Distributed Game-Theoretical Route Navigation for Vehicular Crowdsensing

En Wang
Jilin University
Changchun, China
wangen@jlu.edu.cn

Dongming Luan
Jilin University
Changchun, China
luandm17@mails.jlu.edu.cn

Yongjian Yang
Jilin University
Changchun, China
yyj@jlu.edu.cn

Zihe Wang
Renmin University of China
Beijing, China
wang.zihe@ruc.edu.cn

Pengmin Dong
Jilin University
Changchun, China
dongpm19@mails.jlu.edu.cn

Dawei Li
Montclair State University
Montclair, USA
dawei.li@montclair.edu

Wenbin Liu*
Jilin University
Changchun, China
liuwenbin@jlu.edu.cn

Jie Wu
Temple University
Philadelphia, USA
jiewu@temple.edu

ABSTRACT

Vehicular CrowdSensing (VCS) has become a powerful sensing paradigm by selecting users driving vehicles to perform tasks. In most existing research, the platform centrally allocates tasks according to the collected user information. We argue that the information collection process results in user privacy leakage, and the centralized allocation leads to a heavy computation complexity. We propose to apply a distributed task allocation method in the widely-used route navigation system. The navigation system recommends several routes to a user and each route may cover some tasks. Then, the user distributively selects a route according to the route profit (task reward minus route cost). Since the task reward is shared by users, the route selections of users may influence each other. Hence, it remains unclear how to design a distributed route navigation approach to reach an equilibrium state (i.e., each user is satisfied with the selected route), while guaranteeing a good total profit. To this end, we formulate the problem as a multi-user potential game and propose a distributed route navigation algorithm. The trace-based simulation results verify that the proposed algorithm achieves a Nash equilibrium, while achieving a total user profit performance close to that of the optimal solution.

KEYWORDS

Vehicular CrowdSensing, route navigation, potential game, Nash equilibrium.

*The corresponding author is Wenbin Liu.

ICPP '21, August 9–12, 2021, Lemont, IL, USA

ACM ISBN 978-1-4503-9068-2/21/08…$15.00
https://doi.org/10.1145/3472456.3472498

ACM Reference Format:
En Wang, Dongming Luan, Yongjian Yang, Zihe Wang, Pengmin Dong, Dawei Li, Wenbin Liu, and Jie Wu. 2021. Distributed Game-Theoretical Route Navigation for Vehicular Crowdsensing. In *50th International Conference on Parallel Processing (ICPP '21), August 9–12, 2021, Lemont, IL, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3472456.3472498

1 INTRODUCTION

With the proliferation of mobile devices, Mobile CrowdSensing (MCS) [18, 23, 30], has become a powerful sensing technique. As a typical paradigm of MCS, Vehicular CrowdSensing (VCS) [9, 17], which takes advantage of the mobility of vehicles to perform sensing tasks in large-scale areas, has received much attention.

In VCS, the platform needs to assign tasks to suitable users, which raises the fundamental task allocation problem [12, 26]. Most of the existing task allocation strategies are centralized [10, 15, 19, 25, 28, 31], i.e., the platform centrally collects the users' information and makes the task allocation decision. We argue that the information collection process results in privacy leakage for the users, and the centralized allocation leads to a heavy computation complexity for the platform. More importantly, the centralized task allocation may fail to satisfy all the users with the allocation results. For example, a user may be unwilling to deviate from the original route to perform a remote allocated task, even though there is a good task reward. On the other hand, if users perform sensing tasks totally according to their own plans, some tasks with low rewards or remote locations may not be finished.

In light of this, considering the drivers usually utilize the map navigation systems (e.g., Google Maps [16]), we are inspired to consider a distributed task allocation with the help of the route navigation. As shown in Fig. 1 (left part), when the users input the initial locations and the destinations in their smart phones, several routes are recommended to them. Each route may cover some MCS tasks. When a user selects a route, it can complete the tasks on this route and get the corresponding task rewards. In this way, the users distributively select routes to perform tasks instead of uploading their information to a centralized platform

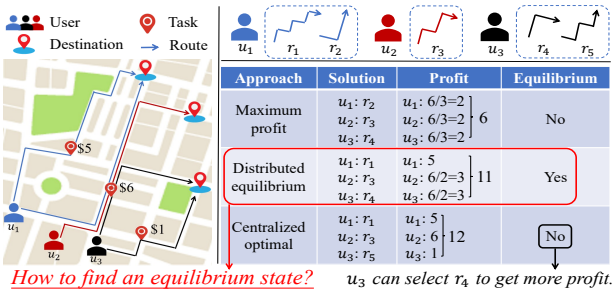


Figure 1: An illustrative example.

