IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 3, 2019, accepted September 24, 2019, date of publication September 30, 2019,
date of current version October 18, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2944739

Dynamic Pick-Up and Delivery Optimization With Multiple Dynamic Events in Real-World Environment

BAOFENG SUN, YUE YANG, JUNYAN SHI, AND LILI ZHENG
College of Transportation, Jilin University, Changchun 130025, China

Corresponding author: Lili Zheng (lilizheng@jlu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 51308249, and in part by the Transportation Science and Technology Project of Jilin Province of China under Grant 20160112.

ABSTRACT Real-time city distribution strategies are highly dependent on dynamic environments, requiring timely responses to real-time changes due to various dynamic events that take place in the distribution system. Considering the influence of four kinds of real-time information on vehicle routing and vehicle scheduling, including new requests arriving gradually, old requests being modified or canceled, traffic congestion and vehicle breakdowns, a dynamic vehicle routing model based on a dynamic pick-up and delivery problem considering multiple dynamic events in a real-world environment (DPDP-MDE) is established. A dynamic algorithm framework is designed to solve the problem, the tabu search (TS) algorithm and the adaptive large neighborhood search (ALNS) algorithm are adopted to improve the quality of the initial solution, and the dynamic insertion method is adopted to solve the synchronization problem of unfixed requests (that is, unaccepted customer requests and modified requests) and new requests. The experimental results show that the model and dynamic algorithm framework proposed in this paper can effectively solve the dynamic pick-up and delivery problem with time windows (DPDP-TW). At different scheduling time horizons T, the TS algorithm improves the initial solution by an average of 3.11% and the ALNS algorithm by an average of 9.98%. Under different degrees of urgency, compared to the ALNS algorithm, the quality of the solution produced by the TS algorithm is not high, but the computation time is very small and it is relatively stable. Under different request sizes, the TS algorithm can obtain optimization results in 60s under four request levels, which gives it a significant advantage over the ALNS algorithm.

INDEX TERMS Dynamic pick-up and delivery problem, dynamic events, dynamic algorithm framework, constructive algorithm, tabu search algorithms, adaptive large neighborhood search algorithms.

I. INTRODUCTION

Real-time urban distribution is characterized by point-to-point, small batch and multi-frequency requests, that is, unaccepted customer requests, new additional dynamic requests and modifications of requests, which means there are higher technical requirements for timely responses and high-level flexibility. Real-time city distribution strategies are highly dependent on dynamic environments, requiring timely responses to real-time changes caused by various dynamic events occurring in the distribution system. Therefore, the dynamic pick-up and delivery problem with time windows (DPDP-TW) deserves more attention. Research on the DPDP-TW has attracted the attention of enterprises and scholars, and real-time urban distribution has developed rapidly. In this paper, we considered the influence of four kinds of real-time information, including new requests arriving gradually, old requests being modified or canceled, traffic congestion, and vehicle breakdowns, based on the traditional DPDP-TW issues, in what is known as the dynamic pick-up and delivery problem considering multiple dynamic events (DPDP-MDE) in a real-world environment. With this approach, we attempt to reveal the dynamic influence of dynamic events in a real-world scenario and improve real-time dynamic decision-making and rapid response.

The associate editor coordinating the review of this manuscript and approving it for publication was Baozhen Yao.

For DPDP-TW and DPDP-MDE modeling, most researchers have considered the impact of single dynamic events to simplify the problem. A few works, such as Ferrucci and Bock and Stefan Bock (2014) [1], introduced the dynamic pick-up and delivery problem with real-time control (DPDPRC) in order to map urgent real-world transportation services. This is a proposed real-time control method for the simultaneous response and effective handling of three dynamic events - new request arrivals, traffic congestion and vehicle disturbance. The method includes synchronous control of route planning and the execution of transport services (requests) and adopts the tabu search (TS) algorithm for planning adjustment.

For a single dynamic event of a new request, Fabri and Recht (2006) [2] adopted a new insertion method for the arrival of new requests. Once a new request occurs, the algorithm updates the current route plan. At this time, the new request is continuously allocated to each vehicle, which converts the problem into a single-vehicle route-planning problem. That is, when a feasible route plan can be obtained, the request is accepted and assigned to the vehicle which generates the lowest additional cost; otherwise, the request is refused. Mitrovic and Laporte [3] adopted the waiting strategy of the cheapest insertion procedure to insert a new request. They compared four waiting strategies: drive-first, wait-first, dynamic waiting and advanced dynamic waiting (ADW). The best empirical results were achieved with ADW, while the other strategies had advantages and disadvantages, which were discussed. In addition to the waiting strategy, Branke, Middendorf, Middendorf *et al.* [4]–[7] adopted the buffering strategy, in which request buffers are put in place for a period of time or the vehicle is moved to a node where future requests can easily reach it when they arrive. Furthermore, Pureza and Laporte [8] demonstrated the advantages of using the waiting strategy and the buffering strategy to integrate new requests into an unlimited capacity DPDP-TW.

For a single dynamic event regarding traffic congestion, Kok *et al.* [9] adopted alternative route selection, changing customer access sequences and changing vehicle assignments to avoid predictable traffic congestion in a planned route. They proposed a model of vehicle speeds during peak traffic congestion and adopted the improved Dijkstra algorithm and limited dynamic programming algorithm to eliminate about 87% of traffic congestion in route planning. Sun *et al.* [10] optimized the DPDP-TW for a transportation service by considering two aspects. On the one hand, the transportation service provider can choose the transportation requests it serves in order to maximize profit. On the other hand, it can take advantage of periods of light traffic by dictating to drivers when their routes should begin.

For a single dynamic event involving vehicle breakdown, the situation is a small probability event, so there is less research on this aspect. Mamasis *et al.* (2013) [11] studied the problem of vehicle breakdown in the urban single product distribution process, considering the maximum time constraint, maximum distance constraint and vehicle capacity constraint based on the Team Orienteering Problem model. The routing model is re-constructed, and the solution is almost instantaneous in real time based on the fast labelling algorithm.

Regarding the DPDP-TW problem under time-varying conditions, Paolo and Daniele [12] systematically analyzed the routing problem under time-varying conditions for the modeling, application and solving of an optimal solution for driving time. Sun *et al.* [10] studied a series of DPDP-TW problems under time-varying conditions, using a branch and price algorithm to improve the algorithm framework according to various acceleration techniques. They evaluated the effectiveness of the improved algorithm framework and the impact of the acceleration techniques using numerical examples. Kritzinger *et al.* [13] presented an experimental evaluation of an algorithm for time-dependent vehicle routing problems using real-world traffic information. They adopted Dijkstra's algorithm to integrate time-dependent travel times and used efficient data structures in order to minimize its run times. A Variable Neighborhood Search algorithm was applied and improved the solution quality significantly.

