A decomposition algorithm to solve dynamic city-scale ridesharing problem

H Wang, W X Wu* and L N Qin
School of electrical and control engineering, north China University of technology, Beijing, 100144, China
*Corresponding author’s e-mail: wuwenxiang@ncut.edu.cn

Abstract. This paper proposes a computational efficient, rapid, dynamic ridesharing algorithm. Firstly, a designed pre-processing procedure reduces size of the input date of potential nodes for vehicle to pass through. The proposed decomposition algorithm decomposes the large-scale problem with pre-processing date into independent sub-problems, and the method of parallel computing is adopted to solve the sub-problems. We call the dynamic programming algorithm to deal with the vehicle routing problem involved in sub-problems. Greedy random adaptive search algorithm is compared with the decomposition algorithm to verify the algorithm. Finally, a mathematical example shows that the algorithm can effectively solve large-scale dynamic ridesharing problems.

1. Introduction

With the worsening of environmental problems, road congestion, and the high cost of vehicle owners, people recently focus on the empty load rate of vehicles [1]. Ride-sharing can provide travelers with a low-cost and convenient way to travel, leading to reduce emissions, congestion, etc. [2][3]

There is a variety of literature concerning ridesharing problems. This problem is an NP problem, so it cannot be solved by precise algorithm [4]. Therefore, it is a challenging problem to design an effective algorithm that can solve such a large-scale, real-time and complex NP problem in order to realize the real-time requirement of dynamic ridesharing [5~18]. Thus, the goal of this paper is to design an algorithm to solve the problem.

When the scale of a problem is huge, an approximate solution or a heuristic solution can get the desired result [19]. For example, Bian (2019) has come up with a heuristic algorithm to solve the first-mile ridesharing, called the solution pooling approach (SPA). The idea is to build a model for every passenger, and solve the model in a pool. However, because of many models to solve, the algorithm takes a long time and is not applicable in large-scale networks. Masoud (2017) proposed a decomposition algorithm to solve the Multi-Hop peer-to-peer ride-sharing problem. This algorithm needs to calculate the possible routing of each participant. For large-scale network, the routing of participant is extremely huge, so it cannot solve the ridesharing problem of large-scale road network. At the same time, this paper could not plan the routing of the vehicles.

The current research has not made a good solution for large-scale dynamic ridesharing problem. How to make reasonable matching between vehicles and passengers under the condition of large-scale network and make effective routing planning for vehicles is the key point of this paper. The core of our decomposition algorithm is to decompose large-scale problems into small sub-problems, which are independent of each other and can be solved in parallel. In the planning routing stage of decomposition algorithm, we use dynamic programming algorithm to solve. In order to reduce the number of nodes that
vehicle may pass through, we preprocess the data before decomposition algorithm. Mathematical example shows that our algorithm can solve the ridesharing problem of large-scale networks.

This paper is structured as follows. In section 2, the model of dynamic multiplication in a time interval is established. Section 3 preprocesses the input data to reduce the size of the model. We design a decomposition algorithm to decompose large-scale problems into small sub-problems in section 4. In section 5, the dynamic programming algorithm is applied to solve the vehicle routing problem, and as a comparison, we design another algorithm called greedy random adaptive search problem (GRASP) algorithm in section 6. Finally, a numerical example is proposed in section 7 to verify the algorithm.

2. ridesharing system

We consider a ride-sharing system that including matching and routing planning. We build a model to pursue the highest matching success rate. We have noticed that different objective functions can be set to achieve the different objectives, including minimizing the system cost, minimizing the travel time and so on. In order to show the effectiveness of the algorithm, we set the maximum matching rate.

After discretization of continuous time, in each time interval, the ridesharing is a static problem, and we build a static model for each time interval. At the same time, we know that there is no absolute advantage in the long run between one vehicle matching one passenger and one vehicle matching many passengers in a time interval [20]. In this article, we address the first case, where one vehicle matches one passenger in a time interval.

