Hindawi
Wireless Communications and Mobile Computing
Volume 2019, Article ID 8035167, 13 pages
https://doi.org/10.1155/2019/8035167

Research Article

SMF-GA: Optimized Multitask Allocation Algorithm in Urban Crowdsourced Transportation

Pengfei Wang and Ruiyun Yu

Software College, Northeastern University, Shenyang, 110169, China

Correspondence should be addressed to Ruiyun Yu; yury@mail.neu.edu.cn

Received 26 October 2018; Accepted 14 February 2019; Published 17 March 2019

Academic Editor: Tony T. Luo

Urban crowdsourced transportation, which can solve traffic problem within city, is a new scenario where citizens share vehicles to take passengers and packages while driving. Differing from the traditional location based crowdsourcing system (e.g., crowdsensing system), the task has to be completed with visiting two different locations (i.e., start and end points), so task allocation algorithms in crowdsensing cannot be leveraged in urban crowdsourced transportation directly. To solve this problem, we first prove that maximizing the crowdsourcing system's profit (i.e., maximizing the total saved distance) is an NP-hard problem. We propose a heuristic greedy algorithm called Saving Most First (SMF) which is simple and effective in assigning tasks. Then, an optimized SMF based genetic algorithm (SMF-GA) is devised to jump out of the local optimal result. Finally, we demonstrate the performance of SMF and SMF-GA with extensive evaluations, based on a large scale real vehicle traces. The evaluation with large scale real dataset indicates that both SMF and SMF-GA algorithms outperform other benchmark algorithms in terms of saved distance, participant profits, etc.

1. Introduction

Transportation exists everywhere; a busy traffic has become a basic feature of large prosperous cities. However, the traffic congestion has become a headache problem for people thanks to the large amount of vehicles on the road, bad planed city, etc. Worse still, the number of vehicles is still keeping growing fast especially in many developing countries where the urban plan is much more terrible. For example, 28.03 million cars in 2016, an increase of 13.7% on the previous year, are sold in China [1]. Mexico City, which has the worst traffic condition in the world, makes drivers cost nearly 97% extra time during the morning hours to arrive their destinations [2]. Even though in Los Angeles which has a good urban planning, a driver can hit a traffic jam at a 40% rate. How to accommodate such large amount of cars on roads at the same time is really a vexed problem for urban planners.

To solve unlimited increasing vehicles, many solutions have been proposed, such as expanding roads, optimizing signals lights, setting up on-way streets, and building viaducts. However, no matter what they do, the road still has a maximum service traffic volume. The congestion can occur more easily when the amount of vehicles surpasses the maximum volume of the road. Besides, vehicle emissions contribute to the air pollution and are major ingredients in the creation of smog in cities. The recent report says that traffic pollution kills 5,000 people a year in UK [3].

How to reduce the usage of cars and satisfy people's transport demands at the same time is a key idea to solve all these problems. Transport crowdsourcing could be an appropriate way to alleviate traffic pressure and meet the travel requirement. Recent studies show that more than 29% trucks run on empty [4], and the occupancy rates (i.e., average number of passengers in a vehicle) of vehicles on the road are falling steadily [5]. What is worse, the occupancy rate is between 1.1 and 1.2 when they commute from/to work during rush hours. Increasing vehicle utilization can decrease the number of cars on roads, and it also has a great growth space. The urban crowdsourced transportation application can gather passengers or packages that have similar transport demands together, and corresponding drivers can pick them up when they drive to the destinations.

With the emergence of the concept of the sharing economy [6], urban crowdsourced transportation is a new

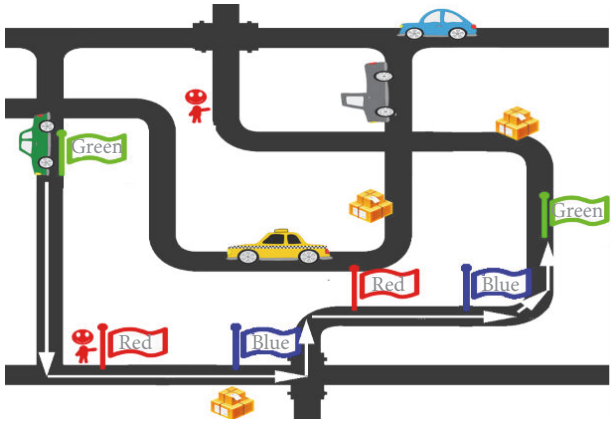


Figure 1: An example of urban crowdsourced transportation.

paradigm to handle traffic problems. An example is shown in Figure 1. Drivers can upload planned routes to the system; meanwhile, other participants post their transportation demands (i.e., to-do tasks). The urban crowdsourced transportation system then assigns suitable tasks to every planned route. This process is alike other crowdsourcing systems such as crowdsensing [7–10]. In crowdsensing, the system assigns suitable sensing tasks to participants, and many related task allocation algorithms for crowdsensing have been proposed taking different kinds of factors into account [11–18].

However, we cannot apply task allocation algorithms of crowdsensing into urban crowdsourced transportation systems directly. Although tasks have location information in both two systems, the task in urban crowdsourced transportation usually has two different locations: start and end points to pass. To fulfill an urban crowdsourced transportation task, one has to pass start and end points in sequence. In addition, a participant has to perform tasks one by one in crowdsensing systems; nevertheless, one can execute several tasks together in urban crowdsourced transportation. For instance, to collect noise data of five different locations, a person will go to these five points one after another. However, a driver can take three passengers in urban crowdsourced transportation in parallel. Also, due to the limited capacity of a vehicle, we cannot assign all suitable tasks to a single planned route.

Besides these differences, a driver also has to detour to pick up passengers or packages and take them to their destinations. A detour must be not nearer than the distance without detouring, and our goal is to influence driver's original plan as low as possible (i.e., minimizing the distance after detouring)and, meanwhile, complete to-do tasks as many as possible. Based on this, we propose optimized multitask allocation algorithm for urban crowdsourced transportation overcoming all these difficulties mentioned above. We first formulate the problem with considering realistic constraints and prove that this is a NP-hard problem. Then, we propose the greedy based Saving Most First (SMF) algorithm to assign tasks which have two-location-information to every planned route. The result of SMF algorithm is leveraged as an input to set up the Saving Most First based genetic algorithm (SMF-GA) to find the global optimum. Finally, we use the real world taxi data [19] and request generators [20] to do the extensive evaluation. The evaluation shows that SMF-GA outperforms all other contrast algorithms and can achieve the goal of saving distance with a relative smaller cycle times comparing with other genetic based algorithms under different scenarios. More narrowly, it can reduce more than half genetic cycle times with the input of the results of SMF. Meanwhile, SMF, which is a simple idea, can also get a relative good output without complex computation.

In our previous work [21], we designed the initial SMF-GA algorithm. In this paper, we prove the proposed task allocation problem is NP-hard and elaborate the SMF-GA algorithm in detail with showing how to calculate the distance between given route and allocated tasks, how to mutate and cross over, etc. In addition, more experiments are conducted with large scale real dataset.

Toward a comprehensive solution for task allocation in urban crowdsourced transportation systems, we make the following contributions.

(i) We propose a very simple but efficient heuristic algorithm called Saving Most First (SMF) which is the first specific greedy algorithm to solve the two-location tasks in urban crowdsourced transportation.

(ii) Based on SMF, we also devise the Saving Most First based genetic algorithm (SMF-GA) which can achieve a much better assignments with a smaller cycle times. According to our evaluation, it can achieve to convergence with less than half time cycles of other genetic algorithms.

