A Real-Time Ride-Sharing Matching Framework Using Simulated Annealing Genetic Algorithm

Jie Xu[1], Yong Zhang[1], Chunxiao Xing[1], Guigang Zhang[2]
[1]Research Institute of Information Technology, Beijing National Research Center for Information Science and Technology, Department of Computer Science and Technology, Institute of Internet Industry, Tsinghua University, Beijing 100084, China
[2]Institute of Automation, Chinese Academy of Sciences
{xuj15}@mails.tsinghua.edu.cn, {zhangyong05, xingcx}@tsinghua.edu.cn, {guigang.zhang}@*ia.ac.cn*

***Abstract*: Recently, ride-sharing services are becoming more popular in commercially application, which have attracted many researchers' attention. The ride-sharing provides significant economic, societal and environmental benefits in a sharing economy, such as reducing pollution, travel costs and traffic congestions. This kind of problem essentially is to maximize the matched ride-sharing pairs, and obviously is an optimization problem. There are lots of platforms for dynamic peer to peer ride-sharing, however, existing researches can be improved better. For example, how to reduce the algorithm computing time, and maximize the matching effectiveness to gain more economic benefits. In this work, we first introduce the problem of ride-sharing with time guarantee on road network, and then design a novel heuristic simulated annealing genetic algorithm. In addition, we carefully adjust the parameters with different constraints, and conduct extensive verification experiments with realistic datasets derived from Beijing car services, the results demonstrate the advancement of our methodologies.**

Keywords—Road Network, Ride-sharing, Simulate Annealing Algorithm, Gentic Algorithm

I. INTRODUCTION

Recently, we witness the rapid growth of the sharing economy, in which the resource owners temporarily transfer their properties to someone to acquire the benefits, the ride-sharing is the one of the most representative applications, that means someone acts as a driver and offers the ride-sharing service to bring together travelers with similar trip. Nowadays, the ride-sharing has become very prevalent application, consequently appearing lots of startups companies, typical like DIDI, Uber, and Lyft. There are lots of increasing researches, however, most existing methods have several limitations, for example, the efficiency is not high, and the methods need deep analysis of specific issues of the problems, or the extendibility is weak. Generally speaking, solving this kind of optimal problem is not easy. People have studied various approaches to address it [1,2,3], such as the integer linear programming, the branch-and-bound problem and the approximation algorithm [4].

In this work, we focus on the ride-sharing problem with time windows. There are many challenges to address such problems: First, it has been proved as an NP hard problem [10]; Second, to calculate the sharing ratio, it needs to calculate the shortest path, it is time-consuming; Third, for the [1]drivers, their main concerns are the economic benefits, thus it is not only needed to minimize detour distance at most in carry services, but also to maximum the number of matched pairs, it is a bi-objective problem, which can be solved by the simulated annealing(abbr.SA) or genetic algorithm(abbr.GA). Simulated annealing is a probabilistic technique for approximating the global optimum in a large search space, and the genetic algorithm is derived from the process of natural selection and genetic evolutionary. With the above mentioned knowledge, we present a framework which combines the simulated annealing algorithm with the genetic algorithm. The contributions of this work are summarized as follows: First, we establish the map grid, it can effectively reduce the shortest path calculation time. Second, we designed a formulation to describe the problem, and then develop a novel simulated annealing genetic algorithm(abbr. SAGA). Third, we conduct extensive experiments on the real dataset, and our results show the superiority of our approach.

The rest of this paper is organized as follows: we propose the related important work on ride-sharing and kinds of approaches in Section 2, and then model the problem in Section 3, afterwards, we illustrate our algorithm in detail in Section 4. Experimental results are presented in Section 5. We outline the conclusions of the paper in Section 6.

II. RELATED WORKS

The ride-sharing problem has attracted increasing attention, In this section, we introduce the related work and summarize the key technologies for different algorithms [5,6]. The studies can be classified into three categories, approximation algorithms, filter and refine algorithms, heuristic algorithms.

