One-Shot Global Optimization of Ridesharing Using A Set-Based Differential Evolution Algorithm

Abstract

The rapid proliferation of private cars brings the problems of traffic congestion, air pollution, etc. Ridesharing provides an efficient way to solve the above issues by allocating riders to drivers with similar itineraries. The existing methods typically perform ridesharing scheduling in two phases to match rider-rider pairs and rider-driver pairs separately, while restricting one driver to serve no more than two riders. The two-phase optimization mechanism limits the general performance of the ridesharing system. To address the issue, this paper proposes a one-shot global optimization algorithm for ridesharing. First, we formulate a ridesharing model and encode the solutions based on sets. The model is comprehensive, which not only poses less restrictions than the literature does but also considers more objectives including service quality and cost. Then, we develop a set-based differential evolution algorithm to search the global optimum for the comprehensive model. From the algorithm aspect, we design new operators, such as the greedy initialization, the inter-vehicle mutation, and the route-sensitive selection, to enhance the performance of differential evolution for dealing with this specific ridesharing problem. The experiment results show that our method outperforms state-of-the-art methods on metropolitan transport datasets.

Introduction

The excessive use of cars in urban areas has brought many serious issues such as environmental pollution, traffic congestion, and gasoline shortage. Research shows that 78% American people commute alone and that each vehicle takes 1.5 people on average, which causes a huge waste of transportation resources (Dakroub et al. 2013). Ridesharing brings together people with similar travel schedules to increase the vehicle occupancy, which inherently solves the above-mentioned issues (Dakroub et al. 2013; Chan and Shaheen 2012; Cookson and Pishue 2017). In addition, ridesharing provides an economical and convenient transport way for people. With these considerations, ridesharing is worth application and popularization (Furuhata et al. 2013). Currently, many car-hailing applications like Uber and Didi provide ridesharing service for users. Many countries such as America even build high-occupancy vehicle lanes that provide a travel time advantage for carpool to encourage ridesharing.

The ridesharing problem can be different according to the working scenarios. Accordingly, different methods are utilized to meet the various ridesharing demands (Agatz et al. 2010; 2012; Amey, Attanucci, and Mishalani 2011; Berbeglia et al. 2007). This study focuses on the commuting and long journey ridesharing, which are the most common ridesharing model (Agatz et al. 2010). This kind of ridesharing system has two types of users: drivers and riders, who will designate their trip information including the starting point, the destination, as well as the number of seats they would provide/take. Integrating the above information, the server is desired to provide a ridesharing scheme that assigns one or multiple riders to each driver and plans the route for each driver to deliver the assigned riders.

In the literature, the ridesharing problem is commonly solved by greedy algorithms (Zheng, Chen, and Ye 2018; Zheng, Cheng, and Chen 2019; Coltin and Veloso 2014; Agatz et al. 2011). Generally, these algorithms have a few limitations. First, they usually follow a two-phase mechanism that matches similar riders into a package in the first phase and then assigns the packages to the drivers in the second phase. However, the method ignores the dependency between the two tasks, which reduces the overall performance of the ridesharing system. Second, when packing the riders, the algorithms make an assumption that no more than two riders can be packed together, which may be inefficient. Third, the existing works commonly intends to minimize the costs or maximize the profit from the perspective of the car-hailing company, which neglects the user experience, i.e., the quality of services (QoS). Last but also important, the greedy algorithms only provide local optimal solutions, which may endure low performance when the problem landscape contains inferior local optima.

This paper develops a novel ridesharing algorithm to address the above-mentioned issues. The algorithm utilizes a set-based method to allocate riders to drivers in a one-shot manner, which is different from the previous two-phase method. Meanwhile, it allows not only two riders to share

drivers as long as there are available seats in the vehicles, which helps to maximize the utilization rate of traffic resources. Considering the optimization objectives, our model considers both QoS and the travel cost, which is more comprehensive. Finally, a global optimization algorithm, the differential evolution (DE), is applied to tackle the optimization task (Storn and Price 1997). Nevertheless, noting that we formulate the problem model by set operation, the traditional DE is unsuitable for this set-based problem. For this consideration, we improve a set-based DE algorithm to accomplish the proposed one-shot global optimization task, which results in a DE-based ridesharing algorithm (DERA). Different from the traditional DE, DERA is characterized by its set-based evolution operation, greedy initialization, inter-vehicle mutation, route-sensitive selection , and individual repairing and refactoring. The experiments show that the proposed DERA outperforms the state-of-the-art.

Related Work

Ridesharing is an efficient way to increase the utilization rate of transportation facilities, which has received significant attention in recent years. In the literature, various methods have been developed to deal with this issue, many are based on spatial search or heuristic algorithms.