| Table 1. Notations |
|---------------------|
| **Notations** | **Descriptions** |
| $G$ | Road network, $G = (N, A)$ |
| $N$ | Set of points in the road network |
| $A$ | Set of links in the road network |
| $i$ | Passenger requests |
| $I$ | Set of passenger requests, $I = \{1, 2, \ldots, I_1, \ldots, B\}$ |
| $p_i$ | Number of passenger request $i$ |
| $s_i$ | Departure point of passenger request $i$, $s_i \in N$ |
| $I$ | Set of departure points of all passenger requests |
| $R$ | Arrival point of passenger request $i$, $r_i \in N$ |
| $RI$ | Set of arrival points of passenger points, $r_i \in N$ |
| $\bar{t}_i$ | The earliest departure time for passenger request $i$, $\bar{t}_i \in N$ |
| $\bar{t}_i$ | The latest arrival time for passenger request $i$, $\bar{t}_i \in N$ |
| $k$ | Vehicle |
| $K$ | Set of vehicles, $K = \{1, 2, \ldots, k, \ldots, M\}$ |
| $s_k$ | Departure point of vehicle $k$, $s_k \in N$ |
| $SK$ | Set of departure points of all vehicles |
| $r_k$ | Arrival point of vehicle $k$, $r_k \in N$ |
| $RK$ | Set of arrival point of all vehicles |
| $e_k$ | The earliest departure time of vehicle $k$ |
| $e_k$ | The latest arrival time of vehicle $k$ |
| $q_k$ | The seat capacity of vehicle $k$ |
| $VI$ | The union of the departure points and arrival points set of all passenger requests, $VI = SI \cup RI$, $VI \subset N$ |
| $VK$ | The union of the departure points and arrival points set of all passenger request, $VK = SK \cup RK$, $VK \subset N$ |
| $W$ | The union of the departure points and arrival points set of all passenger request and vehicles, $W = VI \cup VK$, $W \subset N$ |
The shortest distance from point \( a \in N \) to point \( b \in N \)

\[ d_{ab} \]

The shortest travel time for vehicle from point \( a \in N \) to point \( b \in N \)

\[ t_{ab} \]

| Notations | Description |
|-----------|-------------|
| \( L_k \) | The routing of vehicle \( k \) |
| \( x_{ab} \) | \( x_{ab} = 1 \), if vehicle \( k \) travels to location \( b \) after location \( a \) immediately, otherwise, \( x_{ab} = 0 \), \( a, b \in S \) |
| \( LN_{ka} \) | The vehicle just left the capacity of the node, \( b \in W \) |
| \( T_{ka} \) | The time when the vehicle arrives at the node \( a \), \( a \in N \) |

The problem can be formulated as the following model. For the notations, please refer to Table 1 and 2.

\[
\text{max} \sum_{i \in I} \sum_{k \in K} x_{ipk} \\
\text{Subject to} \\
\sum_{k \in K} x_{ik} = 1, \quad \forall k \in K \\
\sum_{i \in I} x_{ik} = 1, \quad \forall k \in K \\
\sum_{k \in K} x_{ik} \leq 1, \quad \forall i \in I \\
\sum_{i \in I} x_{ipk} \leq 1 \\
\sum_{a \in W} x_{kap} - \sum_{b \in W} x_{ab} = 0, \quad \forall a \in W / (S_k \cup T_k), \quad \forall k \in K \\
x_{kap} \in \{0, 1\}, \quad \forall k \in K, \quad \forall a \in W, \quad \forall b \in W \\
(\bar{T}_k - T_{ac}) x_{ac} \geq 0, \quad \forall i \in I, \quad \forall k \in K \\
e_k \leq T_{ac} \leq L_k, \quad \forall k \in K \\
x_{ac} = 1 \Rightarrow LN_{ac} = LN_{ac} + p_i, \quad \forall a \in W, \quad \forall i \in I \\
x_{ac} = 1 \Rightarrow LN_{ic} = LN_{ic} - p_i, \quad \forall k \in K, \quad \forall a \in W, \quad \forall i \in I \\
0 \leq LN_{ic} \leq q_k, \quad \forall k \in K, \quad b \in W \\
LN_{ik} = 0, \quad \forall k \in K \\
LN_{ik} = LN_{ik} + 1 \]

Formula (1) is the objective function that maximum matching rate. Formulas (2) and (3) represent that the passenger's departure point and arrival point must be in the vehicle routing. Formula (4) represents that one passenger can only be matched to one car, i.e., no transfer. Formula (5) represents that one vehicle can only match one passenger at most. Equation (6) represents the flow balance constraint. Formula (7) represents that the decision variables can only take 0 and 1. Formulas (8), (9) and (10) represent time window constraints. Formulas (11), (12), (13) and (14) represent vehicle load constraints.

