A Parallel Ant Colony System Based on Region Decomposition for Taxi-Passenger Matching

Xin Situ[1], Wei-Neng Chen[2], Yue-Jiao Gong[2,*], Ying Lin[3], Wei-Jie Yu[4], Zhiwen Yu[2] and Jun Zhang[2]
[1] School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China
[2] School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
[3] Department of Psychology, Sun Yat-sen University, Guangzhou, China
[4] School of Information Management, Sun Yat-sen University, Guangzhou, China
[*] Corresponding Author: gongyuejiao@gmail.com

***Abstract*—Taxi dispatch is a critical issue for taxi company to consider in modern life. This paper formulates the problem into a taxi-passenger matching model and proposes a parallel ant colony optimization algorithm to optimize the model. As the search space is large, we develop a region-dependent decomposition strategy to divide and conquer the problem. To keep the global performance, a critical region is defined to deal with the communications and interactions between the subregions. The experimental results verify that the proposed algorithm is effective, efficient, and extensible, which outperforms the traditional global perspective greedy algorithm in terms of both accuracy and efficiency.**

Keywords—taxi dispatch; ant colony system (ACS); Taxi-Passenger Matching (TPM); region-dependent decomposition (RDD)

I. INTRODUCTION

Traditionally, the taxi drivers hunt for passengers on street. The inexperienced drivers would cruise on street blindly, which would increase a good deal of idle driving distance. Conversely, the experienced drivers would try to maximize their own profit by reducing their idle driving distance based on their experience. However, this subjective behavior would bring out a problem that some regions appear an excess of taxi supply over passenger demand while vacant taxis are in short supply in other regions. These two phenomena are due to the 'blindness' and 'subjectivity' of taxi driver.

With the proliferation of smart phones and the increasing demands of taxi market, vehicle booking APPs like UBER are being used more and more commonly [1]. It becomes available to dispatch the taxis from a control center, namely, matching passengers for taxis from a global perspective, in order to avoid a battle of hunting the same passenger, which could cause the waste of resources. Therefore, a taxi-dispatch system is required to match taxis and customer service requests [2]-[3]. In this way, the global profit of the taxi company could be improved.

The first-come-first-serve (FCFS) is commonly adopted, i.e., whenever a passenger demand appears, it is matched with the nearest vacant taxi, which was proved an inefficient strategy in [4]. In addition, for the sake of the total profit of the taxi company, some other works implement the greedy strategy over the global perspective, which gives priority to the most profitable pair of taxi and passenger. However, although increase the total profit, this kind of methods also increase the expense of the matching time. More critically, the greedy algorithm can only provide locally optimal results, which may not be accurate, especially when the number of taxis is large. As taxi dispatch is a critical issue, recent researches have studied about the modern taxi networks based on the vehicular Global Positioning System (GPS) of each taxi [5]-[11]. It was proved that GPS technology is beneficial to enhance service quality to customers [5]. Some driving direction systems were built leveraging the intelligence of experienced drivers [6]. Taxi-hunting recommendation system were proposed in [7] and [8], in order to provide passengers with a waiting time to get a taxi ride in a particular location and helps the drivers to make more profit respectively. Meanwhile, Zhang *et al*. had analyzed taxi service strategies through the GPS data set, aiming to provide useful guidance to taxi drivers [9]. To predict the distribution of taxis and passengers in a short-term time horizon, a novel methodology was proposed in [10].

Considering that the task of taxi dispatch has a requirement of real-time response, we need an effective algorithm to find acceptable solutions. In [11], Miao *et al*. proposed a dispatching scheme based on administrative subdistricts segmentation, which can reduce the total idle driving distance of taxi network and keep a balance between the passenger demand and taxi supply in each sudistrict. But they only consider the dispatch between the subdistricts without matching the vacant taxi to an exact passenger one by one, so the individual taxi in subdistrict still endures the 'blindness' and 'subjectivity' problems.

Ant colony system (ACS) was first proposed by Dorigo and Gambardella in 1997, which is an efficient version of ant colony optimization (ACO) [12]. ACS simulates the foraging process of ant colony to optimize the discrete combinatorial optimization problem. A number of research results have shown that ACS can solve the real-world problem effectively, such as the travelling salesman problem (TSP) [12], the aircraft arrival sequencing and scheduling problem [13], and grid workflow scheduling problem [14]. The Taxi-Passenger Matching (TPM) Problem defined in the paper is also a combinatorial optimization problem, for which the ACS algorithm is naturally applicable. Besides, since we deal with practical city scenarios, the problem space is extremely large. For dimension reduction, the city is divided into several administrative regions according to the geographical locations. Dividing by geographical locations enables the parallel optimization of different regions, which is beneficial to reduce the matching time and travelling costs for the taxis.

Accordingly, ACS is performed in parallel based on a region-dependent decomposition (RDD) strategy. The purpose of decomposition is to reduce the number of taxis in each region,

This work was supported by the National Natural Science Foundation of China under Grant Nos. 61502542, 61622206, 61379061, and 61332002.

978-1-5090-4601-0/17/$31.00 ©2017 IEEE

making the evolution more sufficiently within the same number of iterations. In addition, the critical regions are given fully consideration, which contributes to the evolution between regions and improves the quality of solutions. In the meantime, as taxi dispatch is a practical problem, it is necessary to consider its timeliness. As afore-mentioned, the region decomposition method enables parallel evolution, i.e., each sub-region evolves concurrently. In this way, the running time for optimizing the whole region is greatly reduced. Note that we use the Message Passing Interface (MPI) to coordinate the different regions and achieve parallel computation.

The rest of this paper is organized as follows. In Section II, the TPM problem is formulated. Section III presents the RDD-improved ACS algorithm for the TPM in detail. Experiments are carried out in Section IV, and test results verify that the proposed algorithm is effective, efficient, and extensible. Finally, conclusions and future work are summarized in Section V.

II. Problem Fomulation

Traditionally, people used to hail a taxi on street when they need taxi service. But with the rapid development of mobile technologies, many transportation service providers come out, like Uber, Lyft, Ola, GrabTaxi, etc. Nowadays, the younger generation prefers to hail a taxi by a mobile phone application when they need a taxi. The present researches have modeled the taxi dispatch problem [11], but the emphasis of our work is to optimize the total profit of the taxi network by matching the vacant taxis and passengers when taxi resources are insufficient. The proposed Taxi-Passenger Matching (TPM) problem is formulated in this section. A list of parameters used in the TPM model is shown in Table I.