In a literature review, Berbeglia [14] summarized two optimization methods for DPDP-TW under time-varying conditions: the real-time method and the rolling time horizon method. Gendreau *et al.* [15] used the real-time method to solve the DPDP-TW. In the real-time method, overall re-optimization occurs when a new message appears. The disadvantage of this method is that it takes up too much computation time, making it unsuitable for real-time scenarios.

The rolling time horizon method was proposed and applied by Snezana et al. [3], [5]. They divided the scheduling time horizon into smaller time intervals and found an initial solution according to the static problem algorithm at the beginning of the scheduling time horizon. When the information arrives, the initial solution is updated by a heuristic algorithm such as the insertion method or deletion heuristics. The rolling time horizon method is superior to the real-time method in terms of calculation time, so it can be widely applied to real-time scenarios. Montemanni *et al.* [17] used a rolling time horizon framework and a segmentation-scheduling time horizon method to solve the dynamic vehicle routing problem. An advanced commitment strategy is adopted in which only specific time intervals in the future are planned. The request selection and insertion order are determined by the available slack time, the arrival time of requests, and the delivery time window. It can be seen that the combination of the rolling time horizon method and the algorithm of the static problem has significant advantages for solving the DPDP-MDE in a real-world environment. In a dynamic algorithm framework, the static problem-solving algorithm and its optimization are important for the solution of real-time problems. In this paper, the dynamic algorithm framework is improved, the TS algorithm and adaptive large-scale neighborhood search (ALNS) algorithm are used to improve the quality of the initial feasible solution, and the dynamic insertion method is adopted to solve the

synchronization problem of unfixed requests and the arrival of new requests.

Based on the literature review and analysis, this paper comprehensively considers the influence of dynamic events in a real-world environment of new requests arriving gradually, old requests being modified or canceled, traffic congestion and vehicle breakdown to establish the DPDP-MDE model. The rolling time horizon method is combined with the TS algorithm and the ALNS algorithm to improve the dynamic algorithm framework. The remainder of the paper is divided along the following lines. First, the DPDP-MDE is redefined based on the literature review in Section 1. Then, we describe the problem and establish the DPDP-MDE mathematical programming model in Section 2. The dynamic algorithm framework and its step flow is designed in Section 3. The numerical experiment and analysis are presented in Section 4. Finally, the research conclusions and prospects are provided in Section 5.

II. MODELING

A. PROBLEM DESCRIPTION

A dynamic vehicle routing optimization model based on real-time information is established as follows. All requests in this model are completed by vehicle dispatching. Each vehicle starts from the depot at the beginning of the dispatch period and must return to the depot before the end of the dispatch period. Each vehicle must first access the requested pick-up point and then travel to the appropriate delivery point to complete a request. Each request has a time window requirement for the pick-up point and the delivery point, and the requests appear dynamically throughout the model. For the vehicle routing, this model considers four factors beyond those in the traditional model: new requests arriving, request changes, traffic congestion, and vehicle breakdown. The vehicle routing scheme and scheduling plan are continuously updated throughout the scheduling time horizon, and the total vehicle distribution cost in the scheduling time horizon forms the objective function. The total vehicle distribution cost includes the sum of the vehicle operating cost and the penalty cost of any delay in service caused by the vehicle exceeding the time window.

Assuming each interval is τ, the scheduling time horizon T is partitioned into p intervals, $t_1, t_2, \ldots, t_p$, where $p = T/\tau$.

B. NOTATION

For convenience of description, notation is defined as follows:

1) PARAMETERS

K	the total number of vehicles;
C	the maximum length of the scheduling time horizon;
V	a set of all geographically distributed nodes;
$V = \{v_0, v_1, \cdots, v_n\}$,	where v_0 is the depot, and n is an even number;
Q	the maximum capacity of each vehicle;
D	the maximum driving distance of each vehicle;
t	the time point at which route planning is optimized in the scheduling time horizon;
$N(t)$	the location set together with unaccepted customer requests, new additional dynamic requests and modifications of requests,
$N(t) = V/v_0$,	which can be divided into two subsets of the same size;
N^+	the set of pick-up nodes;
N^-	the set of delivery nodes;
$N^+ \cup N^- = N(t)$,	$N^+ \cap N^- = \emptyset$;
$\lvert N^+\rvert, \lvert N^-\rvert$	the number of requests; $\lvert N^+\rvert = \lvert N^-\rvert$;
k	the k-th vehicle, $k = 1, 2, \ldots, K$;
d_{ij}	the distance between each pair of nodes $(v_i, v_j)(0 \le i \ne j \le n)$;
$s_k(t)$	the speed of vehicle k at time t, $k = 1, 2, \ldots, K$;
$t_{ij}^k(t)$	the vehicle travel time needed to move from node v_i to node v_j at time t, $t_{ij}^k(t) = d_{ij}/t_k(t)$, $k = 1, 2, \ldots, K$;
$D_k(t)$	the cumulative travel distance of vehicle k after departing from the depot at time t, $k = 1, 2, \ldots, K$;
$Q_k(t)$	the surplus capacity of vehicle k at time t, $k = 1, 2, \ldots, K$;
$W_p(t)$	the set of locations of all the vehicles in the network at time t;
$W_{p0}(t)$	the set of locations of all the vehicles and depots in the network at time t;
$W_{u0}(t)$	the set of locations of nodes related to unaccepted customer requests, newly added dynamic requests and depots at time t;
$W_{up0}(t)$	the set of locations of all vehicles, locations of unserved customer requests, newly added dynamic requests, modified requests and depots at time t;
ss_i	the operating time at node v_i, $v_i \in N$;
q_i	the demand quantity of node v_i, $v_i \in N$. For any pick-up node, $v_i \in N^+$, $q_i > 0$; for any delivery node, $v_i \in N^-$, $q_i < 0$;
$[e_i, l_i]$	the service time window at node v_i; e_i is the earliest time service may time start at node v_i; l_i is the latest service may start at node; at depot v_0, $q_0 = 0$, $ss_0 = 0$, $e_0 = 0$, $l_0 = C$;

a penalty cost of arriving later than the end of the time window;
β fixed cost per vehicle, including depreciation fee, maintenance cost, etc.;
c the unit cost of vehicle travel.
α the degree of urgency, $\alpha \in [0.1, 1]$.

2) VARIABLES

y_i^k the load of vehicle k after serving node v_i;
a_i^k the time needed for vehicle k to arrive at node v_i;
w_i^k the waiting time of vehicle $k$at node v_i, $w_i^k = \max\{0, e_i - a_i^k\}$;
de_i^k the total time it takes until vehicle k leaves node v_i, $de_i^k = a_i^k + w_i^k + ss_i$;
tl_i^k the delay with which vehicle k arrives at node v_i, $tl_i^k = \max\{0, de_i^k - l_i\}$;
d_i^k the travel distance after vehicle k has served node v_i, $d_0^k = 0$;

$$u_k = \begin{cases} 1, & \textit{vehicle } k \textit{ is used}; \\ 0, & \textit{else}. \end{cases} \quad k = 1, 2, \ldots, K;$$

n_t the number of vehicles planned for use at time t, $n_t = \sum_{k=1}^{K} u_k, k = 1, \ldots, K$;

$$z_{ij} = \begin{cases} 1, & \textit{if } v_i \textit{ and } v_j \textit{ are respectively the pick-up and delivery node for one request}; \\ 0, & \textit{else}. \end{cases}$$

$(0 \le i \ne j \le n)$.