3. Pre-processing procedure

The goal of the pre-processing procedure is to build a set, \( G_k \), for each vehicle, \( k \in K \), which is a set of points that the vehicle can potentially arrive. It must be noted that the potential arrive point of the vehicle not only satisfies the time window constraint, but also satisfies the time window constraint of the passengers inside of the vehicle.

As explained before, we present two points in time, \( e \) and \( l \). \( e \) is the earliest departure time of
vehicle k from point a. \( l \) is the latest arrive time of vehicle k to point b. Our detailed definitions of \( e \) and \( l \) are shown in the following table.

| Definition                                                                 | \( e \)   | \( l \)   |
|----------------------------------------------------------------------------|-----------|-----------|
| \( a \) is the departure point of vehicle k. \( b \) is the arrival point of vehicle k. | \( e = e_k \) | \( l = l_k \) |
| \( a \) is the departure point of vehicle k. \( b \) is the arrival point of passenger i. | \( e = e_i \) | \( l = l_i \) |
| \( a \) is the departure point of vehicle k. \( b \) is the departure point of passenger i. | \( e = e_i \) | \( l = l_i - t_{a-b} \) |
| \( a \) is the arrival point of passenger i. \( b \) is the arrival point of passenger i. | \( e = e_i \) | \( l = l_i \) |
| \( a \) is the arrival point of passenger i. \( b \) is the arrival point of vehicle k. | \( e = e_i \) | \( l = l_i \) |

We define a ellipse in the network. The departure and arrival points, \((a, b)\), travel time windows of participants can be used as the focal of the ellipse and long axis of the ellipse. According to the definition, the sum of the distances from two focal points of the ellipse, that is, the departure point and arrival point of the vehicle in the section, to any point on the ellipse satisfies the corresponding time windows constraint.

| Table 4. Reduced graph algorithm |
|----------------------------------|

**Algorithm 1** reduced graph algorithm

01 **step1. initialize**

02 \( W = \{\bar{S}, \bar{T}\} \), \( \forall i \in I \)

03 \( G_k = W \), \( \forall k \in K \)

04 \( x_{ab} \leftarrow 1 \), \( \forall k \in K' \)

05 **step2. update or initialize reduced graph for vehicles**

06 for each vehicle \( k \in K \)

07 \( G_k' = \emptyset \)

08 for each \( x_{ab} = 1 \), \( \forall a \in L_k \) \( \forall b \in L_k \)

09 \( l = \) the earliest departure time of vehicle k from point a

10 \( e = \) the latest arrive time of vehicle k to point b

11 for \( w \in G_k \)

12 if \( t_{a-w} + t_{w-b} \leq l - e \)

13 \( G_k' = G_k' \cup \{w\} \)

14 end if

15 end for

16 end for

17 \( G_k = G_k' \)

18 end for

As shown in the figure 1, all points in the shaded part are potential points that can be reached by vehicles. The set of these points is \( G_k \).

We start the algorithm by defining a set \( W \) that contains all passenger’s departure and arrival points (line 02), and initializing the set \( G_k \) that each vehicle can potentially reach as \( W \) (line 03). For each vehicle \( k \in K \), initialize \( G_k' \) as an empty set (line 05-06). According to the results of the last interval iteration, for each \( x_{ab} = 1 \) (line 07), We look for points that satisfy the formula, \( t_{a-w} + t_{w-b} \leq l - e \), which are potentially reachable by vehicles(line 08-14). Finally, for each vehicle, we set \( G_k = G_k' \).