Ta et al. [7] proposed a new ride-sharing model that each driver has a requirement restriction with the shared route percentage, two variants of the ride-sharing problem are proposed, i.e., multiple drivers and a single rider and multiple drivers and multiple riders. To improve the efficiency, an approximate method with error bound guarantee was proposed, that meant updating the lower and upper bounds of the maximal graph matching until the bounds gap was small. The ides was very innovative and useful.

In [8], a highly generalized model for the taxi and delivery services in the market economy was proposed. The model can efficiently used with surge pricing mechanism. To solve the above problem, they proposed an approximation algorithm which was transferred to the multiple disjoint paths (MDP) problem, and carefully proved their algorithm had a tight approximate ratio with the original problem.

DOI reference number: 10.18293/SEKE2018-079

Furuhata et al. [9] proposed a review of ride-sharing to understand the key aspects of existing ride-sharing systems. The main characteristics to describe different aspects of ride-sharing systems were described. They classified the existing problem into six categories depending the target segments and criteria. The objective of the work was to identify key challenges thus helped to build the effective formal ride-sharing mechanisms.

Ma et al. [10] demonstrated a large-scale taxi ride-sharing for a dynamic ride-sharing problem, the framework included two steps：the first was a definition of a searching algorithm by taking advantage of a spatial-temporal index, then a scheduling algorithm was proposed. To address the heavy computational load, a lazy method was described by partitioning the road network into several grids. The approximated distance finally approved the effectiveness of their approach.

Thangaraj et al. [11] described a platform named Xhare-a-Ride (XAR) system for dynamic peer-to-peer ride-sharing. Three kinds of geographical hierarchical discretization including region using grids, landmarks and clusters, were used to help to eliminate shortest path computation during search. The notions were valuable for integration with multi modal trips.

Alarabi et al. [12] presented an effective framework named SHAREK which can embed inside existing approaches to improve the performance and quality. The model took into consideration the users' maximum willing waiting time and the cost of both riders and drivers. Three consecutive phases named Euclidian temporal pruning, Euclidian cost pruning, and Semi-Euclidean skyline-aware pruning were used to avoid the expensive shortest path computations.

Qian et al. [13] introduced SCRAM in order to provide recommendation fairness without sacrificing driving efficiency, the by-product of a framework named SCRAM was the capability of offering actual driving routes rather than rough driving directions. The recommended routes were evaluated from three aspects, i.e., priority principle, decaying principle, sharing principle. The successive probability of picking up customers had higher priority than the driving cost, and the probabilities of road sections were appraised decreasingly with the distance from the starting point.

Huang et al. [14] proposed a large scale service guarantee real-time ride-sharing, and demonstrated kinetic tree algorithms which can satisfy dynamic scheduling requests. They depicted a hotspot-based algorithm to avoid duplicates computation, the drop-off locations were grouped which led to a large number of valid schedules to satisfy the rider-driver constraints.

In [15,16], a genetic and insertion heuristic algorithm for a single rider was proposed. The main idea was to fine the best trips which can be shared by more than one driver with time constraints. They first modeled the ride-sharing problem with the time constraints, then illustrated the detailed implementation steps of the genetic algorithm. They also defined five different mutation operators basing on diverse services time constraints. The experimentation results indicated their algorithm was feasible.

III. PROBLEM DEFINITION

In this section, we formally propose the preliminaries and define the notions of our ride-sharing problem, and then depict the framework of our system.

A. Preliminaries

Definition 1. $G = (V, E)$ is used to model a road network, where a vertex set V is associated with a geographical position, including the longitude and latitude. An edge set E is associated with the weight such as the distance or travel cost between two different vertices.

Definition 2. (Rider). A rider $r_i=(r^s_i, r^e_i, t_r^e, t_r^L, t_{r\text{-}r}{}^L)$ is defined as a rider who intends to go to the destination r^e_i from r^s_i within an interval bounded by the earliest time t_r^e and the latest time t_r^L. (Driver). A driver $d_i=(d^s_i, d^e_i, t_d^e, t_d^L, d^s_i, t^e_i, \lambda_{max})$ is defined as a driver who plans to go to the destination d^e_i from d^s_i, at the latest start time t_d^L with a acceptable detour distance ratio λ_{max}. D denotes the drivers set, R denotes the riders set.

