheduled in both $sched_{i-1}$ and $sched_i$, ACS-DVRP initializes the pheromone between o and w following

$$\tau(o,w) = (1-\gamma)\times\tau^{i-1}(o,w)+\gamma\times\tau_0 \qquad (34)$$

where $\tau^{i-1}(o, w)$ refers to the pheromone accumulated between o and w when $sched_{i-1}$ is finished, τ_0 is the initial value of pheromone in $sched_i$.

c) MACS-VRPTW [31]: The multiple ACS for VRP with time windows (MACS-VRPTW) is an ACS-based approach that is firstly proposed in [34] for solving the VRPTW and is adopted in [31] to solve DVRP. Two colonies are employed in the MACS-VRPTW, where the first one is used to minimize the total distribution cost, and the second one is used to minimize the number of vehicles being used. When MACS-VRPTW is used to solve the DCCL model, the two colonies will be reinitialized once a solution that outperforms the historical best solution is found by the second colony.

d) P-ACO [21]: A memory with limited capacity is used in P-ACO to store the best solution obtained in each generation. If the capacity of memory has been run out, the solution that enters the memory first will be removed to make room for the best solution obtained in the following generations. That is, the best solution of each generation enters and leaves the memory in a first-in-first-out manner. The pheromone on edges that are traversed by a solution is increased when the solution enters the memory and is decreased when the solution leaves the memory. Besides, when the scheduling environment changes from $sched_{i-1}$ to $sched_i$, the KeepElitist strategy [35] is used to repair the solutions in the memory. Specifically, to repair a solution, the orders that have been delivered are deleted, and the new orders are inserted to the most feasible position of the distribution routes.

TABLE I
COMPARISON RESULTS ON FITNESS OF COMPARED APPROACHES AND BACS ON INSTANCES WITH SCALES OF 50 AND 100

Instance	ACS	ACS-DVRP	MACS-VRPTW	P-ACO	ALNS	BACS
S50–C101	1.15E+04 (3+)	**1.08E+04** (1–)	1.29E+04 (6+)	1.16E+04 (5+)	1.14E+04 (2≈)	1.15E+04 (4)
S50–C102	1.05E+04 (2–)	1.06E+04 (5≈)	1.06E+04 (4–)	1.08E+04 (6+)	**1.03E+04** (1≈)	1.06E+04 (3)
S50–C201	7.90E+03 (2≈)	7.90E+03 (2≈)	**6.94E+03** (1–)	7.92E+03 (5+)	7.97E+03 (6+)	7.90E+03 (2)
S50–C202	6.72E+03 (5+)	6.71E+03 (4+)	8.06E+03 (6+)	6.69E+03 (3+)	**6.62E+03** (1≈)	6.62E+03 (2)
S50–R101	1.59E+04 (2≈)	1.60E+04 (3≈)	1.60E+04 (5≈)	**1.59E+04** (1–)	1.62E+04 (6+)	1.60E+04 (4)
S50–R102	1.50E+04 (5+)	1.44E+04 (2+)	1.55E+04 (6+)	1.48E+04 (4+)	1.45E+04 (3+)	**1.37E+04** (1)
S50–R201	1.13E+04 (5≈)	1.11E+04 (4≈)	1.09E+04 (2≈)	1.16E+04 (6+)	1.10E+04 (3≈)	**1.07E+04** (1)
S50–R202	8.63E+03 (3+)	8.62E+03 (2+)	8.64E+03 (4+)	8.64E+03 (5+)	8.78E+03 (6+)	**8.48E+03** (1)
S50–RC101	1.29E+04 (3–)	1.30E+04 (4≈)	1.30E+04 (5≈)	1.26E+04 (2≈)	**1.20E+04** (1–)	1.31E+04 (6)
S50–RC102	1.09E+04 (2–)	1.10E+04 (3–)	1.19E+04 (5≈)	**1.04E+04** (1–)	1.15E+04 (4–)	1.22E+04 (6)
S50–RC201	1.10E+04 (3≈)	1.09E+04 (2≈)	1.22E+04 (5+)	1.26E+04 (6+)	1.21E+04 (4+)	**1.07E+04** (1)
S50–RC202	9.54E+03 (3+)	9.53E+03 (2+)	9.90E+03 (4+)	1.01E+04 (5+)	1.06E+04 (6+)	**9.40E+03** (1)
S100–C101	2.38E+04 (2≈)	2.40E+04 (4+)	2.42E+04 (5+)	2.39E+04 (3+)	2.50E+04 (6+)	**2.32E+04** (1)
S100–C102	**2.20E+04** (1–)	2.21E+04 (3–)	2.21E+04 (2≈)	2.23E+04 (5≈)	2.29E+04 (6+)	2.22E+04 (4)
S100–C201	2.36E+04 (3≈)	2.36E+04 (3≈)	2.50E+04 (6+)	**2.36E+04** (1≈)	2.41E+04 (5+)	2.36E+04 (2)
S100–C202	1.60E+04 (3+)	1.57E+04 (2≈)	1.65E+04 (5+)	1.67E+04 (6+)	1.61E+04 (4+)	**1.53E+04** (1)
S100–R101	3.01E+04 (3+)	3.02E+04 (4+)	3.12E+04 (6+)	2.98E+04 (2+)	3.12E+04 (5+)	**2.88E+04** (1)
S100–R102	1.86E+04 (2+)	1.89E+04 (3+)	1.90E+04 (5+)	1.90E+04 (4+)	1.99E+04 (6+)	**1.81E+04** (1)
S100–R201	1.16E+04 (4+)	1.13E+04 (2+)	1.14E+04 (3+)	1.17E+04 (5+)	1.18E+04 (6+)	**1.09E+04** (1)
S100–R202	1.23E+04 (3+)	1.22E+04 (2≈)	1.28E+04 (6+)	1.26E+04 (4+)	1.27E+04 (5+)	**1.20E+04** (1)
S100–RC101	**1.78E+04** (1≈)	1.81E+04 (2≈)	1.86E+04 (6≈)	1.82E+04 (4≈)	1.82E+04 (3≈)	1.83E+04 (5)
S100–RC102	1.28E+04 (3≈)	1.31E+04 (4≈)	**1.27E+04** (1≈)	1.28E+04 (2≈)	1.46E+04 (6+)	1.32E+04 (5)
S100–RC201	1.27E+04 (5+)	1.26E+04 (3+)	1.24E+04 (2≈)	1.27E+04 (4+)	1.37E+04 (6+)	**1.21E+04** (1)
S100–RC202	8.57E+03 (2≈)	8.62E+03 (3≈)	9.57E+03 (5+)	8.91E+03 (4+)	9.74E+03 (6+)	**8.27E+03** (1)
num of best	2	1	2	3	3	**13**
sum of rank	70	69	105	93	107	**56**
num of +/≈/–	11/9/4	9/12/3	14/8/2	17/5/2	17/5/2	\

