A Hybrid Discrete Differential Evolution Algorithm to Solve the Split Delivery Vehicle Routing Problem

LI-NING XING,1,2 YUANYUAN LIU,2,3 HAIYAN LI,3 CHIN-CHIA WU,4 WIN-CHIN LIN,4 AND WU SONG5

1School of Logistics, Central South University of Forestry and Technology, Changsha 410004, China
2Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
3College of Computer Science and Engineering, Northeastern University, Shenyang 110819, China
4Department of Statistics, Feng Chia University, Taichung 40724, Taiwan
5School of Computer Science and Technology, Hainan Tropical Ocean University, Sanya 572022, China

Corresponding author: Wu Song (songwu_el@sina.com.cn)

This work was supported in part by the National Natural Science Foundation, China, under Grant 61773120, in part by the Innovation Team of Guangdong Provincial Department of Education under Grant 2018KCXTD031, in part by the Hunan Key Laboratory of Intelligent Logistics Technology under Grant 2019TP1015, in part by the State Key Laboratory of Digital Manufacturing Equipment and Technology under Grant DMETKF2020030, and in part by the Hainan Provincial Natural Science Foundation of under Grant 617154.

ABSTRACT For the split delivery vehicle routing problem in the distribution link of logistics where the optimization objective is to minimize the total distance of the transportation path, a hybrid discrete differential evolution (HDDE) algorithm is adopted in this study. The algorithm applies the encoding and decoding method based on the remaining vehicle capacity to split the demand of customer point and adjust the splitting node to the optimal position of its path. In order to avoid the prematurity of the algorithm, the perturbation mechanism is added, i.e., the three-point insertion method is used in the crossover stage. To ensure the higher quality of the solution, the idea of tabu search is introduced in three cases: initial population generation, selection operation and local optimal situation. Finally, through the numerical experiments based on benchmark data, the HDDE algorithm is compared with other competitive algorithms to demonstrate its domination.

INDEX TERMS Split delivery vehicle routing problem, differential evolution, meta-heuristic algorithm, optimization.

I. INTRODUCTION

With a mass outbreak of COVID-19, medical resources are seriously deficient, and hospitals have higher requirements for the transportation speed of medical supplies. More and more hospitals introduce intelligent logistics equipment, and intelligent transport vehicles such as unmanned delivery vehicle and unmanned warehouse transport vehicle have begun to enter people’s vision. In order to minimize human contact and reduce the possibility of human-to-human transmission of the virus, unmanned delivery vehicles transport medical supplies to many hospitals in urban areas. The demand of each hospital may be of great quality or volume, even exceed the transport capacity of unmanned vehicles. To reduce the empty load rate and improve the utilization rate of transport resources, the supplies of hospitals in great demand can be split and distributed to multiple unmanned delivery vehicles to transport. Such transport problem can be classified as the split delivery vehicle routing problem (SDVRP). Dror and Trudeau [1] first proposed the split delivery vehicle routing problem, a relaxation of the classical capacitated vehicle routing problem. In its seminal paper, Dror and Trudeau found two characteristics of the optimal solution if the travel cost of any pair of customer points satisfies the triangle inequality: (1) at most one common customer exists in any two paths; and (2) $k$-split ($k \geq 2$) cycle does not exist, where $k$-split cycle is defined as follows: for $k$ demand points 1, 2, ..., $k$ and $k$ routes, the points 1 and 2 are included in route 1, the points 2 and 3 are included in route 2, ..., the points $k-1$ and $k$ are included in route $k-1$, and the points $k$ and 1 are included in route $k$. This subset of $k$ demand points is called a $k$-split cycle. Then in 1990, the two authors [2]
first presented the mixed integer linear programming model of SDVRP and demonstrated that the number of vehicles and the travel distance can be significantly reduced by splitting if demand is over 10 percent of vehicle capacity. Belenguer et al. [3] indicated that SDVRP has a strict lower bound if the customer demand is less than the vehicle loading capacity and the actual delivery volume is integer. Archetti and Mansini et al. [4] pointed out that when the distance values satisfy the triangle inequality and the vehicle loading capacity \( Q = 2 \), the solution of SDVRP can be obtained in polynomial time under the general cost condition. When \( Q > 2 \), SDVRP is NP-hard. Archetti and Savelsbergh et al. [5] studied the situation where the demand of customers exceeds the loading capacity of vehicle and analyzed the cost savings brought by SDVRP from the aspects of vehicle number and driving distance. They proved that the relationship between average customer demand and vehicle loading capacity was the biggest factor affecting cost savings in SDVRP. The cost savings brought by SDVRP are not related to the distribution of customer points but are affected by the variance of customer demands. Maximum benefit can be achieved in the case that the average demand is slightly more than 50% of the vehicle capacity and customer demands have a smaller variance [6].

Since SDVRP is one of the NP-hard problems, its solution mainly devoted to two aspects, i.e., exact algorithms and heuristic ones. For the former aspect, Dror et al. [7] added sub-tour elimination constraints and designed a branch and bound algorithm with proper constraint combinations, and considerably reduced the differences between the lower and upper bounds at the root of the branching tree. Then, Lee et al. [8] considered SDVRP as a dynamic programming problem and proposed a shortest path search method, greatly reducing the solution space. Jin et al. [9] proposed a two-phase method equipped with valid inequality constraints including customer point clustering stage and TSP path arrangement stage, and their algorithm significantly reduce iterations and computational time required to reach an optimal solution. Moreno et al. [10] designed a column generation and tangent plane method which obtain a more effective lower bound. Archetti and Bianchessi et al. [11] classified and defined the split delivery vehicle routing problem with unlimited fleet (SDVRP-UF) and the split delivery vehicle routing problem with limited fleet (SDVRP-LF), presenting a branch-and-price-and-cut method decomposing the problem and a heuristic whose initial solution results from the solution of subproblem (column) to get the upper bound. They presented two branch-and-cut algorithms based on two relaxed formulas that find the lower bound [12].

However, due to the computational limitation, exact algorithms can only solve small instance of SDVRP. Therefore, heuristic ones, especially meta-heuristic algorithms, have become the main research direction of SDVRP. Archetti et al. [13] designed a tabu search algorithm based on the insertion neighborhood construction method which is proved to be more efficient than the previous heuristic algorithm proposed by Dror et al. Aleman and Hill et al [14] presented the tabu search with vocabulary building approach (TSVBA) which is a learning process, i.e., new solutions are generated from a set of initial solutions according to common properties and novel features that can generate superior solutions. Derigs [15] used a variety of neighborhood operators and compares several meta-heuristic algorithms, proving that the attribute-based hill climber heuristic has the best performance and the relocate neighborhood operator has the greatest contribution. Wilck and Cavalier [16] designed two kinds of hybrid genetic algorithms, one of which is based on the shortest path algorithm, while the other is combined with the ratio relationship of unit customer demand and unit distance. Berbotto [17] introduced granularity computing technique into tabu search algorithm, presenting the idea of exploring neighborhoods of infeasible solutions that does not satisfy the capacity constraint. Silva et al. [18] proposed an iterative local search heuristic with multiple initial positions, including a novel perturbation mechanism for both two cases of SDVRP-UF and SDVRP-LF. Zhang et al. [19] designed a forest-based tabu search Algorithm with three neighborhood operators, applying a novel solution representation to combine a routes and forest. Rajappa et al. [20] presented an ant colony optimization (ACO) and a hybridization of ACO, genetic algorithm and a heuristic. Shi et al. [21] designed a particle swarm optimization approach including two sets of local searches for optimum solutions, which can determine whether they are performed on a specific solution.