In practice, due to the requirement of traffic regulations, many modern cities have set temporary taxi stands, where a taxi can pickup or dropdown a passenger. So the road network is formulated as a Triple $G = (V, E, T)$, where V represents the set of vertices which denote the temporary taxi stands in practice, E represents the set of edges between every reachable intersection, and T represents the weight set of the edges, namely, the path length between every two reachable intersections. All taxis and passengers are generated on the vertices of road network.

The TPM problem is a combinatorial optimization problem (COP). There are taxis and passengers who require taxi service in an area, and we need to match the taxis and passengers in a global perspective. Assume that the quantity of passengers N_C is slightly more than the quantity of taxis N_S, there will be $A_{N_C}^{N_S}$ different assignments.

The scheduling of taxis to pick up passengers is defined as

$$x_{i,j} = \begin{cases} 1, & \text{if taxi } i \text{ is scheduled to serve passenger } j \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

Based on this, we define $y_j = \sum_{i=1}^{N_S} x_{i,j}$, and thus have

$$y_j = \begin{cases} 1, & \text{if passenger } j \text{ is scheduled to be served} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

The objective is to maximize the total profits. For each deal with $x_{i,j} = 1$, there are three critical points to calculate the profits: the position of taxi i ($Tpos(i)$), the position of passenger j ($Ppos(j)$), and the destination of the passenger j ($des(j)$). These three points are called major information of taxi and passenger.

TABLE I. Parameters Of The TPM

Parameters	Description
S	set of vacant taxis
N_S	quantity of vacant taxis
C	set of passengers
N_C	quantity of passengers
$Tpos(i)$	the current position of taxi i
$Ppos(j)$	the current position of passenger j
$des(j)$	the destination of passenger j
$D(a,b)$	the distance between intersection a and b
tip_j	the tips provided by passenger j
ξ	a coefficient of the taxi fare per distance unit
ζ	a coefficient of traverse fee per distance unit

In real life, not only major information, we can also get some additional information from the software or the mobile phone application, such as discounts, the tip provided by the passenger, the credit of passenger, and so on. These information can also be taken into account while calculating the profits. To simplify the model, we divide the profit into two parts, the income part and the cost part. The income part comes from the passenger paying for the service, which can be calculated as

$$income_{i,j} = \xi \cdot D\big(Ppos(j), des(j)\big) + tip_j \quad (3)$$

The cost part comes from the taxi traverse fee, which can be calculated as

$$cost_{i,j} = \zeta \cdot (D(Tpos(i), Ppos(j)) + D(Ppos(j), des(j))) \quad (4)$$

Thus, we can deduce that the profit of a deal is

$$profit_{i,j} = income_{i,j} - cost_{i,j} \quad (5)$$

We call $profit_{i,j}$ is the profit pair between taxi i and passenger j. The total profits of all taxi-passenger pairs is then deduced as

$$P_1 = \sum_{i=1}^{N_S} \sum_{j=1}^{N_C} x_{i,j} \cdot profit_{i,j} \quad (6)$$

However, if only the profit term is considered, it may give priority to the most profitable passengers, leading to the starvation of the other passengers whose destination is not far away. For example, suppose that there are two passengers and one taxi. The distance between the first passengers and the taxi is 3 times than second passenger, while the first trip's length is twice than the other. It is obvious that, if $\xi > 3\zeta > 0$, the profit got from the first passenger is more than the second one. However, it will cost two and a half times time to finish the trip (considering that the time consumption is proportional to the length of path). So, in the proposed TPM model, the objective is to maximize the total profits of all taxi-passenger pairs per unit time, namely,

$$P = \sum_{i=1}^{N_S} \sum_{j=1}^{N_C} x_{i,j} \cdot \frac{profit_{i,j}}{D(Tpos(i), Ppos(j)) + D(Ppos(j), des(j))} \quad (7)$$

subject to

$$\sum_{i=1}^{N_S} x_{i,j} \le 1, \sum_{j=1}^{N_C} x_{i,j} \le 1 \quad (8)$$

The constraints in (8) restrict that each taxi could not serve more than one passenger at once, and that each passenger could not be served by more than one taxi concurrently.

III. The Porposed Algorithm

A. ACS Solution Construction

Ant colony system, proposed by Dorigo and Gambardella in 1997 [12], is an improved version of the original ant colony optimization algorithm. The framework of ACS algorithm is appropriate for the discrete combination optimization problems. In the Taxi-Passenger Matching Problem, ACS needs to optimize the matching pairs of the taxis and passengers to maximize the total profits per unit time. As described in the above Section II, there will be $A_{N_C}^{N_S}$ possible assignments. The Taxi-Passenger Matching problem can be instantiated as a permutation problem, which means to assign a passenger for each taxi in practice.

During the ACS solution construction process, the details about the initialization, transition rules, and updating the pheromone are described as below.

1) Initialization

In the initialization phase, the ants are uniformly distributed among the taxis sequence as their starting-point, and each of the ants will traverse all taxis according to their serial number, and allocate one of the suitable passengers to each of them. The pheromone values, which indicate the probability of assigning a given taxi with a specific passenger, are initialized to an appropriate value τ_0.

