

# Open-source VRPLite Package for Vehicle Routing with Pickup and Delivery: A Path Finding Engine for Scheduled Transportation Systems

Xuesong Zhou<sup>1</sup> · Lu Tong<sup>2,3</sup> · Monirehalsadat Mahmoudi<sup>1</sup> · Lijuan Zhuge<sup>4</sup> · Yu Yao<sup>4</sup> · Yongxiang Zhang<sup>5</sup> · Pan Shang<sup>6</sup> · Jiangtao Liu<sup>1</sup> · Tie Shi<sup>5,7</sup>

Received: 2 May 2018 / Revised: 17 May 2018 / Accepted: 23 May 2018  
© The Author(s) 2018

**Abstract** Recently, automation, shared use, and electrification are viewed as the “three revolutions” in the future transportation sector, and the traditional scheduled public transit system will be greatly enhanced with flexible services and autonomous vehicle scheduling capabilities. Many emerging scheduled transportation applications include the fully automatic operation system in urban rail transit, joint line planning, and timetabling for high-speed rail as well as emerging self-driving vehicle dispatching. The vehicle routing problem (VRP) holds promise for seeking an optimal set of vehicle routes and schedules to meet customers’ requirements and plays a vital role in

optimizing services for feature scheduled transportation systems. Due to the difficulty of finding optimal solutions for large-scale instances, enormous research efforts have been dedicated to developing efficient algorithms, while our paper presents a unique perspective based on a time-dependent and state-dependent path searching framework. An open-source and light-weight VRP with pickup and delivery with time windows (VRPPDTW) modeling package, namely VRPLite, has been developed in this research to provide a high-quality and computationally efficient solution engine for transportation on demand applications. This paper describes the space–time–state

---

✉ Lu Tong  
ltong@buaa.edu.cn

Xuesong Zhou  
xzhou74@asu.edu

Monirehalsadat Mahmoudi  
mmahmoudi@asu.edu

Lijuan Zhuge  
16120948@bjtu.edu.cn

Yu Yao  
yaoyu1@bjtu.edu.cn

Yongxiang Zhang  
bk20100249@my.swjtu.edu.cn

Pan Shang  
Shangp16@mails.tsinghua.edu.cn

Jiangtao Liu  
jliu215@asu.edu

Tie Shi  
tshi2005@my.swjtu.edu.cn

- <sup>1</sup> School of Sustainable Engineering and the Built Environment, Arizona State University, Tempe, AZ 85281, USA
- <sup>2</sup> School of Electronic and Information Engineering, Beihang University, Beijing 100091, China
- <sup>3</sup> National Engineering Laboratory for Comprehensive Transportation Big Data Application Technology, Beijing 100091, China
- <sup>4</sup> School of Traffic and Transportation, Beijing Jiaotong University, Beijing 100044, China
- <sup>5</sup> School of Transportation and Logistics, Southwest Jiaotong University, Chengdu 610031, Sichuan, China
- <sup>6</sup> Department of Civil Engineering, Tsinghua University, Beijing 100084, China
- <sup>7</sup> National United Engineering Laboratory of Integrated and Intelligent Transportation, Southwest Jiaotong University, Chengdu 610031, Sichuan, China

---

Communicated by Jing Teng.

modeling process of VRPPDTW using a hyper-network representation. This solution framework can be embedded in a column generation or Lagrangian relaxation framework to handle many general applications. A number of illustrated examples are presented to demonstrate the effectiveness of the path search algorithm under various traffic conditions and passenger travel requirements.

**Keywords** Vehicle routing problem with pickup and delivery · Space–time–state network modeling · Column generation · Lagrangian relaxation

## 1 Introduction

As population and personal travel activities continue to increase, traffic congestion has remained as one of the major concerns for transportation system agencies with tight resource constraints. The next generation of transportation scheduling initiatives aims to integrate various demand management strategies and traffic control measures to actively achieve mobility, environment and sustainability goals. Various approaches hold promises of reducing the undesirable effects of traffic congestion due to driving-alone trips. In this research, we mainly focus on providing a time-dependent and state-dependent path searching engine to serve the demand-responsive ride-sourcing/urban transit services in next-generation transportation on demand applications.

In general, there are two classes of the vehicle routing problem (VRP): (1) designing line haul services for customers from the depot and back haul services for customers to the depot, and (2) transporting passengers or goods between specific origins and destinations with possible requested time windows. VRPLite can cover the above two types of problems, but our discussion below focuses on the second class without loss of generality. There are a number of excellent reviews on vehicle routing problems with pickup and delivery by Cordeau et al. [1], Parragh et al. [2] and Psaraftis et al. [3]. When each transporting request is defined by determinate pickup and delivery points, the VRP becomes the vehicle routing problem with pickup and delivery (VRPPD). Practical applications of the VRPPD can be commonly found in urban rail transit management, to name a few, rail transit line planning [4, 5], policy decision making [6, 7], train operation management [8, 9], train timetabling [10, 11], and metro-based freight transportation [12]. In the emerging peer-to-peer ride-sharing service, a passenger can ask the driver to take him/her directly to the destination, and the passenger may also share this ride with one or more passengers. The ride-sharing problem can be mathematically modeled by the classic vehicle routing problem with pickup and delivery

with time windows (VRPPDTW) [13]. Previous research has made a number of important contributions along different formulations or solution approaches. On the other hand, there are a number of modeling and algorithmic challenges for a large-scale deployment of vehicle routing and scheduling algorithms, especially for regional networks with various road capacity and traffic delay constraints on freeway bottlenecks and signal timing on urban streets.

In the field of operations research, a few previous studies directly consider the underlying transportation network with time of day traffic congestion, while the majority of studies define the VRPPDTW on a directed graph with fixed shortest travel distance or least travel time routes between origin–destination pairs. Due to the complexity of variables and constraints, it is difficult to seek optimal solutions for large-scale VRPs. The generalized VRP problem also has a number of rich applications on the urban transit and other related scheduling or logistics problems [14, 15]. For example, the most featured one is the transit vehicle assignment and routing problem, which is usually studied as the generalized VRP problem. Other related problems include crew scheduling, aircraft fleet and routing, fleet itinerary scheduling as well as network-wide train timetabling which assigns trains to different routes.

In this paper, we introduce our proposed algorithm from the perspective of shortest path algorithms, which has rich applications in the field of transportation network modeling [16]. The physical network can be denoted by  $(N, A)$ , where  $N$  is the set of nodes and  $A$  is the set of arcs, and then the shortest path problem is to find a path with the minimal cost that consists of a sequence of links  $(i, j) \in A$  connecting origin node  $o$  to destination node  $d$ . The shortest path problem can be represented as the minimal cost network flow problem by sending one unit of flow from  $o$  to  $d$ . A binary variable  $x_{ij}$  is used to denote whether the flow passes the link  $(i, j)$  or not. The mathematical formulation of the shortest path problem with node flow balance constraints is listed below [17].

$$\min z = \sum_{(i,j) \in A} c_{ij} x_{ij} \quad (1)$$

s.t.

$$\sum_{\{j:(i,j) \in A\}} x_{ij} - \sum_{\{j:(j,i) \in A\}} x_{ji} = \begin{cases} -1 & \forall i = o \\ 1 & \forall i = d \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A \quad (3)$$

In general, there are mainly two algorithmic approaches to deal with the shortest path problem, namely label setting and label correcting algorithm [17]. Both approaches

iteratively update the label cost of nodes at each step. The label cost of nodes is designated as permanent in the label setting algorithm at each iteration, while the label correcting algorithm can change the label cost of nodes at any step. This feature determines that label setting algorithm cannot solve the shortest path problem with negative arc costs, but the label correcting algorithm can deal with the shortest path problem with arbitrary arc costs. The static shortest path problem can be further extended to dynamic version with time-dependent arc costs  $c_{ij,t}$  on link  $(i, j)$  for vehicles leaving at time  $t$ , and time-dependent link travel time  $TT_{ij,t}$ . Ziliaskopoulos and Mahmassani [18] proposed an efficient label correcting-based algorithm to find the time-dependent shortest paths from all nodes in the network to the single destination node. Chabini [19] aims to reduce the worst-case complexity in solving the time-dependent shortest path problem by adopting the dynamic programming approach. Recently, the shortest path problem in three-dimensional networks or hyper-networks has attracted significant interests in transportation optimization problems. Mahmoudi and Zhou [20] developed a time-dependent forward dynamic programming algorithm to reformulate the VRPPD as a special version of the shortest path problem with time-dependent and state-dependent arc costs where the state represents the number of passengers on the vehicle. Liu et al. [21] further considered an extended version of household activity scheduling problem

by using a cumulative space–time–state representation to reduce the number of complex constraints. A space–time–speed hyper-network and the corresponding dynamic programming algorithm are introduced by Zhou et al. [22] with applications in joint optimization of train timetables and speed profiles. A resource-space–time network was adapted by Lu et al. [23] to consider time-dependent routing applications with energy resource considerations. As an alternative representation to the high-dimension network, a decomposition approach is considered by Tong et al. [14] and Ruan et al. [24] with two coupled sub-problems, i.e., a space–time network-based shortest path problem and a general assignment problem, for customized bus applications. By extending the space–time network-based time-dependent shortest path algorithms, this VRPLite package aims to solve the time-dependent and state-dependent shortest path problem. As a particular note, the cumulative service state of passengers is presented as the “state” dimension in the VRPLite package.

Table 1 provides a detailed comparison of the above shortest path algorithms from the perspectives of network representation, arc cost, data structure to the solution methods. In Table 2, we further list different state representations in the space–time–state modeling framework to show the potential of this modeling method in solving difficult transportation problems.

The fully functional, open-source VRPLite package can be downloaded from <https://github.com/xzhou99/VRPLite>.