e) ALNS [36]: the adaptive large neighborhood search (ALNS) heuristic approach is also adopted in our experiments because it is designed for DVRP optimization. To obtain a better solution, various pairs of destroy and repair heuristic operations are randomly selected according to the selection probability and then applied to a basic solution. Note that the selection probabilities of the destroy/repair operations are periodically updated in the search process according to the performance of the operations.

2) Parameter Settings: The parameters of BACS are set following the suggestions of both [23] and the results of parameter investigation in Section IV-E. Specifically, the ACS-related parameters in BACS are set as $NP = 10$, $q_0 = 0.9$, $\rho = 0.1$, $\varepsilon = 0.1$, and $\beta = 2$, and the *delta* and the *period* used in BS and PLS are set as 2 and 15, respectively.

As for compared approaches, the parameter settings basically follow the recommendation of the corresponding references. Hence, the parameters of compared approaches are not listed due to the limited space of this paper. Note that to ensure that ALNS can repair solutions within reasonable execution time, the number of orders to be removed by the destroy operators is a random number ranging from 1 to 5. Besides, the repair operators of ALNS are conducted on both the removed orders and rejected orders. The same heuristic information is provided to all ACO-based approaches except MACS-VRPTW since it has its own heuristic information. Furthermore, the pheromone of ACS, ACS-DVRP, and MACS-VRPTW are initialized in the same way as BACS.

Considering that the execution time, i.e., T_{sched}, for instances with various scales should be various, the value of T_{sched} is determined as 0.3, 0.5, 1.0, and 1.5 minutes for instances with scales of 50, 100, 200, and 400, respectively. Besides, *M* is set as 5 for each instance. To alleviate the influence of stochastic factors, all approaches are run 10 times independently on each instance, and the average results are used for comparison. All approaches are implemented using C++ and run on a PC with Intel i5-7400 CPU and 8.00 GB RAM.

C. Comparison With Other Approaches

The average fitness of BACS and all compared approaches on instances with scales of 50 and 100 and instances with scales of 200 and 400 are shown in Tables I and II, respectively. The ranking of each approach based on fitness and the statistical test results are recorded in the brackets. Herein, the Wilcoxon rank-sum test is adopted for the statistical test, and the significance level is 0.05. The statistical test results that are shown with the '+', '≈', or '–' represent that BACS has significantly better, similar, or significantly worse performance when compared with another approach. Besides, the best results are shown in **boldface**.