2) State Transition

Each time when an ant selects a passenger for taxi r, it applies the following rule

$$s = \begin{cases} \arg\max_{u \in J_r} \{[\tau(r,u)] \cdot [\eta(r,u)]^{\beta}\}, \text{ if } q \le q_0 \text{ (exploitation)} \\ S, \text{ otherwise (biased exploration)} \end{cases} \quad (9)$$

where J_r is a set of passengers which can be chosen by the ant on the current taxi r, making sure that none of N_C passengers have been chosen more than once; $\tau(r,u)$ and $\eta(r,u)$ represent the pheromone and heuristic information between taxi r and passenger u respectively; $\beta > 0$ is a parameter that determines the relative importance of the pheromone versus the heuristic [12]; q_0 is a parameter in [0,1], which is used to control the ants' behavior of exploitation and exploration. And if q, a random value generated uniformly in [0,1], is smaller than q_0, the best passenger will be chosen, otherwise the passenger s will be determined by a random variable S which is selected based on the probability distribution as

$$p(r,s) = \begin{cases} \frac{[1+\tau(r,s)] \cdot [1+\eta(r,s)]^{\beta}}{\sum_{u \in CAN_r} [1+\tau(r,u)] \cdot [1+\eta(r,u)]^{\beta}}, \text{ if } CAN_r \neq \emptyset \wedge s \in CAN_r \\ \frac{[1+\tau(r,s)] \cdot [1+\eta(r,s)]^{\beta}}{\sum_{u \in J_r} [1+\tau(r,u)] \cdot [1+\eta(r,u)]^{\beta}}, \text{ if } CAN_r = \emptyset \wedge s \in J_r \\ 0, \text{ otherwise} \end{cases} \quad (10)$$

where CAN_r is a candidate list of taxi r and it contains the most profitable passengers of taxi r. The quantity of passengers in the candidate list is a constant number, and the list can be initialized during initializing the profit pair between taxis and passengers. If and only if the passengers in the candidate list have all been chosen, the ant will choose the passenger in J_r.

In (9) and (10), η, which is called heuristic information in ACS, represents the profit of each taxi, which is defined as

$$\eta(r,u) = profit_{r,u} \quad (11)$$

3) Pheromone Update

In a general ACS process, both the local and global pheromone update rules are performed. In the proposed algorithm, the local update of pheromone is implemented when any ant is scheduled with the taxi-passenger pair , while the formula is given as

$$\tau(r,s) \leftarrow (1 - \rho) \cdot \tau(r,s) + \rho \cdot \tau_0 \quad (12)$$

where ρ is the factor of local evaporation, τ_0 is initial pheromone.

On the other hand, the global update of pheromone is only conducted on the best solution so far. Only the taxi-passenger pairs (r,s), which appear in the best-so-far solution, are updated as

$$\tau(r,s) \leftarrow (1 - \alpha) \cdot \tau(r,s) + \alpha \cdot \Delta\tau \quad (13)$$

where α is the factor of global evaporation, $\Delta\tau$ represents the pheromone increment on the global best solution, and $\Delta\tau$ is calculated as

$$\Delta\tau = \mathrm{CDF}\left(\frac{P_{gb}}{N_S}, \mu, \sigma\right) \quad (14)$$

where function CDF is a cumulative distribution function; P_{gb} is global highest profit; and $\frac{P_{gb}}{N_S}$ is a dynamic parameter of CDF; μ and σ are two parameters of CDF, namely, the mean and variance.

With these two rules of updating pheromone, the search behaviors of ants are guided. The local update of pheromone is used to evaporate the pheromone of the matched pair by the previous ants. It is beneficial to increase the diversity of the population since the probability of choosing the matched pair is reduced for the remaining ants. Conversely, the global update of pheromone is used to strengthen the pheromone of the best matched pairs so far, which will be beneficial to enhance the convergence speed.

B. Parallel Framework

In real life, a city could be divided into several administrative regions according to the geographical location. In the proposed algorithm, dividing by geographical location is beneficial to reduce the matching time for taxi, since we can deal with separated regions in parallel.

With the region-dependent decomposition strategy, we suppose a whole region has been divided into several subregions uniformly, and the area between two or more adjacent subregions is called critical region. Let κ be ratio of each critical region to the corresponding subregion. Each passenger outside the critical regions belong to the subregion where they currently locate, while the passenger inside a critical region could belong to one of its adjacent subregions. In this way, the taxi could pick up not only the passenger in the local region but also the passenger near the subregion border, which is more reasonable and realistic.

In this paper, we adopt a master-slave model [15] with the Message Passing Interface (MPI) as a parallel strategy to deal

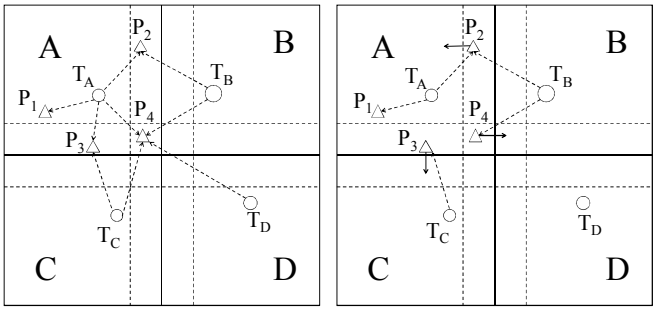


Fig. 1. Illustrations of the region division and passenger assignment.

Algorithm 1: Fast matching process:

procedure fast matching
1. Initialization
2. **for** each taxi i (i=1,2,..., N_S)
3. match the most profit passenger from unmatched passenger set
4. remove the most profit passenger from unmatched passenger set
5. **end for**

end procedure

with the subregions. In this strategy, the number of processors is identical as the number of subregions. These processors, called SLAVEs, take charge of matching the passengers for the taxis in their responsible subregions. In the whole evolution process, the SLAVEs should know which taxis and passengers they can deal with. However, the passengers in critical region could be evolved in both adjacent subregions. As illustrated in Fig. 1(a), there are 4 taxis (T_A~T_D) and 4 passengers (P_1~P_4) in 4 the subregions (A~D). Because P_2~P_4 are in critical region, P_2 can be picked up by T_A and T_B, P_3 can be picked up by T_A and T_C, P_4 can be picked up by T_A~T_D. If one passenger is concurrently evolved in two or more subregions, there is a probability that this passenger is chosen by more than one taxi which will violate the constraint in (8).

So we need a processor, called MASTER, to control the assignment of passengers in critical regions. Initially, all passengers will be assigned to a subregion by running a fast matching process given in Algorithm 1 in MASTER. The passengers that are matched to a certain taxi in the fast matching process would be assigned to the same subregion with the taxi. The other passengers that are not in critical region will be assigned to the subregions they located. The remaining passengers will be assigned to a subregion according to the linear probability distribution of distance to each adjacent subregion. Fig. 1(b) shows an example of possible assignments. As a taxi can only choose the passengers who are assigned to the same subregion with itself, TA can pick up either P_1 or P_2, T_B can pick up P_4, and T_C can pick up P_3.