**Table 1** Comparison of different shortest path algorithms (network representation, arc cost, data structure, and solution methods)

Dimension	Vertex	Arc	States	Cost	Network building	Method
Space based [17]	Physical nodes	Physical arcs	–	Negative or positive	Scan eligible list	Label correcting
	Physical nodes	Physical arcs	–	Positive	Heap for find the minimal temporal labels	Label setting
Space-time, time-dependent [18]	Physical nodes	Physical arcs, time-dependent arc cost	–	Negative or positive	Without explicitly building the space–time network	Label correcting
Space–time, time-dependent [19]	Physical nodes	Physical arcs, time-dependent arc cost	–	Negative or positive	Without explicitly building the space–time network	Dynamic Programming
Space–time–state, time and state dependent [20]	Space–time–state nodes	Space–time–state links	Seats, number of passengers	Negative or positive	Without explicitly building the space–time–state network	Dynamic Programming, but need to enumerate the vehicle carrying states
Space–time–state, time and state dependent (this research)	Space–time–state nodes	Space–time–state links	0, 1, 2, for cumulative service states of passengers	Negative or positive	Without explicitly building the space–time–state network	Dynamic Programming, dynamically generate the passenger cumulative service states

**Table 2** Typical different state representations

State	Applications	Related papers
Vehicle carrying	VRPPDTW	Mahmoudi and Zhou [20]
Passenger cumulative service	VRPPDTW	Mahmoudi et al. [25]
Resource, e.g., energy or emissions	Green VRP	Lu et al. [23]
Activity performing states	Household activity pattern problem	Liu et al. [21]
High-speed train speed	Train scheduling	Zhou et al. [22]
Cumulative working hours in crew scheduling, cumulative running distance in electric multiple unit (EMU) maintenance scheduling	Crew scheduling, EMU maintenance scheduling	Chen et al. [26]

The goal of this package includes the following three aspects.

1. Provide an open-source code base that enables transportation researchers and software developers to expand its range of capabilities to various traffic/transportation management applications.
2. Provide a free, educational modeling framework for students and researchers to understand the complex space–time–state network modeling process for transportation optimization applications, e.g., train time-tabling, urban rail transit scheduling and ride-sharing applications.
3. We introduce our solution approach and related software implementation from a time-dependent and state-dependent shortest path approach.

## 2 Space–Time–State-Based Models

### 2.1 Problem Statement

The VRPPDTW problem studied in this paper can be formally defined by the following statement. Consider a physical transportation network  $(N, M)$  with a finite set of nodes  $N$  and a finite set of links  $M$ , where nodes  $i, j \in N$  and directed link  $(i, j) \in M$ . A space–time network  $G = (E, A)$  can be constructed for transportation network  $(N, M)$  under planning time horizon  $T$  considering passenger carrying state  $W$  with a finite set of space–time–state vertices  $V$  and a finite set of space–time–state arcs  $A$  according to optimization requirements. In a space–time–state setting, each vertex  $(i, t, w) \in V$  simultaneously represents time, location, and vehicle carrying state; each arc  $(i, j, t, s, w, w') \in A$  indicates a directed space–time–state path from node  $i$  departing at time  $t$  with passenger carrying state  $w$  to node  $j$  arriving at time  $s$  with passenger

carrying state  $w'$ . Because of the three-dimensional network structure, it is easy to model passengers' travel requests, vehicles' travel times changing over time, and available passenger carrying states. Given a set of passengers  $P$  and their travel requests, as pickup/delivery locations,  $o_p/d_p$ , and space–time windows,  $[a_p, b_p]$  and  $[a'_p, b'_p]$ , as well as vehicle capacity constraint  $Cap_v$  and other routing constraints, the VRPPDTW problem aims to find optimal passenger-to-vehicle assignment, vehicle routes and timetables for each vehicle  $v$  in the vehicle set  $V$  under certain traffic conditions. The notations used in this paper are listed in Tables 3 and 4.

### 2.2 Modeling Methodology Based on the Space–Time–State Network Representation

A simple example with two travel requests and one vehicle in our previous study [20] is used to demonstrate key modeling features of space–time–state network representation. Consider a general physical transportation network containing six nodes. Each link in this network is associated with time-dependent travel time  $TT(i, j, t)$ . Two requests are with the same pickup node (node 2) and the same drop-off node (node 3), but with different pickup and drop-off time windows. Only one vehicle is available for serving the two requests. Moreover, we assumed that the vehicle starts its route from node 4 and ends it at node 1. One dummy origin node and one dummy destination node need to be built to correspond to its pickup node and drop-off node for each request. As shown in Fig. 1, Passenger 1 should be picked up from dummy node  $o_1$  in time window [4,7] and dropped off at dummy node  $d_1$  in time window [11,14], while Passenger 2 should be picked up from dummy node  $o_2$  in time window [8,10] and dropped off at dummy node  $d_2$  in time window [13,16]. Vehicle 1 also has

**Table 3** Indexes and variables used to describe the VRPPDTW problem

Symbol	Definition
$V$	Set of physical vehicles
$V^*$	Set of virtual vehicles
$P$	Set of passengers
$N$	Set of physical transportation nodes in the physical traffic network
$M$	Set of physical transportation links in the physical traffic network
$W$	Set of possible passenger carrying states
$v$	Vehicle index
$v_p^*$	Index of virtual vehicle exclusively dedicated for passenger $p$
$p$	Passenger index
$w$	Passenger carrying state index
$(i, j)$	Index of physical link between adjacent nodes $i$ and $j$
$TT(i, j, t)$	Link travel time from node $i$ to node $j$ starting at time $t$
$Cap_v$	Maximum capacity of vehicle $v$
$a_p$	Earliest departure time from passenger $p$ 's origin
$b_p$	Latest departure time from passenger $p$ 's origin
$a'_p$	Earliest arrival time at passenger $p$ 's destination
$b'_p$	Latest arrival time at passenger $p$ 's destination
$[a_p, b_p]$	Departure time window for passenger $p$ 's origin
$[a'_p, b'_p]$	Arrival time window for passenger $p$ 's destination
$o'_v$	Dummy node for vehicle $v$ 's origin
$d'_v$	Dummy node for vehicle $v$ 's destination
$e_v$	Vehicle $v$ 's earliest departure time from the origin depot
$l_v$	Vehicle $v$ 's latest arrival time to the destination depot
$o_p$	Dummy node for passenger $p$ 's origin (pickup node for passenger $p$ )
$d_p$	Dummy node for passenger $p$ 's destination (delivery node for passenger $p$ )

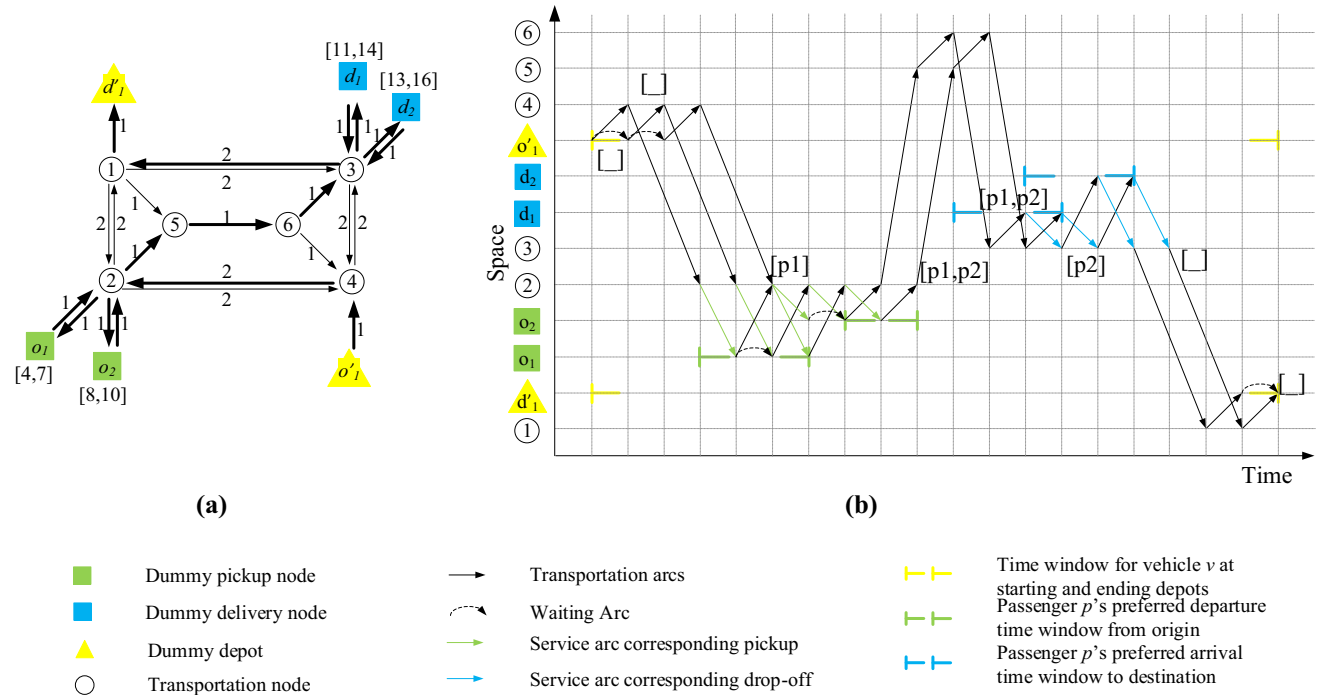
**Table 4** Indexes and variables used for the model formulation

Symbol	Definition
$(i, t, w), (j, s, w')$	Indexes of space–time–state vertexes
$(i, j, t, s, w, w')$	Index of a space–time–state arc indicating that one can travel from node $i$ at time $t$ with passenger carrying state $w$ to the node $j$ at time $s$ with passenger carrying state $w'$
$V$	Set of vertices in the proposed space–time–state network
$V^*$	Set of virtual vehicles
$B_v$	Set of space–time–state arcs in vehicle $v$ 's network
$c(v, i, j, t, s, w, w')$	Routing cost of arc $(i, j, t, s, w, w')$ traveled by vehicle $v$ , including transportation costs, passenger waiting time and vehicle waiting time, converted through the values of time
$TT(v, i, j, t, s, w, w')$	Travel time of arc $(i, j, t, s, w, w')$ traveled by vehicle $v$
$\Psi_{p,v}$	Set of pickup service arcs of passenger $p$ in vehicle $v$ 's networks
$y(v, i, j, t, s, w, w')$	$= 1$ if arc $(i, j, t, s, w, w')$ is used by vehicle $v$ ; $= 0$ otherwise

the earliest departure time from its starting depot,  $t = 1$ , and the latest arrival time at its ending depot,  $t = 20$ .