and deviating from the daily routes. Even so, it remains unclear how to guide users to select the suitable routes in order to maximize the users' profits (task reward minus route cost). Especially when the reward of each task is equally shared by users [6], the route selections of participating users may influence each other. To this end, some obvious route selection approaches are shown in Fig. 1 (right part). An intuitive idea (Maximum reward) for each user is to select the route with the maximum reward. However, this leads to the least total reward ($6), because users need to share the reward of their overlapped task. An ideal approach (Centralized optimal) achieves the highest total reward ($12). However, it does not reach an equilibrium state because u_3 can select route r_4 to get more profit. A trade-off (Distributed equilibrium) is our target solution, where a relatively high reward ($11) and an equilibrium state are achieved, and no user has the incentive to change the decision to get more profit unilaterally. Hence, the problem in this paper is how to design a distributed route navigation approach to reach an equilibrium state while guaranteeing a good profit performance.

Some existing works [5, 6] have proposed to use distributed algorithms to solve the multi-user task allocation problem. However, they do not take the users' original routes into consideration. Hence, the users still have to *deviate from their daily routes* to finish tasks. Moreover, these works also fail to consider *the users' individual preferences* such as task reward, detour distance and congestion level. To the best of our knowledge, this is the first research utilizing the route navigation to perform the MCS task allocation. Hence, there must be some unresolved technical challenges. Our purpose is to find the distributed equilibrium state in route navigation problem. Even if we could prove the existence of an equilibrium state, how to construct a distributed model to achieve *the equilibrium while guaranteeing the profit performance* is the first challenge. Moreover, during the route navigation process, each user has an individual preference (e.g., task reward, detour distance and congestion level). Similarly, the platform also has its specific task allocation purpose (e.g., maximizing task completion). Hence, the second challenge is how to design *a unified distributed algorithm* such that it could take the requirements of both the platform and users into consideration. Finally, even if we find the distributed equilibrium state, the profit performance is not always optimal (as shown in Fig. 1). Hence, the third challenge is how to *guarantee a lower performance bound* with respect to the centralized optimal solution.

To deal with the above challenges, we first formulate the route navigation problem as a multi-user route navigation game, where each user selects a route to maximize its own profit function separately. Then, we prove that the formulated game is a potential game by constructing a global potential function. The change of the profit functions of all the users can be uniformly mapped into the change of the global potential function. Through continuing to approach the maximum value of the global potential function, we achieve an equilibrium state where each user's profit function achieves a local maximum value. Furthermore, we design a distributed game-theoretical route navigation algorithm to achieve the Nash equilibrium. For the profit function, the weighting parameters could be modified by the users based on their individual preferences, as well as by the platform according to the task allocation purpose. Finally, we utilize the metric of Price of Anarchy to guarantee the lower bound of total user profit performance to the centralized optimal solution.

In summary, the contributions are listed as follows:

- *Multi-User Route Navigation Game Formulation:* We first prove that it is NP-hard to find the centralized optimal solution of the route navigation problem in MCS. Instead, we formulate the distributed route navigation problem as a multi-user route navigation game. To the best of our knowledge, it is the first research utilizing the route navigation to perform the task allocation in MCS.
- *Distributed Route Navigation Algorithm:* We prove that the formulated multi-user game is a potential game. Furthermore, we design a distributed route navigation algorithm to reach an equilibrium state, where users could modify the parameters of the profit function to satisfy the individual preferences and the platform could also do the same to achieve the specific task allocation purpose.
- *Theoretical Performance Analysis:* We show that the proposed distributed route navigation algorithm can converge to a Nash equilibrium within a finite number of update steps. Furthermore, we derive the upper bound for the number of update steps and the lower bound for the total user profit with respect to that of the optimal centralized solution.
- *Extensive Trace-based Simulations:* We perform extensive trace-based simulations based on three data sets. The results verify that our proposed algorithm can achieve a Nash equilibrium, while achieving a total user profit close to that of the optimal solution.