3) DECISION VARIABLE

$$x_{ij}^k = \begin{cases} 1, & \textit{if vehicle } k \textit{ is dispatched from } v_i \textit{ to } v_j; \\ 0, & \textit{else}. \end{cases}, \quad k = 1, \ldots, K.$$

C. PROPOSED MODEL

The pick-up and delivery problem (PDP) in real time with a single mathematical programming model considering four kinds of information – new requests arriving gradually, old requests being modified or canceled, traffic congestion and vehicle breakdown – is proposed as follows:

$$\min Z = \sum_{i \in W_{up0}(t)} \sum_{j \in W_{i0}(t)} \sum_{k=1}^{K} c d_{ij} x_{ij}^k + \sum_{i \in W_{up0}(t)} \sum_{k=1}^{K} a t l_i^k + \beta n_t \tag{1}$$

$$\text{s. t.} \sum_{k=1}^{K} \sum_{j \in N(t)} x_{ij}^k = 1, \quad \forall i \in N(t) \tag{2}$$

$$\sum_{k=1}^{K} \sum_{j \in N(t)} x_{ij}^k = 1, \quad \forall i \in W_{p0}(t) \tag{3}$$

$$\sum_{i \in N(t) \cup W_p(t)} x_{i0}^k = 1, \quad \forall k = 1, \ldots, K \tag{4}$$

$$\sum_{i \in N(t)} x_{ih}^k - \sum_{j \in N(t)} x_{hj}^k = 0, \quad \forall h \in N(t), \quad \forall k = 1, \ldots, K \tag{5}$$

$$\sum_{l \in N(t)} x_{li}^k z_{ij} - \sum_{p \in N(t)} x_{pj}^k z_{ij} = 0, \quad \forall i, j \in N(t), \quad \forall k = 1, \ldots, K \tag{6}$$

$$y_j^k \le Q, \quad \forall j \in N(t), \ \forall k = 1, \ldots, K \tag{7}$$

$$y_j^k = Q_k(t), \quad \forall j \in W_p(t), \ \forall k = 1, \ldots, K \tag{8}$$

$$x_{ij}^k (y_j^k - y_i^k - q_j) = 0, \quad \forall i, j \in N(t), \ \forall k = 1, \ldots, K \tag{9}$$

$$x_{ij}^k (de_i^k + t_{ij}^k) \le a_j^k, \quad \forall i, j \in N(t), \ \forall k = 1, \ldots, K \tag{10}$$

$$de_i^k = \max\{a_j^k, e_i\} + ss_i, \quad \forall i \in N(t) \tag{11}$$

$$z_{ij} a_i^k \le a_j^k, \quad \forall i, j \in N(t), \ k = 1, \ldots, K \tag{12}$$

$$x_{i0}^k (de_i^k + t_{i0}^k) \le C, \quad \forall i \in N(t), \ \forall k = 1, \ldots, K \tag{13}$$

$$d_j^k \le D, \quad \forall j \in N(t), \ \forall k = 1, \ldots, K \tag{14}$$

$$d_j^k = D_k(t), \quad \forall j \in W_p(t), \ \forall k = 1, \ldots, K \tag{15}$$

$$x_{ij}^k (d_j^k - d_i^k - d_{ij}) = 0, \quad \forall i, j \in N(t), \ \forall k = 1, \ldots, K \tag{16}$$

$$x_{i0}^k (d_i^k + d_{i0}^k) \le D, \quad \forall i \in N(t), \ \forall k = 1, \ldots, K \tag{17}$$

Objective function (1) is the minimum of the total cost (Z) of the vehicle scheduling. Z includes three parts: the total travel cost of the vehicles, the penalty cost incurred by vehicles that serve nodes outside the time window, and the fixed costs of the scheduled vehicles.

The constraints described by expressions (2) to (17) are divided into six parts as follows:

1) BASIC CONSTRAINTS ON VEHICLE ROUTING PROBLEMS

Constraint (2) guarantees that each node is only accessed once by a vehicle, and the nodes in the dynamic PDP are the pick-up and delivery nodes at which new requests are added, requests are modified, and unmet requests occur due to vehicle breakdown; constraints (3) and (4) guarantee that each vehicle departs from the depot or the vehicle location and finally returns back to the depot. The position of the vehicle in the dynamic modeling problem can be one of three types: depots, the vehicle's position of the end of the previous interval, or the vehicle breakdown position, in which case it is removed from the routing problem. Constraint (5) concerns the flow balance, guaranteeing that the vehicle accesses a node and then must leave from the same node.

2) PICK-UP AND DELIVERY CONSTRAINTS

Constraint (6) is a specific pairing constraint in the PDP, guaranteeing that the pick-up point and delivery point in a given request must be dispatched by the same vehicle; constraint (12) is a specific priority constraint, guaranteeing that the pick-up point of a request is serviced before the delivery point.

3) TIME WINDOW CONSTRAINTS

These constraints relate to the quality of the vehicle routing and scheduling service. The vehicle operating time must satisfy the customer's time window requirements, as specified in constraints (10) and (11). In dynamic PDPs, real-time vehicle congestion information can lead to changes in vehicle routing and scheduling. Here, vehicle speed is described in constraint (22).

4) CAPACITY CONSTRAINTS

Constraints (7), (8) and (9) ensure that the loads of the vehicles do not exceed their maximum capacities at any time. Within dynamic problems, the load might be reassigned, unlike in the static problem.

5) SCHEDULING TIME HORIZON CONSTRAINT

Constraint (13) guarantees that the service time of a vehicle does not exceed the maximum length of the scheduling time horizon of the dynamic problem.

6) DISTANCE CONSTRAINTS

Constraint (14) guarantees that the travel distance after vehicle k has served node v_j does not exceed the maximum driving distance of each vehicle. Constraint (15) guarantees that the travel distance after vehicle k has served node v_j is equal to the cumulative travel distance of vehicle k after departing from the depot at time t. Constraints (16) and (17) guarantee that the vehicle is returned within the maximum travel distance. Within the dynamic problem, a vehicle that has performed a request has already traveled a certain distance, but re-planning may then occur so that it is used again, which is different from the static problem.

III. DYNAMIC ALGORITHM FRAMEWORK

A dynamic algorithm or on-line algorithm is essentially an algorithmic framework in a dynamic environment [12]. In this paper, the dynamic algorithm framework is as shown in Fig. 1. The problem is segmented into many static sub-problems to construct an improved initial solution for the static sub-problem using a constructive algorithm and improved algorithm such as the TS algorithm and the ALNS algorithm.