(iii) SMF-GA is universal for all two-location based task allocation systems which aim to shorten the total traveling distance, for instance, carpooling, and logistics sharing system.

The rest of the paper is organized as follows. Section 2 formulates the problem and proves it is a combinational optimization problem which is NP-hard. The optimized multitask allocation algorithm is proposed in Section 3. Section 4 presents an evaluation of the algorithm. We discuss the related work in Section 5 and finally conclude the paper in Section 6.

2. Problem Formulation

Users generate two kinds of datasets, which are called planned route set and to-do task set, in urban crowdsourced transportation. The planned route set includes the travel information (e.g., when one user will drive from home to the work place), and the to-do task set contains the transport demands (e.g., one user wants to send a package from one location to another). The planned route set is denoted as $X = \{x_1, x_2, \ldots, x_M\}$. For the mth ($1 \leq m \leq M$) route, it has associated attributes $\{s_m, d_m, t_m, l_m, p_m\}$. Accordingly, the to-do task set generated by requesters is denoted as $Y = \{y_1, y_2, \ldots, y_N\}$. Each to-do task y_n, where $1 \leq n \leq N$, also has corresponding attributes $\{s_n, d_n, t_n, k_n, p_n\}$. All notations are elaborated in Table 1. Manhattan distance (taxicab metric)

Table 1: Main notations.

Notation	Explanation
X	Set of planned routes in the system
Y	Set of to-do tasks in the system
G	Graph
E	Edge set
V	vertex set
x	Element in planned route set X
y	Element in to-do task set Y
s_*	Start location of the route/task $*$
d_*	End location of the route/task $*$
t_*	Start time of the route/task $*$
l_*	Maximum capacity of the route $*$
k_*	Occupation of task $*$
p_*	The initiator of the route/task $*$
$Y*$	To-do tasks which have been assigned to the route $*$
e	edge
v	vertex

is used to measure the distance between two points since it is more suitable in the urban scenario, and the distance between s_m and d_m can be computed using

$$dis(s_m, d_m) = \left|lat_{d_m} - lat_{s_m}\right| + \left|lon_{d_m} - lon_{s_m}\right| \quad (1)$$

where lat_{s_m}, lat_{d_m} are latitudes of s_m and d_m; lon_{s_m} and lon_{d_m} denote the longitudes of s_m and d_m separately. We use $dis(x_m)$ and $dis(y_n)$ to denote the distance of route and task from start point to the end point. Here, we consider the urban crowdsourced transportation system has enough planed routes to assign to-do tasks. Moreover, $Ym = \{ym_1, ym_2, \ldots, ym_J\}$ is leveraged to denote the to-do task set assigned to the route x_m. To fulfill these assigned to-do tasks, the participant p_m is required to go through all to-do tasks' start and end points under the way to p_m's destination. Meanwhile, p_m has to pass the start point first then go through the end point to finish one assigned to-do task. But, one can go to different start points first and then take the passenger who has already been in car to the end point. The distance traveled by p_m when finishing all assigned to-do tasks is denoted as $dis(x_m, Ym)$, and it depends on the visiting sequence of locations. There exists the minimum traveling distance which can be denoted as $\min dis(x_m, Ym)$.

The goal is to improve the total vehicles' utilization as much as possible while satisfying all travelers' requests. Fulfilling to-do tasks will make the drivers' traveling distance longer than before since one usually has to detour to take extra passengers or packages to their destinations. Nevertheless, we can cut down the entire distance comparing with that all users drive their own cars separately at the same time. In other words, improving the vehicles' utilization is to maximize the sum of saved distance satisfying all users' demands, and the problem can be formulated as follows:

$$\max \quad \sum_{m=1}^{M} \left\{ \left[dis(x_m) + \sum_{j=1}^{J} dis\left(ym_j\right) \right] - \min dis(x_m, Ym) \right\} \quad (2)$$

$$\text{Subject to:} \quad \forall ym_j \in Ym,\ t_j \geq t_m \quad (3)$$

$$\sum_{j=1}^{J} k_j \leq l_m \quad (4)$$

$$\forall ym_j \in Ym,\ d_j \text{ can be selected to pass if and only if } s_j \text{ has been passed} \quad (5)$$

The problem is to compute (2) subject to (3), (4), and (5), where the to-do task should start no earlier than the planned route; the planned route should have enough space to accommodate the to-do task and the destination location of the to-do task must be passed after the start location is passed. Obviously, this is a combinatorial optimization problem. We can further simplify equation (2) after calculating the outer most summation, and it can be rewritten as

$$\sum_{m=1}^{M} dis(x_m) + \sum_{n=1}^{N} dis(y_n) - \min \sum_{m=1}^{M} \min dis(x_m, Ym) \quad (6)$$

Obviously, the value of $\sum_{m=1}^{M} dis(x_m) + \sum_{n=1}^{N} dis(y_n)$ is a fixed value for a specific route and task set, so the problem can be converted as

$$\min \sum_{m=1}^{M} \min dis(x_m, Ym) \quad (7)$$

So the goal turns to compute (7) subject to (3), (4), and (5). We can prove that the formulated problem is NP-hard, and the proof is as follows.

NP-Hard Proof. The problem is harder than computing equation (7) without any limits; further, (7) is harder than $\sum_{m=1}^{M} \min dis(x_m, Ym)$ which is no easier than $\min dis(x_m, Ym)$. The original problem will be NP-hard if we can prove that $\min dis(x_m, Ym)$ is NP-hard. The problem now is converted to find the shortest distance given the fixed start point v_s, end point v_d, and a list of other passing points and each point only can be passed one time. Add a dummy point v_o which only connects to the start point v_s and end point v_d; meanwhile, the distance between v_o and v_s, v_d is 0. Assume the graph $G = (V, E)$ is a graph, where the vertex set V includes all points including dummy, start, end, and other passing ones; the edge set E contains the

distances among all points. Given graph G, we hope to find the shortest possible route that visits each point exactly once, starting from v_o and returning back to v_o. This question is the same as our problem. We create another graph $G' = (V, E')$, where $E' = \{(i, j) \in V - \{v_o\},\ i \neq j,\} \cup \{(v_o, v_s), (v_o, v_d)\}$, and we have $\deg(v') + \deg(v'') \geq n$ for every two vertexes v' and v''. Based on Ore's theorem, the graph G' is a Hamiltonian graph, so the original problem is at least a NP-hard problem according to the Ore's theorem.

3. Optimized Multitask Allocation Algorithm

In this section, we propose the optimized task allocation algorithm for urban crowdsourced transportation. Traditional greedy algorithms do not work in this scenario, so we first propose a heuristic greedy algorithm called Saving Most First (SMF) defining the distance between route and to-do task taking two locations (i.e., start and end points) of the to-do task into consideration. However, the greedy algorithm is usually suboptimal because it always selects the local best at each step and sometimes could be stuck in the local optimum. By imitating the evolution process, it is proved that GA can improve the effect of the suboptimal issue through the process of crossover and mutation. To overcome the local optimum problem, the Saving Most First enhanced genetic algorithm (SMF-GA) is devised to find the global optimum utilizing the result of the SMF algorithm as the input of the genetic algorithm.