Compared with above heuristic algorithms, Differential Evolution (DE) algorithm has four advantages, which are simple structure, easy implementation, quick convergence, as well as robustness. Given that the idea of hybridization of DE and tabu search have been successfully applied to solve the job-shop scheduling problem by Ponsich and Coello [22], it may be also beneficial to solve SDVRP. To our best knowledge, there is no known research applied DE algorithm to solve the SDVRP, therefore, we propose a hybrid discrete differential evolutionary (HDDE) algorithm in this paper.

The rest of this paper is organized as follows. The mathematical formulation of the SDVRP is provided in Section II, and our HDDE algorithm is presented in Section III. Section IV is devoted to the numerical experiments. Finally, Section V concludes this paper.

II. PROBLEM STATEMENT

The SDVRP studied in this paper is set as single depot, single vehicle type, no time window limit, and pure delivery, which can be described as follows. There are \( n \) customer points, which would be served by multiple vehicles with the maximum load of \( Q \). All vehicles start from a certain depot and return it after finishing the delivery task. Each customer demand can be delivered by one or more vehicles, and the objective is to gain the path of each vehicle that the total travel distance is shortest. The following notations in table 1 are used to describe the problem.

The problem is based on the following assumptions.
TABLE 1. The list of notations and meanings in problem model.

| Symbol | Meaning |
|--------|---------|
| n      | The number of customer points. |
| m      | The number of vehicles required, i.e. minimum number of vehicles. |
| N      | A collection of depot and customer points. |
| M      | A set of vehicles. |
| d_i    | Demand of customer point i, where d_i > 0 (i > 0) and d_0 = 0. |
| Q      | The vehicle capacity. |
| c_yj   | The distance from point i to point j. |
| y_ik   | A binary variable that is 1 when vehicle k travel from point i to point j; 0 otherwise. |

1. The distance between the two points is symmetric: c_ij = c_ji.

2. Distance satisfies the triangle inequality: c_ij + c_jk ≥ c_ik.

The mathematical programming model of the problem is provided as follows:

Minimize \( Z = \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{m} c_{ij} x_{ijk} \) (1)

Subject to:

\( \left\lceil \sum_{i=1}^{n} d_i / Q \right\rceil = m \) (2)

\( \sum_{i=0}^{n} \sum_{j=1}^{m} x_{ijk} \geq 1, j \in N \) (3)

\( \sum_{i=0}^{n} x_{ipk} - \sum_{j=0}^{n} x_{pjk} = 0, p \in N, k \in M \) (4)

\( y_{ik} \leq d_i \sum_{j=0}^{n} x_{ijk}, i \in N', k \in M \) (5)

\( \sum_{i=1}^{n} y_{ik} = d_i, i \in N' \) (6)

\( \sum_{i=1}^{n} y_{ik} \leq Q, k \in M \) (7)

\( \sum_{i \in S} \sum_{j \in S} x_{ijk} = |S| - 1, k \in M, S \subseteq N' \) (8)

\( x_{ijk} \in 0, 1, i \in N, j \in N, k \in M \) (9)

\( y_{ik} \geq 0 \) (10)

The objective is represented by Equation (1), which minimizes the total travel distance. Equation (2) represents the minimum number of vehicles required. Constraint (3) restricts that each customer and the depot must be visited at least once. Constraint (4) ensures flow conservation constraint that the number of visiting a point is the same as the number of leaving the point in each path. Constraint (5) limits that customer point is serviced if and only if a vehicle visited it, and its demand assigned to a vehicle is less than or equal to its total demand. Constraint (6)-(7) guarantees that all customer demand should be satisfied, and the actual load of any vehicle will not exceed the maximum load. Constraint (8) means subtour elimination. Constraints (9)-(10) provides variable restrictions.

III. A HYBRID DISCRETE DIFFERENTIAL EVOLUTION ALGORITHM FOR SDVRP

The differential evolution algorithm presented by Storn and Price [23] in 1997 is one of the global population evolutionary optimization algorithms to solve continuous optimization problems. The basic idea of this algorithm is to start from the initial population and generate new individuals through crossover and mutation operations. New individuals can only be selected as the next generation if their fitness is superior to that of their parents. For applying in a discrete problem, the individual in discrete differential evolutionary (DDE) algorithm is expressed as a customer-based sequence. In order to overcome pre-maturity and obtain a better solution, a hybrid discrete differential evolutionary (HDDE) algorithm is presented, combined with tabu search operation in initial population generation, selection operation and the phase at which the solution cannot be improved, and three-point insertion method is adopted in the crossover stage. The procedure of the HDDE algorithm is as follows.

A. ENCODING AND DECODING

The customer point sequence called the target individual in HDDE is a potential solution. Each individual is encoded as a full array of customer points, and then decoded according to its order by the following three phases: preliminary path generation phase according to capacity constraint, splitting point position adjustment stage and distance calculation stage. Some related variables are showed in table 2.

TABLE 2. Symbols and meanings in decoding phase.

| Phase | Symbol | Meaning |
|-------|--------|---------|
| 1     | d      | Current customer point’s demand quality not arranged. |
|       | Q_i    | The total remaining capacity of remaining paths waiting to be arranged. Before decoding, its value can be calculated as: \( Q_i = m - \sum_{j=1}^{n} d_j \). |
|       | S_Pc   | The splitting point set of the current path which is being adjusted. 0 ≤ |S_Pc| ≤ 2. |
|       | \( \Delta c \) | The distance change value when repositioning. |
(1) Decoding phase 1: generate preliminary paths through capacity constraint relation.

Since distances conform to the relation of triangle inequality, the demand of some customer points should not be split to shorten the total travel distance. In the decoding phase, put customer point one by one into the path according to the order of customer points in an individual, and judge when the path capacity is exceeded: if $Q_c \leq \lambda_c$, customer point is put into next path; otherwise customer point’s demand is split.

(2) Decoding phase 2: reposition splitting points of preliminary paths.