Moreover, at the end of each iteration, the SLAVEs send their information of best solution to the MASTER, the MASTER gather the evolution result from the SLAVEs to calculate the sum of profit, and then send back the information of the best summation so far to the SLAVEs. The SLAVEs update their global pheromone according to the received the information from the MASTER. After that, the MASTER reallocates the attribution of the passengers in critical region and broadcasts to all the SLAVEs. The next iteration will start after the SLAVEs receive the new attribution of passengers.

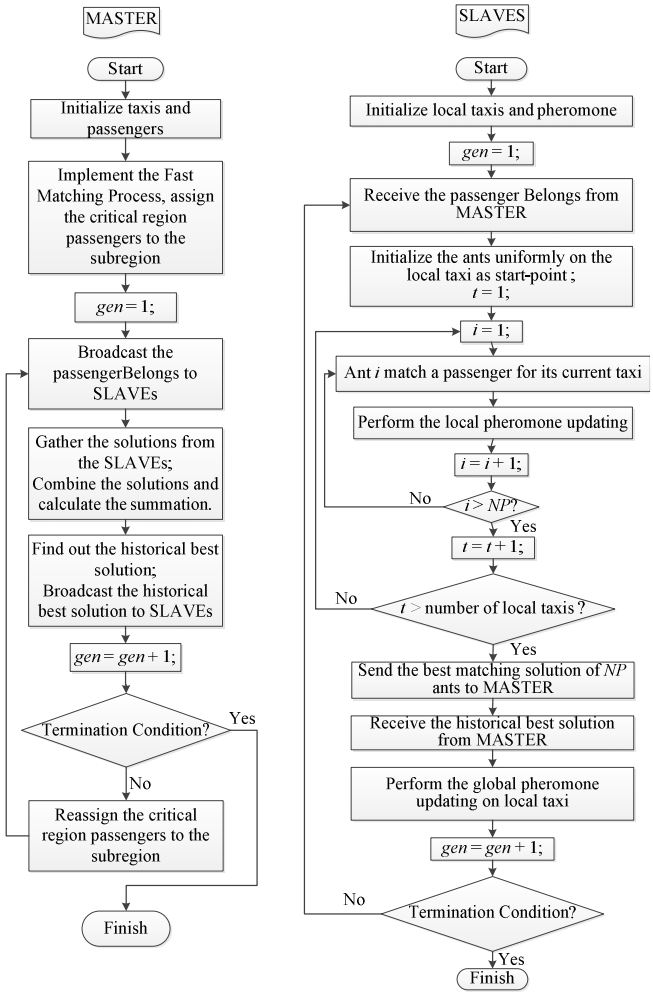


Fig. 2. Flowchart of the RDD-ACS-TPM algorithm. (a) Flowchart of MASTER process. (b) Flowchart of SLAVEs process.

For load balance, we allot the same number of ants to each subregion. In this way, it is beneficial to not only have more probability to get a better result, but it also increases the scalability of dividing the region to finer granularity.

C. *Integrated RDD-ACS-TPM Algorithm*

The RDD-improved ACS algorithm for the TPM (RDD-ACS-TPM) problem is integrated in this section. The flowchart of the integrated RDD-ACS-TPM algorithm is given in Fig. 2. Fig. 2(a) is the process of MASTER and Fig. 2(b) is the process of SLAVEs. In general, it includes the following six steps.

Step 1: Initialization. Initialize parameters, pheromone, ants, and all the profit pair between each taxi and passenger.

Step 2: The MASTER finds all the passengers who locate in the critical region, unless they had matched in the fast matching process.

Step 3: The MASTER randomly assigns passengers in the critical regions to one of the adjacent subregions using a linear probability distribution.

Step 4: The SLAVEs schedule the N_S^{loc} taxis in local subregions using ACS matching process according to (9) and (10) by each ant.

TABLE II.
PARAMETER CONFIGURATIONS FOR THE RDD-ACS-TPM ALGORITHM

Parameters configuration												
NP	***NG***	***CAN***	ρ	α	β	τ_0	q_0	ε	δ	κ	ξ	ζ
$\lfloor \frac{N_S}{400} \rfloor$	1500	10	0.1	0.1	2.0	0.1	0.99	0.50	200	0.10	1.00	0.3077

Step 5: The MASTER gathers all the results from SLAVEs, calculate the global fitness value, and update the global pheromone.

Step 6: Termination check. The MASTER and all the SLAVEs stop the process if one of the termination conditions are met: 1) Variable *gen* has reach the maximal generations; 2) The increment of fitness value is smaller than ε in δ generations. Otherwise, let $gen = gen + 1$ and go to Step 3 for the next generation.

IV. EXPERIMENTS AND COMPARISONS

In this section, the results of experimental tests are carried out. The RDD-ACS-TPM algorithm is compared with a Greedy algorithm in terms of both accuracy and the running time. Note that the Greedy algorithm is implemented over the global perspective, i.e., the algorithm will choose the most profitable pair of taxi-passenger every time, until all taxis are scheduled. All the experiments are implemented in a Linux-based cluster with 14 PCs and each PC has a Core i5-4590 CPU (4 cores) at 3.30GHz with 32 GB random access memory.

A. Test Cases and the Results

There are mainly 2 factors affect the quality of the RDD-ACS-TPM algorithm, the quantity of taxis and passengers, and the intersection number of the testing region. Case 1 is used to investigate the performance of the RDD-ACS-TPM algorithm in the same region, and Case 2 is used to investigate the effect of enlarging the region.

In order to analyze the extensibility of the algorithm, RDD-ACS-TPM is also tested in different granularities of region division. As an extreme case, an ACS-TPM algorithm can be regarded as the non-decomposed version of RDD-ACS-TPM. In other words, ACS-TPM schedules all taxis in the whole region by using ACS algorithm.

The configurations of the related parameters are given in Table II, including the population size *NP* in each subregion, the maximal generation number *NG*, the candidate list number *CAN*, the ratio of critical region κ, the coefficient of the taxi fare ξ, the coefficient of traverse fee ζ, and other ACS-related parameters. The parameters of ACS-related are based on [12], except for q_0, which is suggested to be 0.90. The configuration of parameter q_0 is based on empirical study presented in Sections IV-C.