In the dynamic algorithm, the scheduling time horizon is segmented into many subintervals according to a given interval length, transferring the dynamic problem into a series of static problems. Each subinterval is solved in two stages. In the first stage, the initial solution to the scheduling time horizon subinterval problem is obtained using the construction algorithm, described in detail in Section 3.1. In the second stage, an improving algorithm is used to improve the quality of the initial solution obtained in the first stage, which is described in Section 3.2. Besides constructing and improving on the initial solution, real-time information and its impact on customer requests and vehicle resources is processed using the dynamic insertion method, described in Section 3.3.

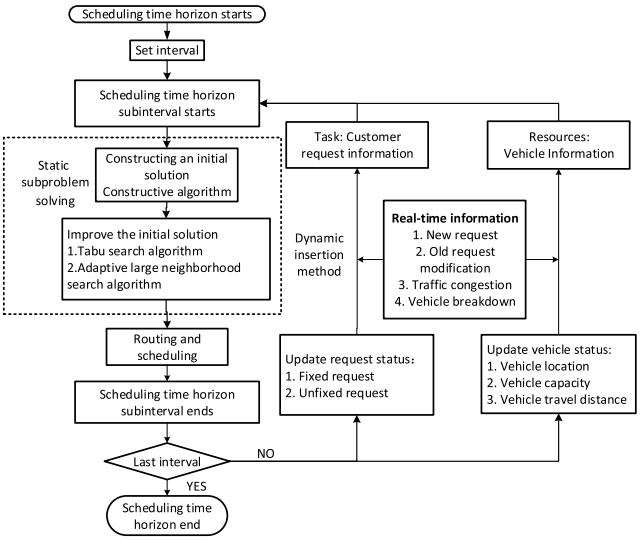


FIGURE 1. Heuristic Dynamic algorithm with rolling time horizon.

A. CONSTRUCTING THE INITIAL SOLUTION

Coming from the classic vehicle routing problem, the dynamic pick-up problem based on real-time information is an NP-hard problem with many constraints, as shown in Section 2.2, which make the problem more complicated. An initial solution is constructed in the first stage. This is improved on in the second stage, using the TS and ALNS algorithms.

In the constructing algorithm, some constraints are applied to improve the initial solution. First, the same vehicle serves two nodes associated by a request and the pick-up point is served before the delivery point. Second, the pick-up point and the corresponding delivery point must be on the same path. Lastly, capacity and time window constraints are applied. The algorithm starts with an empty path, and randomly inserts an unscheduled request. For a request, all available pick-up and delivery point insertion locations of dynamic PDPs are checked in the existing path. All feasible positions are checked and the one with the lowest fitness function value is selected. In this way, all unscheduled requests are inserted into the above path. The fitness function is defined as the objective function (1).

B. IMPROVEMENT OF INITIAL SOLUTION

The initial solution is an NP-hard problem which exists in a huge improvement space. Therefore, we adopt the two following intelligent algorithms to improve the initial solution, trading quality against effectiveness at solving problems.

1) TABU SEARCH (TS) ALGORITHM

The TS algorithm is a kind of simulation of the human thinking process. It accepts some poor solutions by marking

some local optimal solutions as forbidden (''tabu'') to ensure a diversified exploration and global search.

a: NEIGHBORHOOD SHIFT STRATEGY

We adopt the neighborhood shift strategy shown in Fig. 2. A pair of pick-up and delivery points is randomly removed from a path of the initial solution, and then reinserted into this path in the initial solution. After the feasibility of the neighborhood solution has been assessed, those positions with the lowest fitness function value are selected as the adaptive solution from within all the available positions.

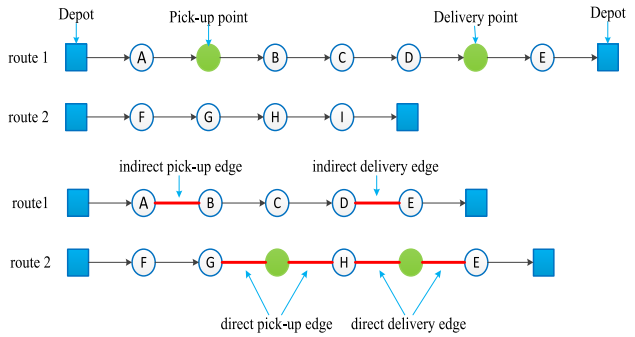


FIGURE 2. Shift operator and tabu object.

b: TABU OBJECT

In Fig. 2, the transformed edges are tabu objects. Edge AB is the indirect pick-up edge, Edge DE is the indirect delivery edge, Edge GH is the direct pick-up edge, and Edge HE is the direct delivery edge. All neighborhood shifts that cause these edges to change are ''tabu''.

c: TABU TABLE LENGTH AND STOPPING CRITERIA

The tabu table length and stopping criteria are proportional while meeting the number of requests in the static sub-problem which needs to be rescheduled. The number of requests that need to be planned in a static sub-problem is dynamic during the solution process, and therefore need to be determined in real-time before each running of the TS algorithm.

d: SELECTION STRATEGY

The candidate solution set is taken as the whole neighborhood, and a solution with the lowest fitness function value is selected as the initial solution of the next iteration.

e: ASPIRATION LEVEL CRITERIA

When the value of the fitness function with a solution in the neighborhood is the smallest and the value of the fitness function is smaller than the initial solution of the iteration, the solution is accepted regardless of whether the movement is ''tabu''.

2) ALNS ALGORITHM

The main rule of the ALNS algorithm is to expand the search range based on the destruction operator and the repair operator. By assigning a weight to both the destruction and the repair operator and adjusting the weights according to the performance of the operators, the usage frequencies of each operator during the search can be controlled. This adaptive adjustment enables better neighborhoods to be found more quickly.

a: DYNAMIC SELECTION OPERATOR

In ALNS, a combination of a destruction and a repair operator is selected based on the previous performance of those operators in generating neighborhoods. In order to expand the search range of the neighborhood, the poorly performing operator will continue to generate the neighborhood with a very low probability. In this paper, the probability of selecting each operator is generated by roulette. Suppose there are n operators, and the weight of each operator is ω_i, reflecting the previous performance of this operator. The probability that each operator is selected in the current solution is defined as $\omega_j / \sum_{i=1}^{n} \omega_i$.

At the beginning of the iteration, each operator has a weight of 1, and each iteration accepts an update of each operator weight based on the operator score. The operator score has a weight of 0 at the beginning of the iteration. After the operator produces a feasible solution, the score of the operator is updated according to Table 1. In practice, the increment should be set to $Q_1 > Q_2 > Q_3 > Q_4$. The score of Q_1 is the biggest because a new solution was found. With Q_2, a new solution is found, and the initial solution is improved upon, which indicates that a new solution space is found. Q_4 is based on the acceptance criteria of the simulated annealing scheme, and the search space is also expanded.