4. decomposition algorithm

The essence of decomposition algorithm shown table 4 is to decompose a large-scale network into small sub-problems, each of which needs less time and space and is easier to solve.
We start to decompose the algorithm by setting sub-problem counters, \( n \), to 0, initializing the set of sub-problems as an empty set (line 02). We set each sub-problem consists of two parts, the first part is a passenger, the second is a collection of vehicles, the collection of potential arrival points of the vehicle includes the departure point and arrival point of the passenger. For each passenger, we put that passenger in the first part of the sub-question (line 05). For each vehicle, we look for whether the passenger's departure point and arrival point are in the vehicle's set of potential arrival points (line 06-09). We create a sub-problem for that passenger (line 11). If the set of vehicles corresponding to this sub-problem is not an empty set, set act=1 (line 12-14). For the routing planning problem of a sub-problem with passengers and vehicles selected in turn, we adopt the method of dynamic programming, which is introduced in detail in the following text (line 17-20). We notice that the elements of set \( \mathcal{R}_n \) are the routings that satisfy the constraints in the sub-problem. We set the set \( \mathcal{Z}_n \) that records the best matching vehicles for each sub-problem to be an empty set (line 22). We set the function \( \text{obj}(L) \) to be the travel time of routing \( L \). for each sub-problem \( \mathcal{S}_p \) with \( \text{act} (\mathcal{S}_p) = 1 \), We select the vehicle whose travel time is the minimum travel time of the elements of routing set \( \mathcal{R}_n \) corresponding to each sub-problem as the solution of the sub-problem (line 24-26). Let \( \mathcal{Z}_n \) be the set of solutions to all sub-problems. After solving all the sub-problems, we determine whether there is any conflict between the solutions. Since we stipulate that one vehicle can only match one passenger in each time interval, the criterion for judging the conflict is whether a vehicle matches two passengers, that is, whether a vehicle is the solution of two sub-problems. We select the sub-problem whose solution is the shortest travel time of the routing corresponding to the sub-problem of \( k \), take the vehicle as the final solution of this sub-problem, and delete the routing corresponding to the vehicle from the routing solution set of other sub-problems (line 31-33). We set \( \text{act} (\mathcal{S}_p) = 0 \) when the vehicle set corresponding to the sub-problem is empty (line 34-36). Output the decision variable according to the routing corresponding to the solution of each sub-problem (line 37-39). The termination criterion is that all \( \text{act} (\mathcal{S}_p) \) are 0 (line 41-43). If the termination criteria is not met, then go back to step 3.
Define vehicle removing the departure point and arrival point of vehicle, add departure point and arrival point empty set, \( a \) is the departure point of the vehicle, \( V' \) is the set of the node of the original routing of the vehicle is in the stage, and \( V' \) to represent the node that the vehicle has not passed. Initialization \( V \) is a define set \( V \) to represent the set of points that the vehicle has passed, \( a \) to represent the point where the vehicle starts.

Dynamic programming algorithm solves the problem of route planning for vehicles and passengers. We define set \( V \) to represent the set of points that the vehicle has passed, \( a \) to represent the point where the vehicle is in the stage, and \( V' \) to represent the node that the vehicle has not passed. Initialization \( V \) is a empty set, \( a \) is the departure point of the vehicle, \( V' \) is the set of the node of the original routing of the vehicle removing the departure point and arrival point of vehicle, add departure point and arrival point of passenger (line 02). Define \( D_s \) as the state of the s phase. State \( D_s \) of phase 1 is initialized as \( \{V, a, V'\} \). The state of the other phases is an empty set (line 03). The purpose of the second step is to find

Figure 2. the decomposition algorithm

In the example shown in figure 2, we set 4 sub-problems. The first step establishes four sub-problems. The red box represents the passenger and the green represents the vehicle. Each sub-problem includes a passenger and numbers of passenger-related vehicles. The second step applies dynamic programming to solve the routing problem of passengers and each vehicle. The shaded part represents the vehicle that meets the constraint conditions. In the third step, the solution of each sub-problem is obtained, and the vehicle with the minimum travel time of the routing is selected as the solution of the travel time. The green shade indicates the solution to the sub-problem. In step 4, determine whether there is a conflict. Vehicle 1 and 2 are matched to two passengers. The best passenger is selected to match the vehicle.

5. Dynamic programming algorithm

Dynamic programming algorithm solves the problem of route planning for vehicles and passengers. We define set \( V \) to represent the set of points that the vehicle has passed, \( a \) to represent the point where the vehicle is in the stage, and \( V' \) to represent the node that the vehicle has not passed. Initialization \( V \) is a empty set, \( a \) is the departure point of the vehicle, \( V' \) is the set of the node of the original routing of the vehicle removing the departure point and arrival point of vehicle, add departure point and arrival point of passenger (line 02). Define \( D_s \) as the state of the s phase. State \( D_s \) of phase 1 is initialized as \( \{V, a, V'\} \). The state of the other phases is an empty set (line 03). The purpose of the second step is to find
out all the state of each stage. Meanwhile, in the same stage, if the set of nodes through which the vehicles in more than two states have passed and the current node where the vehicles are, they are merged into one state. At the same time, if two or more states have the same set V and the current node a, they are merged into one state. For the status of each s-1 stage, the current node of the vehicle is updated successively. If the current node is the arrival point of the passenger, the departure point of the passenger must not be in the collection of the unarrived points of the vehicle (line 05-14). We define function $D(V_i)$ as the travel time in order of set V. The state corresponding to the smallest set of function values is retained, and other states are deleted (line 15-21). From the last stage, the state transition equation is calculated (line 24-34). Output planning results (line 36-41).