2 RELATED WORKS

2.1 Task allocation

The research on the task allocation in MCS can be classified into two categories: the centralized task allocation and the distributed task allocation. For the former [20, 24, 27], Wang *et al.* [24] propose a time-sensitive task allocation approach, which can plan a task execution path to the participant. To tackle the contradiction between user privacy and task allocation in MCS, Ni *et al.* [20] propose a privacy-preserving task allocation scheme that considers users' personal features. Wang *et al.* [27] utilize the spatio-temporal correlation of the sensing tasks to design a multi-task assignment strategy. For the latter [2, 3, 7], Cao *et al.* [3] propose a game-theoretical incentive mechanism and an auction based task migration algorithm, which guarantees the truthfulness, individual

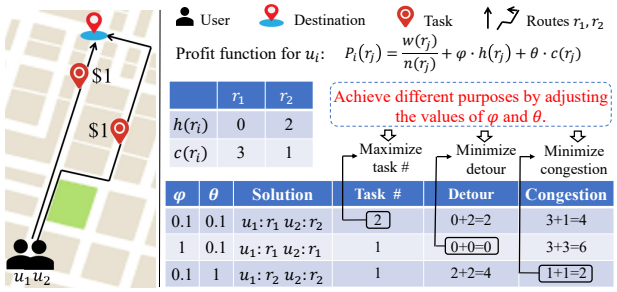


Figure 2: An illustrative example of the influence of ϕ and θ.

rationality, and computational efficiency. Dai *et al.* [7] construct a distributed matching model and design the budget constrained task allocation strategies for MCS based on stable matching theory. Cai *et al.* [2] investigate the joint problem of sensing task assignment and schedule with multi-dimensional task diversity. Furthermore, they propose a distributed auction scheme where each task owner can locally process the auction procedure. However, all the above works fail to consider whether a user is satisfied with the allocation results, i.e., a user may be unwilling to deviate from the original route to perform a remote task. By comparison, in this paper, we propose a distributed route navigation algorithm to utilize the route navigation to perform the MCS task allocation, where each user completes the tasks on their own selected routes.

2.2 Potential Game

Recently, a lot of studies utilize the potential game theory to derive the distributed game-theoretical decisions and achieve the Nash equilibria. Fabiani *et al.* [8] formulate the multi-vehicle driving coordination problem as a mixed-integer potential game and find an equilibrium solution. Hong *et al.* [13] formulate the computation offloading problem as a potential game in which the mobile devices make the offloading decisions in a distributed manner. Raschellà *et al.* [22] propose a novel access point selection approach based on a potential game relying on software-defined networking. Chen *et al.* [29] investigate a multi-user computation offloading problem in mobile-edge cloud computing. Furthermore, they propose a potential game theoretical approach to achieve efficient computation offloading. He *et al.* [11] formulate the edge user allocation problem as a potential game and propose a decentralized algorithm to serve the maximum number of users with minimum overall system cost. However, in the above research, the corresponding potential game does not take the diverse requirements of both the platform and mobile users into consideration. In this paper, we propose a distributed game-theoretical approach, where both the platform and mobile users can achieve different purposes by adjusting the parameters of the profit function.

3 MULTI-USER ROUTE NAVIGATION GAME

3.1 System Model

We first introduce the system model for the route navigation in MCS. We assume that the number of users is enough to cover most of tasks. When a user reports the initial location and destination to the platform, the platform recommends some available routes to the user. Moreover, each route may cover some MCS tasks. The user can check the recommended routes and the covered tasks on the smart phone. Let $\mathcal{U} = \{1, 2, \cdots, M\}$ be the set of users and $\mathcal{L} = \{1, 2, \cdots, N\}$ be the set of tasks covered by the routes of users. In our scenario, each user i will receive a set of available routes from the platform, denoted as R_i and each route $r \in R_i$ may cover a set of MCS tasks, denoted as $\mathcal{L}_r$. Each task $k \in \mathcal{L}$ is associated with a reward w_k, which is defined as follows:

$$w_k(x) = a_k + \mu_k \cdot \ln x, \tag{1}$$

where x is the number of users performing the task k, a_k is the reward when there is only one user performing the task, and μ_k is the weight parameter measuring the reward increment with the growth of the number of users, which is set to be $[0, 1]$. Since the task completion quality is improved when receiving multiple results from different users, we design the reward function as follows: more users performing the sensing task results in a sightly improvement on the reward, which matches the practical scenario.