The spatial search contributes to the taxi dispatch system and also the ridesharing system. A partition-based algorithm is proposed in (Pelzer et al. 2015), which divides the road network into distinct partitions to restrain the searching area. Xhare-A-Ride is a peer-to-peer ridesharing system that utilizes a grid search of hierarchical regions to optimize the match schedule (Thangaraj et al. 2017). Besides, Grabshare searches a nearby area and finds the best solution among several candidate solutions (Tang et al. 2017). Some studies explore the additional information like social relationship and historical trajectory to improve the ridesharing quality. Vshare uses wireless social network to improve the allocation performance (Lin and Shen 2016). (Bistaffa, Farinelli, and Ramchurn 2015) propose an extended graph constrained formation algorithm to tackle social ridesharing problem. Later, (Hasan et al. 2018) propose trip cluster and share-ability graph, and utilize mixed-integer programming to provide an optimal solution. (Bakkal et al. 2017) model the historical trajectories of drivers and do temporal and locational filtering.

The heuristic-based algorithms, which are popular in optimization problems, also take an important position in ridesharing. The greedy algorithms are one branch. In (Bei and Zhang 2018), a rule-based greedy matching algorithm is proposed to do ride-share match which performs well on Atlanta metropolitan benchmarks. (Zheng, Chen, and Ye 2018) develop an order price oriented greedy algorithm and some matching-based methods to maximize the platform's profit and they propose an auction based greedy method for the situation of vehicle shortage in the next year (Zheng, Cheng, and Chen 2019). (Agatz et al. 2011) propose a greedy algorithm which solution is at most 2.5 times the optimal cost for ridesharing problem that constraints each driver is assigned to two riders. Compared with the greedy algorithms, the swarm and evolutionary computation methods are able to search the globally optimal solutions, which are more robust to ridesharing. Herbawi and Weber (Herbawi and Weber 2012) put forward a generational genetic algorithm to work out the matching scheme with a time window in dynamic ridesharing. (Huang, Jiau, and Chong 2017) present a multi-objective ridesharing algorithm with non-domination genetic algorithm and heuristic operator. The ant colony optimization algorithm is also applied to the ridesharing problem considering time window in (Huang, Jiau, and Liu 2018). (Chou, Jiau, and Huang 2016) propose a stochastic particle swarm optimization algorithm to provide carpool service.

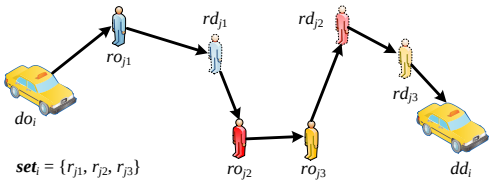


Figure 1: A driver's route traversing a sequence of locations.

Ridesharing System Model

In this section, we mathematically formulate the ridesharing problem. We consider a set of drivers $D = \{d_1, d_2, ..., d_n\}$ and a set of riders $R = \{r_1, r_2, ..., r_m\}$, where n denotes the number of drivers and m denotes the number of riders. Each driver d_i is specified by a tuple (do_i, dd_i, dc_i), where do_i denotes the initial position, dd_i denotes the destination, and dc_j denotes the number of available seats. Similarly, each rider r_j is associated with a tuple (ro_j, rd_j, rc_j), where ro_j denotes the initial position, rd_j denotes the destination, and rc_j denotes the number of requested seats.

The ridesharing schedule consists of rider allocation to each driver and the corresponding route planning. It is required that each driver should pick up the assigned riders and deliver them to their destinations, and in this way the riders need not transfer to another vehicle. We use set_i to denote the set of riders assigned to driver d_i. The route of d_i is then determined by the initial and destination location sequence of d_i and the related locations of the riders in set_i. We utilize the dynamic programming (Bellman 1958) to obtain this shortest feasible route. Note that the rider should not get on the vehicle if there are not enough vacant seats for him. Figure 1 shows an example of the route of d_i, where we assume that $set_i = \{rj_1, rj_2, rj_3\}$.

Usually, a driver has a tolerable detour distance when participating in ridesharing. We use a penalty term dp_i to reflect this requirement:

$$dp_i = \begin{cases} 1, & ddist(d_i) > dthres(d_i) \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where $ddist(d_i)$ is the driving distance for a ridesharing allocation, and $dthres(d_i)$ is the maximum tolerable driving distance of d_i. Similarly, riders have their tolerable waiting

time and detour distance for ridesharing. To reflect this requirement, we define a penalty term rp_j as:

$$rp_j = \begin{cases} 1, & rdist(r_j) > rthres(r_j) \\ 0, & (otherwise) \end{cases} \quad (2)$$

where $rdist(r_j)$ denotes the distance from the driver's initial position to the destination of rider r_j, $rthres(r_j)$ is the maximum tolerable distance of r_j for sharing a ride. Note that $rdist(r_j)$ consists of two parts, the traveling distance from the initial location of the driver to the initial location of r_j (reflecting the waiting time) and the traveling distance from the initial location of r_j to his destination (reflecting the detour distance for ridesharing). For example, in Figure 1, the returned value of $rdist(r_{j2})$ is the route distance between do_i and rd_{j2}, with the route distance between do_i and ro_{j2} being the waiting distance of r_{j2} and the route distance between ro_{j2} and rd_{j2} being the riding distance of r_{j2}.

With the above illustration, we are ready to formulate the objective functions for ridesharing. The primary objective of ridesharing in this paper is to maximize the quality of service (QoS) of the participating drivers.