In paths obtained from phase 1, the splitting point can only be located at the first or the last in customer sequence not including depot due to the encoding method. In this phase, the splitting point’s location will be adjusted to the best location in its path to try to reduce distance and two cases are discussed as follows. The best location is where $\Delta c$ is smallest.

Case 1: $|SPc| = 1$, i.e., a path which is being adjusted exists one splitting point at the first or the last location.

For a path $0 - x_1 - x_2 - \ldots - x_{n*} - 0$ where $x_1$ is a splitting point, delete point $x_1$ and the path is turned to $0 - x_2 - x_3 - \ldots - x_{n*} - 0$ which exists $n*$ insertion positions. The two points left and right of the insertion position are respectively represented as $x^+_1$ and $x^-_1$. Then calculate distance change values of every insertion position $\Delta c$ and find the best position, where $\Delta c = c_{x^+_1, x^-_1} + c_{x^+_{n*}, x^-_{n*}} - c_{x_1, x^+_1}$. If the change value in the first position is the smallest, the splitting point $x_1$ keeps original position; otherwise put $x_1$ into the new best position.

Analogously, for a path $0 - x_1 - x_2 - \ldots - x_{n*} - 0$ where $x_{n*}$ is a splitting point, delete point $x_{n*}$ and the path is turned to $0 - x_1 - x_2 - \ldots - x_{n*+1} - 0$ which exists $n*+1$ insertion positions. The two points left and right of the insertion position are respectively represented as $x^-_{n*}$ and $x^+_{n*}$ then calculate distance change values of every insertion position $\Delta c$ and find the best position, where $\Delta c = c_{x^-_{n*}, x^+_{n*}} + c_{x^+_{n*}, x^-_{n*}} - c_{x_{n*}, x^-_{n*}}$. If the change value in the last position is the smallest, the splitting point $x_{n*}$ keeps original position; otherwise put $x_{n*}$ into the new best position.

Case 2: $|SPc| = 2$, i.e., a path which is being adjusted exists two splitting points at the first and the last location respectively.

For a path $0 - x_1 - x_2 - \ldots - x_{n*} - 0$ where $x_1$ and $x_{n*}$ are splitting points, delete point $x_1$ and point $x_{n*}$. The path is turned to $0 - x_2 - x_3 - \ldots - x_{n*+2} - x_{n*+1} - 0$ which exists $(n* - 1)$ insertion positions, then calculate distance change values when $x_1$ is front of $x_{n*}$ and when $x_{n*}$ is front of $x_1$.

When $x_1$ is front of $x_{n*}$ put $x_1$ into candidate insertion location $\lambda_1$ ($1 \leq \lambda_1 \leq n* - 1$) and put $x_{n*}$ into candidate insertion location $\lambda_{n*}$ ($\lambda_{n*} \leq \lambda_{n*} \leq n* - 1$), then calculate distance change value. If $\lambda_{n*} = \lambda_1$, $\Delta c = c_{x_1, x_{n*}} + c_{x_{n*}, x_{n*+1}} + c_{x_1, x_{n*+1}} - c_{x^-_1, x^+_1}$, otherwise $\Delta c = (c_{x^-_1, x^-_{n*}} + c_{x^+_1, x^-_{n*}} + c_{x^+_1, x^-_{n*}} + c_{x^+_1, x^-_{n*}}) + (c_{x^-_1, x^-_{n*}} + c_{x^+_1, x^-_{n*}} + c_{x^+_1, x^-_{n*}})$. Finally, find the best position and put $x_1$ and $x_{n*}$ into respective best position.

(3) Decoding phase 3: calculate total distance of the paths obtained by phase 2.

For an individual $X = [x_1, x_2, \ldots, x_n]$, the detailed process of decoding is as follows:

Step 1: initialization: set $i = 1, j = 1, d = d_1, Q_c = Q, Q_r = Q_r$.

Step 2: if $d \leq Q_c$: put demand $d$ of customer $x_i$ into path $j$, set $Q_c = Q_c - d$, then go to Step 3; otherwise: go to Step 4.

Step 3: set $i = i + 1$, and if $i > n$: go to Step 5; otherwise: set $d = d_1$, then return to Step 2.

Step 4: if $Q_c \leq Q_r$: set $j = j + 1$, $Q_c = Q_c - Q_r$, $Q_r = Q_r$; then return to Step 2; otherwise: go to Step 5.

Step 5: customer point’s demand is split and use up the remaining capacity of current path. Then set $d = d - Q_c$, $j = j + 1$, $Q_r = Q_r$ and return Step 2.

Step 6: start adjusting locations of splitting points and set $j = 1$.

Step 7: if $|SPc| = 0$ for current path $j$, go to Step 8; otherwise go to Step 9.

Step 8: $j = j + 1$. If $j > n$: go to Step 10; otherwise return Step 7.

Step 9: for the splitting point in $SPc$, find the best adjusting position, then return to Step 8.

Step 10: calculate total travel distance then stop.

An example is given to illustrate the process of decoding:

An example is given to illustrate the process of decoding: it is assumed that there are 7 customers, $Q = 100$ and demand and coordinate of each customer are shown in table 3. For an individual [1]–[7], the decoding results are shown in figures 1 and 2, and each actual path adds 0 to both ends: 0-7-1-0, 0-6-0, 0-4-3-0, 0-5-2-3-0. The objective function value obtained by TS algorithm is as follows:

\[ \text{FIGURE 1. Decoding result obtained by phase 1.} \]

\[ \text{FIGURE 2. Final decoding result.} \]

B. TABU SEARCH OPERATION

Tabu Search (TS) algorithm is a meta-heuristic first presented by Glover in 1986. TS algorithm is an extension of local
TABLE 3. Demands and coordinates of seven customers.

| Customer point | Demand | Coordinate |
|----------------|--------|------------|
| 0              | 0      | (45,15)    |
| 1              | 10     | (51,64)    |
| 2              | 3      | (59,61)    |
| 3              | 116    | (30,54)    |
| 4              | 32     | (28,52)    |
| 5              | 33     | (63,47)    |
| 6              | 73     | (46,46)    |
| 7              | 84     | (61,42)    |

neighborhood search and a simulation of human memory function. Its basic idea is to use a tabu list to record the local optimal solution information found recently and to query tabu list in subsequent search to avoid roundabout search, to jump out of the local optimality and ensure a diversified and effective search [24].

In order to enhance the global optimization capability, Part of the idea of tabu search algorithm is introduced as an operation of HDDE. Before the formal description of HDDE, a brief introduction is presented on the tabu search operation (TSO). TSO consists of four following elements.

1) INITIAL INDIVIDUAL
Different from the previous neighborhood search methods that directly operate paths in the existing literatures, because the designed decoding method has met the capacity constraint, tabu search is performed directly for an individual, and one individual in the process of HDDE is designated as initial solution.