3.1. Saving Most First Algorithm. The greedy algorithm is much more difficult to be applied in transport crowdsourcing than in single-location-task systems (e.g., mobile crowdsensing). Tasks are completed with a specific location in mobile crowdsensing, but to finish a task in urban crowdsourced transportation we need to take the passenger or package from the source point to the destination. It is impossible to measure the distance between two paths with single location measurement method, so this difference leads to the fact that general greedy algorithms (e.g., nearest first) cannot be applied directly to urban crowdsourced transportation. Instead of selecting the nearest point, we compute the saved distance when one to-do task is assigned to a specific planned route as shown in Figure 2. We always allocate the to-do task to the planned route which has the largest saved distance. We compute the saved distance between all to-do tasks and planed routes and then select the local best (i.e., highest saved distance) at each step. Hence, we first propose the method how to calculate the distance matrix; then, the details of SMF are given.

The pseudocode of SMF is presented in Algorithm 1. The SMF algorithm computes saved distances with considering different factors (e.g., time constraints and overlapped paths). First, based on (3), one to-do task cannot be assigned to the route which happens later than it; meanwhile, the start time cannot be earlier too. It is defined that the saved distance between the route and to-do task is negative infinity unless it meets the requirement of (3). The computational complexity of the saved distance should be low enough to ensure the algorithm performance. Assume the planned route x_m has the to-do task y_n, and $dis(s_m, d_m) + dis(s_n, d_n)$ denotes the total distance if x_m and y_m are executed individually. The traveling distance is $dis(s_m, s_n)+dis(s_n, d_n)+dis(d_n, d_m)$ when y_n is allocated to x_m. So, the saved distance $sv(x_m, y_n)$ can be calculated as (8) and the illustration diagram is shown in Figure 2.

$$sv(x_m, y_n) = \begin{cases} dis(s_m, d_m) - dis(s_m, s_n) - dis(d_m, d_n) & t_m \leq t_n \leq t_m + \theta \\ -\infty & t_m \geq t_n,\ t_m \leq t_n + \theta \end{cases} \tag{8}$$

So far, we have got the saving matrix between planned routes and to-do tasks, and the greedy based Saving Most First (SMF) algorithm is designed to be as the baseline. We assign the task y to the route x which has the highest saved distance with y with meeting all constraints during each iteration. The route will be eliminated from the route pool when the volume of tasks reaches its limits.

3.2. SMF-GA Algorithm. The SMF algorithm usually achieves the local optimum since it always selects the local best at every step. Now that we cannot solve this problem within a polynomial time, the genetic algorithm (GA), which is an efficient way to jump out from the suboptimal with lower computation, is utilized to improve the allocation result of SMF algorithm. The key idea of GA is to imitate the process of natural selection in real world, and it mainly includes steps of selection, crossover, and mutation. However, it is difficult to find a satisfying result when the solution space is too big. It has been proved that some heuristic methods can help solve the above problem, so we propose the Saving Most First based genetic algorithm (SMF-GA) to make our result as close as the optimal solution.

Next, we elaborate the details of SMF-GA which mainly includes population initialization, selection, crossover, and mutation. As we know, a better initial population helps to decrease the time that algorithm needs to achieve an acceptable result and could improve the quality of the final answer especially, so it is important to choose a suitable initial population especially when the solution space is too huge. The result of the SMF algorithm is utilized to initialize the population of SMF-GA algorithm. We use a task-route matrix to present each individual, and human-perceived saved distance is proposed to calculate the fitness. In addition, all operations in SMF-GA must follow the constraints in (3), (4), and (5).

Input: The planned route set $X = \{x_1, x_2, \ldots, x_M\}$ and the to-do task set $Y = \{y_1, y_2, \ldots, y_N\}$, all routes' and tasks' corresponding attributes.
Output: The $< route, task >$ tuple

```
Initialize the saved distance matrix between all routes and tasks according to (8)
while X ≠ Φ and Y ≠ Φ do
  for counter = 1 to size(Y) do
    Find the counterth highest saved distance value from the matrix, and get the
    corresponding planned route x_i and to-do task y_j
    if l_i > k_j then
      Select < x_i, y_j >, and assign y_j to x_i
      l_i ⟵ l_i − k_j
      Remove y_j from Y
      if l_i == 0 then
        Remove x_i from X
      end if
      Jump out of the loop
    end if
  end for
  Update saved distance matrix
end while
return < route, task >
```

Algorithm 1: The *SavingMostFirst* algorithm.

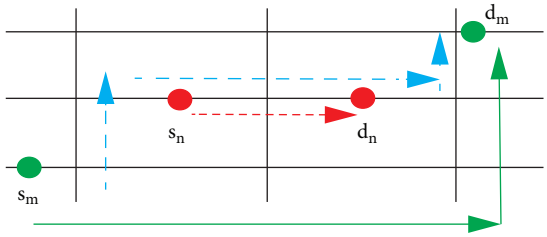
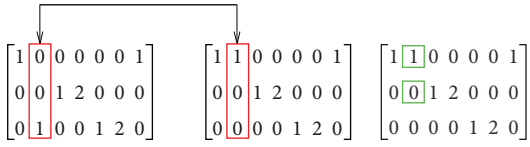


Figure 2: An example of computing saved distance.



(a) Crossover (b) Mutation

Figure 3: The example of crossover and mutation.

3.2.1. Solution Representation. Each route is assigned a subset of tasks, so the matrix is utilized to represent one solution. The rows are tasks, and the columns denote routes. One example of a solution is shown as Figure 3. Each element in the matrix can be nonnegative integer, and 0 indicates that the task with the row index of this element is not assigned to the route with the column index of it. Each task can only be assigned to one route in our scenario (i.e., carpooling system); thus only one element is greater than 0 in each row. The positive integer element in each row indicates how much volume the corresponding task occupies the route with the column index of this element. The solution is feasible when task happens later than the route in a time period and the sum of all assigned tasks' volume cannot exceed the maximum capacity the route can burden.

3.2.2. Population Initialization. According to the previous researches, the initial population plays a very important role on the results of GA since all following steps depend on it. Distributing individuals uniformly in the solution space is usually a common way, but we cannot do that for this problem since its solution space is so large that all initial individuals are far away from the optimal results. To handle this problem, the result of SMF algorithm is utilized to generate the initial population. First, the result of SMF algorithm is as one individual of the initial population. Based on the result of SMF algorithm, we generate other individuals through the mutation, which is also a genetic operator, elaborated below. The number of population usually varies from 60 to 100 relying on the size of planned routes and to-do task set; generally, a bigger number of population will be chosen for a larger set. Others are usually generated randomly, and it can make the individuals distribute uniformly.

3.2.3. Fitness. Maximizing the total saved distance when all to-do tasks are finished is the objective of our algorithm. The saved distance of each route is required to be computed since the total saved distance is the sum of all route's distances when assigned tasks are fulfilled. Based on the above analysis, It is known that the saved distance of the route x_m with fulfilling to-do task set Y_m is $dis(x_m) + \sum_{j=1}^{J} dis(ym_j) - \min dis(x_m, Ym)$. We can calculate $dis(x_m) + \sum_{j=1}^{J} dis(ym_j)$ easily through (1), but $\min dis(x_m, Ym)$ cannot be calculated within a polynomial time since it is NP-hard. Now that we cannot calculate the value of $\min dis(x_m, Ym)$, we use an approximate value to replace it. One is always going to the