Note that all the test cases, which include the information of taxis and passengers, are generated randomly in advance, and each scale has at least three different cases to make sure the reliability of the experiments. The information includes the position of vacant taxis, position of passengers and destination of passengers.

1) Case 1 and the Results

The map we adopt as experimental group is the center region of Beijing, depicted in Fig. 3(a), with longitude from 116.34843 to 116.46 and latitude from 39.88 to 39.96. The spots, which represent the taxi stands in practice, are intersections extracted from the map. The result of Greedy and RDD-ACS-TPM algorithm are presented and compared in Table III. In addition, the results of different problem scales are shown in Fig. 4. The points in the figure are drawn according to the average value of three cases under the same scale. It is worth mentioning that the region of RDD-ACS-TPM has been divided into 4 segments both in longitude and latitude (i.e., a whole region is divided into 16 subregions).

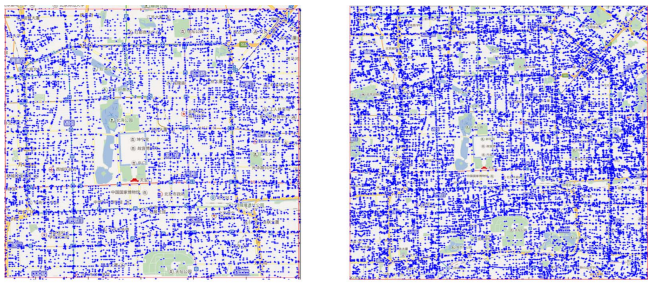

(a) (b)

Fig. 3. Map of Beijing for experiments.
(a) Experimental group: longitude from 116.34843 to 116.46 and latitude from 39.88 to 39.96.
(b) Control group: longitude from 116.329538 to 116.48 and latitude from 39.86 to 39.98.

From Table III, it can be observed that, overall, RDD-ACS-TPM obtains both better solution and less time. Comparing the mean profit, RDD-ACS-TPM performs better in all scale, and the increment is increases with the problem scale. However, there are a few special cases whose worst result is worse than those of the Greedy algorithm, such as case 1 and 3 when the passenger scale is 2100. They may due to the premature convergence of RDD-ACS-TPM. Whenever the increment of fitness value is smaller than ε in δ generations, the process will consider the evolution has stopped. This mechanism is aimed at reducing the running time when the evolution has completely converged. However, there may be a few 'unlucky' situations that the colony converges to a local optimum in the early period. From the value of mean profit, we can see that this situation seldom happens as the mean value is more close to the best value.

From Fig. 4(d) to (f), we can see that, as the number of passengers grows, the Greedy algorithm requires longer running time, while RDD-ACS-TPM has stable running time. It is important to note that, according to Table III, the worse running time of RDD-ACS-TPM algorithm is also half of Greedy's best.

2) Case 2 and the Results

In this case, the map scale is twice of that in Case 1. Shown in Fig. 3(b), its longitude is from 116.329538 to 116.48 and latitude is from 39.86 to 39.98. Fig. 5(a) to (f) are drawn according to the average value of three cases under the same scale of Greedy and RDD-ACS-TPM algorithms results.

From Fig. 5, we can see that using a bigger map in Fig. 3(b) barely impacts the effect of both two algorithms, as Fig. 5 is very similar to Fig. 4.

B. Analysis of Speedup and Scalability

In this part, the scalability of RDD-ACS-TPM is tested, and we present the speedup of RDD-ACS-TPM algorithm. We carried out 30 independent runs for each case in the taxi scale of 4000 with different granularities of segmentation, while the