Note that the shortest path with node sequence  $(o'_1, 4, 2, o_1, 2, o_2, 2, 5, 6, 3, d_1, 3, d_2, 3, 1, d'_1)$  from vehicle 1's origin to its ending depot is shown by bold arrows when

it serves both requests. To use a space–time–state network representation for model formulation, the time horizon is discretized into a set of time intervals with the same time unit, e.g., 1 min. Selected arcs constituting the shortest paths from vehicle 1's origin to its destination are





### 2.3 Understanding Different Optimization Models Within VRPLite Package

The transportation optimization problem for large-scale instances gives rise to challenges requiring innovative concepts and solution techniques. Our VRPLite package tackles with routing and scheduling problems with time dimensions and incorporates theoretical models and practical methods, such as VRP, the assignment problem, the knapsack problem, time-dependent and state-dependent problem, Lagrangian relaxation solution framework, and column generation framework.

#### 1. Vehicle routing problem

The family of VRP problem is a class of linear programming problems with a special structure of two layers. Specifically, when it comes to the freight transportation, each source has a fixed supply of units, which must be distributed to the destinations. Each destination has a fixed demand for units, which must be satisfied by the sources. In dial-a-ride, transportation on demand problem or VRPPDTW problem for passenger transporting, each passenger needs to be transported from his/her origin to destination in a complex transportation network.

#### 2. Assignment problem

The assignment problem needs to match a number of agents to a number of tasks. Any agent can be assigned to perform any task, incurring some costs that may vary depending on the agent-to-task assignment relationship. It is required to perform all tasks by assigning agents to tasks following certain rules in such a way that the total cost of the assignment is minimized. In the VRPPDTW problem, vehicles need to be assigned to serve passengers. By following a variable splitting method introduced by Fisher [27], the complex space–time–state variables can be decomposed into assignment variables for both passengers and vehicles and space–time routing variables for vehicles. The merit of variable dimensionality reduction lies in avoiding the enumeration of vehicle states. Therefore, the assignment problem-based VRP model is superior when dealing with high-capacity transportation modes, such as customized buses, public transit and etc.

#### 3. Knapsack problem

The knapsack problem refers to the common problem of packing the most valuable or useful items subject to the overall knapsack capacity constraints. In the VRPPDTW, we need to decide which passengers should be served by each vehicle because of the limited carrying capacity of vehicles or limited time budgets.

#### 4. Time-dependent and state-dependent shortest path problem

In graph theory, the shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized. Interested readers could get more information on space–time network construction and computationally time-dependent shortest path algorithms in [17, 18]. In the VRPPDTW, we need to compute time-dependent and state-dependent shortest path for each vehicle.

#### 5. Dynamic programming

Several efficient algorithms have been developed to compute time-dependent shortest paths in networks with time-dependent arc costs. In the path searching engine, a time-indexed dynamic programming algorithm is used to solve the shortest path problem.

#### 6. Lagrangian relaxation solution framework

Lagrangian relaxation is a relaxation method which approximates a difficult problem of constrained optimization by a simpler problem. To find the optimal solution for the Lagrangian dual problem, VRPLite computes time-dependent and state-dependent least-cost path for each vehicle based on updated multipliers by calling the proposed time-dependent forward dynamic programming algorithm.

#### 7. Column generation framework

Column generation, to be detailed further, provides an effective way to find solutions for linear programs with numerous variables. The predominant concept is that the primal problem is split into two problems: the master problem and the subproblem. The master problem is the original problem with only a subset of variables being considered. The subproblem is a new problem created to identify a new variable. The VRPPDTW problem can be split into set partitioning problems and time-dependent shortest path problems, which can be solved by standard optimizer and proposed time-dependent dynamic programming algorithm, respectively.

## 3 Software Architecture, Data Flow and Implementation

### 3.1 Data Flow Chart of VRPLite

The software architecture designed in VRPLite aims to integrate passenger-to-vehicle assignment and time-dependent routing into an open-source VRP modeling package. As shown in Fig. 2, there are three major modeling components, including: (1) dynamic programming based on a node-link network structure and vehicle states transition; (2) Lagrangian relaxation module, which utilizes passenger's prices to determine vehicle's pickup and

delivery decision in (1) dynamic programming module; (3) column (i.e., vehicle paths) generation for large-scale problems. The final output includes passenger costs/profits (e.g., total waiting cost, transportation cost), passenger-to-vehicle assignment and vehicle space–time trajectories.

### 3.2 A Dynamic Programming Solution Framework Implemented as a Beam Search Process

Within a dynamic programming (DP) framework, Fig. 3 shows the data flow chart and the solution framework of VRPLite. In general, dynamic programming is suitable for multistep or sequential decision processes with given state transition. In our case, the time horizon serves as the stages in DP, and at current node and current time, (cumulative service) states are defined by the vehicle's path node sequence, path time sequence, and served passenger sequence.

Figure 4 depicts the pseudo-code of the time-indexed beam search algorithm, which is an improved version compared with the three-loop dynamic programming algorithm by Mahmoudi and Zhou [20]. Essentially, a vehicle starts from its depot at departure time and scans three loops with the index of time, the index of  $k$  for the beam search and the index of outgoing nodes from the current node. At each time, all the possible states are evaluated by the objective function and the best  $K$  partial solutions are selected to move forward. It should be noted that the current node is stored in the time-indexed vector  $td\_state[t][k]$  and only the  $k$  best solutions are selected to move forward. The final solution is output as the vehicle reaches its destination within its time window. Figure 5 depicts the process of the time-indexed beam search algorithm.

To handle the demand satisfaction constraint (8) of our model, we introduce Lagrangian relaxation to relax this constraint, with added passenger profits (i.e., dual price) in the new objective function. Iteratively, passenger profits

are updated according to a subgradient method, that is, checking how many times a passenger has been served during the current iteration to increase or decrease the price accordingly. For a large-scale application, we use a multi-vehicle column generation process to better define and search for feasible solutions.

### 3.3 Column Generation Framework for Finding Multi-vehicle Routing Solutions

The VRPPDTW, in its original arc-based form, can also be reformulated as a set partitioning problem by applying Dantzig–Wolfe decomposition. We could introduce a new set of path-based variables  $x(v, p)$ , which equals 1 if passenger  $p$  is served by vehicle  $v$  and equals 0 otherwise. The set partitioning formulation of the VRPPDTW can be expressed as follows.

$$\text{Min } Z = \sum_{p \in P} \sum_{v \in V} c(v) x(v, p) \quad (10)$$

s.t.

$$\sum_{v \in V \cup V^*} x(v, p) = 1 \quad \forall p \in P \quad (11)$$

$$x(v, p) = \{0, 1\} \quad \forall p \in P, v \in V \quad (12)$$

The objective function (10) minimizes the cost of the chosen paths. The constraint (11) guarantees that each passenger is served exactly once. The linear programming (LP) relaxation of (10)–(12) with a subset of feasible paths is called the Restricted Master Problem (RMP), which can be solved by a standard optimizer. We are also able to obtain dual prices for each passenger  $\pi(p)$  from the solution of the RMP. Those dual prices are then utilized in the objective function of the subproblem, which searches for variables with negative reduced cost. The objective function of the underlying pricing subproblem can be formulated as Eq. (13).