Input: The planned route's start point s_m and end point d_m; start points set $S_m = \{s_1, s_1, \ldots, s_J\}$ and end points set $D_m = \{d_1, d_1, \ldots, d_J\}$ of to-do tasks which are assigned to x_m.
Output: The total distance dm that p_m travels when finishing assigned to-do tasks $dis(x_m, Ym)$

```
1: E_m ⟵ S_m, p ⟵ s_m, dm ⟵ 0, V_m.add(p)
2: while TRUE do
3:    dist_min ⟵ dis(p, E_m[0])
4:    p_min ⟵ E_m[0]
5:    for i = 1 to E_m length-1 do
6:       if dist_min > dis(p, E_m[i]) then
7:          dist_min ⟵ dis(p, E_m[i])
8:          p_min ⟵ E_m[i]
9:       end if
10:   end for
11:   dm ⟵ dm + dist_min
12:   p ⟵ p_min, V_m.add(p), E_m.remove(p)
13:   if p in S_m then
14:      Find corresponding end point d_j of start point p from D_m and add d_j to E_m
15:   end if
16:   if E_m length == 0 then
17:      Exit While
18:   end if
19: end while
20: dm ⟵ dm + dis(p, d_m), V_m.add(d_m)
21: return dm
```

Algorithm 2: Calculate $dis'(x_m, Ym)$.

nearest point first, and it can have a relative better satisfying distance, so we utilize this empirical value, which is denoted as $\min dis'(x_m, Ym)$, to substitute the minimum value we cannot compute with in polynomial time. Considering the real world, we cannot pass the end point before passing its corresponding start point (i.e., constraint in (5)). The pseudocode of computing empirical distance is shown in Algorithm 2. It is assumed that the population I has individuals $i_1, i_2, \ldots, i_Z$, and the total saved distance of the individual $i_z (0 < z \leq Z)$ can be calculated as

$$sd(i_z) = dis(x_z) + \sum_{j=1}^{J} dis(yz_j) - dis'(x_z, Yz) \quad (9)$$

Then, the fitness of the individual i_z can be calculated as (10) which is a key value to decide which individual to drop.

$$F(i_z) = \frac{sd(i_z)}{\sum_{w=1}^{Z} sd(i_w)} \quad (10)$$

3.2.4. Crossover and Mutation. The crossover operator is utilized to generate child individuals by integrating parents' genes. Mutation operator can maintain genetic diversity from one generation's chromosomes to the next. All individuals generated from crossover or mutation should be feasible; in other words, to-do tasks should be assigned to routes with satisfying constraints and rationality. Figure 3 shows an example of crossover and mutation with three planned routes and seven to-do tasks. The value at the *ith* row and *jth* column means that the task j is assigned to the planned route i with occupying corresponding value space. The task is not assigned to the planned when the element value is 0. Each column of matrix is defined as a gene, and we exchange columns between two individuals for crossover. Meanwhile, the work is inseparable (e.g., we cannot divide a task which deliver a package to someplace to two planned routes.), so we do the mutation as Figure 3(b) where we exchange the value which is not zero in one column to another one in the same column. In addition, we also need to ensure the feasibility of task allocation, so we will not exchange the columns or mutate the elements if the total occupation of all assigned tasks exceeds the maximum capacity of the planned route.

3.2.5. Terminal Condition. The terminal condition of the genetic algorithm we adopt here is the number of cycle times since the output of the algorithm is the best single individual and we do not know the global optimum beforehand. The algorithm will stop if it exceeds the time limit which is set beforehand.

4. Evaluation

To evaluate the effectiveness of SMF-GA algorithm, a large scale real world dataset is used in this paper. We first elaborate the details of the dataset, basic settings, and benchmarks we use in the evaluation. Then, the saved distance and computational complexity are studied. Finally, we conclude that SMF-GA can assign tasks to suitable routes maximizing saved distances with much lower computational complexity.

4.1. Dataset and Experiment Settings

4.1.1. Dataset. The T-drive data [19], which contains six-day trajectories of 10,357 taxis within Beijing, is used to model the location and movement of vehicles in the city. The average sampling interval of taxis in dataset is about 177 seconds with a distance of about 623 meters. We choose periods of path of each vehicle to generate the planned route set from T-drive data with selecting start and end points based on trajectories. Since each sampling only includes taxi id, date time, longitude, and latitude, we also add maximum capacity of vehicles with using uniform distribution from 1 to 4. Corresponding taxi query generator [20] is applied to generate task requests as realistic as possible. Analogously, the generator also does not have the occupation information (i.e., how much space the task will occupy), so the information is generated randomly. To be specific, we have two kinds of tasks: transporting people and packages. It is just assumed that the maximum occupation value is 4 for simplicity since the maximum value of the capacity is 4. For task of transporting people, the value of occupation is integer from 1 to 4; for task of transporting packages, the value of occupation is floating point number from 1 to 4. At this point, we have the planned route set and to-do task set generated from large scale real world dataset.

4.1.2. Settings. To verify SMF-GAs performance under different conditions, we leverage various to-do task sets. First, we have task sets with different number of task. They are 0.5, 1, 1.5, and 2 times of the planned route set. But the total required occupation of all to-do task sets is smaller than the planned route set. Here, we do not consider the case that the sum of capacities is smaller than the occupation of to-do tasks since we cannot assign all tasks to planned routes if the capacity of all planned routes is not enough to perform all to-do tasks. How to maximize the saved distance when the planned routes are not enough for all to-do tasks could be a future study. In addition, we also care about the performance of SMF-GA under different task distributions. Here, we use another three task sets including compact (congregated in the train station, shopping malls, etc.), scattered (distributed uniformly) and hybrid distribution (Integrate the first two kinds distribution).

Besides, to evaluate the effectiveness of the proposed algorithm, the following algorithms are used as benchmarks.

(1) *Nearest first algorithm (NF)*: it is a greedy based heuristic algorithm which always allocate the nearest tasks to each route, and it is used widely since it is simple and easy to apply.

(2) *Saving most first algorithm (SMF)*: this refers to the proposed Saving Most First algorithm without the part of the genetic algorithm.

(3) *Genetic algorithm (GA)*: this is the ordinary genetic algorithm which does not use the Saving Most First algorithm as an input. The inputs of GA are assignments which is generated at random.

(4) *Nearest first based genetic algorithm (NF-GA)*: it is the combination of nearest first algorithm and genetic algorithm. It utilizes the result of nearest first algorithm as an input of the genetic algorithm.

(5) *Enumerate algorithm (EA)*: it can search the whole solution space which means it will find the optimal solution; however, its computational complexity is very high comparing with other algorithms. We leverage it to evaluate the performance of SMF-GA. The computational complexity increases with exponential function, so we just do within a small scale data.

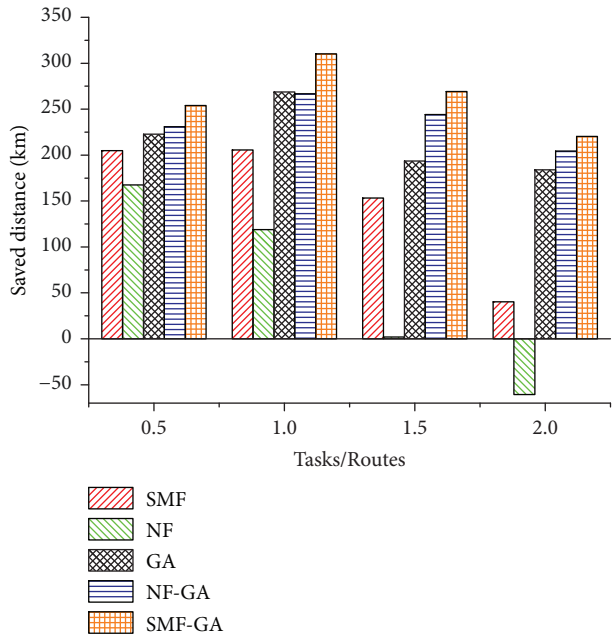


Figure 4: Saved distance under five task allocation algorithms.