2) TABU OBJECTIVE AND TABU LENGTH
For convenience, the objective function value of the individual is tabued in a tabu list. Tabu length is measured experimentally in appendix.

3) NEIGHBORHOOD OPERATOR
Three neighbor operators including relocate operator, exchange operator and 2-opt operator are adopted to increase the diversity of neighborhood solutions (neighborhood individuals).

4) THE NUMBER OF NEIGHBORHOOD SOLUTIONS SELECTED \( (n_s) \)
Unlike the classical tabu search algorithm, TSO doesn’t carry out multiple iterations. After neighborhood operator searching about the initial solution once, a certain number of candidate neighborhood solutions are selected depending on when HDDE is applied.

In subsequent sections, the symbol TSO \(-X_i - n_s - XS\) is applied to indicate the use of TSO, representing that TSO start from the initial individual \(X_i\) to find the \(n_s\) best neighborhood individuals to update the individual set \(XS\), where \(|XS| = n_s\).

C. HYBRID DIFFERENTIAL DISCRETE EVOLUTION ALGORITHM
Two heuristic methods are introduced to generate the initial population. During each iteration, individuals in the current population are successively mutated, crossed and selected. The DE algorithm may stall in the later stage, so the distur-
bance mechanism is designed. The detailed processes of these operations are shown in the following subsections.

1) INITIAL SOLUTION GENERATION

The target population consists of \( \Lambda \) target individuals in which the \( h \)-th target individual is \( X_h^t = [x^t_{h,1}, x^t_{h,2}, \ldots, x^t_{h,s}] \), where \( h = 1, 2, \ldots, \Lambda \). \( \tau \) is the current iteration and \( \Lambda \) is the population size. Firstly, two initial individuals of the initial population are generated by sweep algorithm and nearest neighborhood algorithm.

a: \( X_1^0 \) BASED ON SWEEP ALGORITHM

Given coordinates of depot and all customer points, a coordinate system is established with the depot as the origin, and an arc is sent out from the depot. The arc starts at an angle of zero and rotates counterclockwise, then the sequence that the customer points are scanned is taken as \( X_1^0 \) [25].

\( X_1^0 \) is DE/best/2. The mutant individual obtained by mutation operation can increase the perturbation of target individual, i.e., with \( X_1^0 \) as the initial individual, perform TSO and reserve \((\Lambda - 2)/2\) best neighborhood individuals as \( XS' = \{X_{h'}^0, h' = 3, 5, \ldots, \Lambda - 1\} \).

b: \( X_2^0 \) BASED ON NEAREST NEIGHBORHOOD ALGORITHM

First, put the customer point closest to the depot into \( X_2^0 \), and then put the one closest to the current last customer point into \( X_2^0 \) in turn, until all the customer points are put into \( X_2^0 \) [26]. Initial individuals \( X_1^0 \) and \( X_2^0 \) have been generated, and the remaining individuals are multiple neighborhood individuals of TSO based on the two initial individuals, where \( \Lambda \) is assumed as the even number. Namely, the following two operations are performed:

1) TSO \(-X_1^0 - (\Lambda - 2)/2 - XS' \), i.e., with \( X_1^0 \) as the initial individual, perform TSO and reserve \((\Lambda - 2)/2\) best neighborhood individuals as \( XS' = \{X_{h'}^0, h' = 3, 5, \ldots, \Lambda - 1\} \).

2) TSO \(-X_2^0 - (\Lambda - 2)/2 - XS'' \), i.e., with \( X_2^0 \) as the initial individual, perform TSO and reserve \((\Lambda - 2)/2\) best neighborhood individuals as \( XS'' = \{X_{h''}^0, h'' = 4, 6, \ldots, \Lambda \} \).

The purpose of interlacing generation is to evenly distribute the two groups of neighborhood solutions. Therefore, the objective value of individuals in the initial population can be calculated by the decoding method.

2) MUTATION OPERATION

Near-optimal solution is updated as the population iterates. The mechanism of mutation operation adopted in this study is DE/best/2. The mutant individual obtained by mutation operation is \( V_h^t = [v_{h,1}^t, v_{h,2}^t, \ldots, v_{h,s}^t] \).

For each target individual, a mutant individual \( V_h^t \) is obtained by the following formula:

\[
V_h^t = X_{\text{best}}^t + Z \circledast (X_{a_1}^{t-1} - X_{b_1}^{t-1}) \circledast Z \circledast (X_{a_2}^{t-1} - X_{b_2}^{t-1})
\]

where \( X_{\text{best}}^{t-1} \) is the best individual, \( X_{a_1}^{t-1}, X_{b_1}^{t-1}, X_{a_2}^{t-1} \) and \( X_{b_2}^{t-1} \) are four different target individuals at iteration \((\tau - 1)\), \( \alpha_1, \beta_1, \alpha_2 \) and \( \beta_2 \) are random integers in \([1, \Lambda]\), and \( Z \in [0,1] \) is a mutation probability. The operator \( \circledast \) is calculated as follows:

\[
G_h^t = Z \circledast (X_{a_1}^{t-1} - X_{\text{best}}^{t-1}) \leftrightarrow g_{h,q}^t
\]

where \( G_h^t = \{g_{h,1}^t, g_{h,2}^t, \ldots, g_{h,s}^t\} \) is a temporary individual. The operator \( \circledast \) is calculated as follows:

\[
V_h^t = X_{\text{best}}^{t-1} \oplus G_{h_1}^t \oplus G_{h_2}^t \leftrightarrow v_{h,q}^t \equiv \mod \left( \left\lfloor \left( X_{\text{best}}^{t-1} + g_{h_1,q}^t + g_{h_2,q}^t + n - 1 \right) / n \right\rfloor \right) + 1
\]

where operator \( \mod \) is modulus. Individual \( V_h^t \) maybe not valid because some encoding elements may repeat or not exist. An instance is provided in Table 4-6 to explain the mutation process.