Definition 3. (Sharing ratio for driver) considering the driver's minimum detour distance λ_{max}, we introduce the sharing ratio to measure the ride-sharing effectiveness.

$$\xi(d_i, r_j) = \frac{\delta(d_i, r_j)}{\delta(d_i, r_j) + detour(d_i, r_j)} \quad (1)$$

$\delta(d_i, r_j)$ stands for the shortest distance of rider r_j, the $detour(d_i, r_j)$ refers to the cost of detour distance to pick up or put down the riders. We assume that if the sharing ratio is less than the ratio bound (λ_{max}), the match is valid. Some key notations are summarized in Table 1 [7]:

Table 1. Some Key Notaions

Notation	Definition
r^s_i	rider's start point
r^e_i	rider's destination point
t_r^e	rider's earliest start time
t_r^L	rider's latest start time
$t_{r\text{-}r}{}^L$	rider's latest reach time
t_d^e	driver's earliest start time
t_d^L	driver's latest start time
d^s_i	driver's start point
d^e_i	driver's destination point
λ_{max}	driver's maximum acceptable detour ratio
$\xi(d, r)$	detour distance for matched pair

B. Problem definition

The ride-sharing problem is defined as follows: For a set of drivers and a set of riders on a road network $G(V,E)$, when the riders send series requests, we aim to satisfy the requests

and acquire the maximum sharing matching ratio denoted by $\sum_{\forall d_i \in D, r_j \in R} \text{detour}(d_i, r_j)$ under the temporal and distance factors [9]. In Figure 1, there are 3 drivers and 3 riders, we aim to match the drivers and riders with the minimum detour distance.

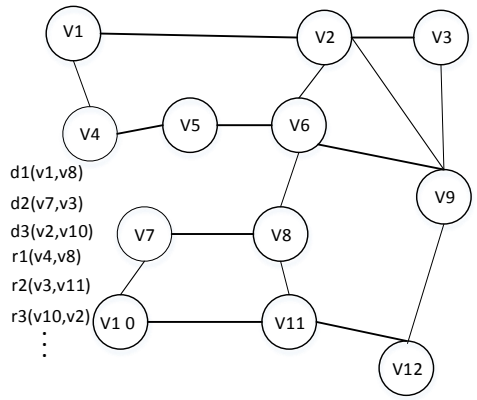

Fig. 1. Example of ride-sharing between drivers and riders

IV. A SIMULATED ANNEALING GENETIC ALGORITHM

Our goal is to minimize the detour distance for whole riders' requests, it is obvious a global optimization problem. A brute force method is to enumerate all possible driver-rider pairs. However, traditional approaches are rather time consuming and inefficiency. To tackle the challenges, we first divide the map into many regular squares to quickly find the approximate shortest distance, and then introduce a heuristic simulated annealing genetic algorithm (SAGA) to address the optimal driver-rider pairs matter.

A. Grid construction

To calculate the shortest path on road networks, Ma et al.[10] presented a spatial gird which did pre-calculation and stored the shortest path by dividing the map into small fixed square areas [18]. Their main ideas lay in the approximate path and distance. Similarly, we also utilize this basic ideas, select the history hottest visited places on behalf of the grid's location, and then construct the shortest distance matrix and the shortest path matrix. For any two arbitrary points, we first figure out which girds they geographically belong to, then we acquire the shortest distance by querying the corresponding grid matrix.