We evaluate SMF-GA with saved distance, user activity level, and computational complexity. According to our optimization purpose, the saved distance is a critical metric, and a larger saved distance means a better performance of the algorithm. In addition, we also calculate the rewards one user can get after fulfilling tasks in the system. Generally speaking, the user will be more active when earning more money. Meanwhile, we compare the performance and computational complexity between SMF-GA and EA algorithm which can get the best output within a small scale of planned route set and to-do task set since the computational complexity is very large for EA.

4.2. Saved Distance. We evaluate the performance of saved distance under different scenarios in this subsection. We first show saved distance performance of SMF-GA comparing with SMF, NF, GA, and NF-GA for a quick look. Then, we study the performance of SMF-GA in saved distance with the increase of cycle times contrasting with the other two genetic algorithms. Finally, we show our algorithm has excellent performance in different kinds of task distributions.

The performance comparison on total saved distance between SMF-GA and the other four baselines is presented and the result is shown in Figure 4. EA is not available

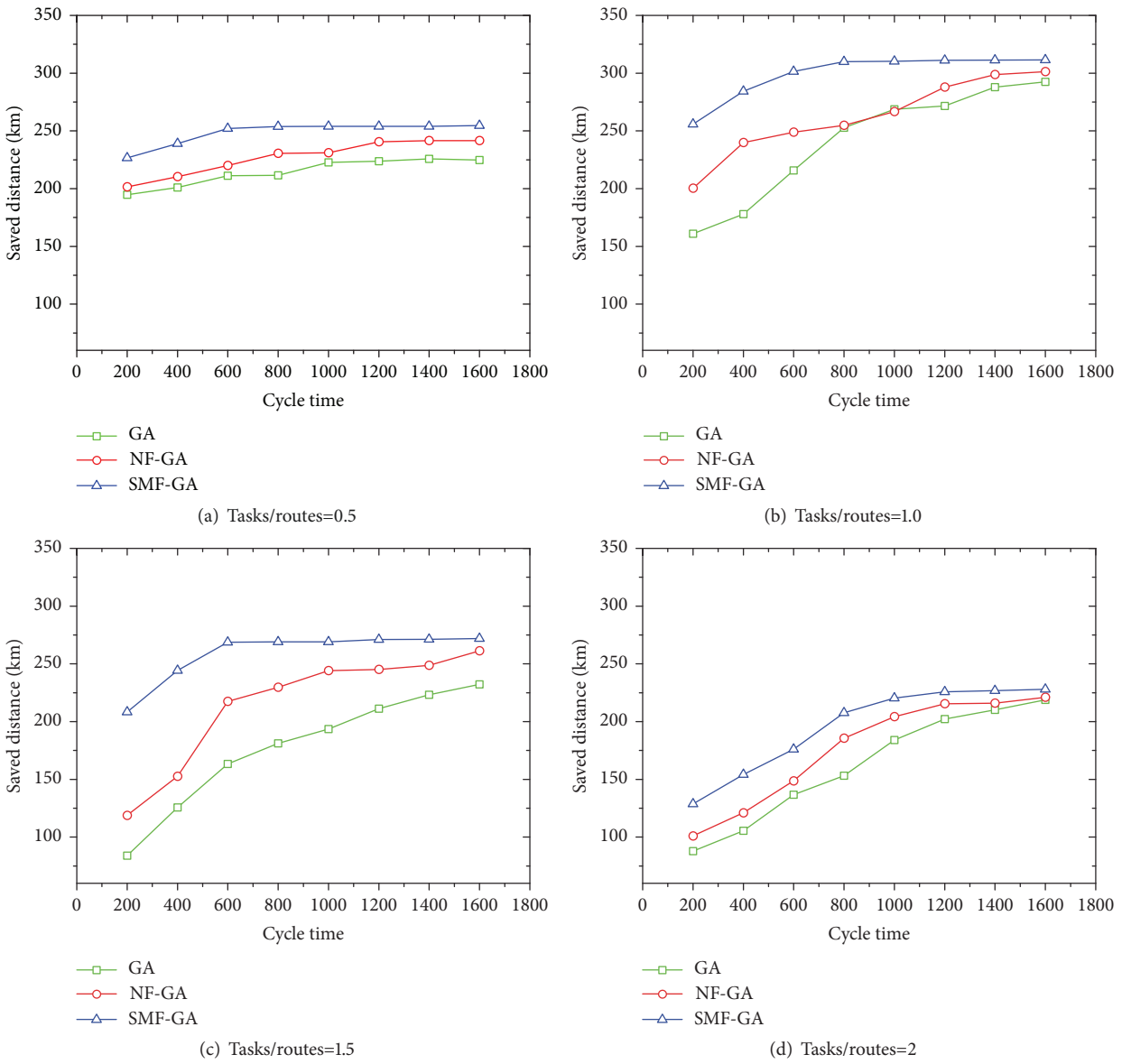


Figure 5: Convergences of genetic algorithms.

here since its high computational complexity and we cannot get the results for such large scale datasets. We know that the cycle times of the genetic algorithm can influence its performance, so we adopt the cycle times when the algorithms reach to their own convergence. We can conclude that SMF performs much better than NF; moreover, the SMF-GA outperforms all other four algorithms including SMF, NF, GA, and NF-GA in terms of saved distance. The reason why SMF is better than NF is because SMF always chooses the task which can maximize the saved distance; however, NF chooses the nearest task which may not save the distance. When the number of to-do tasks grows, especially it is twice as many as planned routes, NF assigns too many tasks to the route which could go a different direction, and it leads to that the total distance is longer than the condition that all people travel separately. The reason why SMF-GA outperforms SMF is that SMF-GA utilizes SMF as an input and search the global optimum through the crossover and mutation, so its result is at least as good as SMF. Meanwhile, the reason why SMF-GA outperforms GA is that, instead of using a random input, SMF-GA uses the result of SMF as the input which is closer to the global optimum. NF-GA and GA have similar performance and it indicates that NF does not work for urban crowdsourced transportation, especially when the number of tasks is very large.

Then, we study the convergence feature of SMF-GA comparing with GA and NF-GA. A better genetic algorithm should reach convergence much more quickly. We still use the route set and four task sets which we used in last simulation. These three genetic algorithms are run with different cycle times separately and the result is shown in Figure 5. It indicates that the value of cycle times which can make the