Definition 1 (QoS)**.** *The QoS of driver d_i is defined by the following function:*

$$QoS_i = \sum_{r_j \in set_i} (rc_j - rp_j) - dp_i \quad (3)$$

where set_i is the rider set that includes all the riders assigned to driver d_i; rc_j denotes the seat occupancy, i.e., the number of seats r_j reserves; rp_j and dp_i are the penalty items to adjust the influence of travel distance.

Note that the quality of ridesharing service in Eq. (3) concludes the user's satisfaction as well as the efficiency of the carpool. The user's satisfaction is correlated to whether the detour distance is out of tolerance and the efficiency of the carpool is correlated to whether the seat occupancy is high.

The secondary objective of ridesharing is to minimize the costs of the planned routes. Generally, the costs are caused by the detour distance of riders and drivers.

Definition 2 (Cost)**.** *The cost of driver d_i for delivering the assigned rider set set_i in ridesharing is defined as:*

$$Cost_i = \sum_{r_j \in set_i} rdist(r_j) + ddist(d_i) \quad (4)$$

where $rdist(r_j)$ and $ddist(d_i)$ are the penalty terms caused by detour.

For clearer expression, we define $f(set_i) = (QoS_i, Cost_i)$ as the composite objective function that returns QoS_i and $Cost_i$ of set_i. To compare two solutions with respect to the composite objective, we define the partial order as follows.

Definition 3 (Partial Order)**.** *Given two rider sets set_i and set'_i for a given driver d_i, set_i is better than set'_i, denoted as $f(set_i) \succ f(set'_i)$, if set_i has a larger QoS value than set'_i or set_i has the same QoS value but a smaller Cost value compared with set'_i.*

Finally, we formulate the ridesharing problem.

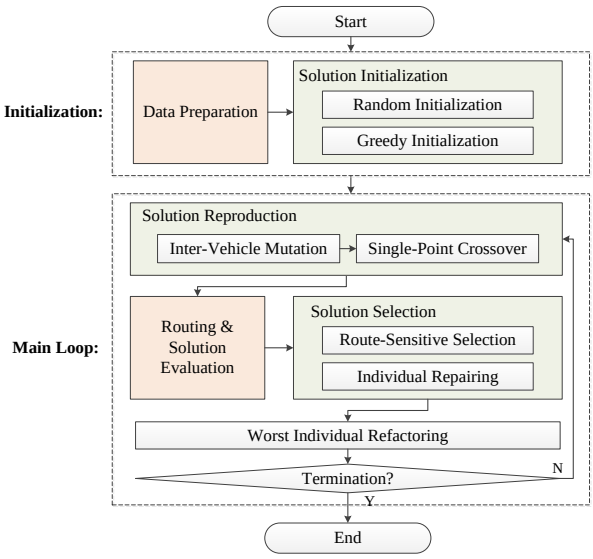


Figure 2: The flowchart of DERA.

Definition 4 (Ridesharing Problem)**.** *The ridesharing problem is a combinational optimization problem that maximizes the composite objective function:*

$$\max \sum_{d_i \in D} f(set_i) \quad (5)$$

$$s.t. \sum_{r_j \in R} x_j^i \leq RS, \qquad i = 1, ..., n \quad (6)$$

$$\sum_{d_i \in D} x_j^i \leq 1, \qquad j = 1, ..., m \quad (7)$$

$$\forall j, x_j^i \times rc_j \leq dc_i, \qquad i = 1, ..., n \quad (8)$$

where x_j^i is a binary variable denoting whether rider r_j is assigned to driver d_i:

$$x_j^i = \begin{cases} 1, & r_j \in set_i \\ 0, & otherwise \end{cases} \quad (9)$$

In Eq. (6), RS is the load restriction that defines the maximum size of a rider set. Each driver is limited to take no more than RS riders, such that his route will not be too long. The constraint in Eq. (7) illustrates that one rider can be allocated to at most one driver. Eq. (8) ensures that a driver will not take riders who request more seats than he can provide.

Differential Evolution Based Ridesharing Algorithm

In this section, we propose a differential evolution based ridesharing algorithm (DERA) to solve the ridesharing problem modeled in the previous section. Developed from the set-based DE (Liu et al. 2013), DERA employs a set to simulate the allocation scheme of each driver, applies set-based

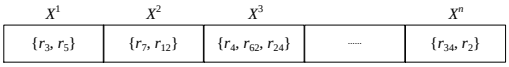


Figure 3: Individual representation.

operations to realize the reassignment of riders and iteratively eliminates improper assignment to optimize the fitness function, i.e., the composite function defined in Eq. (5).

The main process of DERA is sketched in Figure 2. In each iteration, the algorithm updates individuals, i.e., the seat allocation solutions through inter-vehicle mutation, single-point crossover, route-sensitive selection, individual repairing, and the refactoring of the worst individuals. Each time after determining the new rider-to-driver assignments, the routes of each individual are automatically planned by dynamic programming for the shortest driving distance. Then, according to the ridesharing schedules (including the assignments of riders to drivers and the routes of drivers), the solutions are evaluated by the fitness function. Through the evolution of the population which consists a number of individuals at each iteration, the ridesharing results can be optimized. In the following, we describe the details of the aforementioned operations.