TABLE 1. Adaptive adjustment of operator scores.

Increment	Conditions on solution obtained by the operators
Q_1	New solution and improved optimal solution
Q_2	New solution and improved initial solution
Q_3	Not a new solution but improves the initial solution
Q_4	Does not improve the initial solution, but is accepted by the simulated annealing scheme

In order to update the weight, let α_i be the quotation time of operator i, β_i be the result score of operator i, and $\eta \in [0, 1]$ the speed at which the weight ω_i reacts to the operator. The weight is adjusted by the following formula:

$$\omega_{i,seg+1} = \begin{cases} \omega_{i,seg} & \text{when } \beta_{i,seg} = 0, \\ (1-\eta)\omega_{i,seg} + \eta\beta_{i,seg}/\alpha_{i,seg} & \text{else} \end{cases} \tag{18}$$

When $\eta = 1$, the previous performance of the operator is completely ignored when adjusting the weight, and only the score of the operator obtained last time is taken into consideration. When $\eta = 0$, the value of the previous time is completely ignored, and the new weight is only related to the previous performance of the operator.

b: DESTROYING OPERATORS

i) WORST REMOVAL

Firstly, the reduction in the cost is calculated when each request is removed, and then a randomly selected r% of requests are removed. The probability of being selected here increases as the amount of cost reduction increases.

ii) RANDOM REMOVAL

The requests in the randomly selected $r\%$ are removed from the current solution.

iii) RELATED REMOVAL

Related requests are removed. Two requests [5] are considered related based on the distance between their pick-up and delivery points, the difference between their service times, and the difference in demand. To solve this problem, we modify the relatedness measure of Pisinger and Ropke [6] $R(i, j)$:

$$R(i,j) = \varphi(d_{P_i,P_j} + d_{D_iD_j}) + \chi(\left|T_{P_i} - T_{P_j}\right| + \left|T_{D_i} - T_{D_j}\right|) + \psi(\left|q_i - q_j\right|) \quad (19)$$

where P_i represents the pick-up point of request i, D_i represents the delivery point of request i, T_{P_i} represents the time at which the pick-up point of request i is accessed, q_i represents the demand quantity of the pick-up point of request i, and the range of values of φ, χ and ψ is [0,1].

c: REPAIR OPERATORS

Two insertion methods are adopted: the best insertion method and the regret heuristics method:

i) BEST INSERTION

The best insertion cost for each removal request is calculated at each iteration, and the request with the lowest insertion cost is placed at the last insertion location until all requests are inserted into the route.

ii) REGRET HEURISTICS

Regret heuristics is a kind of algorithm based on regret. Δf_i^j denotes the insertion cost of an unplanned request i at the best position in the j-th best route. At each iteration, the request which meets $i^* = \arg\max_{i \in U}(\sum_{j=2}^{k} \Delta f_i^j - \Delta f_i^1)$ is selected to be inserted into the best position until all requests have been inserted.

d: ACCEPTANCE AND STOPPING CRITERIA

If the fitness function value of the new solution is smaller than the current solution, the solution is accepted. When the fitness function value of the new solution is larger than the current solution, the new solution is accepted according to the simulated annealing criterion. At each iteration, the simulated annealing scheme accepts inferior solutions with a probability that decreases as the temperature T decreases. After each iteration, the temperature T is multiplied by θ to lower the temperature, where $\theta \in [0, 1]$. Actually, θ should be close to 1 to slow down the speed of cooling. The simulated annealing (SA) scheme thereby enables us to expand the search space of the neighborhood and avoid falling into a local optimal solution.

e: ADAPTIVE ASPECT

The destruction and repair operators are both used to generate neighborhoods i which are related to weights W_i and scores ϕ_i. The roulette method is used to select the neighborhood. In the first iteration, all neighbors have the same weight. If a neighborhood is referenced once, the score is updated once based on the effect of the neighborhood. If an optimal solution is generated in the neighborhood, the score of the neighborhood is increased by σ_1; if the generated solution is better than the current solution, the score is increased by σ_2; if a new solution is accepted as the optimal current solution, the score is increased by σ_3.

The whole search process is segmented into multiple time intervals. 100 time intervals are considered in this paper. At the end of each time interval, the weight of the neighborhood is updated according to the expression $w_i \leftarrow (1-\alpha)w_i + \alpha\pi_i/\theta_i$, where θ_i is the number of times the neighborhood was referenced in the previous interval, and $\alpha \in [0, 1]$ is the factor that controls the weights σ_1, σ_2 and σ_3, which are the same as Pisinger and Ropke's [20] α of 0.5.

The parameter scores of the ALNS algorithm in this case are summarized in Table 1.

C. DYNAMIC INSERTION METHOD

At this point, we already have completed paths, new requests, and unvisited requests from the previous interval between two adjacent intervals. Re-planning the route is carried out by the dynamic insertion method, exactly as in the unfixed request method, and is shown in Fig. 3.

At time t, if a pair consisting of a pick-up and a delivery point is not accessed, then the request corresponding to this pair of points is not fulfilled. Otherwise, the request is a fulfilled request. For example, if the pick-up point of a request has been visited, but the delivery point has not, then at this time the request is considered a fixed request. Alternatively, if both the pick-up and the delivery point of a request have been visited, then that request is deemed fixed.

The unfixed request insertion method proposed here includes two steps. In the first step, all unfixed requests are removed from the original path between intervals. In the second step, these unfixed requests and any new requests are reinserted into the route using the constructing algorithm from Section 3.2. An example of reinserting a dynamic new request based on the unfixed request insertion method is

TABLE 2. Parameters of ALNS algorithm.

Symbols	Definition	Scores
$nsegs$	Number of subdivisions	Number of requests
$niters$	Number of iterations in each segment	Number of requests
Q_1	New solution and improved score for optimal solution	0.5
Q_2	New solution and improved score for initial solution	0.4
Q_3	Not a new solution but improved the score of the original solution	0.3
Q_4	No improvement in the initial solution, but score accepted by SA	0.2
η	Score response factor	0.8
δ	Maximum number of iterations without improved solution	100
ρ	Remove request factor	0.2
T_beg	SA initial temperature	0.3
θ	SA cooling factor	0.99

shown in Fig. 3. The blue dashed line here represents the completed route, and the black solid line represents the planned but not yet traveled path. ''Depot'' represents the car yard, the nodes labeled with a number followed by ''p'' represent pick-up points, and the nodes labeled with a number followed by ''d'' represent delivery points. Corresponding numbers represent a pick-up point and a delivery point from the same request.

In Fig. 3, request 3 in the original path of Fig. 3(a) is removed, while requests 1 and 2 remain unchanged as fixed requests. The route before insertion is shown in Fig. 3(b). The new request 4 and the unfixed request 3 from the original path are both reinserted into the path according to the constructing algorithm from Section 2.1. The resulting route after insertion is shown in Fig. 3(c).