Table 5a. The decomposition algorithm

| Algorithm 3 the decomposition algorithm |
|------------------------------------------|
| **Step 1. Initialize and construct sub-problems** |
| 01 $n \leftarrow 0$, $SP = \emptyset$ |
| 02 for each rider $i \in I$ |
| 03 $n = n + 1$ |
| 04 set $I_n = \{i\}$, $\bar{I}_n = \emptyset$, $R_n = \emptyset$ |
| 05 for each driver $k \in K$ |
| 06 if $(\bar{x}_n, \bar{y}_n) \in G_k$ |
| 07 $\bar{I}_n = \bar{I}_n \cup \{k\}$ |
| 08 end if |
| 09 end for |
| 10 $SP_0 = \{I_n, \bar{I}_n\}$, $SP = SP \cup \{SP_n\}$ |
| 11 if $\bar{I}_n \neq \emptyset$ |
| 12 act($SP_n$) = 1 |
| 13 end if |
| 14 end for |
| **Step 2. Solve the routing problem for each sub-problem** |
| 15 for each driver $k \in \bar{I}_n$ |
| 16 call the dynamic programming algorithm and output $L'_k$ |
| 17 $L'_n = L'_k$, $R_n = R_n \cup \{L'_n\}$ |
| 18 end for |
| **Step 3. Solve sub-problems** |
| 19 $Z_n = \emptyset$ |
| 20 for each sub-problem $SP_n$ with act($SP_n$) = 1 |
| 21 if $obj(L'_n) = \min\{obj(L'_n)\}$, $\forall L'_n \in R_n$ |
| 22 $z_n \leftarrow k1$ |
| 23 end if |
| 24 $Z_n = Z_n \cup \{z_n\}$ |
| 25 end for |
Table 5b. The decomposition algorithm

| Step 4. termination criterion and output results |
|-----------------------------------------------|
| for each driver $k \in Z_m$                   |
| $\text{obj}(L_{nk}) = \min \{\text{obj}(L_{nk'})\}$, $\text{act}(SP_n) = 1$ |
| $R_n = R_n \setminus \{L_{nk}\}$, $\forall n \neq n1$ |
| $L_n = L_{nk}$                                |
| if $\bar{I}_n = \emptyset$, $\forall n \neq n1$ |
| $\text{act}(SP_n) = 0$                        |
| end if                                        |
| for $j = 1 : \text{length}(L_{nk}) + 1$       |
| let $x_{nkj} = 1$                             |
| end for                                       |
| if $\sum_{SP \subseteq SP} \text{act}(SP_n) = 0$ |
| stop                                          |
| else                                          |
| $m = m + 1$, go back to step 3                |
| end                                           |

Table 6. Dynamic programming algorithm

algorithm 3 dynamic programming algorithm

| Step 1. initialize for rider $i$ and driver $k$ |
|-----------------------------------------------|
| $V = \emptyset$, $a = \emptyset$, $V' = L_0 \cup \{\overline{s}, \overline{T} \} / \{\emptyset, L_0\}$ |
| $D_1 = \{V, a, V'\}$, $D_0 = \emptyset$, $s = 2, 3, \ldots, \text{length}(L_0) + 2$ |

| Step 2. forward movement                      |
|-----------------------------------------------|
| for $s = 2 : \text{length}(L_0) + 1$          |
| for each $\{V, a, V'\} \in D_{s-1}$          |
| for $b \in V'_i$                             |
| if $b$ is rider $i$’s departure point, or $\overline{a} \in V_i$ such that $b$ is rider $i$’s arrival point |
| if $T_{s\emptyset} \leq \overline{T}$ and $LN_{s\emptyset} \leq a_i$ |
| $V_i = V \cup \{a\}$, $a \leftarrow b$, $V'_i = V' / \{b\}$, $D_{s} = D_{s} \cup \{(V_i, b, V'_i)\}$ |
| end if                                        |
| end for                                       |
| for each $\{V, a, V'\} \in D_s$              |
| for each $\{V_i, a, V'_i\} \in D_s / \{V, a, V'\}$ |
| if $V_i = V$ and $D(V_2) = \min(D(V_1), D(V'))$, $V_2 \in \{V, V_i\}$ |
| $D_s = D_s / \{V_2, a, V'_i\}$, $V = V_2$     |
| end if                                        |
end for
end for