Individual Representation and Evaluation

Figure 3 shows the design of an individual, which is an n-dimensional vector. We encode the driver d_i as the i-th dimension and encode the rider set set_i as the value of the i-th dimension. In this manner, each individual represents an allocation scheme that assigns riders to drivers. For simplicity, we define a population P with PS individuals. Each individual $X \in P$ is denoted by $X = \{X^1, X^2, ..., X^n\}$, where X^i denotes the i-th dimension and it is the allocation scheme of driver d_i.

After forming the population, we utilize function $f(set_i)$ to evaluate the fitness of each dimension d_i. The overall fitness of an individual is simply the summation of the fitness values in all dimensions.

Population Initialization

We use both random and greedy methods to generate the initial population. Random assignment, which generates individuals distributed uniformly in the feasible region, is a common method for population initialization since it tends to investigate the whole feasible region. On the other hand, the greedy initialization method makes the population evolve from a better initial point and it is widely used to improve the quality of the solution. Specifically, to initialize the population, we first allocate riders to the drivers in random order until the rider sets of all the drivers are fully matched or all riders are assigned. Then we randomly get one individual in the whole population generated by the greedy method.

We develop our greedy method by assigning each driver a new rider iteratively until the drivers are fully matched or the fitness value cannot be improved by adding a new rider. A total number of RS iterations are needed according to Eq. (6). In each iteration, the assignment order of unassigned riders is determined by the fitness of the pairwise matching between the driver and the rider set. In other words, we sort the driver-rider pair candidates in descending order according to the fitness values. Then we examine the feasibility of adding the driver-rider pair in order and make assignment decision accordingly. It should be noticed that the assignment at each iteration is fixed, which means that the assigned rider will never be reassigned to another driver during iterations.

Inter-Vehicle Mutation

In traditional DE mutation, the operations on the individual's different dimensions are independent. Hence dimensions cannot share information. However, this mechanism is inefficient for solving the ridesharing problem since the riders are not limited to specific drivers (i.e., dimensions). Instead, it is suggested to share information among different drivers so as to identify the best allocation of riders. Therefore, we propose a set-based mutation operator to realize the interactions among different vehicles. Specifically, the mutant individual can be generated according to the following formulation based on the rules in Definition 5:

$$V^i = X^i_{r1} + F \times (X^i_{r2} - X^i_{r3}) + S_{i,k} \times (X^k_{r4} - X^k_{r5}) \quad (10)$$

where V denotes the mutant individual; X_{r1}, X_{r2}, X_{r3}, X_{r4}, and X_{r5} are five different individuals that are randomly selected in the population; F is a differential weight relating to the differential set of X_{r2} and X_{r3} in i-th dimension; $S_{i,k}$ is a weight that measures the similarities of i-th dimension and k-th dimension relating to the differential set of X_{r4} and X_{r5}, where k is a random integer from 1 to n.

Definition 5 (Set Operations)**.** *Given two sets A and B, and a scalar $c \in (0, 1)$, we have:*

- *Rule 1: The difference of A and B, denoted by $A - B$, is a set containing all the elements in A but not in B.*
- *Rule 2: The aggregation of A and B, denoted by $A + B$, is a set containing all the elements in A or in B.*
- *Rule 3: The product of A and c, denoted by $c \times A$, is a set whose elements are randomly selected in set A with possibility c.*

It can be noticed from Eq. (10) that the mutation allows the communication between dimensions i and k, which is different from the traditional DE mutation. The inter-dimension communication is weighted by the similarity factor $S_{i,k}$. For each driver, the communication with a similar dimension should get a higher weight than communication with a dissimilar dimension. Namely, the more similar the two drivers, the more likely they are going to share the riders, which accords with the practice. Here we utilize the chi-square test (Pearson 1900) to calculate the similarities between two different dimensions. The properties of i-th dimension are represented by an m-dimension vector dr^i, which is calculated according to

$$dr^i_j = \sum_{X \in P} x^i_j \quad (11)$$

According to the properties, we then calculate the chi-square distance between dr^i and dr^k and then obtain the similarities of i-th dimension and k-th dimension by table look-ups.

Single-Point Crossover

We perform the single-point crossover according to Eq. (12). It operates on a mutant individual V and a parent individual X with the crossover constant CR and a random crossover point i_{rand}. Function $rand(0, 1)$ generates a random number between 0 and 1. The crossover here generates the offspring U with some random dimensions from the mutant individual V and the other dimensions from the parent individual X. On doing this, we realize the individual recombination.

$$U^i = \begin{cases} V^i, & \text{if } i = i_{rand} \text{ or } rand(0,1) < CR \\ X^i, & \text{otherwise} \end{cases} \quad (12)$$

Route-Sensitive Selection

The traditional DE performs individual-wise selection, which is not comprehensive for the ridesharing problem. Each dimension of the individual for ridesharing is a relatively independent sub-scheme of the whole allocation scheme that can be evaluated separately. Moreover, the performance of the dimensions generated by mutation and crossover has randomness, and most of the times, the mutually exclusive selection of individuals misses some superior dimensions of the unselected individuals. Therefore, we propose a route-sensitive selection operator according to Eq. (13), which conducts dimension-wise selection (i.e., the route selection of each driver) firstly, and then performs selection between individuals according to Eq. (14). In this way, the operator considers dimension-wise comparisons of the routes to improve the selection result. We notice that a better dimension allocation does not always correspond to a better individual allocation, and hence we define the competence probability CP that denotes the possibility of competitive dimension selection according to Eq. (13). On the other hand, with probability $(1 - CP)$, the random dimension selection is applied. Offspring U' is the result of the dimension-wise selection, and then the individual-wise selection is performed to reserve the optimal individual between U' and X.