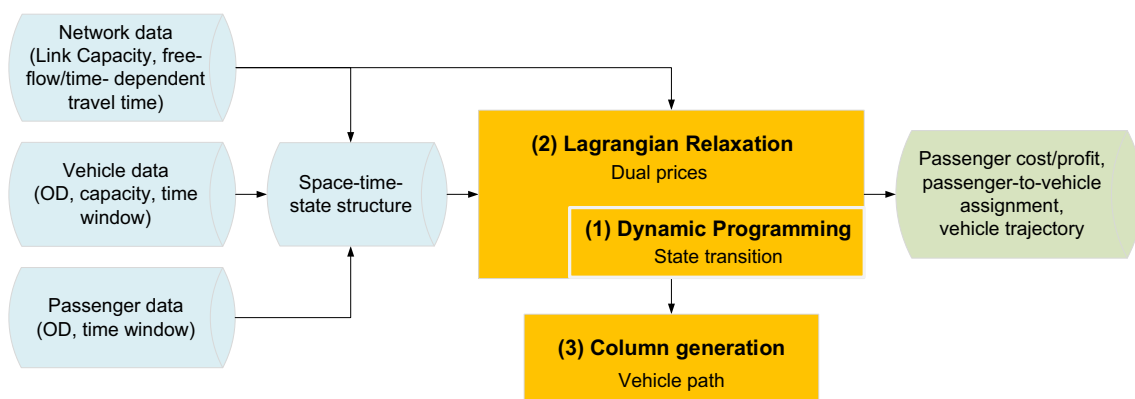


Fig. 2 Software architecture with key modeling components



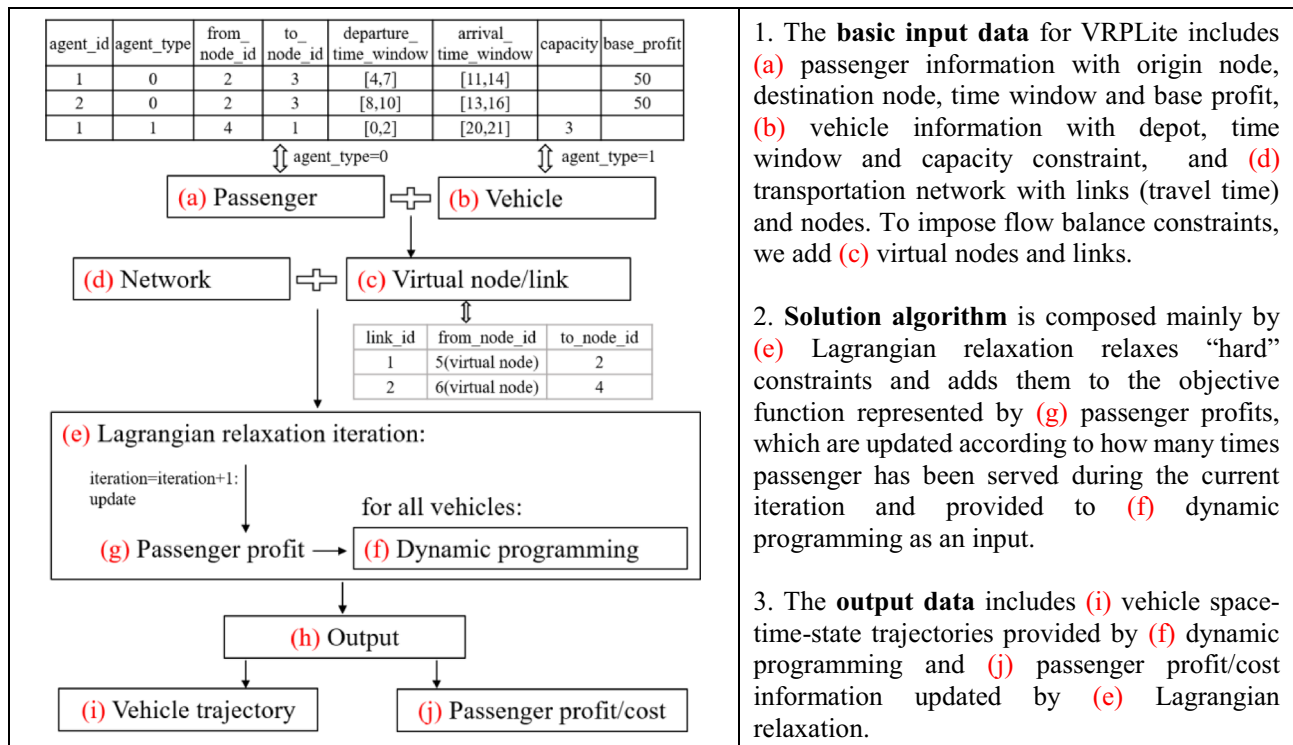


Fig. 3 Data flow chart of VRPLite

$$\text{Min } \sigma = \sum_{i,j,t,s,w,w'} c(v,i,j,t,s,w,w') y(v,i,j,t,s,w,w') - \sum_p \pi(p) x(v,p) \quad (13)$$

The pricing problem is essentially a time-dependent shortest path problem in a network with time-dependent arc costs, while several efficient algorithms have been developed to solve such a problem. Along this line, the time-dependent and state-dependent path searching engine developed based on a dynamic programming framework can be used to compute the subproblem within the general column generation framework.

As shown in Fig. 6, by solving the Restricted Master Problem, the outputting optimal solution with dual prices of passengers being its byproduct results could be served as the input of the pricing subproblem. Iteratively, the pricing subproblem is solved to generate a new path of the vehicle, as a new column to be added to the master problem. Finally, the optimized vehicle routes are obtained. For more details of column generation algorithms, especially about the branch and price framework and its final convergence criteria, interested readers are referred to Lübbecke and Desrosiers [28].

In the column generation algorithm, the RMP is relaxed into a linear programming problem; thus, the optimal solution we obtain might be fractional. However, the variables  $x(v,p)$ , which represent if the vehicle  $v$  serves

passenger  $p$  or not, should be binary variables. Thus we have to design branching strategies to find feasible integer solutions. When the algorithm branches on variables  $x(v,p)$ , it indicates that  $x(v,p)$  is fractional and typically imposes two branches  $x(v,p) = 0$  and  $x(v,p) = 1$ .

## 4 Numerical Experiments

### 4.1 The First Toy Example on a Corridor

A small corridor network consisting of 6 physical nodes and 5 physical links is shown in Figs. 7 and 6 dummy nodes and 10 dummy links are also included. The internal number of physical and dummy nodes is determined with a given labeling rule. First, all of the physical nodes are labeled from 1 to  $N$  where  $N$  is the number of physical nodes. Second, the pickup and delivery nodes of each passenger are numbered sequentially from  $N+1$  to  $N+2P$  where  $P$  represents the total number of passengers. Finally, we mark the number of origin and destination depots of each vehicle in sequence from  $N+2P+1$  to  $N+2P+2M$ . In addition, the departure and arrival time windows are specified directly beside those dummy nodes. In this example, one vehicle will travel from node 1 to node 6 to serve two passengers on its way. The first passenger A departs from node 2 and needs to alight the vehicle at node

Initialize vector  $td\_state[t_0] = [o]$ ,  $end\_state[v] = []$ ,  $KBestSize$ , build virtual node for each passengers' and vehicle's origin-destination node

**Do while**  $t$  in  $(T_0, T)$

**Sort**  $td\_state[time\ t]$  according to the overall cost (transportation cost + profit for serving passengers + passenger and vehicle waiting cost)

**Do while** index  $k < \min(td\_state[t_0].size(), KBestSize)$

current\_node =  $td\_state[t_0][k].current\_node$

**Do while** to\_node in current\_node.outbound\_node\_vector.size()

to\_node\_time =  $t + \text{outbound\_link's travel time when entering link at time } t$

**Case 1:** to\_node is a dummy pick up or drop-off node and the arrival time to\_node\_time is within the service time window. If passenger  $p$  is not in  $td\_state[t_0][k]$ , then  $v$  can pick up passenger  $p$ ; if  $p$  has been in vehicle  $v$ 's carrying state  $td\_state[t_0][k]$ , then  $v$  would drop off passenger  $p$ .

Update the vehicle carrying states in  $td\_state[to\_node\_time]$  with its current\_node = to\_node and new cost with possible service profit.

**Case 2:** to\_node is a physical node.

Update  $td\_state[to\_node\_time]$  with its current\_node = to\_node and updated cost.

**Case 3:** to\_node is vehicle's destination node  $d$ .

Update  $end\_state[v]$  with  $td\_state[to\_node\_time]$  and updated cost.

**End // downstream node**

**End // index k**

**End // time t**

Sort  $end\_state[v]$  according to its cost, choose the first one as our solution.

#### Parameter and variable definitions:

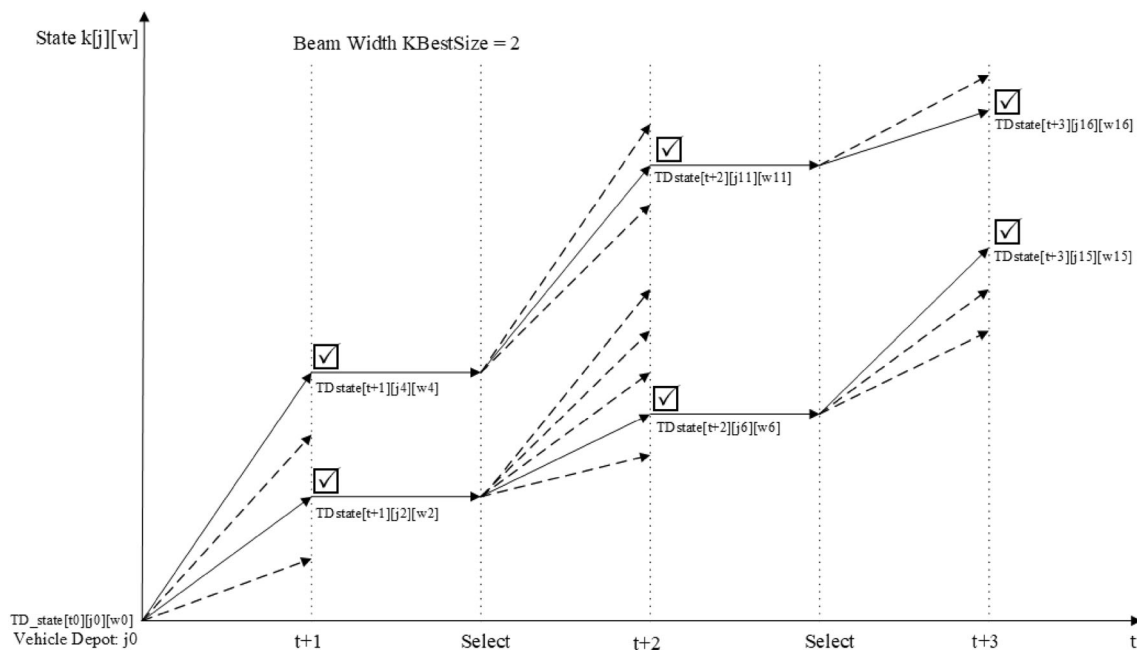
$td\_state$ : vehicle's time-dependent state