Step 3. backward movement

\[ s = \text{length}(L_a) + 1 \]

while \( s \neq 0 \)

for each element \( \{ V, a, V' \} \in D_s \)

if \( s = \text{length}(L_a) + 1 \)

\[ d\left(\{ V, a, V' \}\right) = d_{aV} \]

else

\[ d\left(\{ V, a, V' \}\right) = \min\left\{ d_{aV} + d\left(\{ V \cup \{ a \}, b, V' / \{ b \}\}\right), \ \forall b \in V' \right\} \]

end if

end for

\[ s = s - 1 \]

end while

Step 4. output results

for \( j = 1 : \text{length}(L_a) + 2 \)

if \( d\left(\{ V, a, V' \}\right) = d_{aV} + d\left(\{ V \cup \{ a \}, b, V' / \{ b \}\}\right) \)

\[ L'_j(j) = b \]

end if

\( V = V \cup \{ a \}, \ V' = V' / \{ b \}, \ a \leftarrow b \)

end for

6. Greedy random adaptive search algorithm

In order to verify the effectiveness of the algorithm, we designed the greedy random adaptive search algorithm. As shown in the table 7, the greedy random adaptive search algorithm consists of two parts: the first part is to find an initial solution, and the second is searched locally and the exact solution is obtained.

First, we initialize set \( P \) to passengers (line 02). When the set \( P \) is not empty, we do the following (line 03). Route planning is carried out for each vehicle take turns for each randomly selected passenger (line 04-06). We select two points in the routing of vehicle \( k \) in turn, \( a \) and \( b \). We insert the passenger departure point after point \( a \) and the passenger arrival point after point \( b \). Noting that if \( a \) and \( b \) are the same point, the passenger departure point is inserted after \( a \) and the passenger arrival point is inserted after the passenger starting point (line 06). Determine whether the time window constraint and load constraint are satisfied, and we record the planned routing if constraints are satisfied. Delete the passenger from set \( P \) (line 11).

In the local search stage, we adopt the classical exchange operator algorithm. Remove two passengers from their original routing and insert them into other's routing, if greedy function get better, exchange of match (line 16-24).
Table 7. Greedy random adaptive search algorithm

| Step 1. structural initial solution |
|------------------------------------|
| $P = I$                             |
| while $P = \emptyset$              |
| for $i \in I$                      |
| for $k \in K$                      |
| $L'_{ki} = \{a, \tilde{a}, b, \tilde{b}, \ldots\}$ |
| If $T_{ab} \leq T$ and $LN_{ab} \leq q_l$ |
| $L_{qi} = L'_{ki}$                 |
| end                                |
| $D(L_{ki}) = \min(D(L_{ki}))$      |
| $P = P \setminus \{i\}$            |
| end                                |
| Step 2. Local search               |
| for $i_1 \in I$                    |
| for $i_2 \in I$                    |
| $L'_{k1/2} = \{a, \tilde{a}, \ldots\}$, if $L'_{k1/2} \neq \emptyset$ |
| $L'_{k1/2} = \{a, \tilde{a}, \ldots\}$, if $L'_{k1/2} \neq \emptyset$ |
| If $D(L_{k1/2}) + D(L_{k2/1}) < D(L_{k1/1}) + D(L_{k2/2})$ |
| $L_{k2} = L'_{k2/1}$, $L_{k1} = L'_{k1/2}$ |
| end if                             |
| end for                            |

7. Numerical examples
In order to verify the effectiveness of the algorithm, we designed a simulation experiment. Since we only verify the validity of the algorithm, we do not have too much requirement on the authenticity of the data. We built a simulated road network consisting of 10 longitudinal roads and 20 transverse roads with 200 nodes. Starting point, arrival point, departure time and arrival time of passengers and drivers are generated by a random generator.

The ridesharing problem instances are solved on a PC with core i5 and 8GB of RAM. We use MATLAB to program the algorithm.