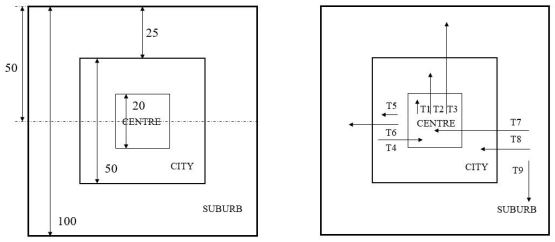
$$U'^i = \begin{cases} U^i, & \text{if } f(U^i) > f(X^i) \\ X^i, & \text{otherwise} \end{cases} \quad (13)$$

$$X = \begin{cases} U', & \text{if } f(U') > f(X) \\ X, & \text{otherwise} \end{cases} \quad (14)$$

Individual Repair

Particularly, the mutation operator may insert some illegal riders that request more seats than the driver provides into the rider set, which violates the constraint in Eq. (8). We can simply remove the illegal riders to repair this issue. In the mutation, crossover, and dimension-wise selection operators, it may appear that one rider is allocated to two or more drivers, which violates the constraint in Eq. (7). We utilize a heuristic function to avoid this issue. The heuristic function $h(r_j, d_i)$ is defined in Eq. (15), which returns the driving distance after assigning rider r_j to driver d_i. The driver with the minimum return value of this heuristic function is more likely to keep the rider as his cost of delivering the rider is the least among all competitors. Considering that the driver with the least cost is sometimes not the best driver for the rider, we define the following operation: with probability CP the heuristic repair strategy is applied, and with probability $1 - CP$ the rider is assigned to a random driver.

$$h(r_j, d_i) = dist(do_i, ro_j) + dist(ro_j, rd_j) + dist(rd_j, dd_i) \quad (15)$$



(a) City partition model (b) City transition model

Figure 4: City model.

Individual Refactoring

In the process of population evolution, it often happens that the population converges too early or is trapped in a local optimum. Therefore, we design an individual refactoring strategy to enhance the population diversity. Specifically, the worst evaluated individuals in each iteration are chosen and regenerated with the random initialization method. In this way, every individual and every dimension of the individual can communicate with the entire rider set R. With the progress of evolution, the generation of new riders helps the algorithm to explore new search horizon and jump out of the local optimal.

Experimental Evaluation

Dataset for the Ridesharing Problem

The dataset is generated according to the statistical data of the actual environment in Taipei gathered by Taipei City Government Department of Transportation (Chou, Jiau, and Huang 2016). The statistical data are summarized in Table 1, which divides people's trip into three movement models, CI, CL, CO, and nine transition categories, T1 to T9, according to the city partition of the center zone, city zone, and suburb zone (as shown in Figure 4). The nine transitions differ from each other considering the transition origin zones and destination zones: T1, T2, and T3 denote the movements from the center zone, T4, T5, and T6 denote the movements from the city zone, T7, T8, and T9 denote the movements from the suburb zone. The three movement models simulate different travel behaviors:

- CI simulates the movement of riding or driving to the office or school. As school and office buildings are mostly located in the city center, this movement model favors transition T4 and T7.
- CL simulates the movement of delivery men. This model is focused on lateral movement due to the delivering distance limitation.

Table 1: Metropolitan data statistics. ***NO.*** denotes the number of drivers or riders joining ridesharing, ***Average*** and ***STD*** denote the average number of seats offered by drivers or requested by riders and the corresponding variance, respectively.

Model	Transition Proportion(%)									Driver			Rider		
	T1	*T2*	*T3*	*T4*	*T5*	*T6*	*T7*	*T8*	*T9*	*No.*	*Average*	*STD*	*No.*	*Average*	*STD*
CI102V1	29	1	2	24	11	7	16	7	3	36	3.02	1.58	66	2.34	1.23
CI303V2	25	1	1	26	15	4	18	8	2	100	2.98	1.69	203	1.92	1.36
CI108F1	26	1	1	23	15	6	19	6	3	41	2.95	0.56	67	1.14	0.54
CI301F2	24	2	1	27	17	7	14	6	2	107	3.08	0.62	194	1.25	0.44
CL107V1	54	4	3	1	24	6	1	4	3	38	2.83	1.48	69	2.27	1.31
CL306V2	60	3	1	3	17	5	2	5	4	111	2.87	1.57	195	2.56	1.49
CL106F1	58	4	2	2	21	6	2	3	2	42	3.12	0.38	64	1.12	0.59
CL302F2	57	3	3	1	20	5	3	4	4	103	3.03	0.65	199	1.17	0.45
CO104V1	25	20	22	3	12	7	2	7	2	32	3.09	1.61	72	1.83	1.2
CO309V2	23	23	20	3	13	7	2	6	3	108	3.26	1.72	201	2.15	1.28
CO104F1	25	26	19	2	14	6	2	4	2	38	3.02	0.45	66	1.08	0.55
CO303F2	22	27	21	1	12	8	1	6	2	104	2.96	0.4	199	1.15	0.58

Table 2: QoS performance of DERAs.