IV. ANALYSIS OF ALGORITHMS

A. EXPERIMENTAL DATA

First, we obtained the static DPDP-TW data from Li and Lim [7] (see https://www.sintef.no/projectweb/top/pdptw/li-lim-benchmark/), and we turned the static data into dynamic data by incorporating real-time information, including increased occurrences of requests, a vehicle speed matrix, and vehicle breakdown events. All the data were programmed using MATLAB 9.5. Our experimental environment was Intel(R) Core (TM) i7-4790 CPU 3.60GHz.

1) RELEASE TIME OF REQUESTS

Compared to the normal static problem, each request in the dynamic problem has a release time. According to the definition of a release time in Holborn [8] and Pureza and Laporte [3], we calculate the release time of a request as follows:

$$t_r^{latest} = \min\left\{l_i, l_j - t_{ij} - s_i\right\} - t_{0i} - \beta \tag{20}$$

where t_r^{latest} is the latest arrival time, l_i, l_j are the lower bounds of the latest service time window at v_i and v_j, s_i and s_j are the service times at v_i and v_j, t_{ij} is the travel time between the pick-up location and the delivery location, v_i and v_j are the pick-up and delivery locations of request r, v_0 is the depot, β is the reaction time after the request is received, which we set equal to 10 in the subsequent analysis, and each request has a time stamp of $t_r = \alpha \times t_r^{latest}$ where α is the degree of urgency, $\alpha \in [0.1, 1]$.

2) VEHICLE SPEED IN REAL-TIME TRAFFIC CONDITIONS

When solving dynamic problems, we divide the scheduling time horizon into many intervals to support routing and scheduling. The speed of each vehicle is obtained from the Intelligent Transportation System (ITS) in terms of the exact time period and exact location. The shorter the interval is, the more accurate the vehicle speed will be and the more accurate the planning scheme. Vehicle speed is expressed as follows:

$$v_{cT} = a_{cT} \times v \tag{21}$$

where v is the normal speed in static traffic conditions, and a_{cT} is a speed adjustment factor related to the location of the vehicle and the time period, with $a_{cT} \in (0, 2)$.

3) VEHICLE BREAKDOWN

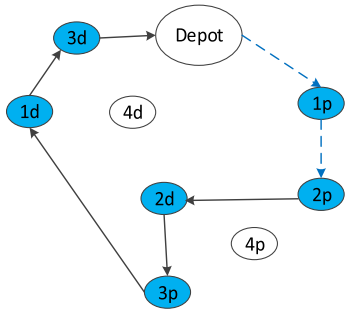
We assume that only one vehicle breaks down during the dispatching time horizon. Since it is very rare to have multiple vehicle breakdowns in one day, such an assumption is realistic and reasonable.

B. NUMERICAL ANALYSIS AND COMPARISON

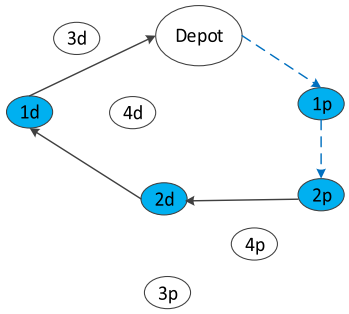
We take the improvement degree of solution I as the evaluation criterion to compare the TS algorithm with the ALNS algorithm. We calculate it as follows:

$$I = \frac{f(s_{constructed}) - f(s_{improved})}{f(s_{constructed})} \times 100\% \tag{22}$$

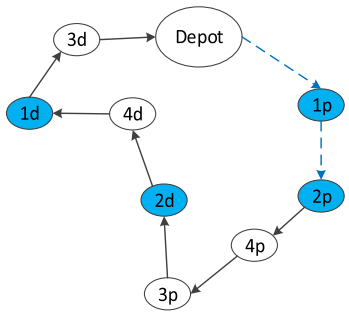
where $f(s_{constructed})$ refers to the objective function value based on the initial constructed solution, and $f(s_{improved})$



(a) Initial rout



(b) Route before insertion



(c) Route after insertion

FIGURE 3. Dynamic insertion procedure.

refers to the objective function value based on the improved solution.

1) COMPARISON OF IMPROVEMENT DEGREES OF SOLUTIONS AT TIME HORIZON *T*

The numbers of new requests at different scheduling time horizons T are shown in Fig. 4. The arrival of new requests is rare after a scheduling time horizon of 900. The improvement degrees of the initial solutions in different time horizons T produced by the TS and ALNS algorithms respectively are shown in Fig. 5. The line with diamonds represents the TS algorithm, and the one with squares the ALNS algorithm.

Comparing Fig. 5 with Fig. 4, the TS algorithm and the ALNS algorithm both improve the initial solution. The line with the diamonds is always lower than the one with the squares up until the scheduling time horizon reaches 900, meaning that the ALNS algorithm improves the initial

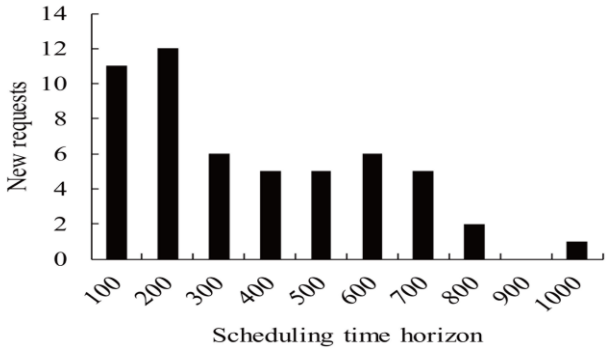


FIGURE 4. New requests under different scheduling time horizons.

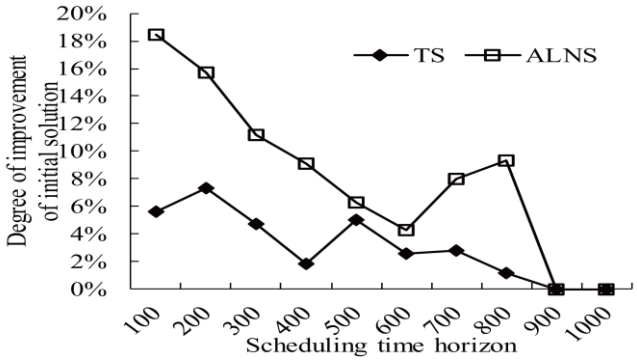


FIGURE 5. Degree of improvement by TS and ALNS under different scheduling time horizons.

solution by more than the TS algorithm. By calculation, we know that the TS algorithm improves the initial solution by an average I of 3.11% across different intervals, while the ALNS algorithm improves the initial solution by an average I of 9.98% across different intervals. Thus, the ALNS algorithm is better than the TS algorithm at improving the initial solution.