TABLE III
EXPERIMENTAL RESULT COMPARISONS IN CASE 1 WITH DIFFERENT SCALE OF TAXI

taxi scale	passenger scale	case	profit				running time (sec)					
				Mean	*Best*	*Worst*	*Mean*		*Best*		*Worst*	
			Greedy	*rdd-acs-tpm (16 subregions)*	*rdd-acs-tpm (16 subregions)*	*rdd-acs-tpm (16 subregions)*	*Greedy*	*rdd-acs-tpm (16 subregions)*	*Greedy*	*rdd-acs-tpm (16 subregions)*	*Greedy*	*rdd-acs-tpm (16 subregions)*
2000	2100 (+5%)	1	1247.53	**1256.74**	1259.74	1247.05	22.4885	**11.5861**	22.1926	3.9627	24.4069	12.4391
		2	1250.56	**1264.34**	1267.18	1255.24	22.1881	**11.325**	21.9314	4.3047	24.3329	13.2331
		3	1226.6	**1231.93**	1238.23	1225.97	22.5484	**8.76021**	22.3117	2.67396	24.5279	13.6086
	2200 (+10%)	1	1265.25	**1281.94**	1282.99	1280.3	23.6502	**12.8832**	23.4345	10.7821	25.7862	14.7588
		2	1253.99	**1268.46**	1271.61	1264.07	23.6392	**11.5947**	23.4	6.76677	25.8183	12.5822
		3	1264.78	**1279.94**	1281.04	1278.44	23.8543	**12.1041**	23.6204	9.73755	25.9418	13.3763
	2300 (+15%)	1	1282.71	**1292.44**	1293.27	1291.27	25.0097	**11.228**	24.5601	8.57642	27.0843	13.3501
		2	1278.06	**1289.54**	1291.22	1287.06	24.9706	**12.7855**	24.7272	10.054	27.1232	15.0468
		3	1274.44	**1285.77**	1287.58	1282.51	24.9076	**13.592**	24.658	9.62677	27.1673	14.662
	2400 (+20%)	1	1289.14	**1299**	1300.01	1297.76	25.5209	**12.3902**	25.2581	9.09614	27.96	15.7569
		2	1293.6	**1302.75**	1303.82	1301.58	25.6781	**11.2029**	25.4273	8.64556	28.0886	14.0574
		3	1293.58	**1304.08**	1304.84	1303.12	25.6783	**11.181**	25.4089	8.28082	28.078	13.47
3000	3150 (+5%)	1	1911.42	**1925.65**	1929.22	1915.82	76.5814	**38.7763**	75.9251	21.3053	82.9381	40.1469
		2	1897.07	**1918.44**	1921.8	1910.51	76.9095	**34.9685**	76.203	17.1397	83.2099	36.9779
		3	1897.31	**1928.5**	1931.74	1919.91	76.6263	**33.3317**	75.9182	16.4762	83.0529	35.471
	3300 (+10%)	1	1926.96	**1949.48**	1950.99	1944.99	80.8535	**39.3786**	80.1094	29.8607	87.6955	40.8662
		2	1936.33	**1956.73**	1958	1954.58	80.8413	**36.0215**	80.115	28.4382	87.6817	38.8449
		3	1931.97	**1954.5**	1955.94	1951.38	81.8626	**41.5364**	81.0976	36.8272	88.45	42.5729
	3450 (+15%)	1	1947.61	**1967.63**	1968.67	1966.47	84.2886	**38.6078**	83.5291	32.5057	91.6783	44.4536
		2	1955.88	**1976.2**	1976.89	1974.36	84.4178	**34.9348**	83.6537	25.9661	91.7295	43.0266
		3	1954.37	**1972.23**	1973.05	1970.08	84.8626	**38.1238**	84.122	32.1249	92.0272	44.1105
	3600 (+20%)	1	1967.35	**1979.97**	1980.61	1979.25	87.5633	**35.2581**	86.8268	27.9104	95.2809	43.2914
		2	1968.86	**1983.65**	1984.53	1981.65	88.7724	**40.9418**	87.9438	34.7791	96.3907	47.4262
		3	1953.55	**1969.07**	1969.93	1967.65	87.9175	**35.2181**	87.0972	27.9608	95.5019	41.1761
4000	4200 (+5%)	1	2556.83	**2593.84**	2597.58	2588.36	184.213	**95.9508**	182.828	59.908	198.659	99.7557
		2	2559.49	**2570.51**	2581.3	2559.23	184.299	**74.7332**	182.848	18.9259	198.583	95.5478
		3	2566.46	**2587.85**	2595.75	2568.59	184.116	**96.1397**	182.652	20.6753	198.949	103.725
	4400 (+10%)	1	2601.78	**2626.34**	2629.08	2623.95	192.117	**104.992**	190.459	99.0993	207.896	191.504
		2	2588.87	**2617.1**	2621.49	2610.7	193.1	**98.8697**	191.548	57.1328	208.186	107.47
		3	2599.55	**2629.55**	2630.58	2624.69	193.919	**104.43**	192.313	92.6316	209.286	107.409
	4600 (+15%)	1	2623.31	**2646.38**	2647.42	2644.53	202.165	**96.3396**	200.475	80.2067	218.317	109.307
		2	2623.73	**2646.08**	2646.87	2644.01	201.376	**101.375**	199.717	85.0373	218.543	107.047
		3	2625.36	**2649.99**	2651.22	2647.6	201.292	**93.0845**	199.572	76.0958	217.75	103.882
	4800 (+20%)	1	2648.32	**2664.61**	2665.21	2663.21	209.169	**88.6413**	207.459	66.8558	227.764	103.727
		2	2640.81	**2658.71**	2659.24	2656.98	209.954	**95.45**	208.119	76.394	227.858	114.987
		3	2643.44	**2660.78**	2661.63	2658.87	210.498	**91.7538**	208.681	74.6502	228.288	112.114

other configurations are kept as the same as Section IV-A. The mean results of each scale are plotted in Fig. 6.

From Fig. 6(a) and (b), we can see that, as the number of subregions increases, the running time overall presents a decreasing tendency in each passenger scale, while the profit gradually maintains on a certain level. The leftmost point of each line in Fig. 6(a) also represents the profit of using ACS-TPM which means scheduling all taxis in the whole region by using ACS algorithm. We also present the speedup of RDD-ACS-TPM algorithm in Fig. 6(c). The speedup is calculated as *speedup*(N) = ACS-TPM / RDD-ACS-TPM(N) where N represents the number of subregions (cores). From Fig. 6(c), it can be observed that, as a whole, the speedup has an ascend trend, but there is an 'ebb' when the number of subregions is 9. It is due to a phenomenon of insufficient evolution. For example, when the number of subregion is 4, there are about 1000 taxis in each subregion, and the evolving process will be too slow. An interrupt may perform in early phase to save the running time, because the increment of fitness value is easily smaller than ε in δ generations. On the contrary, when the number of subregions is 9, the evolving process mostly stops when it meets the maximal generations. From Fig. 6(a) and (b), running the algorithm in 9 subregions has earned more profits but costed more time than in 4 subregions. But overall, as the number of subregions/cores increases, the running time of the parallel algorithm has appeared a decreasing trend while profit value has showed smooth and stable.

C. Parameter Investigation

In the RDD-ACS-TPM algorithm, q_0 is a crucial parameter. The investigation of parameter q_0 is helpful to find a suitable configuration for the algorithm as well as to improve the performance. We set q_0 equals to 0.90, 0.95, 0.98, 0.99, and 1.00 respectively (note that 0.90 is suggested in [12]). For each configuration, the algorithm is used to optimize the preceding test cases group whose taxi scale is 3000 in Case 1, and each test case will carry out 30 independent runs. The results are presented in Table IV and Fig. 7.