To begin with, it can be observed from Table I that BACS performs best on 13 of 24 instances with scales of 50 and 100,

TABLE II
COMPARISON RESULTS ON FITNESS OF COMPARED APPROACHES AND BACS ON INSTANCES WITH SCALES OF 200 AND 400

Instance	ACS	ACS-DVRP	MACS-VRPTW	P-ACO	ALNS	BACS
S200–C101	5.45E+04 (3+)	5.45E+04 (3+)	5.42E+04 (2+)	5.49E+04 (5+)	6.22E+04 (6+)	**5.34E+04 (1)**
S200–C102	3.88E+04 (4≈)	3.86E+04 (3≈)	**3.82E+04** (1≈)	4.63E+04 (5+)	4.68E+04 (6+)	3.83E+04 (2)
S200–C201	4.77E+04 (4+)	4.75E+04 (2+)	4.92E+04 (5+)	4.76E+04 (3+)	4.92E+04 (6+)	**4.62E+04 (1)**
S200–C202	3.06E+04 (2+)	3.06E+04 (3+)	3.12E+04 (5+)	3.12E+04 (4+)	3.32E+04 (6+)	**2.95E+04 (1)**
S200–R101	5.73E+04 (4+)	5.72E+04 (3+)	5.85E+04 (5+)	5.68E+04 (2≈)	6.44E+04 (6+)	**5.50E+04 (1)**
S200–R102	5.10E+04 (4≈)	5.12E+04 (5≈)	5.06E+04 (2≈)	**5.05E+04** (1≈)	5.74E+04 (6+)	5.08E+04 (3)
S200–R201	4.31E+04 (5+)	4.28E+04 (2+)	4.31E+04 (3+)	4.31E+04 (4+)	4.49E+04 (6+)	**4.13E+04 (1)**
S200–R202	3.53E+04 (4+)	3.52E+04 (3+)	3.48E+04 (2+)	3.53E+04 (5+)	3.61E+04 (6+)	**3.35E+04 (1)**
S200–RC101	6.46E+04 (2+)	6.48E+04 (3+)	6.69E+04 (5+)	6.56E+04 (4+)	7.24E+04 (6+)	**6.31E+04 (1)**
S200–RC102	4.70E+04 (3≈)	**4.66E+04** (1≈)	4.69E+04 (2≈)	5.23E+04 (6+)	5.16E+04 (5+)	4.72E+04 (4)
S200–RC201	6.74E+04 (4≈)	6.74E+04 (3≈)	6.98E+04 (5+)	6.73E+04 (2≈)	7.08E+04 (6+)	**6.71E+04 (1)**
S200–RC202	4.77E+04 (3≈)	4.76E+04 (2≈)	5.27E+04 (6+)	4.93E+04 (5+)	4.92E+04 (4+)	**4.73E+04 (1)**
S400–C101	1.11E+05 (2+)	1.11E+05 (2+)	1.11E+05 (4+)	1.34E+05 (5+)	1.53E+05 (6+)	**1.10E+05 (1)**
S400–C102	8.31E+04 (2≈)	8.36E+04 (3≈)	8.41E+04 (4≈)	1.12E+05 (5+)	1.24E+05 (6+)	**8.29E+04 (1)**
S400–C201	8.30E+04 (2+)	8.34E+04 (3+)	8.48E+04 (4+)	9.29E+04 (5+)	9.73E+04 (6+)	**8.12E+04 (1)**
S400–C202	5.84E+04 (2≈)	5.86E+04 (3≈)	6.19E+04 (4+)	7.03E+04 (6+)	6.96E+04 (5+)	**5.79E+04 (1)**
S400–R101	1.33E+05 (3+)	1.33E+05 (4+)	1.31E+05 (2+)	1.49E+05 (5+)	1.66E+05 (6+)	**1.29E+05 (1)**
S400–R102	9.85E+04 (3≈)	1.00E+05 (4+)	**9.72E+04** (1≈)	1.21E+05 (5+)	1.37E+05 (6+)	9.82E+04 (2)
S400–R201	9.83E+04 (2≈)	9.94E+04 (3+)	1.00E+05 (4+)	1.03E+05 (5+)	1.05E+05 (6+)	**9.72E+04 (1)**
S400–R202	8.16E+04 (2+)	8.24E+04 (3+)	8.42E+04 (4+)	8.98E+04 (6+)	8.77E+04 (5+)	**8.03E+04 (1)**
S400–RC101	1.36E+05 (3+)	1.36E+05 (4+)	1.34E+05 (2+)	1.60E+05 (5+)	1.78E+05 (6+)	**1.30E+05 (1)**
S400–RC102	1.11E+05 (3≈)	1.11E+05 (2≈)	1.17E+05 (4+)	1.35E+05 (5+)	1.53E+05 (6+)	**1.09E+05 (1)**
S400–RC201	1.25E+05 (3≈)	1.24E+05 (2≈)	1.32E+05 (4+)	1.36E+05 (5+)	1.49E+05 (6+)	**1.23E+05 (1)**
S400–RC202	9.95E+04 (3≈)	9.87E+04 (2≈)	1.09E+05 (4+)	1.09E+05 (5+)	1.17E+05 (6+)	**9.72E+04 (1)**
num of best	0	1	2	1	0	**20**
sum of rank	72	68	84	108	139	**31**
num of +/≈/–	12/12/0	14/10/0	19/5/0	21/3/0	24/0/0	\