2) COMPARISON OF IMPROVEMENT DEGREE OF SOLUTION UNDER DIFFERENT DEGREES OF URGENCY

The degree of improvement of the initial solution by the two algorithms under different levels of urgency $\alpha(\alpha \in [0.1, 0.9])$

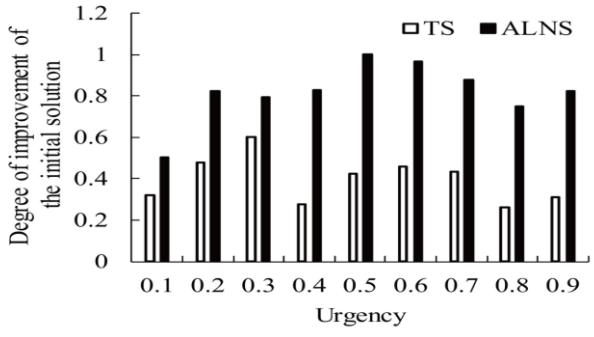


FIGURE 6. Comparison of improvement degree of initial solution under different urgencies.

TABLE 3. Comparison of degrees of improvement and computation times of the two intelligent algorithms.

Request size	Data source	Initial solution	Improved solution		Improvement degree		Computation time(s)	
			TS	ALNS	TS	ALNS	TS	ALNS
50	LC1_0_1	13499.29	12685.16	10765.53	6.03%	20.25%	0.72	8.87
	LR1_0_1	5334.35	4463.41	4405.10	16.33%	17.42%	12.72	66.74
	LRC1_0_1	5690.40	5170.15	4953.49	9.14%	12.95%	23.47	93.83
100	LC1_2_1	28761.62	27062.05	25699.91	5.91%	10.65%	4.01	741.44
	LR1_2_1	17411.13	15640.55	14755.93	10.17%	15.25%	5.74	633.92
	LRC1_2_1	18524.11	16154.23	15645.90	12.79%	15.54%	12.26	1026.40
200	LC1_4_1	17411.13	15640.55	15003.17	10.17%	13.83%	10.16	1218.88
	LR1_4_1	18524.11	16154.23	15856.64	12.79%	14.40%	15.46	4711.36
	LRC1_4_1	43170.27	40406.39	38145.25	6.40%	11.64%	68.78	12203.84
300	LC1_6_1	77561.53	73369.65		5.40%		19.39	
	LR1_6_1	90873.69	83317.97		8.31%		46.48	
	LRC1_6_1	48895.00	43278.83		11.49%		58.03	
400	LC1_8_1	193450.67	186192.5		3.75%		435.15	
	LR1_8_1	118986.36	110324.4		7.28%		33.92	
	LRC1_8_1	94205.02	86117.61		8.58%		122.93	

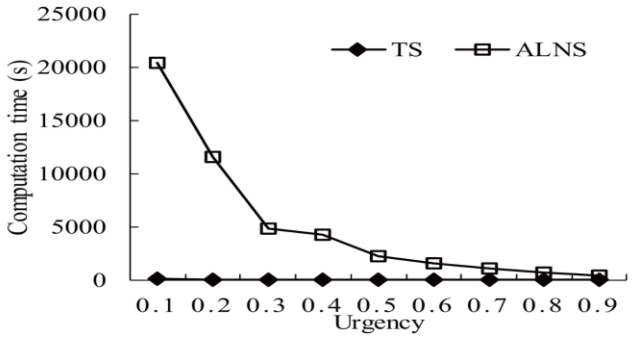


FIGURE 7. Computational time of two algorithms under different urgencies.

is shown in Fig. 6. The black columns are much higher than the white columns regardless of the degree of urgency, which indicates that the ALNS algorithm improves the initial solution by more than the TS algorithm. The computation times of the two algorithms are shown in Fig. 7.

The line with the squares is generally higher than the one with the diamonds regardless of the degree of urgency, meaning that the ALNS algorithm takes more computation time than the TS algorithm. However, when the degree of urgency is higher (such as 0.9), the two algorithms' computation times are fairly close. As the urgency increases, for example, as the urgency increases from 0.5 to 0.9, i.e., the requests become closer to dynamic requests, the two trend lines showing the computation times become closer.

However, when the degree of urgency α is low, the proportion of uncompleted requests is very high. For example, the proportion of uncompleted requests is 86.79% when the urgency is 0.1 and T is at timestamp 200. This means that urgency has a large influence on the computation time of the ALNS algorithm. Especially when the urgency is lower than 0.3, the computation time of the ALNS decreases sharply, with 20465.80s for an urgency of 0.1, 11604.83s for an urgency of 0.2, and 4874.15s for an urgency of 0.3.

In summary, compared to the ALNS algorithm, the quality of the solution of the TS algorithm is not high, but the computation time is very small and relatively stable. Therefore, a large number of previous studies have adopted the TS algorithm. However, the computation time of the ALNS algorithm is similar to that of the TS algorithm when the urgency is high, and the quality of the solution is greatly improved at the same time.

3) COMPARISON OF DIFFERENT REQUEST SIZES

A comparison of the degrees of improvement and computation times of the two intelligent algorithms is shown in Table 3. We use five levels of request size, 50, 100, 200, 300, and 400. A blank in the table means that the computation time of the ALNS was too long, at more than one hour, and therefore the specific time is not listed.

It can be seen from Table 3 that the TS algorithm can obtain optimization results under four request levels, and its computation time is basically less than 60s. The ALNS algorithm takes more than 3600s when the request size reaches 300. This is not suitable for handling dynamic problems. However, when the request size is less than 200, the ALNS algorithm shows a greater degree of improvement of the initial solution than the TS algorithm.

In the case of a request size of 50, the computation time of the ALNS is below 100s. When the request size reaches 100, the computation time of the ALNS reaches 1026.40s,

but when the request size reaches 200, the computation time increases to 12203.84s. This means that the computation time of the ALNS algorithm increases sharply with an increase in request size, which verifies the known fact that the computational complexity of the NP-hard problem increases with an increase in data size.

V. CONCLUSION

This paper redefines the dynamic pick-up and delivery problem considering multiple dynamic events (DPDP-MDE) in a real-world scenario, and attempts to reveal the dynamic influence of multiple dynamic events with the classical dynamic pick-up and delivery problem with time windows (DPDP-TW) model, which can be used to improve real-time dynamic decision making and response times. The model for the DPDP-TW is constructed based on new requests arriving, old requests being canceled or modified, traffic congestion and vehicle breakdown, forming the so-called DPDP-MDE in a real-world environment. The model takes the minimum of the total cost of vehicle scheduling in time horizon T as the objective function, and refines the constraints on vehicle position, time windows, the dynamic time horizon and so on, considering four kinds of dynamic events. It is a useful expansion of the classical DPDP-TW.

The dynamic algorithm framework is improved, and the dynamic insertion method is used to solve the synchronization problem of unfixed and new requests. The TS algorithm and the ALNS algorithm are used to optimize the quality of the initial solution. A numerical experiment shows that the model and its algorithm are effective. Without considering the computation time, the ALNS algorithm improves the initial solution by more than the TS algorithm for any time interval and urgency level. Considering the computation time, the computation time of the TS algorithm is lower and relatively more stable than that of the ALNS algorithm. Therefore, the TS algorithm is a better choice than the ALNS when a particular degree of improvement of the initial solution is not required. Under different request sizes, when the computation time is important, the TS algorithm performs better than the ALNS algorithm; when the degree of improvement of the initial solution is important, the ALNS algorithm performs better than the TS algorithm.