$end\_state$ : vehicle's ending state at destination

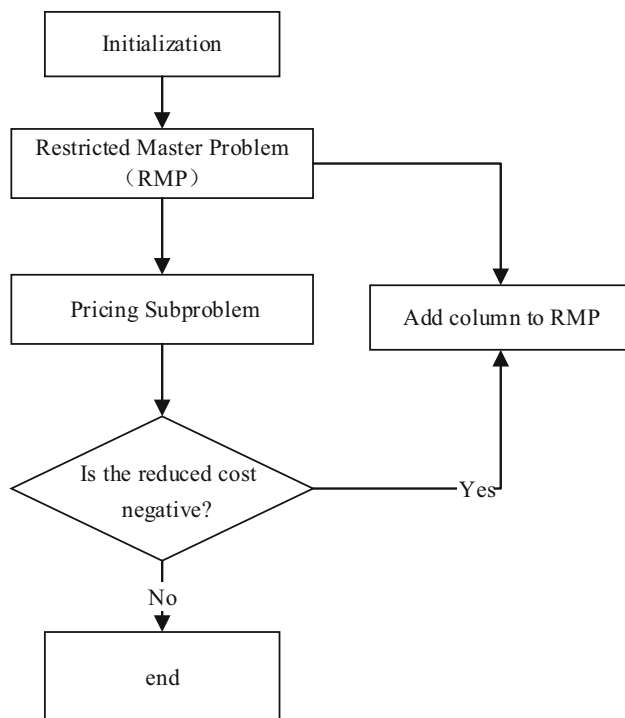
to\_node\_time: vehicle arrival time at the downstream node of the current node.

Index  $k$ : the index for beam search with the  $KBestSize$  as beam width

**Fig. 4** Pseudo-code of time-indexed beam search process for each  $v$  with origin  $o$  departure time  $T_0$ , destination  $d$  arrival time  $T$



**Fig. 5** The process of the time-indexed beam search algorithm, with the time dimension as the horizontal axis



**Fig. 6** Column generation framework for finding multi-vehicle routing solutions

4, while the second passenger B needs to travel from node 3 to node 5. The internal number of nodes shown in Fig. 7 is labeled in Table 5 by following our cost updating rule. In Figs. 7, 9, 11 and 13, texts in circles, rectangles and triangles denote node numbers, and texts on links represent corresponding travel costs.

Both the upper bound and lower bound of the small example are equal to 20, so it is proved that the optimal solution is 20. In addition, the optimal routes and schedules for the single vehicle are shown in Table 6, and the space–time trajectory of the vehicle for the basic example is illustrated in Fig. 8. It is remarked that there is no

**Table 5** Internal number of the nodes shown in the basic example

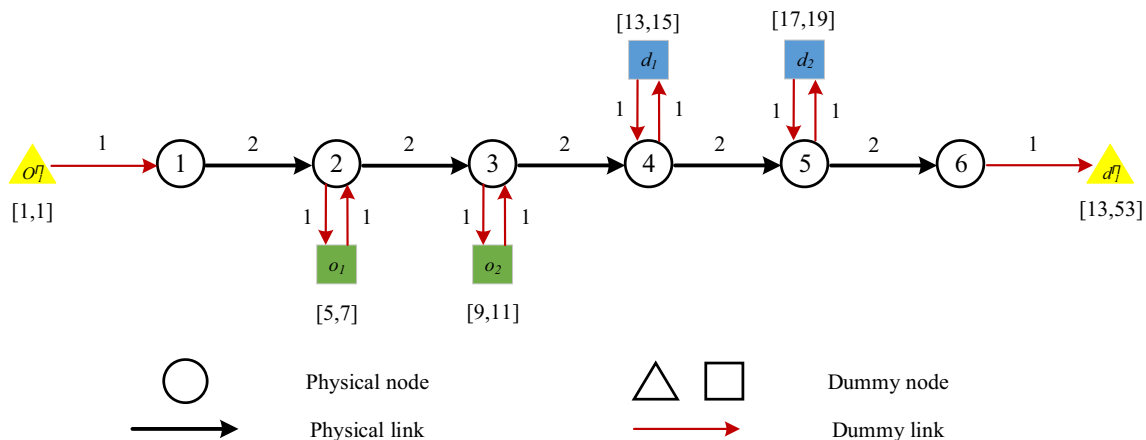
Name	Internal node number	Type
1	1	Physical node
2	2	Physical node
3	3	Physical node
4	4	Physical node
5	5	Physical node
6	6	Physical node
$o_1$	7	Dummy node, passenger A pickup node
$d_1$	8	Dummy node, passenger A delivery node
$o_2$	9	Dummy node, passenger B pickup node
$d_2$	10	Dummy node, passenger B delivery node
$o'_1$	11	Dummy node, vehicle 1 origin depot
$d'_1$	12	Dummy node, vehicle 1 destination depot

passenger waiting time or vehicle waiting time in this example. That is, in the optimal solution the vehicle can arrive at the service points just at the time when the service time windows start.

## 4.2 The Second Example with Extended Link Travel Time

The travel time of links (2, 3) and (4, 5) is extended to 4, as shown in Fig. 9, and the departure and arrival time windows of passengers A and B are updated accordingly, so that no vehicle or passenger waiting time is introduced, as shown in Table 7, and the space–time trajectory of the vehicle is shown in Fig. 10.

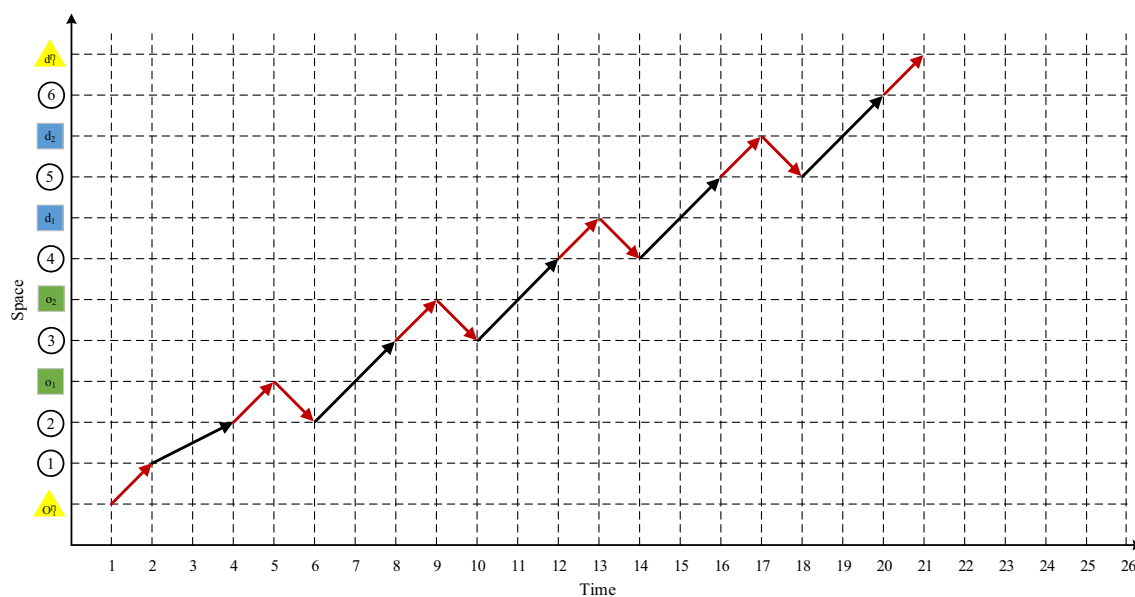
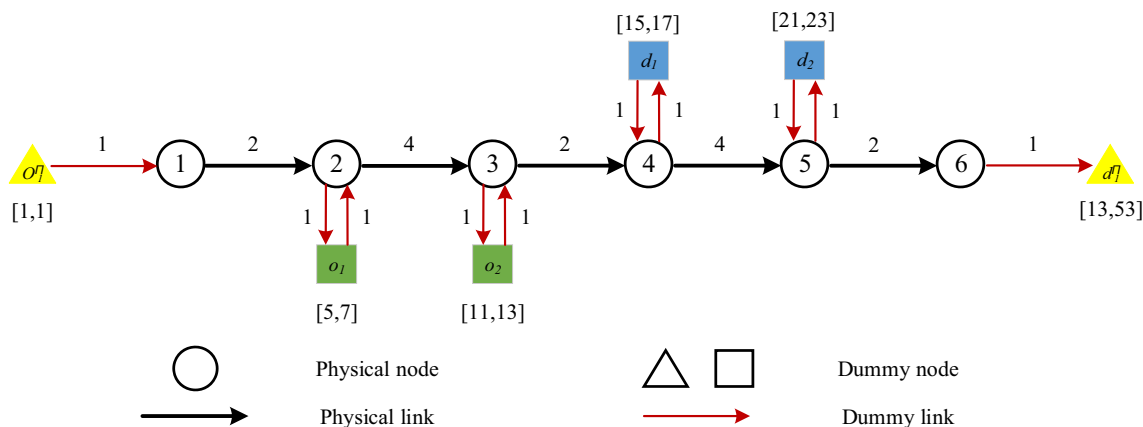
Both upper bound and lower bound of this problem are equal to 24, which increases by 4 compared to the optimal solution of the first example. Obviously, it is for the reason that the travel time of links (2, 3) and (4, 5) increases by 4