Model	Upperbound	DERA-R		DERA-G	
	QoS	QoS	RT	QoS	RT
CI102V1	165	151.3	0.92	**152.6**	**0.92**
CI303V2	418	384.1	0.92	**387.3**	**0.93**
CI108F1	80	75.1	0.94	**76.8**	**0.96**
CI301F2	236	223.7	0.95	**229.1**	**0.97**
CL107V1	154	143.1	0.93	**144**	**0.94**
CL306V2	479	434.5	0.91	**439.4**	**0.92**
CL106F1	80	77.1	0.96	**77.6**	**0.97**
CL302F2	246	233.9	0.95	**237.5**	**0.97**
CO104V1	142	123.1	0.87	**124.4**	**0.88**
CO309V2	459	420.7	0.92	**425.6**	**0.93**
CO104F1	80	71	0.89	**72**	**0.9**
CO303F2	252	237.8	0.94	**242.6**	**0.96**

- CO simulates the movement of going home after work or after school. In this model, the proportion of T2 and T3 is more than other transitions as the residential areas are mostly in the city or suburb zone.

Furthermore, each movement model is refined by the seat conditions and the traffic flow conditions. For example, the model "CI102V1" in Table 1 means that 102 users are collected for the CI model, "V" denotes a large variance of provided and request seat numbers, "1" denotes the data are collected in non-peak hours. As comparison, the "CI301F2" means that 301 users are collected for the CI model, "F" denotes a small variance of provided and request seat numbers, "2" denotes the data are collected in peak hours. There are totally 12 such models listed in Table 1. We use these models to generate the test instances of civic ridesharing and test the performance of the algorithms on these instances.

Experimental Settings

In the experiment, we implement two versions of DERA, i.e., the version with random initialization (termed as DERA-R) and the version with greedy initialization (termed as DERA-G). The compared algorithms include the particle swarm optimization algorithm PSCSA proposed in (Chou, Jiau, and Huang 2016), the genetic algorithm GCRMA proposed in (Huang, Jiau, and Lin 2014), the hill-climbing algorithm (HC) (Russell and Norvig 2016), and the greedy method. All the tested algorithms share the same maximum size of rider set $RS = 4$. The evolutionary computation-based algorithms adopt the same population size $PS = 50$ and they all terminate at 1500 iteration. The parameters in DERA are as follows: $F = 0.5$, $CR = 0.8$, $CP = 0.9$. In PSCSA, the accelerating coefficient is set as 2 and the inertia weight is set as 0.9. Besides, PSCSA does a local search among $\tau = 5$ candidate solutions on dimension to get better performance. For GCRMA, the crossover and mutation possibilities are set as 0.9 and 0.5, respectively. The selection of GCRMA remains the best $es = 5$ chromosomes and utilizes the tournament selection strategy to decide the other chromosomes that would be reserved to do crossover. The HC performs the optimization by iteratively doing local search on each dimension. The greedy method is introduced in the section of population initialization. We independently run the algorithms for 20 epochs and obtain the average results of these epochs to avoid casual results.

Performance on QoS

Here, we examine the performance of DERA-R and DERA-G on QoS. As QoS is the primary objective, the better performance of QoS denotes the better performance on fitness. The upper bound of QoS is calculated according to Eq. (16). In Table 2, we listed the QoS performance of DERAs on the datasets of different models and the ratio of our obtained QoS value to the upperbound QoS_{upper} (denoted as RT in the table). It can be seen that DERA-R and DERA-G find the near optimal solution comparing to the upperbound. Specifically, the ratio of DERA-G is larger than 0.9 on 11 models out of the 12 models. For the models with small variance of provided and requested seat numbers (i.e., the models with "F" in the name sequence), the ratio of DERA-G is close to

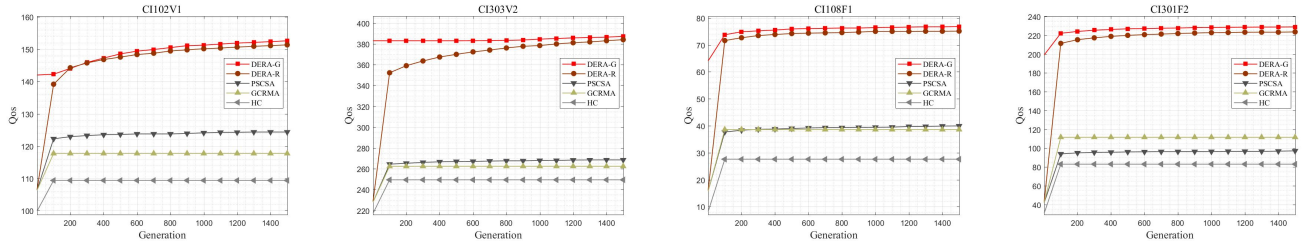


Figure 5: Convergence performance on QoS.