while the compared approaches only obtain the best result on 11 instances in total. For instances with scales of 200 and 400, it is presented in Table II that BACS surpasses the compared approaches on 20 of 24 instances. However, the compared approaches can only outperform BACS on 4 instances. Besides, BACS ranks first in terms of the sum of rank no matter for instances with scales of 50 and 100 or instances with scales of 200 and 400. Meanwhile, the total rank of BACS on instances with scales of 200 and 400 is much better than that of BACS on instances with scales of 50 and 100. That is, the superiority of BACS is further presented on solving instances with larger scales. Hence, BACS can be regarded as a better approach for solving the DCCL scheduling problem when compared with the compared approaches. At the same time, BACS shows more advantages when instance scales increase.

The statistical test results displayed in Tables I and II also support the conclusion that BACS outperforms the compared approaches. Specifically, BACS is significantly superior to ACS, ACS-DVRP, MACS-VRPTW, P-ACO, and ALNS on 23, 23, 33, 38, and 41 of 48 instances, and it has competitive performance on 21, 22, 13, 8, and 5 of 48 instances. In contrast, the performances of ACS, ACS-DVRP, MACS-VRPTW, P-ACO, and ALNS are not so satisfying since they only get better results on 4, 3, 2, 2, and 2 of 48 instances when compared with BACS.

To further analyze the characteristic of the compared approaches and BACS, some findings based on the comparison results shown in Tables I and II are summarized as follows. First, although the performance of ACS and ACS-DVRP are fairly similar, the latter slightly outperforms the former, which may owe to the pheromone conservation procedure adopted in ACS-DVRP. However, both ACS and ACS-DVRP are inferior to BACS, suggesting that the adoptions of BS and PLS are more promising. Second, even though MACS-VRPTW is an ACS-based approach, it shows poor performance when compared with BACS. This may be caused by the employment of time-consuming local search operations for most newly generated solutions. Hence, the time consumption of local search operations should be concerned when they are integrated into the heuristic approaches. Third, the poor performance of P-ACO may result from the repair procedure of solutions, which can also be very time-consuming, meaning that little execution time is left to P-ACO for further search. Thus, the balance of time used between solution exploitation and further search is important. Last but not least, it can be observed that the performance of ALNS is competitive on instances with scales of 50 when compared with BACS. Nevertheless, the performance of ALNS declines rapidly with the scales of instances increase. Thus, compared with the heuristic approaches that search the optimal solution through the repetitive application of destroy and repair operations, the combination of meta-heuristic approaches and local search operators seems to be a better choice for solving the DCCL scheduling problem.

For an intuitive comparison, the average improvement rates of BACS on instances in the four scales are listed in Table III, and the improvement rate of BACS to approach A is calculated as (Mean(A) – Mean(BACS)) / Mean(A). The results in Table III

TABLE III
AVERAGE IMPROVEMENT RATES OF BACS

Scale	ACS	ACS-DVRP	MACS-VRPTW	P-ACO	ALNS
50	0.57%	–0.44%	3.65%	1.74%	1.43%
100	1.88%	2.04%	4.31%	3.16%	6.78%
200	2.21%	2.00%	3.80%	4.82%	10.13%
400	1.61%	1.98%	4.07%	15.20%	20.91%
Avg	1.57%	1.39%	3.95%	6.23%	9.81%

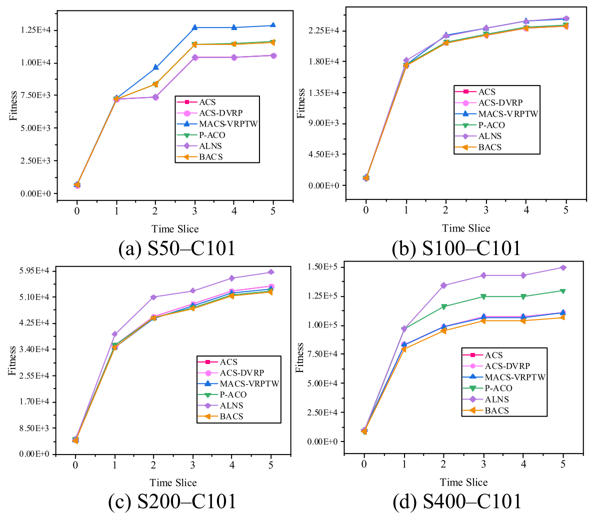

Fig. 6. Dynamic change of solutions of compared approaches and BACS.