**Fig. 7** Network layout and time-window settings for the basic example

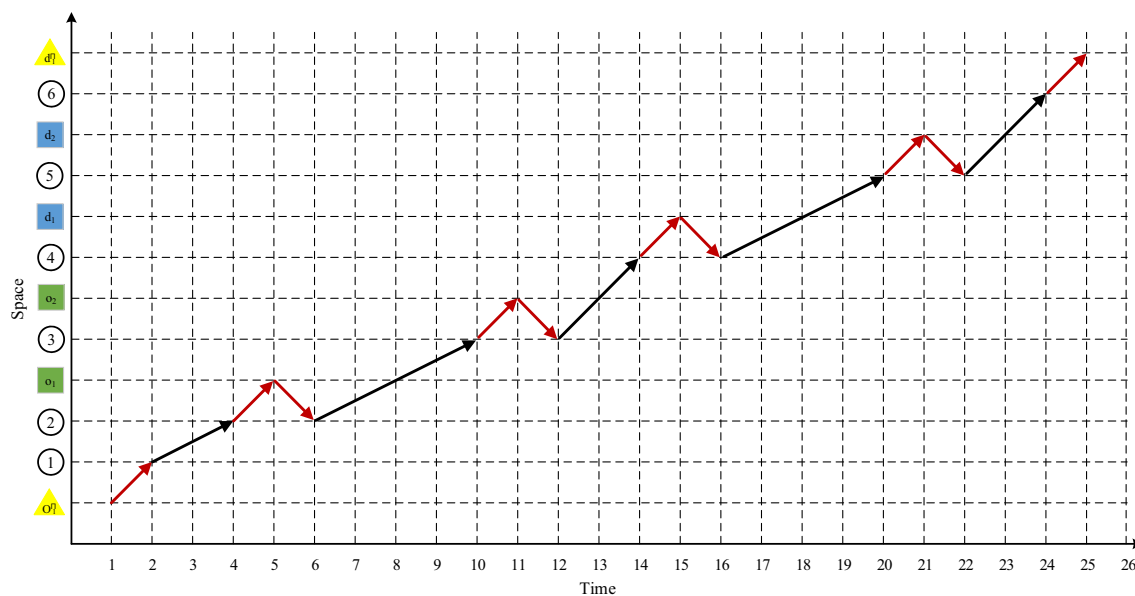
**Table 6** The evolution of path node sequences and path time sequences for the basic example

LR iteration	Step size	Node sequence	Time sequence
0	1	11;1;2;3;4;5;6;12	1;2;4;6;8;10;12;13
1	1	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;6;8;9;10;12;13;14;16;17;18;20;21
2	0.5	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;6;8;9;10;12;13;14;16;17;18;20;21
3	0.333333	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;6;8;9;10;12;13;14;16;17;18;20;21
4	0.25	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;6;8;9;10;12;13;14;16;17;18;20;21
5	0.2	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;6;8;9;10;12;13;14;16;17;18;20;21
6	0.166667	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;6;8;9;10;12;13;14;16;17;18;20;21
7	0.142857	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;6;8;9;10;12;13;14;16;17;18;20;21
8	0.125	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;6;8;9;10;12;13;14;16;17;18;20;21
9	0.111111	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;6;8;9;10;12;13;14;16;17;18;20;21
Upper bound:		11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;6;8;9;10;12;13;14;16;17;18;20;21

**Fig. 8** Vehicle space-time trajectory for the basic example**Fig. 9** Network layout and time-window settings for the example with extended link travel time

**Table 7** The evolution of path node sequences and path time sequences for basic example with extended link travel time

LR iteration	Step size	Node sequence	Time sequence
0	1	11;1;2;3;4;5;6;12	1;2;4;6;8;10;12;13
1	1	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;6;8;9;10;12;13;14;16;17;18;20;21
2	0.5	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;6;8;9;10;12;13;14;16;17;18;20;21
3	0.333333	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;6;8;9;10;12;13;14;16;17;18;20;21
4	0.25	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;6;8;9;10;12;13;14;16;17;18;20;21
5	0.2	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;6;8;9;10;12;13;14;16;17;18;20;21
6	0.166667	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;6;8;9;10;12;13;14;16;17;18;20;21
7	0.142857	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;6;8;9;10;12;13;14;16;17;18;20;21
8	0.125	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;6;8;9;10;12;13;14;16;17;18;20;21
9	0.111111	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;6;8;9;10;12;13;14;16;17;18;20;21
Upper bound:		11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;6;8;9;10;12;13;14;16;17;18;20;21

**Fig. 10** Space–time trajectory of the vehicle for the example with extended link travel time

and no extra vehicle or passenger waiting time is introduced.

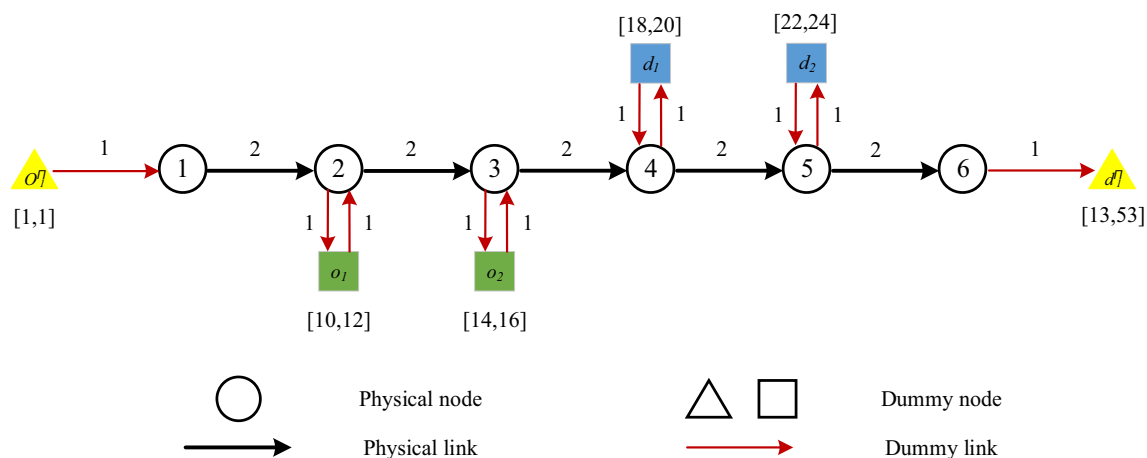
### 4.3 The Third Example with Extra Vehicle Waiting Time

In order to test the influence of vehicle waiting time on the optimal vehicle routes and schedules, the departure time window of passenger A is delayed from [5, 7] to [10, 12]. Therefore, the vehicle will have to wait at node  $o_1$  until passenger A gets ready to depart at time 10. In addition, the arrival time window of passenger A and the time windows of passenger B are modified accordingly, so that the vehicle will not wait at other places. The updated time window settings are shown in Fig. 11.

The upper bound and lower bound of this example are equal to 22.5, which turns out to be the optimal solution. The optimal routes and schedules for the vehicle are shown in Table 8, and the space–time trajectory of the vehicle is shown in Fig. 12. It can be observed that the vehicle arrives at node  $o_1$  at time 5 and waits until time 10, so the vehicle waiting time is 5. Besides, the total travel time of the vehicle is 25, but the cost ratio of vehicle waiting time is only 0.5, and then the optimal value of total cost is equal to  $25 - 0.5 \times (10 - 5) = 22.5$ .

### 4.4 The Forth Example with Extra Passenger Waiting Time

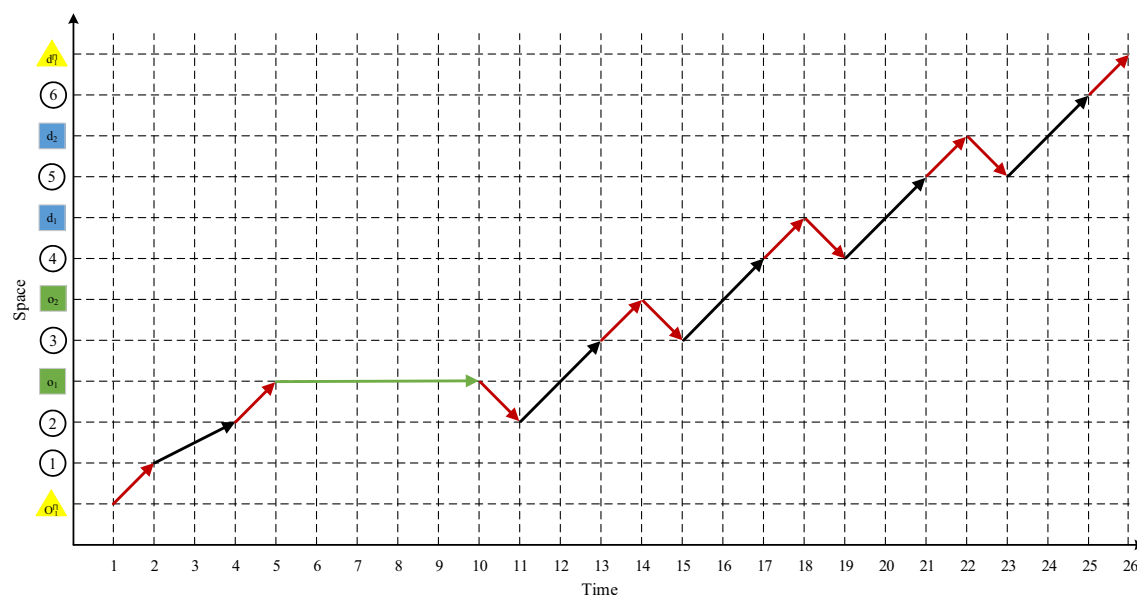
If the vehicle departs late from the origin depot, then the passengers will have to wait until the vehicle arrives.