Given the strategy profile of all users $\mathbf{s} = (s_i, s_{-i})$, where s_i denotes the route decision of user i and s_{-i} represents the route decisions of all the users except user i, the profit of a user i under the strategy profile $\mathbf{s}$ is shown as follows:

$$P_i(\mathbf{s}) = \alpha_i \cdot \sum_{k \in \mathcal{L}_{s_i}} w_k(n_k(\mathbf{s}))/n_k(\mathbf{s}) - \beta_i \cdot d(s_i) - \gamma_i \cdot b(s_i), \tag{2}$$

where $\alpha_i, \beta_i, \gamma_i$ are the user weight parameters used to measure the preference of the user, which are adjusted by the user. $e_{min} < \alpha_i, \beta_i, \gamma_i < e_{max}$, where $e_{min} > 0$. For example, if user i prefers a high reward, it can increase the value of α_i. We use $n_k(\mathbf{s})$ to represent the number of users performing task k under strategy profile $\mathbf{s}$. $d(s_i)$ is the cost incurred by traveling the detour distance of s_i, which is defined as follows:

$$d(s_i) = \varphi \cdot h(s_i), \tag{3}$$

where $h(s_i)$ is the detour distance of the selected route s_i compared to the shortest route between the initial location and the destination. φ is a weight parameter adjusted by the platform, where $0 < \varphi < 1$. $b(s_i)$ in Eq. (2) is the cost incurred by the congestion on the route s_i, which is defined as follows:

$$b(s_i) = \theta \cdot c(s_i), \tag{4}$$

where $c(s_i)$ is used to measure the congestion level of the selected route s_i. θ is a weight parameter controlled by the platform and $0 < \theta < 1$. Since the number of mobile users is finite, it has little impact on the congestion level of a route. Hence, we assume that the congestion level of route strategy s_i selected by user i is irrelevant to other users' route decisions. In other words, the congestion level of a route is the same under different route strategy profiles.

Through adjusting the values of φ and θ, the platform can achieve different task allocation purposes. As shown in Fig. 2, we consider a simple case to demonstrate the influence of the weight parameters φ and θ, where each route only contains one task and the two users are at the same initial location. We observe the number of covered tasks, the detour distance and the congestion level with the change of φ and θ. The platform can decrease the values of both φ and θ to maximize the number of tasks covered by users. Moreover, by increasing the values of φ and θ, the platform can guide users to select the routes with short detour distance and low congestion level, respectively. Similarly, user i can also achieve the individual

Table 1: Main notations

Symbol	Meaning
$\mathcal{U}$, $\mathcal{L}$	the sets of mobile users, MCS tasks.
$\mathcal{L}_r$	the set of tasks that is covered by route r
$\mathbf{s}$, s_i, s_{-i}	the strategy profile, the strategy of user i, and the strategies of users except user i.
R_i	the recommended route set of user i.
$P_i(\mathbf{s})$	the profit of user i under the strategy profile $\mathbf{s}$.
$n_k(\mathbf{s})$	the number of users performing the task k under strategy profile $\mathbf{s}$.
$w_k(x)$	the reward of task k when the number of users performing the task is x.
$d(s_i)$, $b(s_i)$	the cost incurred by detour distance, and the congestion level under strategy s_i.
$h(s_i)$, $c(s_i)$	the detour distance, and the congestion level under strategy s_i.
α_i, β_i, γ_i	the weight parameters controlled by user i.
φ, θ	the weight parameters controlled by the platform.
$\phi(\mathbf{s})$	the potential function.

preference by adjusting the values of α_i, β_i, γ_i in Eq. (2). In other words, when the platform has set the system parameters (φ and θ), user i can adjust its user weight parameters (α_i, β_i and γ_i) under these settings of the platform to achieve the individual preference.

3.2 NP-hardness of The Centralized Problem

We first consider the centralized optimization problem of finding a solution to maximize the total profit of all users. Mathematically, given the strategy profile $\mathbf{s} = (s_i, s_{-i})$, the problem can be formulated as follows:

$$\max_{\mathbf{s}} \sum_{i\in\mathcal{U}} P_i(\mathbf{s}), \quad \text{subject to } s_i \in R_i, \forall i \in \mathcal{U}. \tag{5}$$

Then, we try to prove that finding the optimal solution of the formulated centralized optimization problem is quite difficult, as shown in Theorem 1.

Theorem 1. *The problem of finding the solution with the maximum total profit in a centralized manner is NP-hard.*

Proof. The main idea is to reduce the maximum set cover problem, which is NP-complete [4], to a special case of our centralized profit maximization problem.

First, we give a definition of the maximum set cover problem. Given a universal set E of n elements, a collection of subsets of E, and an integer h, select h subsets so as to maximize the number of covered elements.

Then, we construct a special case of our centralized profit maximization problem with the following restrictions. Let $\mu_k = 0, a_k = a, \forall k \in \mathcal{L}$; that is, the reward of all tasks is a fixed value and exactly the same. Moreover, all users have the same recommended route set. Finally, set φ, θ to zero, and set $\alpha_i = 1, \forall i \in \mathcal{U}$. Correspondingly, $P_i(\mathbf{s}) = \sum_{k\in\mathcal{L}_{s_i}} \frac{a}{n_k(\mathbf{s})}$.