In summary, the proposed model and the dynamic algorithm framework can effectively solve the DPDP-TW problem. Both the TS algorithm and the ALNS algorithm can solve the DPDP-TW based on real-time information. Compared with the ALNS algorithm, the TS algorithm is more efficient, but the solution quality is poorer; meanwhile, the ALNS algorithm has better solution efficiency and better solution quality when there is a higher degree of urgency and a shorter number of intervals.

REFERENCES

[1] F. Ferrucci and S. Bock, "Real-time control of express pickup and delivery processes in a dynamic environment," *Transp. Res. B, Methodol.*, vol. 63, pp. 1–14, May 2014.

[2] A. Fabri and P. Recht, "On dynamic pickup and delivery vehicle routing with several time windows and waiting times," *Transp. Res. B, Methodol.*, vol. 40, no. 4, pp. 335–350, May 2006.

[3] S. Mitrović-Minić and G. Laporte, "Waiting strategies for the dynamic pickup and delivery problem with time windows," *Transp. Res. B, Methodol.*, vol. 38, pp. 635–655, Aug. 2004.

[4] J. Branke, M. Middendorf, G. Noeth, and M. Dessouky, "Waiting strategies for dynamic vehicle routing," *Transp. Sci.*, vol. 39, no. 3, pp. 298–312, Aug. 2005.

[5] S. Mitrović-Minić, R. Krishnamurti, and G. Laporte, "Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows," *Transp. Res. B, Methodol.*, vol. 38, no. 8, pp. 669–685, Sep. 2004.

[6] S. Ichoua, M. Gendreau, and J.-Y. Potvin, "Exploiting knowledge about future demands for real-time vehicle dispatching," *Transp. Sci.*, vol. 40, no. 2, pp. 211–225, May 2006.

[7] B. W. Thomas, "Waiting strategies for anticipating service requests from known customer locations," *Transp. Sci.*, vol. 41, no. 3, pp. 319–331, Aug. 2007.

[8] V. Pureza and G. Laporte, "Waiting and buffering strategies for the dynamic pickup and delivery problem with time windows," *Inf. Syst. Oper. Res.*, vol. 46, no. 3, pp. 165–176, 2008.

[9] A. L. Kok, E. W. Hans, and J. M. J. Schutten, "Vehicle routing under time-dependent travel times: The impact of congestion avoidance," *Comput. Oper. Res.*, vol. 39, no. 5, pp. 910–918, May 2012.

[10] P. Sun, L. P. Veelenturf, M. Hewitt, and T. V. Woensel, "The time-dependent pickup and delivery problem with time windows," *Transp. Res. B, Methodol.*, vol. 116, pp. 1–24, Oct. 2018.

[11] K. Mamasis, I. Minis, and G. Dikas, "Managing vehicle breakdown incidents during urban distribution of a common product," *J. Oper. Res. Soc.*, vol. 64, no. 6, pp. 925–937, 2013.

[12] P. Toth and D. Vigo, *The Vehicle routing problem*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2001.

[13] S. Kritzinger, K. F. Doerner, R. F. Hartl, G. Y. Kiechle, H. Stadler, and S. S. Manohar, "Using traffic information for time-dependent vehicle Routing," *Procedia-Soc. Behav. Sci.*, vol. 39, pp. 217–229, Jun. 2012.

[14] G. Berbeglia, J. Cordeau, and G. Laporte, "Dynamic pickup and delivery problems," *Eur. J. Oper. Res.*, vol. 202, no. 1, pp. 8–15, Apr. 2010.

[15] M. Gendreau, F. Guertin, J. Potvin, and R. Seguin, "Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries," *Transp. Res. C, Emerg. Technol.*, vol. 14, no. 3, pp. 157–174, Jun. 2006.

[16] R. Montemanni, L. M. Gambardella, A. E. Rizzoli, and A. V. Donati, "Ant colony system for a dynamic vehicle routing problem," *J. Combinat. Optim.*, vol. 10, no. 4, pp. 327–343, Dec. 2005.

[17] D. Pisinger and S. Ropke, "A general heuristic for vehicle routing problems," *Comput. Oper. Res.*, vol. 34, no. 8, pp. 2403–2435, Aug. 2007.

BAOFENG SUN was born in 1970. She received the B.S. degree in metal material, the M.S. degree in law, and the Ph.D. degree in management from Jilin University, Changchun, China, in 1992, 1999, and 2002, respectively.

She visited the Transportation and Transportation Center, Technical University of Denmark, as a Visiting Scholar, in 2004. She visited the School of Industrial and Systems Engineering, Georgia Institute of Technology, as a Senior Research Scholar, in 2013. She is currently a Professor with the Transportation College, Jilin University, where she is also the Director of the Modern Logistics Research Institute. She has completed more than 30 research projects totally. She holds four national invention patents. She has published over 30 journal and conference proceedings papers in her research areas. Her research interests include Logistics park planning and logistics system design, logistics system simulation, digital supply chain management, service network design and application.

Dr. Sun serves as a frequent Reviewer for more than ten international journals and Chinese journals.

YUE YANG was born in Changchun, China, in 1988. She received the B.S. degree in electronic information engineering from the Changchun University of Technology and the M.S. degrees in systems engineering from Jilin University, Changchun, China, in 2011 and 2014, respectively, where she is currently pursuing the Ph.D. degree with the Transportation College.

She has published an article at the Communications in Computer and Information Science. Her research interests include route optimization, complex networks, and cascading failure.

JUNYAN SHI was born in Shanxi, China, in 1994. She received the B.S. and M.S. degrees in logistics engineering from Jilin University, Changchun, China, in 2017 and 2019, respectively.

Her research interests include route optimization, dynamic pick-up and delivery problem, and algorithm optimization.

LILI ZHENG received the B.S. and M.S. degrees from Jilin University and the Ph.D. degree from Tongji University.

She visited the University of Minnesota as a Visiting Scholar, from 2014 to 2015. She is currently an Associate Professor with the Transportation College, Jilin University. Her main research interests include design and planning of cooperative vehicle-infrastructure systems, multimode traffic system analysis and coordinate optimization, design and planning of intelligent parking guidance systems. She has granted the second prize of the National Scientific and Technological Progress Award, the first prize of the Jilin Province Scientific and Technological Invention Award, the second prize of the Science and Technology Awards of China Highway and Transportation Society, and the second prize of the Jilin Province Natural Science Scholastic Achievement Award. She has hosted or participated in over ten national level and provincial level research projects. She has published more than ten EI/SCI indexed journal articles and has be authorized seven patents for invention.