TABLE IV
COMPARISON RESULTS OF COMPARED APPROACHES AND BACS ON NUMBER OF REJECTED ORDERS

Scale	ACS	ACS-DVRP	MACS-VRPTW	P-ACO	ALNS	BACS
50	8.15	8.03	8.48	8.13	**8.00**	8.18
100	12.81	12.93	13.06	12.89	12.89	**12.71**
200	39.30	39.20	39.69	39.91	40.80	**39.09**
400	81.00	80.96	81.85	93.4	94.88	**80.23**
Avg	35.31	35.28	35.77	38.58	39.14	**35.05**

show that the improvement rates of BACS are minor on instances with scales of 50, and BACS is slightly surpassed by ACS-DVRP. However, the improvement rates of BACS become more remarkable with the instances scales increase, especially when compared with P-ACO and ALNS. This may be due to the BS adopted in BACS, which can reduce the redundant scheduling and allow BACS to focus on scheduling orders that ought to be delivered first. Meanwhile, examples for presenting the dynamic changes of scheduling results of all approaches in a working day are shown in Fig. 6, which also display the advantages of BACS in solving the DCCL model, especially on instances with larger scales.

Considering that both order rejection and vehicle usage play important roles in the total distribution cost, comparison results in terms of the number of rejected orders and the number of vehicles being used are shown in Tables IV and V, respectively. Likewise, the optimal results are presented using **boldface**. As can be observed, the average number of rejected orders of

TABLE V
COMPARISON RESULTS OF COMPARED APPROACHES AND BACS ON NUMBER OF VEHICLES BEING USED

Scale	ACS	ACS-DVRP	MACS-VRPTW	P-ACO	ALNS	BACS
50	5.41	5.42	5.68	5.65	6.11	**5.31**
100	9.36	**9.27**	9.84	**9.27**	11.31	9.29
200	12.83	12.96	15.12	13.38	16.85	**12.03**
400	25.23	25.67	30.18	26.96	35.24	**23.56**
Avg	13.21	13.33	15.21	13.81	17.38	**12.55**

BACS is minimal among the six approaches, and the same is true for the average number of vehicles being used. Hence, it can be concluded that the scheduling result of BACS for the DCCL model is conducive to improving the service quality and reducing the vehicle cost.

D. *Effectiveness Investigation of BS and PLS*

To investigate the effectiveness of BS and PLS, three new BACS variants are proposed. The first variant named BACS-v1 is developed by removing the PLS from BACS, and the second one named BACS-v2 is proposed by removing the BS from BACS. The third variant is named BACS-v3 and is set up by removing both PLS and BS from BACS. In other words, BACS-v1 is the ACS with BS, BACS-v2 is the ACS with PLS, and BACS-v3 is the pure ACS. Therefore, with these three BACS variants, the effectiveness of BS can be evaluated through the comparison between BACS-v1 and BACS-v3, while the effectiveness of PLS can be evaluated through the comparison between BACS-v2 and BACS-v3.

Firstly, Table S.II in the supplementary material lists the comparison results of BACS-v1 and BACS-v3 on all test instances, and the best results are presented in **boldface**. The comparison results show that BACS-v1 outperforms BACS-v3 on 38 of 48 instances in terms of average fitness. Besides, the performance of BACS-v1 is significantly superior to or comparable with that of BACS-v3 on 47 of 48 instances. At the same time, BACS-v1 is inferior to BACS-v3 on only 1 instance. Hence, it is believed that BS has a positive effect on improving the performance of ACS. Consequently, it is worthwhile to employ BS in BACS.

Secondly, Table S.III in the supplementary material displays the comparison results of BACS-v2 and BACS-v3, and it presents the best results in **boldface**. As can be observed from Table S.III, BACS-v2 ranks first on 30 of 48 instances. Meanwhile, BACS-v2 can make significant improvements on 10 out of the 48 instances in comparison with BACS-v3. In contrast, BACS-v3 surpasses BACS-v2 on only 1 instance. Therefore, it is reasonable to consider that the PLS can also enhance the performance of ACS.