3) Crossover Operation

Mutant individual can increase the perturbation of target individual and jump the local optimum [27]. The crossover operation is performed after mutation operation to obtain a trial individual \( U_h^t = [u_{h,1}^t, u_{h,2}^t, \ldots, u_{h,n}^t] \) that is valid. In the crossover operation phase, selective parts of mutant individual \( V_h^t \) are put into a target individual \( X_h^{t-1} \) to produce

\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
\text{TABLE 4. Instance of the mutation operation (Z = 0.4) (1).} \\
\hline
X_{a_1}^{t-1} - X_{b_1}^{t-1} & X_{a_2}^{t-1} - X_{b_2}^{t-1} \\
\hline X_{a_1}^{t-1} & 7 & 1 & 4 & 3 & 2 & 6 \\
X_{a_2}^{t-1} & 3 & 1 & 6 & 5 & 7 & 2 & 4 \\
X_{a_1}^{t-1} - X_{b_1}^{t-1} & 4 & 4 & -5 & -1 & -4 & 0 & 2 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
\text{TABLE 5. Instance of the mutation operation (Z = 0.4) (2).} \\
\hline
X_{a_1}^{t-1} - X_{b_1}^{t-1} & X_{a_2}^{t-1} - X_{b_2}^{t-1} \\
\hline X_{a_1}^{t-1} & 2 & 7 & 5 & 3 & 6 & 4 & 1 \\
X_{a_2}^{t-1} & 4 & 1 & 2 & 3 & 6 & 7 & 5 \\
X_{a_1}^{t-1} - X_{b_1}^{t-1} & -2 & 6 & 3 & 0 & 0 & -3 & -4 \\
G_{h_1}^t & 0 & 4 & 0 & 0 & 0 & 0 \\
G_{h_2}^t & 2 & 0 & 3 & 0 & 0 & 0 & 0 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
\text{TABLE 6. Instance of the mutation operation (Z = 0.4) (3).} \\
\hline
V_h^t & X_{\text{best}}^{t-1} & X_{a_1}^{t-1} & X_{b_1}^{t-1} & X_{a_2}^{t-1} & X_{b_2}^{t-1} \\
\hline X_{\text{best}}^{t-1} & 1 & 3 & 6 & 5 & 4 & 2 & 7 \\
G_{h_1}^t & 0 & 4 & 0 & 0 & 0 & 0 \\
G_{h_2}^t & 2 & 0 & 3 & 0 & 0 & 0 & 0 \\
V_h^t & 6 & 7 & 2 & 5 & 4 & 2 & 7 \\
\hline
\end{array}
\]

where \( G_h^t = \{g_{h,1}^t, g_{h,2}^t, \ldots, g_{h,s}^t\} \) is a temporary individual. The operator \( \circledast \) is calculated as follows:

\[
V_h^t = X_{\text{best}}^{t-1} \oplus G_{h_1}^t \oplus G_{h_2}^t \leftrightarrow v_{h,q}^t \equiv \mod \left( \left\lfloor \left( X_{\text{best}}^{t-1} + g_{h_1,q}^t + g_{h_2,q}^t + n - 1 \right) / n \right\rfloor \right) + 1
\]

where operator \( \mod \) is modulus. Individual \( V_h^t \) may not valid because some encoding elements may repeat or not exist. An instance is provided in Table 4-6 to explain the mutation process.
a trial individual \( U^*_h \). Three-point insertion method is used to increase the disturbance. The insertion points \( \rho 1, \rho 2 \) and \( \rho 3 \) are three different positions of target individual, where three parts of mutant individual are inserted, and they are different random integers in \([1, n]\). For each encoding element of \( V^*_h \), if a random number \( \text{rand}() < Y \), it can be placed into \( X^r_{h} \), where \( Y \in [0,1] \) is a crossover probability; otherwise, it is deleted. The encoding element appear more than once in \( U^*_h \) are omitted from right to left, which controls that each customer number in \( U^*_h \) exists once only.

For \( V^*_h \) in Table 6, the crossover process of \( X^r_{h} \) is presented in Table 7. The crossover probability \( Y = 0.7 \) and the insertion points are \( \rho 1 = 1, \rho 2 = 3 \) and \( \rho 3 = 6 \).

| \( V^*_h \) | \( U^*_h \) | \( V^*_h \) | \( U^*_h \) |
|---|---|---|---|
| 2 5 1 7 6 4 3 | 2 5 1 7 6 4 3 | 6 7 2 5 4 2 7 | 0.35 0.89 0.73 0.24 0.67 0.91 0.52 |
| \( V^*_h \) | \( U^*_h \) | \( V^*_h \) | \( U^*_h \) |
| 6 5 4 7 | 2 6 5 5 1 4 7 6 4 7 3 |

**TABLE 7. Instance of the crossover operation \((Y = 0.7)\)**

4) **SELECTION OPERATION**

The new generation of individuals comes from trial individuals or the last generation’s neighborhood individual. Trial individuals with better objective values can be selected as the next generation of individuals. To make full use of the neighborhood solutions generated in the iterative process, if a trial individual \( U^*_h \) can’t update the individual \( X^r_{h} \) of last generation, perform TSO \(-X^r_{h} - 1 - (X^r_{h} \alpha_{h} = 1) \), i.e., with \( X^r_{h} \) as the initial individual, perform TSO and reserve \( \alpha \) best neighborhood individuals to replace \( \alpha \) best neighborhood individual to replace \( \{X^r_{h}\} \). The best individual of the current population may need to be updated.

5) **DISTURBANCE OPERATION**

The selection operation of the DDE algorithm adopts the greedy selection strategy, which can accelerate the convergence, but easily result in the local optimum. To enhance the global optimization ability, DDE algorithm is improved as follow:

(1) If there is no improvement on best individual of current population within two consecutive iterations, perform TSO \(-X^r_{\text{best}} - (\Lambda - 1) - (X^r_{h} \alpha_{h = 1} - X^r_{\text{best}}) \), i.e., with \( X^r_{\text{best}} \) as the initial individual, perform TSO and reserve \( \alpha \) best neighborhood individuals to replace the current population except \( X^r_{\text{best}} \). It can enhance the local optimization ability and improve the quality of the whole population once.

(2) If there is no improvement on best individual of current population within ten consecutive iterations, perform TSO \(-X^r_{\text{rand}} - (\Lambda - 1) - (X^r_{h} \alpha_{h = 1} - X^r_{\text{rand}}) \), i.e., with a random individual \( X^r_{\text{rand}} \) as the initial individual, perform TSO and reserve \( \alpha \) best neighborhood individuals to replace the current population except \( X^r_{\text{rand}} \). Therefore, the local optimal value can be jumped out by disturbing the population.

In the above two cases, the best individual of the current population may need to be updated.

**D. ALGORITHM FLOWCHART**

The concrete procedures of HDDE are expressed as follows [28].

**Step 1:** Initialize parameters including mutation probability \( Z \), crossover probability \( Y \), population size \( \Lambda \), tabu list length \( TL \) and maximum iteration \( \tau_{\text{max}} \).

**Step 2:** Initialize the population.

**Step 2.1:** Execute two heuristic method to obtain two sequences including all costumers.

**Step 2.2:** Initialize the initial population including two heuristic-based individuals and \( \Lambda - 2 \) target individuals generated by TSO \(-X^r_{0} - (\Lambda - 2) / 2 - X^r \) and TSO \(-X^r_{0} - (\Lambda - 2) / 2 - X^r \). Set the generation \( \tau = 0 \) and find \( X^r_{0 \text{best}} \).