B. Formulation representation

The bi-objective function that minimizes the total distance and maximizes the number of the valid matched pairs is provided in Equation 2. The *α, β, γ* define the relative weight factors. Let $x_{i,j}$=1 if the driver-rider pair is matched from node i to node j. The formulation is,

$$max(-\sum_{\forall d_i \in D, r_n \in R} detour(d_i, r_n) + \sum_{\forall x \in D, y \in R} x_{i,j}) \qquad (2)$$

subject to:

$$x_{ij} \in \{0,1\} \qquad (3)$$

$$\sum_{i \in D} d_i \sum_{j \in R} r_j = D + R \qquad (4)$$

$$\sum x_{ij} <= D_{number} \qquad (5)$$

$$\sum x_{ij} <= R_{number} \qquad (6)$$

$$t_d^{\,L} \le t_r^{\,e} \qquad (7)$$

$$t_d^{\,L} \le t_r^{\,L} \qquad (8)$$

$$\xi(d_i, r_j) <= \lambda_{\max} \qquad (9)$$

where Constraint (3) enforces the matched pairs have only two choices: 0 or 1. Constraints (4) ensures that the number of drivers and riders leaving the starting is equal to the number of drivers and riders arriving at the destination. Constraints (5) and Constraints (6) ensures the total matched driver-rider are less than the total initial given drivers. Constraints (7) ensures driver's latest start time is earlier than rider's earliest start time. Constraints (8) ensures driver's latest start time is earlier than rider's latest start time. Constraints (9) ensures driver's detour distance is less than the driver's default acceptable detour distance.

C. Procedure of SAGA

The procedure of SAGA is described in Figure 2. The GASA process firstly starts by given some important parameters such as population size, initial temperature, number of generations, probability of crossover and mutation. An initial population of chromosome is also generated, every chromosome is a candidate solution for problem [17].

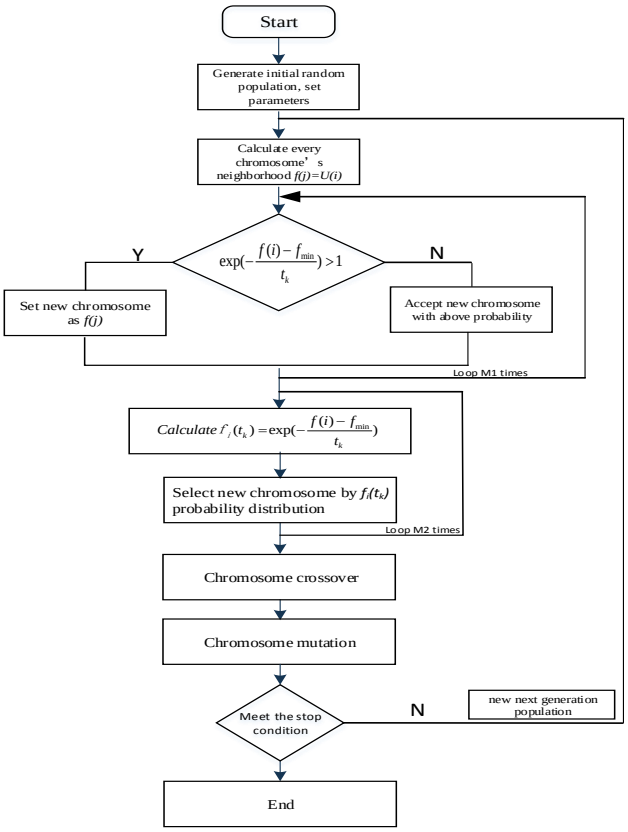

Fig. 2. SAGA procedure for ride-sharing

For every chromosome, we use the objective function (Equation 2) as a fitness function. The selection of fitness function directly affects the convergence speed of genetic algorithm. A random value *U(i)* as *f(j)* is selected from the neighborhood of *f(i)*, then probability function $exp(-\frac{f(j)-f(i)}{t_k})$ is calculated. If the value is greater than constant one, then accept the neighborhood *f(j)* as the new chromosome, else accept the new one with above probability. After above steps have been executed in M_1 cycles, new population named NewPopOne where the size is equal to the initial population is generated. Later, the fitness of each chromosome represented by function $exp(-\frac{f(i)-f_{min}}{t_k})$ is calculated, afterward, new chromosome with the probability distribution determined by above fitness function is generated, that process which is executed in M_2 cycles finally results in the next new population named NewPopTwo. In addition, the crossover and mutation strategies are executed to create offspring until the next new population is generated.