Moreover, to investigate whether the combination of BS and PLS can further improve the performance of ACS, the comparison results on the average fitness of BACS-v1, BACS-v2, and BACS are displayed in Table S.IV in the supplementary material. In general, the comparison results show that BACS performs best on 31 of 48 instances, while BACS-v1 and BACS-v2 only get better results on 14 and 7 instances, respectively. According

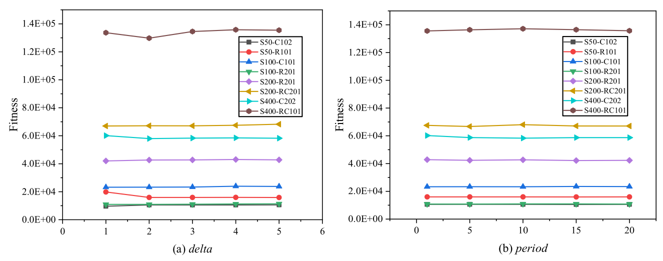

Fig. 7. Parameter investigation of BS and PLS.

to the statistical test, it can be known that the BACS dominates BACS-v1 on 14 instances, while the latter only defeats BACS on 1 instance. Specifically, BACS has significantly better performance on 10 small scale (i.e., 50 and 100) instances and 4 large scale (i.e., 200 and 400) instances in comparison with BACS-v1. Hence, by integrating PLS to BACS-v1, the overall performance of BACS-v1 can be improved, especially on instances with small scales. Meanwhile, BACS also outperforms BACS-v2 on 14 instances, including 5 small scale instances and 9 large scale instances, while BACS-v2 is superior to BACS on 1 instance. Hence, BS is beneficial to strengthen the ability of BACS in solving the DCCL model, especially on instances with large scales. Therefore, the effectiveness of BS and PLS are considered complementary, and the combination of BS and PLS can promote BACS to solve the DCCL model better.

E. Parameter Investigation of BS and PLS

In this part, the parameter investigation is conducted for the *delta* of BS and the *period* of PLS. Herein, eight instances with various scales and categories are randomly selected for parameter investigation. The parameters of BS and PLS are investigated on BACS-v1 and BACS-v2, respectively. The comparison results of the *delta* with different values (i.e., 1, 2, 3, 4, 5) are displayed in Fig. 7(a), and the investigation results of the *period* with different values (i.e., 1, 5, 10, 15, 20) are shown in Fig. 7(b). The overview of the curves shows that the performance is not very sensitive to these two parameters, which is also an advantage of BACS. The detailed discussions on the results are presented as follows.

As Fig. 7(a) shows, the performance of BACS-v1 on S50–R101, S400–C202, and S400–RC101 is improved when *delta* changes from 1 to 2, while the performance of BACS-v1 on S400–RC101 is weakened when *delta* changes from 2 to other values. Besides, the performances of BACS-v1 with different *delta* on other instances are similar. Thus, the *delta* of BS is determined as 2.

The comparison results of Fig. 7(b) present that BACS-v2 can perform better when the *period* is 15 and 20 according to the trend of curves. Besides, although the difference between performances of BACS-v2 with *period* equals 15 and *period* equals 20 is not that noteworthy, the performance of the former is actually better since it ranks first on 3 instances while the latter ranks first on no instance, which is shown in Table S.V in the supplementary material. Therefore, the *period* of PLS is determined as 15.

V. Conclusion

In this paper, a DCCL model to minimize the total distribution cost in a working day is established. Herein, the total distribution cost consists of the transportation cost, the cost of order rejection penalty, and the cost of quality loss of goods. In comparison with the existing static CCL models, the DCCL model is more appropriate for the simulation of real CCL scheduling due to the considerations of dynamic orders (i.e., orders revealed in a working day). To solve the DCCL model, the BACS approach that incorporates both BS and PLS is proposed. With the assistance of BS, the redundant scheduling of non-urgent orders can be reduced. Thus, BACS has sufficient time to search for better solutions in a relatively smaller yet promising search space. Besides, the PLS allows the BACS to further improve the best solution by learning from the periodic best solutions.

The comparison results with the other five scheduling approaches show that BACS is an effective approach for solving the DCCL scheduling problem. To be specific, BACS can better decrease the amount of both rejected orders and used vehicles. This is significant because less order rejection can improve customer satisfaction and fewer vehicle usage can save great operational cost. Furthermore, it is presented that BACS shows more advantages when solving instances with large scales. At the same time, the effectiveness investigation of BS and PLS shows that BS and PLS are complementary because the former can enhance the performance of BACS on instances with large scales while the latter can help BACS to perform better on instances with small scales.

References

[1] H. X. Zhao, S. Liu, C. Q. Tian, G. Yan, and D. Wang, “An overview of current status of cold chain in China,” *Int. J. Refrigeration*, vol. 88, pp. 483–495, 2018.

[2] J. W. Han, M. Zuo, W. Y. Zhu, J. H. Zuo, E. L. Lü, and X. T. Yang, “A comprehensive review of cold chain logistics for fresh agricultural products: Current status, challenges, and future trends,” *Trends Food Sci. Technol.*, vol. 109, pp. 536–551, 2021.

[3] G. B. Dantzig and J. H. Ramser, “The truck dispatching problem,” *Manage. Sci.*, vol. 6, no. 1, pp. 80–91, 1959.