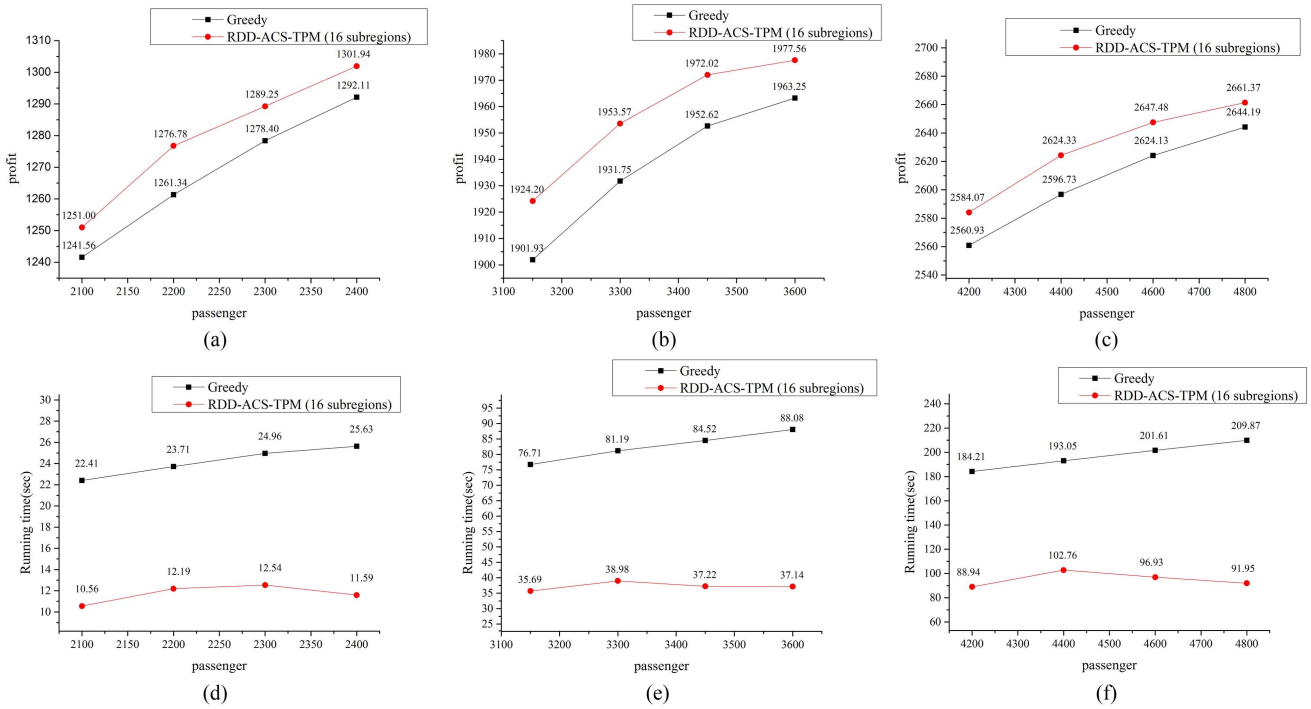


Fig. 4. Average experimental result in case 1 with different scale of taxi. (a)~(c) Average optimal profit. (d)~(f) Average running time.

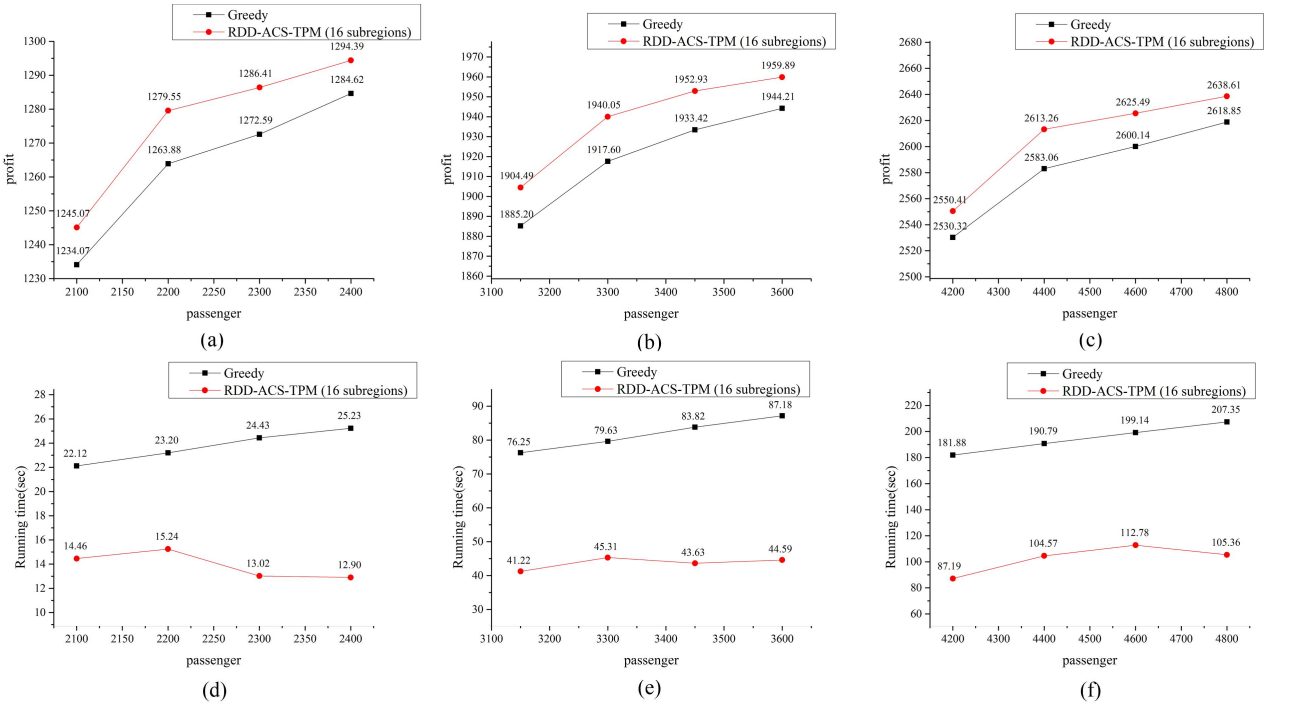
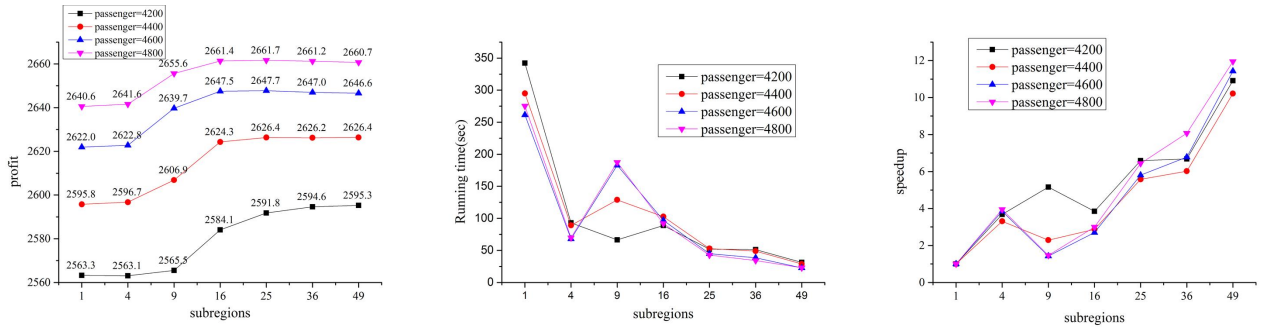


Fig. 5. Average experimental result in case 2 with different scale of taxi. (a)~(c) Average optimal profit. (d)~(f) Average running time.