**Step 3:** Set \( h = 1 \) and start iteration.

**Step 4:** Choose four different random individuals \( X^r_{\alpha_{h_1}} \), \( X^r_{\alpha_{h_2}} \), \( X^r_{\alpha_{h_3}} \) and \( X^r_{\beta_{h_2}} \), where \( \alpha_1, \beta_1, \alpha_2 \) and \( \beta_2 \) are different random integers in \([1, \Lambda]\).

**Step 5:** Perform the mutation and crossover to obtain the trial individual \( U^*_h \).

**Step 6:** If objective values satisfy \( Z(U^*_h) < Z(X^r_{h}) \), then set \( X^r_{\text{best}} = U^*_h \); otherwise, perform TSO \(-X^r_{h} - 1 - (X^r_{h} \alpha_{h = 1}) \) and update \( X^r_{\text{best}} \), otherwise go to Step 8.

**Step 7:** If \( h \leq \Lambda \), set \( h = h + 1 \), and return to Step 4. Otherwise, go to Step 8.

**Step 8:** Update \( X^r_{\text{best}} \). If there is no improvement of current population in two consecutive iterations, perform TSO \(-X^r_{\text{best}} - (\Lambda - 1) - (X^r_{h} \alpha_{h = 1} - X^r_{\text{best}}) \) and update \( X^r_{\text{best}} \), otherwise go to Step 10.

**Step 9:** Update \( X^r_{\text{best}} \). If there is no improvement of current population in ten consecutive iterations, perform TSO \(-X^r_{\text{rand}} - (\Lambda - 1) - (X^r_{h} \alpha_{h = 1} - X^r_{\text{rand}}) \) and update \( X^r_{\text{best}} \).

**Step 10:** If \( \tau < \tau_{\text{max}} \), set \( \tau = \tau + 1 \), then return to Step 3. Otherwise go to Step 11.

**Step 11:** Output the historical optimal sequence as the final solution. Stop.

**IV. COMPUTATIONAL RESULTS**

To demonstrate the domination of the proposed HDDE algorithm, experiments with the following two sets of SDVRP benchmark instances were conducted:

Set 1: this set contains 25 instances generated by Belenguer et al. [3]. The number of customers ranges from 21 to 100. The coordinates originate in TSPLIB instances. The customer demand is randomly generated through uniform distribution according to six scenarios \([0.01, 0.1], [0.1, 0.3], [0.1, 0.5], [0.1, 0.9], [0.3, 0.7] \) and \([0.7, 0.9]\).

Set 2: this set contains 42 instances generated by Archetti et al. [11], derived from the capacitated VRP (CVRP) bench-
mark instances. The number of customers ranges from 50 to 199, and the generation scenarios of customer demands are the same with set 1.

In this paper, HDDE is compared with two competitive algorithms: (1) the forest-based tabu search (FBTS) algorithm by Zhang et al. [19]; and (2) Genetic algorithm (GA). The proposed HDDE algorithm and GA algorithm were coded in C++ language and run on a PC with an Intel Core i7 CPU (2.6GHz × 12) and 8 GB RAM. The parameters of HDDE is tested through orthogonal experimental design: mutation probability: 0.9, crossover probability: 0.6, population size: 50, tabu list length: 300, maximum iterations: 300. The specific process of orthogonal experimental design is shown in appendix.

The experimental results are shown in the tables 8 and 9. The results of FBTS were obtained directly from the literature [19]. The initial population generation method of GA is the same as HDDE. Because the comparison algorithms were carried out on different machines, running times of FBTS and GA are not reported. Column “Cost” of HDDE, column “FBTS” and column “GA” respectively show final objective values (final total travel distance) of three algorithms.

| Instances | HDDE Cost (s) | FBTS Cost (%) | GAP1 (%) | GA Cost (%) | GAP2 (%) |
|-----------|--------------|---------------|----------|-------------|----------|
| eil22     | 375.3        | 102.9         | 375.3    | 0           | 383.5    |
| eil23     | 568.6        | 111.4         | 568.6    | 1.23        | 512.9    |
| eil30     | 512.7        | 202.3         | 519.0    | 0.78        | 512.9    |
| eil33     | 837.7        | 259.7         | 837.1    | 0.07        | 868.8    |
| eil51     | 524.6        | 648.6         | 528.0    | 0.65        | 579.2    |
| eilA76    | 833.0        | 1700.9        | 842.7    | 1.16        | 873.3    |
| eilB76    | 1023.2       | 1811.4        | 1017.1   | 0.60        | 1099.3   |
| eilC76    | 750.1        | 1564.9        | 754.3    | 0.56        | 806.8    |
| eilD76    | 694.1        | 1546.7        | 701.1    | 1.01        | 769.5    |
| eilA101   | 835.4        | 2860.3        | 838.8    | 0.41        | 955.6    |
| eilB101   | 1124.5       | 3153.8        | 1096.1   | 2.53        | 1193.8   |
| set1      | 459.5        | 617.9         | 464.8    | 1.15        | 518.4    |
| set2      | 711.7        | 710.7         | 711.9    | 0.03        | 745.6    |
| set1      | 959.1        | 772.6         | 952.8    | 0.66        | 1037.6   |
| set2      | 1582.7       | 917.2         | 1587.8   | 0.32        | 1597.4   |
| set1      | 1336.2       | 866.5         | 1348.8   | 0.94        | 1418.6   |
| set2      | 2209.1       | 1053.9        | 2202.2   | 0.31        | 2301.4   |
| set1      | 605.7        | 1451.3        | 615.9    | 1.68        | 692.9    |
| set2      | 1110.2       | 1866.6        | 1103.6   | 0.59        | 1197.1   |
| set1      | 1457.7       | 2073.9        | 1449.8   | 0.54        | 1513.4   |
| set2      | 2126.9       | 2437.8        | 2108.6   | 0.86        | 2193.6   |
| set1      | 732.9        | 2800.2        | 745.7    | 1.75        | 860.4    |
| set2      | 1418.1       | 3587.8        | 1394.6   | 1.66        | 1512.1   |
| set1      | 1925.0       | 4136.3        | 1913.3   | 0.61        | 2097.7   |
| set2      | 2833.0       | 4998.0        | 2858.8   | 0.91        | 2939.9   |

Average 0.14 6.67

| Mutation probability | 0.0 0.2 0.4 0.6 0.8 1.0 |
|----------------------|-------------------------|
| Crossover probability| 0.0 0.1 0.2 0.3 0.4 0.5 |
| Population size      | 0 10 20 30 40 50       |
| Maximum iteration    | 0 50 100 150 200 250   |
| Tabu list length     | 0 50 100 150 200 250   |

GAP1 = \frac{Z_{FBTS} - Z_{HDDE}}{Z_{HDDE}} \times 100\% and GAP2 = \frac{Z_{GA} - Z_{HDDE}}{Z_{HDDE}} \times 100\% are employed to measure the performance of algorithm, where Z_{HDDE}, Z_{FBTS} and Z_{GA} denote the final objective values of HDDE, FBTS and GA respectively. When GAP is positive, the final solution obtained by HDDE is better than another algorithm.