[4] Y. L. Lan, F. G. Liu, W. W. Y. Ng, J. Zhang, and M. K. Gui, “Decomposition based multi-objective variable neighborhood descent algorithm for logistics dispatching,” *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 5, no. 5, pp. 826–839, Apr. 2021.

[5] J. Mańdziuk, “New shades of the vehicle routing problem: Emerging problem formulations and computational intelligence solution methods,” *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 3, no. 3, pp. 230–244, Mar. 2019.

[6] C. M. Qi and L. S. Hu, “Optimization of vehicle routing problem for emergency cold chain logistics based on minimum loss,” *Phys. Commun.*, vol. 40, pp. 1–7, 2020.

[7] B. Z. Yao, C. Chen, X. L. Song, and X. L. Yang, “Fresh seafood delivery routing problem using an improved ant colony optimization,” *Ann. Oper. Res.*, vol. 273, no. 1, pp. 163–186, 2017.

[8] Z. X. Zhao, X. M. Li, and X. C. Zhou, “Distribution route optimization for electric vehicles in urban cold chain logistics for fresh products under time-varying traffic conditions,” *Math. Problems Eng.*, vol. 2020, pp. 1–17, 2020.

[9] G. K. Liu, J. Y. Hu, Y. Yang, S. M. Xia, and M. K. Lim, “Vehicle routing problem in cold chain logistics: A joint distribution model with carbon trading mechanisms,” *Resour. Conservation Recycling*, vol. 156, pp. 1–13, 2020.

[10] Y. Li, M. K. Lim, and M. L. Tseng, “A green vehicle routing model based on modified particle swarm optimization for cold chain logistics,” *Ind. Manage. Data Syst.*, vol. 119, no. 3, pp. 473–494, 2019.

[11] J. Chen, P. F. Gui, T. Ding, S. Y. Na, and Y. T. Zhou, “Optimization of transportation routing problem for fresh food by improved ant colony algorithm based on tabu search,” *Sustainability*, vol. 11, no. 23, pp. 1–22, 2019.

[12] L. L. Leng, C. M. Zhang, Y. W. Zhao, W. L. Wang, J. L. Zhang, and G. F. Li, "Biobjective low-carbon location-routing problem for cold chain logistics: Formulation and heuristic approaches," *J. Cleaner Prod.*, vol. 273, pp. 1–18, 2020.

[13] S. Y. Wang, F. M. Tao, and Y. H. Shi, "Optimization of location–routing problem for cold chain logistics considering carbon footprint," *Int. J. Environ. Res. Public Health*, vol. 15, no. 1, pp. 1–17, 2018.

[14] 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.

[15] H. N. Psaraftis, M. Wen, and C. A. Kontovas, "Dynamic vehicle routing problems: Three decades and counting," *Networks*, vol. 67, no. 1, pp. 3–31, 2016.

[16] P. Kilby, P. Prosser, and P. Shaw, "Dynamic VRPs: A study of scenarios," Univ. Strathclyde, Glasgow, U.K., 1998, pp. 1–11.

[17] A. M. F. AbdAllah, D. L. Essam, and R. A. Sarker, "On solving periodic re-optimization dynamic vehicle routing problems," *Appl. Soft Comput.*, vol. 55, pp. 1–12, 2017.

[18] M. Okulewicz and J. Mańdziuk, "The impact of particular components of the PSO-based algorithm solving the dynamic vehicle routing problem," *Appl. Soft Comput.*, vol. 58, pp. 586–604, 2017.

[19] R. J. Kuo, B. S. Wibowo, and F. E. Zulvia, "Application of a fuzzy ant colony system to solve the dynamic vehicle routing problem with uncertain service time," *Appl. Math. Model.*, vol. 40, no. 23/24, pp. 9990–10 001, 2016.

[20] 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.

[21] M. Guntsch and M. Middendorf, "A population based approach for ACO," in *Proc. 2nd Eur. Workshop Evol. Comput. Comb. Optim.*, 2002, pp. 72–81.

[22] M. Mavrovouniotis, C. H. Li, and S. X. Yang, "A survey of swarm intelligence for dynamic optimization: Algorithms and applications," *Swarm Evol. Comput.*, vol. 33, pp. 1–17, 2017.

[23] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Jan. 1997.

[24] Z. H. Zhan *et al.*, "Matrix-based evolutionary computation," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 6, no. 2, pp. 315–328, Feb. 2022.

[25] Z. H. Zhan, L. Shi, K. C. Tan, and J. Zhang, "A survey on evolutionary computation for complex continuous optimization," *Artif. Intell. Rev.*, vol. 55, no. 1, pp. 59–110, 2022.

[26] Z. G. Chen *et al.*, "Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 2912–2926, Aug. 2019.