The above mentioned steps are repeated until the results meet the expectation value or iteration operations are all completed.

D. Problem coding

We randomly assemble the driver-rider matched pairs successively in one chromosome, the length of the chromosome depends on the minimum of drivers or riders. The whole population is set as 50. Figure 3 presents the problem coding.

parent 1:

1	4	8	8	7	3	11	3	2	10	2	10

parent 2:

1	3	11	8	7	10	2	3	2	4	8	10

parent i:

.....											

Fig. 3. Chromsome initial coding

E. Annealing Algorithm Fundamentals

In general, the key to integrate the two algorithms lies in how to select the next generation, the SAGA selects the next generation with the higher probability of getting close to the target in the range of neighborhood, in which the algorithmic process is a constant random walk from one state to another. We can use Markov process to describe the transfer probability, acceptance probability.

The challenge of Annealing Algorithm are as follows:

(1) The initial temperature t_0. When the initial temperature is high, it is more likely to search for the global optimal solution, but it takes a lot of computation time. On the other hand, the computation time can be decrease, but the global search performance may be affected.

(2) The annealing speed. The global search performance is closely related to the number of iterations at each temperature t. "Full" search at the same temperature is quite necessary, but it also takes time to calculate. The increase of the number of loop cycles will inevitably lead to an augment in computation overhead. We use the following function [18]:

$$t_{k+1} = t_k(1+\beta t_k)^{-1} \quad (10)$$

$$\beta = \frac{t_0 - t_f}{M_1 t_0 t_f} \quad (11)$$

the t_0 and the t_f are default value, M_1 is the number of iterations. In theory, SA can solve most of the optimization problems, but in practice, due to the global optimum annealing speed is too slow to be accepted. In this article, we select the cooling strategy as follows:

$$f_i(t_k) = \exp(-\frac{f(i)-f_{min}}{t_k}) \quad (12)$$

while the f_{min} denotes the minimum of fitness value in population. According to the Metropolis criterion [21], the probability that particles tend to equilibrate at temperature t is *exp (-ΔE / (kT))*, where *E* is the internal energy at temperature *T*, *k* is Boltzmann's constant. Metropolis formulation are often expressed as follows:

$$p = \begin{cases} 1 & if\ E(x_{new}) < E(x_{old}) \\ \exp(-\frac{f(j)-f(i)}{t_k}) & if\ E(x_{new}) \geq E(x_{old}) \end{cases} \quad (13)$$

which means the algorithm is more likely to accept sort-of-bad jumps rather than complete refuse to accept it, the probability gradually decreases over time till solution becomes global stability.

F. Crossover

The function of the crossover is to ensure the stability of the population and evolvement towards the optimal solution. Crossover can facilitate avoiding premature convergence. Chromosome crossover doesn't mean descendants are definite better than their parents, but represent the next generations have a better development tendency than the previous generation. There are many kinds of crossover methods, multi-point crossover refers to exchanging the multiple crossover points in an individual chromosome. In this paper, we use the method of position-based multi-point crossover (PBC) [20]:

As in Figure 4, first, we select a random pair of chromosomes (parents) in population, the location may not be continuous, but the two parents chromosomes were selected at the same location; Second, a pro-offspring with the guarantee that the selected gene position is located at the same with the parents position is generated. Third, we find the position of the gene selected in the first step in another parent, and then put the remaining genes sequentially into the pro-offspring generated in the previous step 2.

parent 1:

1	4	8	8	7	3	11	3	2	10	2	10

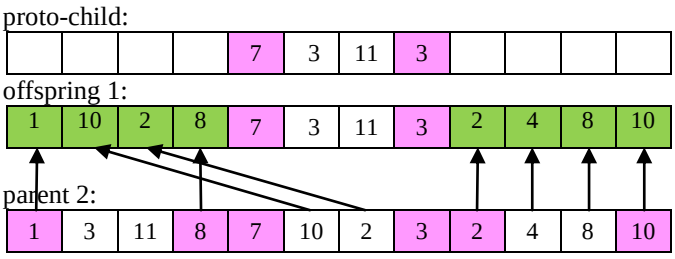

Fig. 4. Chromosome crossover

G. *mutation*

The role of mutation is to ensure the vast diversity of the population with the operation that change the value of a particular gene from one generation of a population, that also can avoid the possible convergence of local convergence. In this work, the probability of mutation is set as 0.01~0.1.

V. Performance Evaluation

In this section, we conduct experiments and evaluate the performance of our algorithm by using real trajectories dataset. We use the Beijing road data with about 300,000 vertices and 400,000 edges, and utilize two historical trajectories datasets, the Taxi, which contains about 100,000 trajectories of user orders generated by more than 5,000 public taxicabs in one month in Beijing, Ucar [7], which contains about 300,000 trajectories of user orders generated by more than 4,000 public taxicabs in two weeks in Beijing. The riders' trajectories include start and destination point, earliest start time, latest start time, latest reach time, the drivers' trajectories includes start and destination point, earliest start time, latest start time. We also simulate some drivers trajectories by using history hottest start and destination points. Experimental settings: All experiments are run on a machine equipped with 3.6 GHz Intel Core i7-4790 CPU, 16GB RAM, window7 OS and algorithms implemented in Python 27.

We evaluate the algorithms mainly from two aspects, the running time, the shared path ratio. We compare our algorithm with the XAR [11], TGA [15]. We conduct our experiments for different number of drivers from 3000 to 15000 with a fixed number of riders (i.e., 9000). By default, $t_0 = 100$, $t_f = 10$, and M_1=1000 for SAGA algorithm. We make use of the grid matrix in Section 4 when compute the shortest distance.

A. *Efficiency*

We assess the efficiency by fixing the number of drivers as 9000 while varying the number of riders from 3000 to 12000 presented by x-axis. In Figure 5a, 5b, we set the same number of iterations as 800 for SAGA and GA. Figure 5 shows the results corresponding to different algorithms. We can see that SAGA is greater than the pure genetic algorithm(GA). Meanwhile, since the SAGA takes advantage of a fast acceleration adaptation function, the running time is not too high. From Figure 5c and Figure 5d, our algorithm running time is superior to XAR and TGA.

B. *Effectiveness*

Figure 6 shows the performance of average shared path ratio. The number of iteration is set as 800 for SAGA and GA. We have the following observations that the matched ratio of SAGA is higher than the GA, because the SAGA search in the range of chromosome neighborhood, thereby avoiding falling into local optimal. Figure 6c and Figure 6d show that our algorithm ratio is within 72%-75%, and is better than baseline algorithms within 65%-54%.

C. *The effects of the parameter : descent speed M_1*

We evaluate how the annealing descent speed affects the running time and the matching ratio. We set parameter M_1 as 500, 1000, 1500 for SAGA_t1, SAGA_t2, SAGA_t3 respectively, the results are shown in Figure 7. The larger the M_1 is, the faster the evolution change. We have the following observations in Figure 7, as M_1 grows, the running time also increase, this is because it requires more loop iteration calculations. Meanwhile, from Figure 7b and Figure 7d, the approximate matched ratio enlarge as well, because it is possible to gain more candidate chromosomes from the neighborhood of old chromosomes.

VI. Conclusion

In this work, we utilize real dataset to model a ride-sharing problem and present a simulated annealing genetic algorithm to address it. The proposed SAGA outperforms commonly baseline algorithms including XAR and TGA. We run a careful iterative turning process, and the extensive experiments on large car services datasets show the advantage of our algorithm.

There are three interesting directions for the future work, (a)we will incorporate traffic conditions to assemble the candidate riders, (b)consider the ride-sharing with social networks, (c)introduce artificial intelligence algorithm to solve the ride-sharing problem.