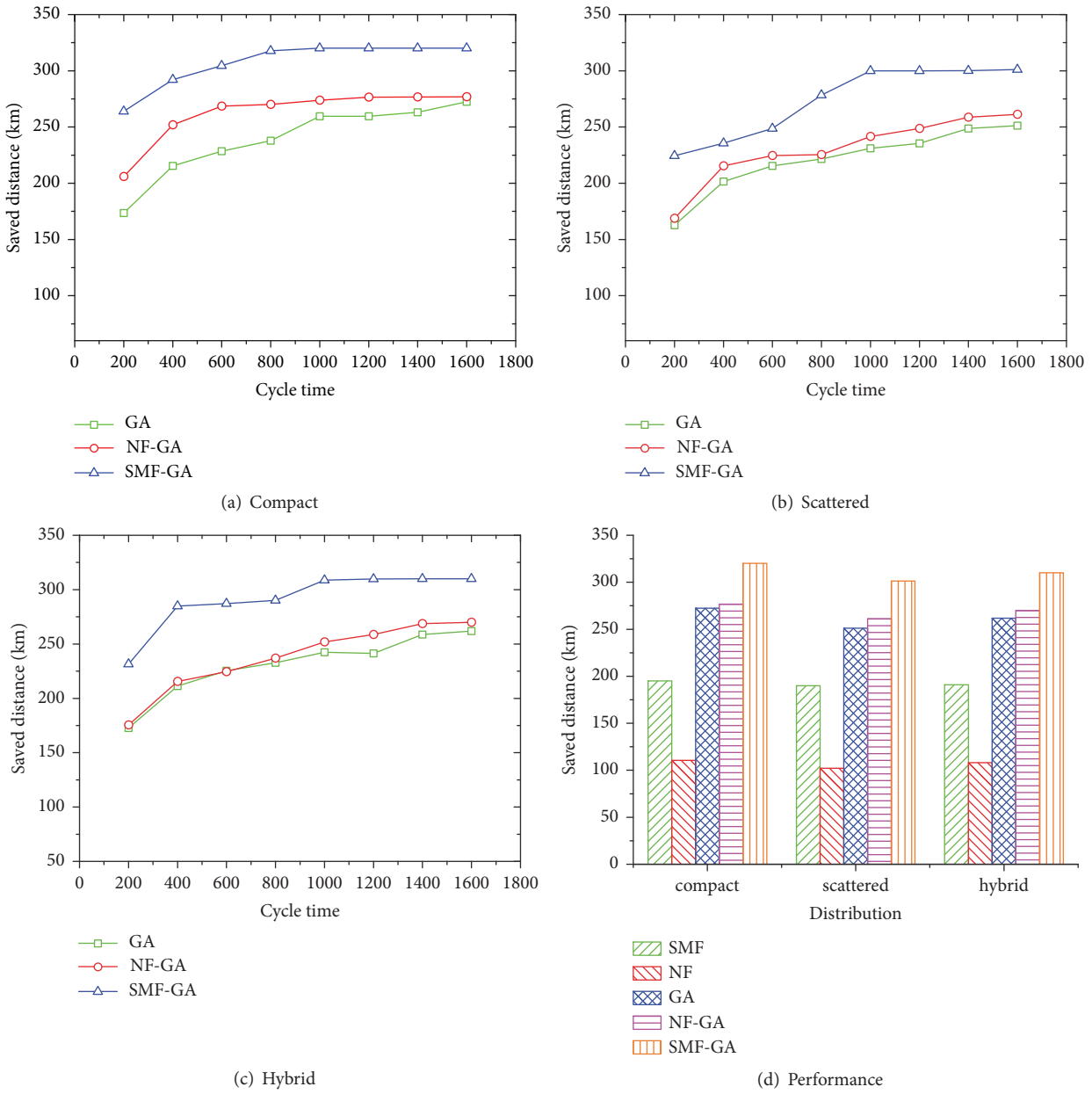
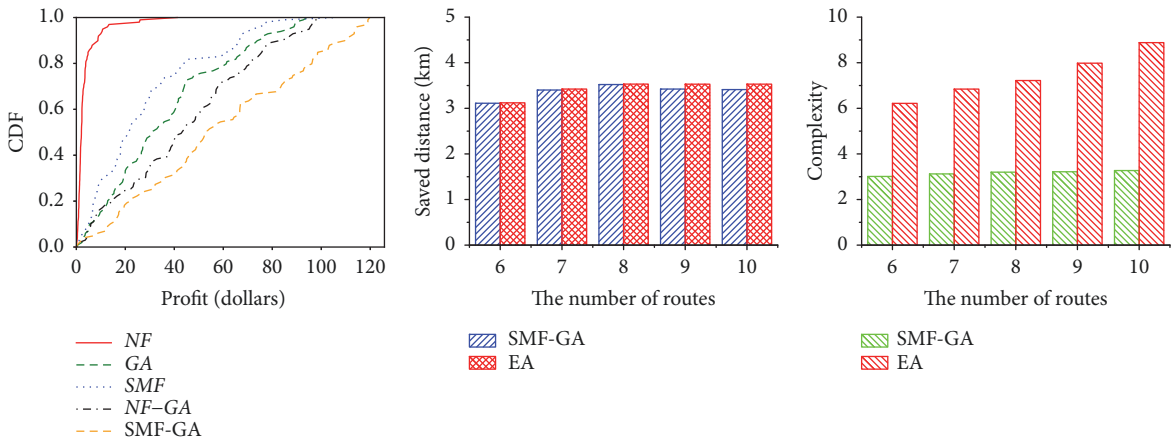


Figure 6: Saved distance under different distributions.

algorithm reach convergence depends on the scale of the system. To be specific, the solution space grows exponentially with the increase of tasks, so more cycle times are needed to search the global optimum from the whole solution space. Moreover, SMF-GA can reach to the convergence much more quickly than NF-GA and GA, and it can obtain a better result than the other two algorithms under different scenarios. For example, SMF-GA achieves convergence before 400 cycle times comparing with GA and NF-GA reaching convergence at about 1200 cycle times in Figure 5(a), and this is similar to three other scenarios. Finally, NF-GA and GA have similar performances which means that NF has little help for GA algorithm, and it is more obvious with the increase of the number of tasks.

Finally, the saved distance under different task distribution scenarios is studied. We run GA based algorithms with the different cycle times with using three task distributions: compact, scattered, and hybrid, and we also test SMF and NF with using these three task sets. The result is shown in Figure 6, and it implies that SMF-GA outperforms all other algorithms under different kinds of task distributions. Moreover, the performance under compact distribution is a little better than that under scattered and hybrid distribution. This may lead from that the distribution of taxis is not uniform, and more taxis are located in hot areas of the city. According to Figures 6(a), 6(b), and 6(c), SMF-GA can reach convergence more quickly than NF-GA and GA under different distributions. Also, SMF outperforms NF



(a) CDF plot of task executer's profit

(b) Saved distance between SMF-GA and EA

(c) Logarithmic complexity between SMF-GA and EA

Figure 7: Results of user enthusiasm and computational complexity.

with saving about two times of distance based on Figure 6(d) under different task distributions.

To conclude, SMF-GA outperforms other four algorithms including SMF, NF, GA, and NF-GA in saving distance under various scenarios. SMF-GA can reach the status of convergence with fewer cycle times and have a better result than other algorithms since the initial input of SMF-GA is much more close to the global optimum. In addition, the simple SMF can also do a good job at saving distance without any genetic algorithm.

4.3. Participant Profit. The adhesiveness to users is very important for the crowdsourcing based system since it can achieve the goal more easily with more participation by people. We suppose that the system will provide corresponding money to the task executor considering the task length, and the user would like to join the urban crowdsourced transportation system more if one can get more money during each journey. To be specific, it is assumed that the task requesters pay 4 dollars per kilometer and the actual cost of the task executor is 2 dollars per kilometer in this simulation. We calculate the earning money under different scenarios. Due to the similar results and limited space, we just show the CDF plot of profit earning by the task executor when $tasks/routes = 1.0$ as the scenario in the last subsection, and the result is shown in Figure 7(a).

It indicates that SMF can make user obtain more profits which leads to the fact that users would like use the system more often since SMF always chooses the local optimal task which can provide user more profit. However, NF is always choosing the nearest task which could have a different heading direction and cost more. SMF-GA is superior in performance to algorithms compared to profit since it overcomes the local optimum and searches the optimal solution given a better SMF input. In addition, GA and NF-GA have similar results since the poor result of NF does not help a lot.