[27] X. Zhang, Z. H. Zhan, W. Fang, P. Qian, and J. Zhang, "Multi population ant colony system with knowledge based local searches for multiobjective supply chain configuration," *IEEE Trans. Evol. Comput.*, to be published, doi: 10.1109/TEVC.2021.3097339.

[28] X. F. Liu, Z. H. Zhan, J. D. Deng, Y. Li, T. Gu, and J. Zhang, "An energy efficient ant colony system for virtual machine placement in cloud computing," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 113–128, Jan. 2018.

[29] T. Wei, J. Zhong, and J. Zhang, "An energy-efficient partition-based framework with continuous ant colony optimization for target tracking in mobile sensor networks," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 5, no. 4, pp. 700–713, Apr. 2021.

[30] B. Ombuki, B. Ross, and F. Hanshar, "Multi-objective genetic algorithms for vehicle routing problem with time windows," *Appl. Intell.*, vol. 24, no. 1, pp. 17–30, 2006.

[31] Z. Yang *et al.*, "Dynamic vehicle routing with time windows in theory and practice," *Natural Comput.*, vol. 16, no. 1, pp. 119–134, 2017.

[32] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Oper. Res.*, vol. 35, no. 2, pp. 254–265, 1987.

[33] J. Homberger and H. Gehring, "A two-phase hybrid metaheuristic for the vehicle routing problem with time windows," *Eur. J. Oper. Res.*, vol. 162, no. 1, pp. 220–238, Apr. 2005.

[34] L. M. Gambardella, E. D. Taillard, and G. Agazzi, "MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows," *New Ideas Optim.*, pp. 63–76, 1999.

[35] M. Guntsch, M. Middendorf, and H. Schmeck, "An ant colony optimization approach to dynamic TSP," in *Proc. Genet. Evol. Comput. Conf.*, 2001, pp. 860–867.

[36] S. F. Chen, R. Chen, G. G. Wang, J. Gao, and A. K. Sangaiah, "An adaptive large neighborhood search heuristic for dynamic vehicle routing problems," *Comput. Elect. Eng.*, vol. 67, pp. 596–607, 2018.

Li-Jiao Wu (Student Member, IEEE) received the B.S. degree in computer science and technology in 2020 from the South China University of Technology, Guangzhou, China, where she is currently working toward the M.S. degree. Her research interests include evolutionary computation, ant colony system, and their applications to real-world optimization problems.

Lin Shi (Student Member, IEEE) received the B.S. degree in 2018 from the South China University of Technology, Guangzhou, China, where she is currently working toward the Ph.D. degree. Her research interests include artificial intelligence, evolutionary computation, swarm intelligence, and their applications in intelligent transportation.

Zhi-Hui Zhan (Senior Member, IEEE) received the bachelor's and Ph. D. degrees in computer science from Sun Yat-Sen University, Guangzhou, China, in 2007 and 2013, respectively. He is currently a Changjiang Scholar Young Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. His research interests include evolutionary computation algorithms, swarm intelligence algorithms, their applications in real-world problems, and in environments of cloud computing and Big Data. Dr. Zhan was the recipient of the IEEE Computational Intelligence Society (CIS) Outstanding Early Career Award in 2021, Outstanding Youth Science Foundation from National Natural Science Foundations of China in 2018, and Wu Wen-Jun Artificial Intelligence Excellent Youth from the Chinese Association for Artificial Intelligence in 2017. His doctoral dissertation was awarded the IEEE CIS Outstanding Ph.D. dissertation and China Computer Federation Outstanding Ph.D. dissertation. He is one of the World's Top 2% Scientists for Career-Long Impact and Year Impact in Artificial Intelligence and one of the Highly Cited Chinese Researchers in computer science. He is the Chair of Membership Development Committee in IEEE Guangzhou Section and the Vice-Chair of IEEE CIS Guangzhou Chapter. He is also an Associate Editor for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the *Neurocomputing*, and the *Memetic Computing*.

Kuei-Kuei Lai received the Ph.D. degree in management science from Tamkang University, Taiwan. He is currently a Professor with the Department of Business Administration, Chaoyang University of Technology, Taiwan. He is the author of more than 90 articles. His research interests include quantitative analysis, patent citation analysis, social network analysis, technology strategy and technological forecasting, and computational intelligence, and applications in management.

Jun Zhang (Fellow, IEEE) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Hong Kong, in 2002. He is currently the Korea Brain Pool Fellow Professor with Hanyang University, Seoul, South Korea. He has authored or coauthored more than 300 technical papers in his research field, which include computational intelligence, cloud computing, high performance computing, operations research, and power electronic circuits. Dr. Zhang was the recipient of the Changjiang Chair Professor from the Ministry of Education, China, in 2013, China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011, and First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is also an Associate Editor for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON CYBERNETICS.