In the experimental results of the two data sets, the percentage of cases that HDDE is superior to FBTS and GA are 61.2% and 100%, respectively. The average GAP1 between the two data sets is 0.49%, where set 1 is 0.14% and set 2 is 0.70%. Although these gaps are all less than 1%, considering that SDVRP has been extensively studied over the past 30 years, the improvements of the proposed HDDE are considerable. The average GAP2 between the two data sets is 5.96%, where set 1 is 6.67% and set 2 is 5.53%. It is found that HDDE algorithm is much better than GA. Therefore, HDDE algorithm designed in this study has achieved good results.

VOLUME 8, 2020
Table 9. Performance comparison based on set 2.

| Instances | HDDE | Cost | Time (s) | GA | GP | GPA |
|-----------|------|------|----------|----|----|-----|
| p1-00     | 524.6| 664.2| 532.0    | 1.41| 556.3| 6.04 |
| p1-110    | 461.8| 608.2| 461.0 -0.17| 490.6| 6.24 |
| p1-1030   | 781.7| 722.7| 759.8 -2.80| 793.8| 1.55 |
| p1-1050   | 1026.9| 809.0| 1026.5 -0.04| 1064.4| 6.35 |
| p1-1090   | 1497.2| 929.0| 1552.1 3.67| 1591.2| 6.28 |
| p1-1095   | 1495.6| 922.9| 1498.1 0.17| 1529.7| 2.28 |
| p1-7090   | 2192.3| 1095.7| 2191.4 -0.04| 2249.6| 2.61 |
| p2-00     | 828.2| 1775.1| 824.7 -10.04| 890.0| 7.46 |
| p2-110    | 620.8| 1516.7| 637.0 2.61| 734.6| 18.33 |
| p2-1030   | 1129.3| 1863.6| 1118.1 -0.99| 1186.1| 5.68 |
| p2-1050   | 1517.4| 2094.1| 1525.7 0.55| 1600.3| 5.46 |
| p2-1090   | 2324.8| 2502.1| 2358.8 1.46| 2439.1| 4.92 |
| p2-3070   | 2253.3| 2466.8| 2280.3 1.20| 2310.5| 2.54 |
| p2-7090   | 3269.1| 3003.5| 3239.0 -0.09| 3349.2| 2.68 |
| p3-00     | 840.1| 2858.2| 847.4 0.87| 963.1| 14.64 |
| p3-110    | 772.7| 2822.3| 792.2 2.52| 857.1| 9.65 |
| p3-1030   | 1506.2| 3758.0| 1476.9 -1.95| 1598.1| 6.10 |
| p3-1050   | 2048.5| 4136.7| 2032.2 -1.24| 2115.7| 3.28 |
| p3-1090   | 3154.3| 5330.3| 3181.3 0.86| 3203.6| 1.56 |
| p3-3070   | 3038.8| 5159.5| 3044.1 0.17| 3122.6| 2.76 |
| p3-7090   | 4422.2| 6342.1| 4441.7 -0.01| 4567.2| 2.00 |
| p4-00     | 1067.1| 7007.3| 1081.6 1.36| 1239.1| 16.12 |
| p4-110    | 941.1| 6959.6| 953.0 1.26| 1123.6| 19.39 |
| p4-1030   | 2067.3| 10225.2| 2060.4 -0.33| 2177.8| 5.35 |
| p4-1090   | 2907.0| 15163.5| 1342.5 0.69| 1482.3| 11.18 |
| p4-1095   | 3465.1| 17992.4| 1342.5 0.69| 1482.3| 11.18 |
| p5-00     | 1333.3| 21887.3| 1342.5 0.69| 1482.3| 11.18 |
| p5-110    | 3269.1| 2858.2| 3239.0 -0.09| 3349.2| 2.68 |
| p5-1030   | 2548.1| 26758.7| 2525.0 -0.91| 2709.0| 6.00 |
| p5-1050   | 3560.0| 31115.1| 3542.5 -0.49| 3716.3| 4.39 |
| p5-1090   | 5589.2| 39806.2| 5700.7 1.99| 5771.3| 3.25 |
| p5-3070   | 501.4| 39564.9| 5585.1 1.52| 5691.5| 3.46 |
| p5-7090   | 3030.7| 5077.3| 3255.4 -0.58| 3437.6| 1.61 |
| p1-00     | 1062.6| 6064.6| 1048.3 -1.35| 1088.8| 2.47 |
| p1-110    | 1067.7| 5974.4| 1119.2 4.08| 1097.1| 2.75 |
| p1-1030   | 2949.2| 7813.9| 2953.1 0.23| 2986.3| 0.65 |
| p1-1050   | 4278.1| 8967.8| 4298.4 0.47| 4508.9| 0.72 |
| p1-1090   | 6890.1| 10994.5| 7206.2 4.59| 6911.8| 0.31 |
| p1-3070   | 6681.3| 10864.5| 6858.1 2.65| 6712.0| 0.46 |
| p1-7090   | 10231.3| 13063.6| 10285.7 0.53| 10270.2| 0.38 |

Average 0.70 5.53

Table 10. Orthogonal factor level table of HDDE.

| Factors | Mutation probability (Z) | Crossover probability (C) | Population size (A) | Maximum iteration (Tmax) | Tabu list length (TL) |
|---------|--------------------------|---------------------------|---------------------|-------------------------|----------------------|
| Level 1  | 0.3                      | 0.3                       | 50                  | 100                     | 300                  |
| Level 2  | 0.6                      | 0.6                       | 100                 | 300                     | 600                  |
| Level 3  | 0.9                      | 0.9                       | 150                 | 500                     | 900                  |

The HDDE algorithm that employs the tabu search operator provides a good balance between the global and local optimization capabilities. Moreover, the capacity constraint is only placed in the decoding part, which simplifies the operation of neighborhood operators. However, the running time of this algorithm is relatively long. The time is mainly spent on the path distance calculation during the decoding phase, so path structure and programming skills need to be further improved.

V. CONCLUSION

SDVRP relieves the constraint that one customer can only be accessed by only one vehicle, which is conducive to save vehicle resources and shorten the travel distance. However, this kind of problem is more complicated than the traditional VRP problem, and solution method is quite different. This article investigates SDVRP to minimize total travel distance. For this NP-hard problem, a hybrid discrete differential evolution algorithm is presented to obtain near-optimal solution of high quality. Numerical simulations verify the dominance of the HDDE.

Future research could mainly focus on two aspects: (1) Customer point location will be considered to determine whether a customer's demand should be split so that further optimization can be carried out in the decoding stage; and (2) future research will consider the additions of time windows, multi-types vehicle and other practical common factors to design an appropriate algorithm.