**Fig. 11** Network layout and time-window settings for the example with extra passenger waiting time with extra vehicle waiting time

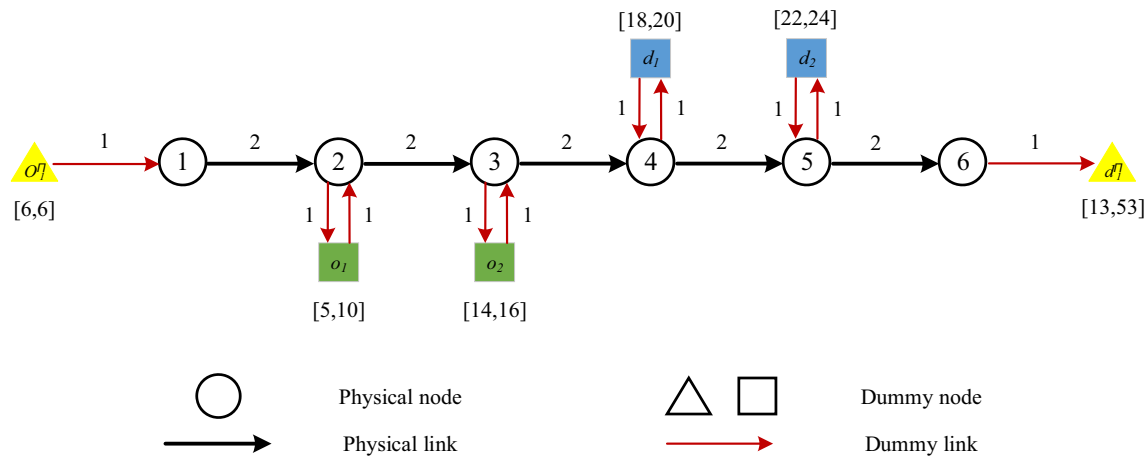
**Table 8** The evolution of path node sequences and path time sequences for the example with extra vehicle waiting time

LR iteration	Step size	Node sequence	Time sequence
0	1	11;1;2;3;4;5;6;12	1;2;4;6;8;10;12;13
1	1	11;1;2;7;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;10;11;13;14;15;17;18;19;21;22;23;25;26
2	0.5	11;1;2;7;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;10;11;13;14;15;17;18;19;21;22;23;25;26
3	0.333333	11;1;2;7;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;10;11;13;14;15;17;18;19;21;22;23;25;26
4	0.25	11;1;2;7;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;10;11;13;14;15;17;18;19;21;22;23;25;26
5	0.2	11;1;2;7;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;10;11;13;14;15;17;18;19;21;22;23;25;26
6	0.166667	11;1;2;7;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;10;11;13;14;15;17;18;19;21;22;23;25;26
7	0.142857	11;1;2;7;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;10;11;13;14;15;17;18;19;21;22;23;25;26
8	0.125	11;1;2;7;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;10;11;13;14;15;17;18;19;21;22;23;25;26
9	0.111111	11;1;2;7;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;10;11;13;14;15;17;18;19;21;22;23;25;26
Upper bound:		11;1;2;7;7;2;3;9;3;4;8;4;5;10;5;6;12	1;2;4;5;10;11;13;14;15;17;18;19;21;22;23;25;26



**Fig. 12** Space-time trajectory of the vehicle for the example with extra vehicle waiting time





**Fig. 13** Network layout and time-window settings for the example with extra passenger waiting time

**Table 9** The evolution of path node sequences and path time sequences for the example with extra passenger waiting time

LR iteration	Step size	Node sequence	Time sequence
0	1	11;1;2;3;4;5;6;12	6;7;9;11;13;15;17;18
1	1	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	6;7;9;10;11;13;14;15;17;18;19;21;22;23;25;26
2	0.5	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	6;7;9;10;11;13;14;15;17;18;19;21;22;23;25;26
3	0.333333	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	6;7;9;10;11;13;14;15;17;18;19;21;22;23;25;26
4	0.25	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	6;7;9;10;11;13;14;15;17;18;19;21;22;23;25;26
5	0.2	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	6;7;9;10;11;13;14;15;17;18;19;21;22;23;25;26
6	0.166667	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	6;7;9;10;11;13;14;15;17;18;19;21;22;23;25;26
7	0.142857	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	6;7;9;10;11;13;14;15;17;18;19;21;22;23;25;26
8	0.125	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	6;7;9;10;11;13;14;15;17;18;19;21;22;23;25;26
9	0.111111	11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	6;7;9;10;11;13;14;15;17;18;19;21;22;23;25;26
Upper bound:		11;1;2;7;2;3;9;3;4;8;4;5;10;5;6;12	6;7;9;10;11;13;14;15;17;18;19;21;22;23;25;26

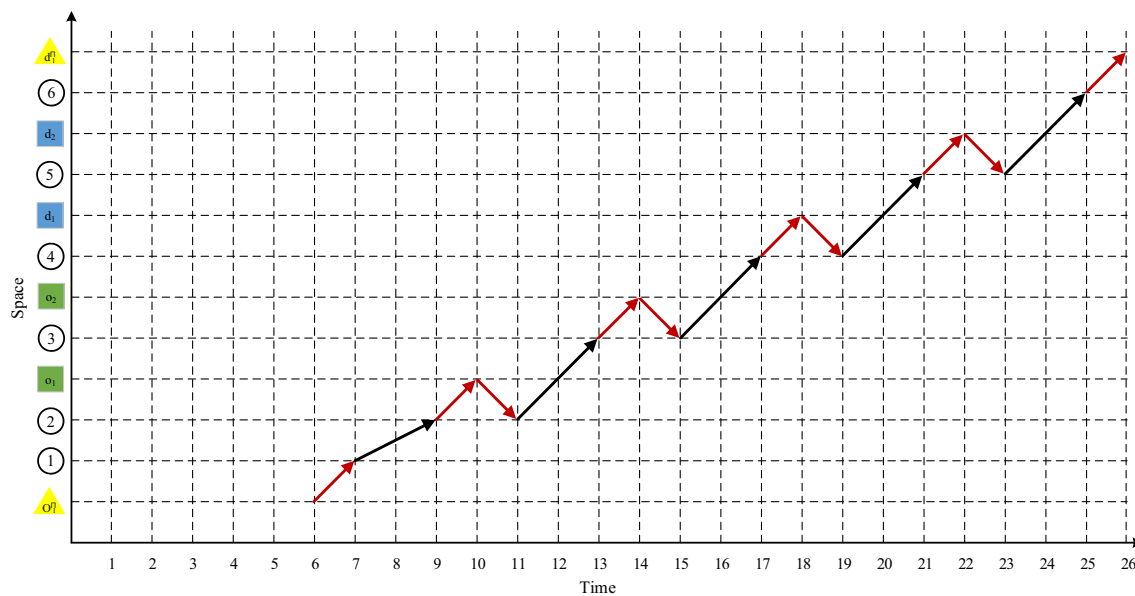
Therefore, the departure time window of the vehicle is delayed by 5 compared with the basic example, and the departure and arrival time windows of those two passengers are adjusted accordingly. The new time-window settings are shown in Fig. 13.

The upper bound and lower bound of this example are equal to 21.5, and the optimal routes and schedules of the vehicle are shown in Table 9, and the space–time trajectory of the vehicle is shown in Fig. 14. It is obvious that passenger A starts to wait at time 5 until the vehicle arrives at time 10, so the waiting time of passenger A is 5. In addition, because the cost ratio of passenger waiting time is 0.5 and the total travel time of the vehicle is 20, the optimal value of the total cost is  $20 + 0.5 \times 5 = 21.5$ .

#### 4.5 The Fifth Example for Branching Process Within a Column Generation Process

We test our column generation algorithm on the six-node transportation network illustrated in Fig. 1 for a scenario with three passengers and two vehicles. Table 10 shows origin–destination pairs, and passengers’ departure and arrival time windows. Terms “TW” and “TH” stand for time window and time horizon, respectively. The code and related data of this example can be downloaded at <https://github.com/YaoYuBJTU/VRPLite-python>.

Based on this scenario, we obtain a fractional optimal solution during solving the RMP. The solution,  $x(1, 1) = x(1, 2) = 0.5$ ,  $x(2, 1) = x(2, 3) = 0.5$ ,  $x(3, 2) = x(3, 3) = 0.5$  shows that three types of paths are used (two passengers can be served through the ride-sharing mode by the vehicle going through each path), but the value of the decision variable = 0.5 means that only 0.5 vehicles go through that path, which has no physical significance. As a



**Fig. 14** Space-time trajectory of the vehicle for the example with extra passenger waiting time

**Table 10** Passengers' origin-destination pairs and corresponding departure and arrival time windows

$o_1$	$d_1$	$o_2$	$d_2$	$o_3$	$d_3$	$o'_1$	$d'_1$	$o'_2$	$d'_2$
2	6	2	3	5	3	4	1	4	1
$TW_{o_1}$	$TW_{d_1}$	$TW_{o_2}$	$TW_{d_2}$	$TW_{o_3}$	$TW_{d_3}$	$TH_{v_1}$	$Cap_{v_1}$	$TH_{v_2}$	$Cap_{v_2}$
[5,7]	[8,11]	[5,8]	[11,15]	[7,9]	[12,14]	[1, 30]	2	[1, 30]	2

result, branching is needed to obtain the feasible solutions, i.e., for the fractional value  $x(1, 1) = 0.5$ , we could force  $x(1, 1)$  to be 1 and 0 as two child nodes, ensuring passenger  $p_1$  is served by vehicle  $v_1$  or not. After a branching step, column generation is used again and a series of new paths are generated through the subproblem. The feasible solution of branch  $x(1, 1) = 1$  is  $x(1, 1) = x(1, 2) = 1, x(4, 3) = 1$ , while for the other branch  $x(1, 1) = 0$ , there are two feasible solutions,  $x(2, 1) = x(2, 3) = 1, x(5, 3) = 1$  and  $x(3, 2) = x(3, 3) = 1, x(6, 1) = 1$ . Finally, we could obtain the feasible optimal solution,  $x(3, 2) = x(3, 3) = 1, x(6, 1) = 1$ , which means  $p_2$  and  $p_3$  are served by one vehicle through the ride-sharing mode and  $p_1$  is served by another vehicle.