4.4. Computational Complexity. We evaluate the performance of SMF-GA comparing with EA algorithm in this subsection, and a relative small scale data set is leveraged since the complexity of EA increases exponentially with the increase of the task and route number. We utilize a fixed task set which includes 4 tasks, and the small scale route sets contain 6-10 routes. The computational complexity of both algorithms under different route sets is shown in Figure 7(b); meanwhile, the performance of saving distance is shown in Figure 7(c).

It suggests that the EA algorithm can maximize the saved distance, but its computational complexity is too large growing much more quickly than SMF-GA with the increase of the number of routes and tasks. Even though SMF-GA cannot get the maximum saved distance, the result is very close to that of EA. The solution space grows exponentially with the increase of the number of routes and tasks. Nevertheless, the computation complexity of SMF-GA does increase much more slowly than the EA algorithm with the rising of routes and tasks. Also, the saved distance will increase with the rising of routes when the number of tasks is fixed since there could be more suitable routes for tasks to increase maximum saved distance. The evaluation shows that SMF-GA outperforms other algorithms, and the result is close to EA in which computational complexity is very high but the result is the optimal. In addition, it can work well under different distributions and reach to global optimum with fewer cycle times comparing with other GA based algorithms.

5. Related Work

We focus on the task allocation problem in urban crowdsourced transportation, which involves a large number of users in solving transport sharing problem inside the city. Many crowdsourcing based platforms have been successful in attracting the participation of millions of users. With the success of urban crowdsourced transportation systems (e.g., UberPOOL [22], Hitch [23]), a large amount of researchers

have been interested in developing new crowdsourcing based intelligence transport prototypes [24–27] in recent years, but few studies are done in task allocation of urban crowdsourced transportation.

By contrast, many task allocation algorithms [15, 28–37] have been proposed in mobile crowdsensing which is similar to urban crowdsourced transportation, and different task allocation algorithms are aimed at different optimization goals. Here, we highlight some most related work. For example, Wang et al. propose [15] CCS-TA framework combining the compressive sensing, Bayesian inference, and active learning techniques to choose a minimum number of subareas for sensing task allocation. Reddy et al. [32] discuss the issue that developing a recruitment framework to enable organizers to identify well-suited participants for data collections based on geographic and temporal availability and their habits. Han et al. [38] studied a participant recruitment problem where the initial set of recruited nodes needs to make an optimal decision on what other nodes to recruit to perform the crowdsourcing task. They also [39] study how to efficiently distribute a crowdsourcing task and recruit participants based on the future 5G networks with the feature of D2D communications. Restuccia et al. [40] propose a scalable and secure trust-based framework to assign the task in crowdsensing, which relies on the concept of mobile security agents and Josang's trust model to rule out incorrect reports and recruit users. Lane et al. [41] devise a system for collecting mobile sensor data from smartphones that lowers the energy overhead of user participation. Zhao et al. [42] devise an online incentive mechanism in mobile crowdsensing to recruit participants. He et al. [43] propose the allocation algorithm with considering the location dependence of tasks and devise an efficient local radio based algorithm (LRBA). Xiong et al. [18] define a novel spatial-temporal coverage metric and k-depth coverage, for mobile crowdsensing problems, and propose a near-optimal task allocation algorithm in piggyback crowdsensing.

The most similar work to ours in mobile crowdsensing is multitask allocation problem since one participant can do several tasks during his/her way to destination in urban crowdsourced transportation. We only find few work bout multitask participant selection algorithm. Song et al. [13] study selecting the most appropriate multiparticipants with considering different incentive requirements, associated capabilities, and uncontrollable mobility. Also, Guo et al. [11] devise ActiveCrowd which is a multitask allocation framework for mobile crowdsensing and study two different sensing environments: time-sensitive and delay-tolerant tasks.

We also find some related work in the field of transportation. Chen et al. [44] propose a two-phase framework called Crowddeliver for the package delivery leveraging the crowd of taxis, and it assigns packages to the corresponding taxis which can deliver them to the destination. SMF-GA differs from above related work in the following aspects. First, compared to MCS task allocation, the urban crowdsourced transportation can execute several tasks in parallel. In mobile crowdsensing, the participant usually fulfills corresponding tasks one by one; however, in urban mobile crowdsourcing, one can execute tasks together. For example, one can pick up several passengers at a time. On the top of it, no method is found considering calculating the distance between task and route since we need to go a period of path fulfilling the task. These two problems are main difficulties we have to overcome in urban crowdsourced transportation.

6. Conclusion

The multitask allocation problem in urban crowdsourced transportation is studied in this paper. We first formulate the problem and prove it is a combinational optimization problem which is NP-hard. Then, we propose a heuristic greedy algorithm called Saving Most First (SMF) which is simple and easy to assign tasks. The optimized SMF based genetic algorithm (SMF-GA) is devised to jump out of the local optimal result of allocation. The extensive simulation indicates that SMF can find a local optimal task allocation result and SMF-GA can obtain the global optimum with a quicker convergence than other GA based algorithms, and both SMF and SMF-GA are simple and effective. In the future, we will develop an urban crowdsourced transportation application and apply our proposed algorithm to verify the effectiveness and devise suitable incentive mechanisms to stimulate participants to be more active in the system.

Data Availability

T-taxi data were used to conduct the experiment. The prior dataset is cited at relevant places within the text as [19].

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

A preliminary version of this work appeared in proceedings of the 15th IEEE International Conference on Ubiquitous Intelligence and Computing (UIC 2018) [21]. The corresponding author of this manuscript is Ruiyun Yu. This work is supported by the Program for Liaoning Innovative Research Team in University (LT2016007), the National Natural Science Foundation of China (61672148, 61502092), the Fundamental Research Funds for the Central Universities (N171702001, N171604016), and the Ministry of Education, China Mobile Research Fund (MCM20160201).

References

[1] China 2016 Car Sales Surge at Fastest Rate in Three Years, 2017, http://www.scmp.com/business/china-business/article//china-2016-car-sales-surge-fastest-rate-three-years.

[2] Cities with the Worst Traffic in the Worlds, 2017, http://www.worldatlas.com/articles/cities-with-the-worst-traffic-in-the-world.html.

[3] Traffic Pollution Kills 5000 a Year in UK, 2012, http://www.scmp.com/business/china-business/article//china-2016-car-sales-surge-fastest-rate-three-years.

[4] Empty Running on the Rise, 2015, https://www.paragonrouting.com/paragon-blog/empty-running-rise.

[5] European Environment Agency, *Occupancy Rates*, 2008.

[6] J. Hamari, M. Sjöklint, and A. Ukkonen, "The sharing economy: why people participate in collaborative consumption," *Journal of the Association for Information Science and Technology*, vol. 67, no. 9, pp. 2047–2059, 2015.

[7] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[8] B. Guo, Z. Wang, Z. Yu et al., "Mobile crowd sensing and computing: the review of an emerging human-powered sensing paradigm," *ACM Computing Surveys*, vol. 48, no. 1, article 7, 2015.

[9] R. Liu, J. Cao, K. Zhang et al., "Understanding mobile users' privacy expectations: a recommendation-based method through crowdsourcing," *IEEE Transactions on Services Computing*, pp. 1-1, 2016.

[10] R. Yu, P. Wang, Z. Bai, and X. Wang, "Participatory sensing: people-centric smart sensing and computing," *Jisuanji Yanjiu yu Fazhan/Computer Research and Development*, vol. 54, no. 3, pp. 457–473, 2017.