From Fig. 7, it is obvious that the RDD-ACS-TPM algorithm obtains the highest profit in every passenger scale when the parameter q_0 is set to 0.99. Although the profit value is increasing from 0.90 to 0.99, q_0 still can't be set to 1.00. Because it will make the algorithm lose the ability of exploration and get bad results with this configuration.

V. CONCLUSION

In this paper, we have modeled the Taxi-Passenger Matching (TPM) problem and proposed a new parallel framework to solve the TPM problem. The Ant Colony System (ACS) algorithm has been developed by incorporating a Region-Dependent



(a) (b) (c)

Fig. 6. Influence of the number of subregions and CPU cores on RDD-ACS-TPM in the taxi scale of 4000. (a) Average optimal profit. (b) Average running time. (c) Speedup of RDD-ACS-TPM.

TABLE IV EXPERIMENTAL RESULT COMPARISONS OF DIFFERENT q_0 ON RDD-ACS-TPM

Taxi scale	Passenger scale	Value of parameter q_0				
		0.90	*0.95*	*0.98*	*0.99*	*1.00*
3000	3150	1894.98	1897.63	1909.97	**1924.20**	1902.04
	3300	1925.13	1927.57	1945.15	**1953.57**	1931.58
	3450	1947.93	1950.24	1966.24	**1972.02**	1953.33
	3600	1956.27	1958.62	1972.40	**1977.56**	1961.51

Decomposition (RDD) strategy for divide-and-conquer. Further, the algorithm is parallelized by applying the MPI to coordinate the different regions. With the help of critical region strategy, the proposed algorithm solves the TPM problem effectively at the global level. From the experimental results, RDD-ACS-TPM outperforms the traditional Greedy algorithm in terms of both optimal value and running time. From the speedup analysis, the algorithm shows a good scalability and stability.

The future research work includes the following aspects: 1) extending the algorithm to a dynamical version; 2) taking quality of service measure from the passenger perspective into consideration; and 3) applying the algorithm to tackle the real data from a practical taxi company.

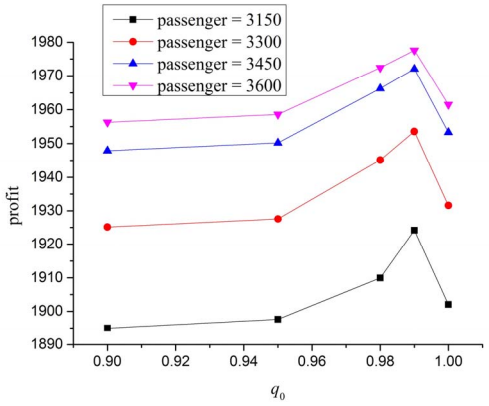


Fig. 7. Influence of the parameter q_0 on RDD-ACS-TPM

REFERENCES

[1] B. Leng, H. Du, J. Wang, L. Li and Z. Xiong, "Analysis of Taxi Drivers' Behaviors Within a Battle Between Two Taxi Apps," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 1, pp. 296-300, Jan. 2016.

[2] K. T. Seow and D. H. Lee, "Performance of Multiagent Taxi Dispatch on Extended-Runtime Taxi Availability: A Simulation Study," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 1, pp. 231-236, March 2010.

[3] K. T. Seow, N. H. Dang and D. H. Lee, "A Collaborative Multiagent Taxi-Dispatch System," in *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 3, pp. 607-616, July 2010.

[4] M. Maciejewski, J. Bischoff and K. Nagel, "An Assignment-Based Approach to Efficient Real-Time City-Scale Taxi Dispatching," in *IEEE Intelligent Systems*, vol. 31, no. 1, pp. 68-77, Jan.-Feb. 2016.

[5] Z. Liao, "Taxi dispatching via Global Positioning Systems," in *IEEE Transactions on Engineering Management*, vol. 48, no. 3, pp. 342-347, Aug 2001.

[6] J. Yuan, Y. Zheng, X. Xie and G. Sun, "T-Drive: Enhancing Driving Directions with Taxi Drivers' Intelligence," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 1, pp. 220-232, Jan. 2013.

[7] X. Xu, J. Zhou, Y. Liu, Z. Xu and X. Zhao, "Taxi-RS: Taxi-Hunting Recommendation System Based on Taxi GPS Data," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1716-1727, Aug. 2015.

[8] N. J. Yuan, Y. Zheng, L. Zhang and X. Xie, "T-Finder: A Recommender System for Finding Passengers and Vacant Taxis," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 10, pp. 2390-2403, Oct. 2013.

[9] D. Zhang *et al.*, "Understanding Taxi Service Strategies From Taxi GPS Traces," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 123-135, Feb. 2015.

[10] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira and L. Damas, "Predicting Taxi–Passenger Demand Using Streaming Data," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1393-1402, Sept. 2013.

[11] F. Miao *et al.*, "Taxi Dispatch With Real-Time Sensing Data in Metropolitan Areas: A Receding Horizon Control Approach," in *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 463-478, April 2016.

[12] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," in *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53-66, Apr 1997.

[13] Z. H. Zhan *et al.*, "An Efficient Ant Colony System Based on Receding Horizon Control for the Aircraft Arrival Sequencing and Scheduling Problem," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 2, pp. 399-412, June 2010.

[14] W. N. Chen and J. Zhang, "An Ant Colony Optimization Approach to a Grid Workflow Scheduling Problem With Various QoS Requirements," in *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 39, no. 1, pp. 29-43, Jan. 2009.

[15] Y. J. Gong *et al.* , "Distributed evolutionary algorithms and their models," in *Applied Soft Computing*, vol. 34, no. 1, pp. 286-300, Sep. 2015.