## 5 Discussions and Conclusions

This research aims to improve the scheduled transportation system performance by enabling better vehicle scheduling capabilities in complex transportation on demand applications. Specifically, the VRPLite package addresses several fundamental research issues in scheduled transportation systems, which offers a set of solution platforms on holistic

traveler mobility optimization, agent-based trajectory control under the new environment of shared self-driving car or automated guided vehicle (AGV) networks. This open-source and educational modeling framework could help researchers understand the complex space-time-state network modeling methodologies, especially from a time-dependent and state-dependent shortest path perspective. Because the shipping of passengers and goods by shared self-driving cars or automated urban rail trains needs to be fully coordinated and cooperative, we hope this algorithm could help to demonstrate how to reduce the transportation cost and improve the efficiency in shipping passengers or goods [29], especially in the area of city logistics [30]. In particular, if the shared self-driving cars are electrified, the gas emission caused by the transportation process could decrease to a large extent [31].

It should be highlighted that, unlike the shared self-driving cars, AGVs usually move on the visual track-based networks with specially required path topologies and two AGVs may conflict on the interactions of their paths [32] where the paths of AGVs are planned in advance to avoid all kinds of obstacles [33, 34]. In this situation, the scheduling and routing of AGVs is very similar to that of scheduled rail systems [35–37] where each spatial and

temporal resource can only be occupied by at most one train. It can be shown that the scheduling and routing of AGVs are also a variant of the vehicle routing problem, and the readers can refer to [32, 38] for detailed reviews on the corresponding solution approach and applications of AGVs.

In addition, the VRPLite package uses a discretized space–time–state modeling approach, so it is natural to consider time-dependent link travel time in the program, such as the vehicle routing problem with time-dependent link travel time and path flexibility in the paper [39], as well as spatial and temporal conflicts between AGVs.

We hope that, the theoretical methodologies, insights and open-source tools developed from this research will be useful for modeling and optimizing new autonomous vehicle operation and control methods for metropolitan regions. In the future, a new class of ubiquitous distributed computing-based algorithms will be further studied, to include joint trip assignment, routing and scheduling problems.

**Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

## References

- Cordeau JF, Laporte G, Potvin JY, Savelsbergh MWP (2007) Chapter 7 transportation on demand. Handbooks in operations research and management science. Elsevier B.V, New York
- Parragh SN, Doerner KF, Hartl RF (2008) A survey on pickup and delivery problems. *Journal Für Betriebswirtschaft* 58(1):21–51
- Psaraftis HN, Wen M, Kontovas CA (2016) Dynamic vehicle routing problems: three decades and counting. *Networks* 67(1):3–31
- Cheng WC, Schonfeld P (2015) A Method for optimizing the phased development of rail transit lines. *Urban Rail Transit* 1(4):227–237
- Lu K, Han B, Zhou X (2018) Smart urban transit systems: from integrated framework to interdisciplinary perspective. *Urban Rail Transit*. <https://doi.org/10.1007/s40864-018-0080-x>
- Bao X (2018) Urban rail transit present situation and future development trends in China: overall analysis based on national policies and strategic plans in 2016–2020. *Urban Rail Transit* 4(1):1–12. <https://doi.org/10.1007/s40864-018-0078-4>
- Kelly J, Marinov M (2017) Innovative interior designs for urban freight distribution using light rail systems. *Urban Rail Transit* 3(4):238–254
- Wang Y, Zhang M, Ma J, Zhou X (2016) Survey on driverless train operation for urban rail transit systems. *Urban Rail Transit* 2(3–4):106–113
- He L, Liang Q, Fang S (2016) Challenges and innovative solutions in urban rail transit network operations and management: China's Guangzhou metro experience. *Urban Rail Transit* 2(1):33–45
- Niu H, Zhou X (2013) Optimizing urban rail timetable under time-dependent demand and oversaturated conditions. *Transp Res Part C Emerg Technol* 36:212–230
- Shang P, Li R, Liu Z, Yang L, Wang Y (2018) Equity-oriented skip-stopping schedule optimization in an oversaturated urban rail transit network. *Transp Res Part C Emerg Technol* 89:321–343
- Dampier A, Marinov M (2015) A study of the feasibility and potential implementation of metro-based freight transportation in Newcastle upon Tyne. *Urban Rail Transit* 1(3):164–182
- Toth P, Vigo D (eds) (2002) The vehicle routing problem. Society for Industrial and Applied Mathematics, Philadelphia
- Tong L, Zhou L, Liu J, Zhou X (2017) Customized bus service design for jointly optimizing passenger-to-vehicle assignment and vehicle routing. *Transp Res Part C Emerg Technol* 85:451–475
- Niu H, Zhou X, Tian X (2018) Coordinating assignment and routing decisions in transit vehicle schedules: a variable-splitting Lagrangian decomposition approach for solution symmetry breaking. *Transp Res Part B Methodol* 107:70–101
- Pallottino S, Scutellà MG (1998) Shortest path algorithms in transportation models: classical and innovative aspects. equilibrium and advanced transportation modelling. Springer, Berlin
- Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: theory, algorithms, and applications. Prentice Hall, Upper Saddle River
- Ziliaskopoulos AK, Mahmassani HS (1993) A time dependent shortest path algorithm for real time intelligent vehicle/highway systems. *Transp Res Rec J Transp Res Board* 1408:94–100
- Chabini I (1998) Discrete dynamic shortest path problems in transportation applications: complexity and algorithms with optimal run time. *Transp Res Rec J Transp Res Board* 1645:170–175
- Mahmoudi M, Zhou X (2016) Finding optimal solutions for vehicle routing problem with pickup and delivery services with time windows: a dynamic programming approach based on state–space–time network representations. *Transp Res Part B* 89:19–42
- Liu J, Kang JE, Zhou X, Pendyala R (2017) Network-oriented household activity pattern problem for system optimization. *Transp Res Part C*. <https://doi.org/10.1016/j.trc.2017.09.006>
- Zhou L, Tong L, Chen J, Tang J, Zhou X (2017) Joint optimization of high-speed train timetables and speed profiles: a unified modeling approach using space–time–speed grid networks. *Transp Res Part B Methodol* 97:157–181
- Lu G, Zhou X, Mahmoudi M, Shi T, Peng Q (2018) Optimizing resource recharging location-routing plans: A resource–space–time network modeling framework for railway locomotive refueling applications. *Computers & Industrial Engineering* (in press)
- Ruan JM, Liu B, Wei H, Qu Y, Zhu N, Zhou X (2016) How many and where to locate parking lots? A space–time accessibility–maximization modeling framework for special event traffic management. *Urban Rail Transit* 2(2):59–70
- Mahmoudi M, Chen J, Shi T, Zhang Y, Zhou X (2018) A cumulative service state representation for the pickup and delivery problem with synchronized transfers (Submitted)
- Chen R, Zhou L, Yue Y, Tang J, Lu C (2018) The integrated optimization of robust train timetabling and electric multiple unit circulation and maintenance scheduling problem. *Adv Mech Eng* 10(3):1–16
- Fisher ML, Jörnsten KO (1997) Vehicle routing with time windows: two optimization algorithms. INFORMS, Catonsville
- Lübbecke ME, Desrosiers J (2005) Selected topics in column generation. *Oper Res* 53(6):1007–1023
- Arslan A, Agatz N, Kroon L, Zuidwijk R (2016) Crowdsourced delivery: A dynamic pickup and delivery problem with ad-hoc

- drivers. Technical report, ERIM Report Series Reference. <http://ssrn.com/abstract2726731>
30. Savelsbergh M, Van Woensel T (2016) 50th anniversary invited article—city logistics: challenges and opportunities. *Transp Sci* 50(2):579–590
  31. Muñoz-Villamizar A, Montoya-Torres JR, Faulin J (2017) Impact of the use of electric vehicles in collaborative urban transport networks: a case study. *Transp Res Part D Transp Environ* 50:40–54
  32. Qiu L, Hsu WJ, Huang SY, Wang H (2002) Scheduling and routing algorithms for AGVs: a survey. *Int J Prod Res* 40(3):745–760
  33. Chen X, Li Y (2006) Smooth formation navigation of multiple mobile robots for avoiding moving obstacles. *Int J Control Autom Syst* 4(4):466–479
  34. Ota J (2006) Multi-agent robot systems as distributed autonomous systems. *Adv Eng Inform* 20(1):59–70
  35. Yin J, Tang T, Yang L, Gao Z, Ran B (2016) Energy-efficient metro train rescheduling with uncertain time-variant passenger demands: an approximate dynamic programming approach. *Transp Res Part B Methodol* 91:178–210
  36. Rao X, Montigel M, Weidmann U (2016) A new rail optimisation model by integration of traffic management and train automation. *Transp Res Part C Emerg Technol* 71:382–405
  37. Yin J, Yang L, Tang T, Gao Z, Ran B (2017) Dynamic passenger demand oriented metro train scheduling with energy-efficiency and waiting time minimization: mixed-integer linear programming approaches. *Transp Res Part B Methodol* 97:182–213
  38. Fazlollahtabar H, Saidi-Mehrabad M (2015) Methodologies to optimize automated guided vehicle scheduling and routing problems: a review study. *J Intell Robot Syst* 77(3–4):525–545
  39. Huang Y, Zhao L, Van Woensel T, Gross JP (2017) Time-dependent vehicle routing problem with path flexibility. *Transp Res Part B Methodol* 95:169–195