[11] B. Guo, Y. Liu, W. Wu, Z. Yu, and Q. Han, "ActiveCrowd: a framework for optimized multitask allocation in mobile crowdsensing systems," *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 3, pp. 392–403, 2017.

[12] S. He, D. Shin, J. Zhang, and J. Chen, "Near-optimal allocation algorithms for location-dependent tasks in crowdsensing," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3392–3405, 2016.

[13] Z. Song, C. H. Liu, J. Wu, J. Ma, and W. Wang, "QoI-aware multitask-oriented dynamic participant selection with budget constraints," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 9, pp. 4618–4632, 2014.

[14] J. Wang, Y. Wang, D. Zhang et al., "Multi-task allocation in mobile crowd sensing with individual task quality assurance," *IEEE Transactions on Mobile Computing*, vol. 17, no. 9, pp. 2101–2113, 2018.

[15] L. Wang, D. Zhang, A. Pathak et al., "CCS-TA: Quality-guaranteed online task allocation in compressive crowdsensing," in *Proceedings of the 3rd ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp 2015*, pp. 683–694, Japan, September 2015.

[16] L. Wang, D. Zhang, D. Yang et al., "SPACE-TA: Cost-effective task allocation exploiting intradata and interdata correlations in sparse crowdsensing," *ACM Transactions on Intelligent Systems and Technology*, vol. 9, no. 2, 2017.

[17] Z. Wang, J. Hu, R. Lv et al., "Personalized privacy-preserving task allocation for mobile crowdsensing," *IEEE Transactions on Mobile Computing*, 2018.

[18] H. Xiong, D. Zhang, G. Chen, L. Wang, V. Gauthier, and L. E. Barnes, "ICrowd: near-optimal task allocation for piggyback crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 2010–2022, 2016.

[19] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11)*, pp. 316–324, August 2011.

[20] S. Ma, Y. Zheng, and O. Wolfson, "Real-time city-scale taxi ridesharing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 7, pp. 1782–1795, 2015.

[21] P. Wang and R. Yu, "Catalyze sharing economy: optimized multi-task allocation for urban transport crowdsourcing," in *Proceedings of the 15th IEEE International Conference on Ubiquitous Intelligence and Computing*, pp. 1118–1123, Guangzhou, China, October 2018.

[22] UberPOOL, 2017, https://www.uber.com/ride/uberpool/.

[23] Hitch, 2017, http://www.hitchit.co/.

[24] H. Lv, F. Wu, T. Luo, X. Gao, and G. Chen, "Achieving location truthfulness in rebalancing supply-demand distribution for bike sharing," in *Algorithmic Aspects in Information and Management*, vol. 11343 of *Lecture Notes in Computer Science*, pp. 256–267, Springer International Publishing, Cham, Switzerland, 2018.

[25] J.-F. Rougés and B. Montreuil, "Crowdsourcing delivery: new interconnected business models to reinvent delivery," in *Proceedings of the 1st International Physical Internet Conference*, p. 19, 2014.

[26] X. Wang, X. Zheng, Q. Zhang, T. Wang, and D. Shen, "Crowdsourcing in ITS: the state of the work and the networking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 6, pp. 1596–1605, 2016.

[27] D. Zhang, Y. Li, F. Zhang, M. Lu, Y. Liu, and T. He, "CoRide: carpool service with a win-win fare model for large-scale taxicab networks," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems, SenSys 2013*, Italy, November 2013.

[28] A. Chakeri and L. G. Jaimes, "An incentive mechanism for crowdsensing markets with multiple crowdsourcers," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 708–715, 2018.

[29] G. Han, L. Liu, S. Chan, R. Yu, and Y. Yang, "HySense: a hybrid mobile crowd sensing framework for sensing opportunities compensation under dynamic coverage constraint," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 93–99, 2017.

[30] A. Khan, S. K. A. Imon, and S. K. Das, "A novel localization and coverage framework for real-time participatory urban monitoring," *Pervasive and Mobile Computing*, vol. 23, pp. 122–138, 2015.

[31] T. Luo, S. S. Kanhere, J. Huang, S. K. Das, and F. Wu, "Sustainable incentives for mobile crowdsensing: Auctions, lotteries, and trust and reputation systems," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 68–74, 2017.

[32] S. Reddy, D. Estrin, and M. Srivastava, "Recruitment framework for participatory sensing data collections," in *Pervasive Computing*, vol. 6030 of *Lecture Notes in Computer Science*, pp. 138–155, Springer, Berlin, Germany, 2010.

[33] F. Restuccia, P. Ferraro, T. S. Sanders, S. Silvestri, S. K. Das, and G. L. Re, "First: a framework for optimizing information quality in mobile crowdsensing systems," *ACM Transactions on Sensor Networks*, vol. 15, no. 1, pp. 1–35, 2019.

[34] F. Restuccia, P. Ferraro, S. Silvestri, S. K. Das, and G. Lo Re, "IncentMe: effective mechanism design to stimulate crowdsensing participants with uncertain mobility," *IEEE Transactions on Mobile Computing*, 2018.

[35] F. Restuccia, N. Ghosh, S. Bhattacharjee, S. K. Das, and T. Melodia, "Quality of information in mobile crowdsensing: survey and research challenges," *ACM Transactions on Sensor Networks*, vol. 13, no. 4, 2017.

[36] J. Wang, J. Tang, D. Yang, E. Wang, and G. Xue, "Quality-aware and fine-grained incentive mechanisms for mobile crowdsensing," in *Proceedings of the 36th IEEE International Conference on Distributed Computing Systems, ICDCS 2016*, pp. 354–363, Japan, June 2016.

[37] S. Yang, U. Adeel, and J. McCann, "Backpressure meets taxes: faithful data collection in stochastic mobile phone sensing systems," in *Proceedings of the 34th IEEE Annual Conference on Computer Communications and Networks, IEEE INFOCOM 2015*, pp. 1490–1498, Hong Kong, May 2015.

[38] Y. Han, T. Luo, D. Li, and H. Wu, "Competition-based participant recruitment for delay-sensitive crowdsourcing applications in D2D networks," *IEEE Transactions on Mobile Computing*, vol. 15, no. 12, pp. 2987–2999, 2016.

[39] Y. Han and H. Wu, "Minimum-cost crowdsourcing with coverage guarantee in mobile opportunistic D2D networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 10, pp. 2806–2818, 2017.

[40] F. Restuccia and S. K. Das, "FIDES: a trust-based framework for secure user incentivization in participatory sensing," in *Proceedings of the 15th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2014*, Australia, 2014.

[41] N. D. Lane, Y. Chon, L. Zhou et al., "Piggyback CrowdSensing (PCS): energy efficient crowdsourcing of mobile sensor data by exploiting smartphone app opportunities," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems, SenSys 2013*, Italy, November 2013.

[42] D. Zhao, X.-Y. Li, and H. Ma, "How to crowdsource tasks truthfully without sacrificing utility: Online incentive mechanisms with budget constraint," in *Proceedings of the 33rd IEEE Conference on Computer Communications, IEEE INFOCOM 2014*, pp. 1213–1221, May 2014.

[43] S. He, D.-H. Shin, J. Zhang, and J. Chen, "Toward optimal allocation of location dependent tasks in crowdsensing," in *Proceedings of the 33rd IEEE Conference on Computer Communications (INFOCOM '14)*, pp. 745–753, IEEE, Toronto, Canada, May 2014.

[44] C. Chen, D. Zhang, X. Ma et al., "CROWDDELIVER: planning city-wide package delivery paths leveraging the crowd of taxis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1478–1496, 2017.