Table 3: Comparison on the composite objective.

Model	DERA-R		DERA-G		PSCSA		GCRMA		Greedy		HC	
	QoS	$Cost$	QoS	$Cost$	QoS	$Cost$	QoS	$Cost$	QoS	$Cost$	QoS	$Cost$
CI102V1	151.3	5984.5	**152.6**	**5944.1**	124.3	9410.7	117.7	7462.7	142	6364	109.4	10693.6
CI303V2	384.1	17796.3	**387.3**	**17607.2**	268.6	33526.4	262.5	19392.1	383	16634	249.5	33596.4
CI108F1	75.1	5110	**76.8**	**5105.8**	40	9159.2	38.6	4989.9	64	4116	27.6	9843.5
CI301F2	223.7	12472.7	**229.1**	**12273.3**	97.1	29378	111.6	13957	199	10881	83	28336.9
CL107V1	143.1	4674.3	**144**	**4637.7**	108.9	7881	107.2	5844.3	135	4830	97	8410.2
CL306V2	434.5	13402.9	**439.4**	**15008.8**	334.3	24546	322.2	17245.7	439	15724	312.7	25252.6
CL106F1	77.1	3246.7	**77.6**	**3286.3**	42.1	6691.7	43.5	3431	67	2581	28.2	7443.5
CL302F2	233.9	11491.8	**237.5**	**11248.6**	98.8	25063.2	116.2	10713.5	202	8101	86.7	24330.7
CO104V1	123.1	7154.8	**124.4**	**7184.9**	92.8	10945.7	85.5	7533.2	113	6897	80.5	11702.7
CO309V2	420.7	20047.2	**425.6**	**20717.1**	311.7	35655.6	303	23604.3	425	21301	291.5	36451.1
CO104F1	71	5382.7	**72**	**5318.6**	40.7	9517.9	38.6	5358.9	63	4890	28.7	10375.1
CO303F2	237.8	16055.9	**242.6**	**15584**	111.9	33682	127.5	16055.6	211	12149	95.9	32850.1

one.

$$QoS_{upper} = \sum_{r_j \in R} rc_j \tag{16}$$

Then we test the algorithm convergence of DERA-G, DERA-R, PSCSA, GCRMA, and HC regarding QoS. The results on instances of CI movement model are shown in Figure 5 We can see that the proposed DERAs are superior to the other three algorithms. Particularly, DERA obtains much higher convergence rate comparing to PSCSA, GCRMA and HC with same initialization. The greedy initialization makes DERA-G start with a high QoS, and DERA-G is not trapped to local optima. In models labeled with "V1", "F1" and "F2", DERA-G demonstrates its global searching ability and get the population out of local optima. for the model labeled with "V2", DERA-G obtains limited improvement, because the high variance of the dataset makes it hard to search a better solution. Also, this could be because the dimension selection missed riders suitable for the drivers. After all, the convergence speed and solution quality of our proposed DERA is superior to other algorithms.

Performance on the Composite Objective

Table 3 shows the performance of DERA and compared algorithms considering the two objectives, i.e., the primary objective QoS and the secondary objective cost of drivers. It can be seen that DERA-G performs better than all the other algorithms. DERA-G outperforms DERA-R with a narrow margin. It is because that, although greedy initialization may cause early convergence to local optima, our improved mutation operator and individual repair strategy help to solve this problem. PSCSA focuses on local best individuals and ignores the communication among non-optimal individuals, which limits the swarm to find better dimensions. GCRMA shows even less communication of riders, the only dimension communication happens in mutation. Their searching is somewhat one-sided and cannot cover the feasible field. HC performs the worst among all algorithms, which indicates that the pure local search is not suitable for the studied optimization problems. To our surprise, our proposed greedy method gets better results than PSCSA, GCMRA, and HC.

Conclusion

In this paper, we studied a ridesharing model with the objectives of optimizing the service quality and cost. The model is appealing to both drivers and riders, as drivers can be matched with multiple riders at one time such that the seats can be made full use of, and riders can reserve more than one seats. We then proposed a set-based differential evolution algorithm named DERA, which incorporates several new operators, such as greedy initialization, inter-vehicle mutation, and route-sensitive selection, to enhance the performance of differential evolution to search the optimal solutions. The experimental results showed that DERA performs better than several state-of-the-art algorithms for the ridesharing algorithm.

References

Agatz, N.; Erera, A.; Savelsbergh, M.; and Wang, X. 2010. Sustainable passenger transportation: Dynamic ride-sharing.

Agatz, N.; Erera, A. L.; Savelsbergh, M. W.; and Wang, X. 2011. Dynamic ride-sharing: A simulation study in metro atlanta. *Procedia-Social and Behavioral Sciences* 17:532–550.

Agatz, N.; Erera, A.; Savelsbergh, M.; and Wang, X. 2012. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research* 223(2):295–303.