From the figure 3, we can see clearly that the running time of the decomposition algorithm is far less than that of the greedy random adaptive search algorithm. It is because that the decomposition algorithm does not search globally and applies parallel computing.
8. Conclusions

In this paper, we designed a decomposition algorithm for dynamic ridesharing problem. Compared with greedy adaptive search algorithm, it can effectively solve large-scale ridesharing problem and greatly reduce the running time. The central idea of this proposed algorithm is to turn large-scale problems into a number of independent small-scale sub-problems. We pursue parallel computation of these sub-problems. In order to solve the problem of vehicle path planning in the decomposition algorithm, we adopt the method of dynamic programming. Before the decomposition algorithm starts to run, we pre-process the input data in order to reduce the size of the points that the vehicle may pass through.

Finally, numerical examples are used to verify the effectiveness of our decomposition algorithm.

References

[1] Masoud, N. (2017) A decomposition algorithm to solve the multi-hop peer-to-peer ride-matching problem. Transportation Research Part B, 99: 1-29.
[2] Furuhata, M. et al. (2019) Ridesharing: the state-of-the-art and future directions. Transportation Research Part B. 57: 28-46.
[3] Cici, B. et al. (2014) assessing the potential of ride-sharing using mobile and social data: a tale of four cities, in: proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing. ACM.
[4] Lenstra, J.K., Kan. (1981) complexity of vehicle routing scheduling problems. Networks 11(2): 221-227.
[5] Kamar, E., Horvitz, E. (2009). Collaboration and shared plans in the open world: studies of ridesharing. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp. 187–194.
[6] Cheng, S.F., Nguyen, D.T., Lau, H.C. (2014) Mechanisms for arranging ride sharing and fare splitting for last-mile travel demands. In: Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems. International Foundation for Autonomous Agents and Multivalent Systems.
[7] Zhao, D. et al. (2014) Incentives in ridesharing with deficit control. In: Proceedings of the 2014 International Conference on Autonomous Agents and Multi–Agent Systems. International Foundation for Autonomous Agents and Multiagent Systems.
[8] Zhao, D. Ramchurn, S.D., Jennings, N.R. (2015) "Incentive design for ridesharing with
uncertainty." arXiv preprint arXiv:1505.01617
[9] Asghari, M. et al. (2016) Price-aware real-time ride-sharing at scale: an auction-based approach. In: Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM.
[10] Asghari, M., Shahabi, C. (2017) An On-line truthful and individually rational pricing mechanism for ride-sharing. In: Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM.
[11] Shen, W., Lopes, C.V., Crandall J.W. (2016) An online mechanism for ridesharing in autonomous mobility-on-demand systems. arXiv: 1603.02208
[12] Nguyen, D.T. (2013). Fair Cost Sharing Auction Mechanisms in Last Mile Ridesharing. Singapore Management University (Singapore).
[13] Zhang, J., Wen, D., Zeng, S. (2016) A discounted trade reduction mechanism for dynamic ridesharing pricing. IEEE Trans. Intell. Transp. Syst. 17 (6): 1586–1595.
[14] Kleiner, A., Nebel, B., Ziparo, V.A. (2011) A mechanism for dynamic ride sharing based on parallel auctions. Twenty-Second International Joint Conferences on Artificial Intelligence.
[15] Lloret B., Masoud, R.N., Nam D. (2017) P2p Ridesharing with Ride-Back on Hov Lanes: Towards a Practical Alternative Mode for Daily Commuting. No. 17-06253.
[16] Masoud, N., Lloret, B.R., Jayakrishnan, R. (2017b) Using bilateral trading to increase ridership and user permanence in ridesharing systems. Transp. Res. E 102: 60–77.
[17] Masoud, N., Lloret B.R. (2016). 95th Annual Meeting of the Transportation Research Board. Washington, DC.
[18] Ma, H., Fang, F., and DC. (2018) Parkes Spatio-temporal pricing for ridesharing platforms. arXiv preprint arXiv:1801.04015.
[19] Bian, Z.Y., Liu, X. (2019) Mechanism design for first-mile ridesharing based on personalized requirements part II: solution algorithm for large-scale problems. Transp. Res. B 120:172–192.
[20] Simonetto, A., Mnteil, J., Gambella, C. (2019). Real-time city-scale ridesharing via linear assignment problems. Transportation Research Part C. 101:208-232.