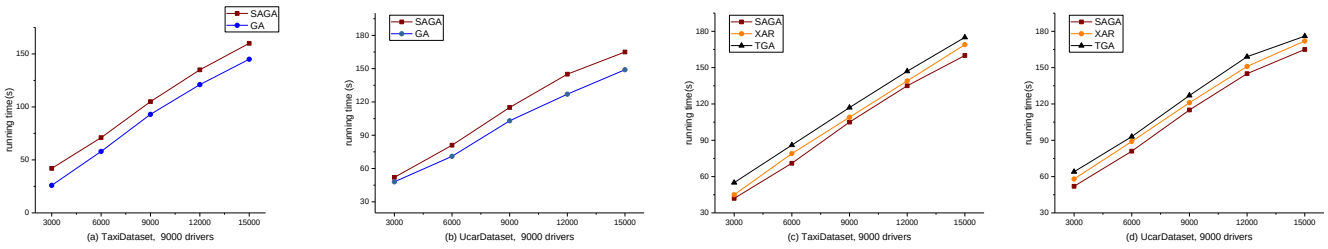

Fig. 5. Comparison of running time taken by SAGA, GA, XAR,TGA

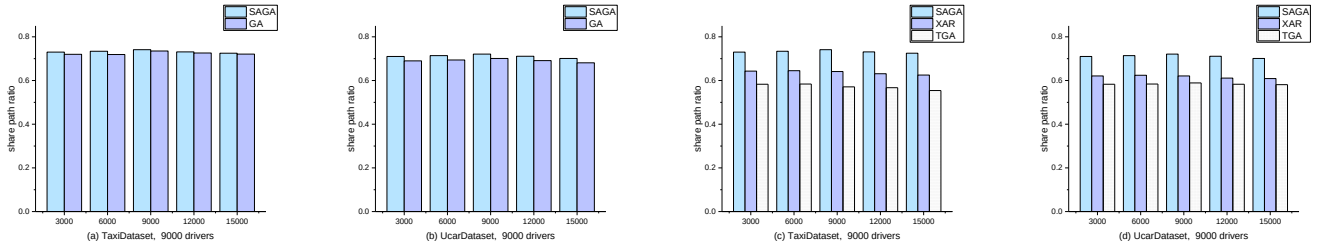

Fig. 6. Comparison of share path ratio taken by SAGA, GA, XAR,TGA

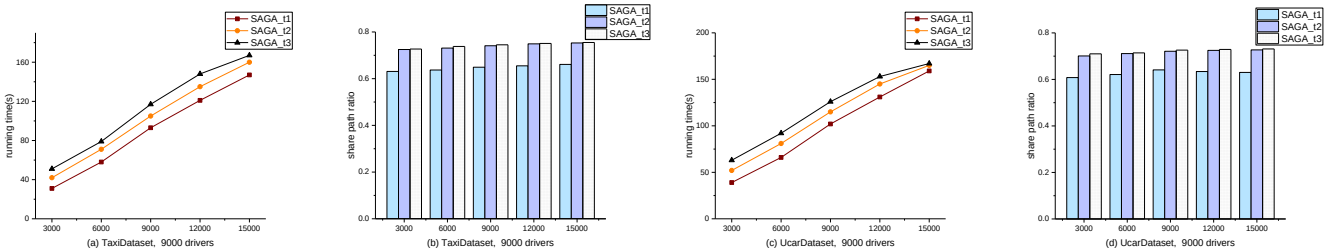

Fig. 7. Performance of running time and share path ratio by varying descent speed M_1

ACKNOWLEDGMENT

This work was supported by NSFC(91646202), the National High-tech R&D Program of China(SS2015AA020102), Research/Project 2017YB142 supported by Ministry of Education of The People's Republic of China, the 1000-Talent program, Tsinghua University Initiative Scientific Research Program.