Amey, A.; Attanucci, J.; and Mishalani, R. 2011. Real-time ridesharing: opportunities and challenges in using mobile phone technology to improve rideshare services. *Transportation Research Record* 2217(1):103–110.

Bakkal, F.; Eken, S.; Savaş, N. S.; and Sayar, A. 2017. Modeling and querying trajectories using neo4j spatial and timetree for carpool matching. In *IEEE International Conference on INnovations in Intelligent SysTems and Applications*, 219–222. IEEE.

Bei, X., and Zhang, S. 2018. Algorithms for trip-vehicle assignment in ride-sharing. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Bellman, R. 1958. On a routing problem. *Quarterly of applied mathematics* 16(1):87–90.

Berbeglia, G.; Cordeau, J.-F.; Gribkovskaia, I.; and Laporte, G. 2007. Static pickup and delivery problems: a classification scheme and survey. *Top* 15(1):1–31.

Bistaffa, F.; Farinelli, A.; and Ramchurn, S. D. 2015. Sharing rides with friends: a coalition formation algorithm for ridesharing. In *AAAI Conference on Artificial Intelligence*.

Chan, N. D., and Shaheen, S. A. 2012. Ridesharing in north america: Past, present, and future. *Transport Reviews* 32(1):93–112.

Chou, S.-K.; Jiau, M.-K.; and Huang, S.-C. 2016. Stochastic set-based particle swarm optimization based on local exploration for solving the carpool service problem. *IEEE transactions on cybernetics* 46(8):1771–1783.

Coltin, B., and Veloso, M. 2014. Ridesharing with passenger transfers. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3278–3283. IEEE.

Cookson, G., and Pishue, B. 2017. Global traffic scorecard. Technical report, Technical report, INRIX.

Dakroub, O.; Boukhater, C. M.; Lahoud, F.; Awad, M.; and Artail, H. 2013. An intelligent carpooling app for a green social solution to traffic and parking congestions. In *IEEE Conference on Intelligent Transportation Systems*, 2401–2408. IEEE.

Furuhata, M.; Dessouky, M.; Ordóñez, F.; Brunet, M.-E.; Wang, X.; and Koenig, S. 2013. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological* 57:28–46.

Hasan, M. H.; Van Hentenryck, P.; Budak, C.; Chen, J.; and Chaudhry, C. 2018. Community-based trip sharing for urban commuting. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Herbawi, W., and Weber, M. 2012. The ridematching problem with time windows in dynamic ridesharing: A model and a genetic algorithm. In *IEEE Congress on Evolutionary Computation*, 1–8. IEEE.

Huang, S.-C.; Jiau, M.-K.; and Chong, K.-H. 2017. A heuristic multi-objective optimization algorithm for solving the carpool services problem featuring high-occupancy-vehicle itineraries. *IEEE Transactions on Intelligent Transportation Systems* 19(8):2663–2674.

Huang, S.-C.; Jiau, M.-K.; and Lin, C.-H. 2014. A genetic-algorithm-based approach to solve carpool service problems in cloud computing. *IEEE Transactions on intelligent transportation systems* 16(1):352–364.

Huang, S.-C.; Jiau, M.-K.; and Liu, Y.-P. 2018. An ant path-oriented carpooling allocation approach to optimize the carpool service problem with time windows. *IEEE Systems Journal* 13(1):994–1005.

Lin, Y., and Shen, H. 2016. Vshare: A wireless social network aided vehicle sharing system using hierarchical cloud architecture. In *IEEE International Conference on Internet-of-Things Design and Implementation*, 37–48. IEEE.

Liu, Y.; Chen, W.-N.; Zhan, Z.-H.; Lin, Y.; Gong, Y.-J.; and Zhang, J. 2013. A set-based discrete differential evolution algorithm. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, 1347–1352. IEEE.

Pearson, K. 1900. X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 50(302):157–175.

Pelzer, D.; Xiao, J.; Zehe, D.; Lees, M. H.; Knoll, A. C.; and Aydt, H. 2015. A partition-based match making algorithm for dynamic ridesharing. *IEEE Transactions on Intelligent Transportation Systems* 16(5):2587–2598.

Russell, S. J., and Norvig, P. 2016. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Storn, R., and Price, K. 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* 11(4):341–359.

Tang, M.; Ow, S.; Chen, W.; Cao, Y.; Lye, K.-w.; and Pan, Y. 2017. The data and science behind grabshare carpooling. In *IEEE International Conference on Data Science and Advanced Analytics*, 405–411. IEEE.

Thangaraj, R. S.; Mukherjee, K.; Raravi, G.; Metrewar, A.; Annamaneni, N.; and Chattopadhyay, K. 2017. Xhare-a-ride: A search optimized dynamic ride sharing system with approximation guarantee. In *IEEE International Conference on Data Engineering*, 1117–1128. IEEE.

Zheng, L.; Chen, L.; and Ye, J. 2018. Order dispatch in price-aware ridesharing. *Proceedings of the VLDB Endowment* 11(8):853–865.

Zheng, L.; Cheng, P.; and Chen, L. 2019. Auction-based order dispatch and pricing in ridesharing. In *IEEE International Conference on Data Engineering*, 1034–1045. IEEE.