APPENDIX

Parameters setting for HDDE:

The five factors that affect the performance of HDDE are tested. Factors and levels needed to test are provided in table 10 and the orthogonal experimental result is shown in table 11. Figure 7 indicate mean of experimental results at each level. Each group of experiment is tested randomly ten times: n = 50 and improved percentage of the algorithm is recorded. The column “Experimental result” is the average improved percentage of 10 random tests. Improved percentage $Z_{IS} - Z_{FS}$ denotes the objective value of the initial solution and the final solution obtained by HDDE respectively.

According to the experimental results in figure 7, the value of each parameter is determined as: mutation probability: $\frac{1\%}{207970}$.
0.9, crossover probability: 0.6, population size: 50, maximum iteration: 300, tabu list length: 300.

REFERENCES

[1] M. Dror and P. Trudeau, “Savings by split delivery routing,” Transp. Sci., vol. 23, no. 2, pp. 141–145, May 1989.

[2] M. Dror and P. Trudeau, “Split delivery routing,” Nav. Res. Logistics, vol. 37, no. 3, pp. 383–402, Jun. 1990.

[3] J. M. Belenguer, M. C. Martínez, and E. Mota, “A lower bound for the split delivery vehicle routing problem,” Oper. Res., vol. 48, no. 5, pp. 801–810, Oct. 2000.

[4] C. Archetti, R. Mansini, and M. G. Speranza, “Complexity and reducibility of the skip delivery problem,” Transp. Sci., vol. 39, no. 2, pp. 182–187, May 2005.

[5] G. Moretti, M. W. P. Savelsbergh, and M. G. Speranza, “Worst-case analysis for split delivery vehicle routing problems,” Transp. Sci., vol. 40, no. 2, pp. 226–234, May 2006.

[6] C. Archetti, M. W. P. Savelsbergh, and M. G. Speranza, “To split or not to split: That is the question,” Transp. Res. e-Logist. Transp. Rev., vol. 44, no. 1, pp. 114–123, Jan. 2008.

[7] M. Dror, G. Laporte, and P. Trudeau, “Vehicle routing with split deliveries,” Discrete Appl. Math., vol. 50, no. 3, pp. 239–254, May 1994.

[8] C.-G. Lee, M. A. Epelman, C. C. White, and Y. A. Bozer, “A shortest path approach to the multiple-vehicle routing problem with split pick-ups,” Transp. Res. B, Methodol., vol. 40, no. 4, pp. 265–284, May 2006.

[9] M. Jin, K. Liu, and R. O. Bowden, “A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem,” Int. J. Prod. Econ., vol. 105, no. 1, pp. 228–242, Jan. 2007.

[10] L. Moreira, M. P. de Araújo, and E. Uchoa, “Improved lower bounds for the split delivery vehicle routing problem,” Oper. Res. Lett., vol. 38, no. 4, pp. 302–306, Jul. 2010.

[11] C. Archetti, N. Bianchessi, and M. G. Speranza, “A column generation approach for the split delivery vehicle routing problem,” Networks, vol. 58, no. 4, pp. 241–254, Oct. 2011.

[12] C. Archetti, N. Bianchessi, and M. G. Speranza, “Branch-and-cut algorithms for the split delivery vehicle routing problem,” Eur. J. Oper. Res., vol. 238, no. 3, pp. 685–698, Nov. 2014.

[13] C. Archetti, M. G. Speranza, and A. Hertz, “A tabu search algorithm for the split delivery vehicle routing problem,” Transp. Sci., vol. 40, no. 1, pp. 64–73, Feb. 2006.

[14] R. E. Aleman and R. R. Hill, “A tabu search with vocabulary building approach for the vehicle routing problem with split demands,” Int. J. Metaheuristics, vol. 1, no. 1, pp. 55–80, May 2010.

[15] U. Derigs, B. Li, and U. Vogel, “Local search-based metaheuristics for the split delivery vehicle routing problem,” J. Oper. Res. Soc., vol. 61, no. 9, pp. 1356–1364, Sep. 2010.

[16] J. H. Wilck IV and T. M. Cavalier, “A genetic algorithm for the split delivery vehicle routing problem,” Amer. J. Oper. Res., vol. 02, no. 02, pp. 207–216, 2012.

[17] L. Berbóto, S. García, and F. J. Nogales, “A randomized granular tabu search heuristic for the split delivery vehicle routing problem,” Ann. Oper. Res., vol. 222, no. 1, pp. 153–173, Nov. 2014.

[18] M. M. Silva, A. Subramanian, and L. S. Ochi, “An iterated local search heuristic for the split delivery vehicle routing problem,” Comput. Oper. Res., vol. 53, pp. 234–249, Jan. 2015.

[19] Z. Zhang, H. He, Z. Luo, H. Qin, and S. Guo, “An efficient forest-based tabu search algorithm for the split-delivery vehicle routing problem,” in Proc. AAI. Austin, TX, USA, 2015, pp. 3432–3438.

[20] G. P. Rajappa, J. H. Wilck, and J. E. Bell, “An ant colony optimization and hybrid metaheuristics algorithm to solve the split delivery vehicle routing problem,” Int. J. Appl. Ind. Eng., vol. 3, no. 1, pp. 55–73, Jan. 2016.

[21] J. Shi, J. Zhang, K. Wang, and X. Fang, “Particle swarm optimization for split delivery vehicle routing problem,” Asia-Pacific J. Oper. Res., vol. 35, no. 02, Apr. 2018, Art. no. 1840006.

[22] A. Ponsich and C. A. C. Coello, “A hybrid differential evolution—Tabu search algorithm for the solution of job-shop scheduling problems,” Appl. Soft Comput., vol. 13, no. 1, pp. 462–474, Jan. 2013.

[23] R. Storn and K. Price, “Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces,” J. Global Optim., vol. 11, no. 4, pp. 341–359, 1997.

[24] F. Glover, “Future paths for integer programming and links to artificial intelligence,” Comput. Oper. Res., vol. 13, no. 5, pp. 533–549, Jan. 1986.
WIN-CHIN LIN received the Ph.D. degree from the Department of Statistics, Iowa State University, USA, in 1998. He is currently an Associate Professor with the Department of Statistics, Feng Chia University, Taiwan. His teaching and research interests include applied statistics, experimental designs, and scheduling.

WU SONG received the B.S. degree in computer science and technology and the M.S. degree in computer software and theory from Xiangtan University, Xiangtan, China, in 2004 and 2007, respectively, and the Ph.D. degree in computer applications from Central South University, Changsha, China, in 2016. He is currently an Associate Professor with Hainan Tropical Ocean University, Sanya, China. His current research interests include evolutionary computation and multi-objective optimization.