REFERENCE

1. Bartolini E, Bodin L, Mingozzi A. The traveling salesman problem with pickup, delivery, and ride - time constraints[J]. Networks, 2016, 67(2): 95-110.
2. Agatz N, Erera A, Savelsbergh M, et al. Optimization for dynamic ride-sharing: A review[J]. European Journal of Operational Research, 2012, 223(2): 295-303.
3. Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE transactions on evolutionary computation, 2002, 6(2): 182-197.
4. Z. Chen, H. T. Shen, X. Zhou. Monitoring path nearest neighbor in road networks. SIGMOD, pages 591–602, 2009.
5. Goel P, Kulik L, Ramamohanarao K. Optimal pick up point selection for effective ride sharing[J]. IEEE Transactions on Big Data, 2017, 3(2): 154-168.
6. Asghari M, Shahabi C. An On-line Truthful and Individually Rational Pricing Mechanism for Ride-sharing[J]. 2017.
7. Ta N, Li G, Zhao T, et al. An Efficient Ride-Sharing Framework for Maximizing Shared Route[J]. IEEE Transactions on Knowledge and Data Engineering (TKDE), 2017.
8. Jia Y, Xu W, Liu X. An Optimization Framework For Online Ride-sharing Markets[C]//Distributed Computing Systems (ICDCS), 2017: 826-835.
9. Furuhata M, Dessouky M, Ordó?ez F, et al. Ridesharing: The state-of-the-art and future directions[J]. Transportation Research Part B: Methodological, 2013, 57: 28-46.
10. Ma S, Zheng Y, Wolfson O. T-share: A large-scale dynamic taxi ridesharing service[C]//Data Engineering (ICDE), 29th International Conference on. IEEE, 2013: 410-421.
11. Thangaraj R S, Mukherjee K, Raravi G, et al. Xhare-a-Ride: A Search Optimized Dynamic Ride Sharing System with Approximation Guarantee[C]//Data Engineering (ICDE), 2017 IEEE 33rd International Conference on. IEEE, 2017: 1117-1128.
12. Alarabi L, Cao B, Zhao L, et al. A demonstration of SHAREK: an efficient matching framework for ride sharing systems[C]//Proceedings of the 24th ACM SIGSPATIAL. ACM, 2016: 95.
13. Qian S, Cao J, Mou?l F L, et al. SCRAM: a sharing considered route assignment mechanism for fair taxi route recommendations[C]//Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2015: 955-964.
14. Huang Y, Bastani F, Jin R, et al. Large scale real-time ridesharing with service guarantee on road networks[J]. Proceedings of the VLDB Endowment, 2014, 7(14): 2017-2028.
15. Herbawi W, Weber M. The ride matching problem with time windows in dynamic ridesharing: A model and a genetic algorithm[C]//Evolutionary Computation (CEC), 2012 IEEE Congress on. IEEE, 2012: 1-8.
16. Herbawi W M, Weber M. A genetic and insertion heuristic algorithm for solving the dynamic ridematching problem with time windows[C]//Proceedings of the 14th annual conference on Genetic and evolutionary computation. ACM, 2012: 385-392.
17. Shen B, Zhao Y, Li G, et al. V-Tree: Efficient kNN Search on Moving Objects with Road-Network Constraints[C]//Data Engineering (ICDE), 2017 IEEE 33rd International Conference on. IEEE, 2017: 609-620.
18. Wenxun Xing, Jinxing Xie. Modern optimization calculation method[M]. Tsinghua University Press, 2005
19. Pham D, Karaboga D. Intelligent optimisation techniques: genetic algorithms, tabu search, simulated annealing and neural networks[M]. Springer Science & Business Media, 2012.
20. Razali N M, Geraghty J. Genetic algorithm performance with different selection strategies in solving TSP[C]//Proceedings of the world congress on engineering. Hong Kong: International Association of Engineers, 2011, 2: 1134-1139.
21. Van Laarhoven PJM. Simulated annealing[M]//Simulated annealing: Theory and applications. Springer, Dordrecht, 1987.