In the constructed special case, each user only chooses one route. Thus, the total number of the selected routes is equivalent to the number of users. Since the reward of all tasks is the same, we can maximize the total profit by selecting the routes to cover the maximum number of tasks. The tasks can be viewed as the elements in the maximum set cover problem. Each recommended route covers a subset of tasks, which has a one-to-one correspondence with the subset of elements in the maximum set cover problem. Therefore, the maximum set cover problem is reduced to a special case of the centralized maximization problem and the problem is NP-hard. □

According to the proof of Theorem 1, finding an optimal solution to our problem in a centralized manner is extremely difficult. Hence, we turn to consider a distributed mechanism with low computation complexity. An idea comes that game theory can be applied for the distributed scenarios and lead to an equilibrium state. This inspires us to formulate the route navigation problem as a multi-user route navigation game.

3.3 Potential Game Formulation

To investigate the existence of Nash equilibrium, we try to formulate the multi-user route navigation game as a weighted potential game. Before the description about the potential game formulation, we first introduce some definitions.

Definition 1. ***(Better and best response update)*** *For a strategy profile $\mathbf{s} = (s_i, s_{-i})$, in the better response update, user i changes the strategy from $s_i \in R_i$ to $s_i' \in R_i$, which can lead to an increase of its profit, i.e., $P_i(s_i', s_{-i}) > P_i(s_i, s_{-i})$. The best response update is a special type of the better response update. Each user $i \in \mathcal{U}$ will select a new strategy s_i', which maximizes the profit among all better response updates.*

Definition 2. ***(Nash equilibrium)*** *In the multi-user route navigation game, a strategy profile $\hat{\mathbf{s}} = (\hat{s}_1, \cdots, \hat{s}_M)$ is a Nash equilibrium if and only if*

$$P_i(\hat{s}_i, \hat{s}_{-i}) = \max_{s_i\in R_i} P_i(s_i, \hat{s}_{-i}) \ \ \forall i \in \mathcal{U}. \tag{6}$$

It is obvious that no user can improve the profit by altering the strategy unilaterally in a Nash equilibrium.

Definition 3. ***(Weighted potential game)*** *A game is defined as a weighted potential game if and only if there exists a potential function $\phi(\mathbf{s})$ such that $\forall i \in \mathcal{U}, \forall s_i, \forall s_i' \in R_i, \forall s_{-i} \in \prod_{j\neq i} R_j$:*

$$P_i(s_i, s_{-i}) - P_i(s_i', s_{-i}) = w_i(\phi(s_i, s_{-i}) - \phi(s_i', s_{-i})), \tag{7}$$

where $(w_i)_{i\in\mathcal{U}}$ constitutes a vector of positive numbers.

The potential game has two significant properties: (1) *Nash equilibrium existence:* there always exists at least one Nash equilibrium in the potential game. (2) *Finite improvement property:* the potential game always converges to a Nash equilibrium in a finite number of decision steps when taking better/best response updates, irrespective of the initial strategy profile or the users' updating order.

Then, Theorem 2 shows that the multi-user route navigation game is a weighted potential game.

Theorem 2. *The multi-user route navigation game is a weighted potential game and has a Nash equilibrium and the finite improvement property.*

Proof. We first construct the potential function as follows:

$$\phi(\mathbf{s})=\sum_{k\in\mathcal{L}}\sum_{q=1}^{n_k(\mathbf{s})}\frac{w_k(q)}{q}-\sum_{i\in\mathcal{U}}\frac{\beta_i}{\alpha_i}d(s_i)-\sum_{i\in\mathcal{U}}\frac{\gamma_i}{\alpha_i}b(s_i). \tag{8}$$

We define the original strategy profile as $\mathbf{s}=(s_i, s_{-i})$ and a new strategy profile $\mathbf{s}'=(s_i', s_{-i})$ after user i changes the decision from s_i to s_i'. We divide the set of tasks $\mathcal{L}$ into three non-overlapping portions: $\mathcal{L}_1=\{k : k \in \mathcal{L}_{s_i}, k \in \mathcal{L}_{s_i'}\}$, $\mathcal{L}_2=\{k : k \in \mathcal{L}_{s_i}, k \notin \mathcal{L}_{s_i'}\}$, and $\mathcal{L}_3=\{k : k \notin \mathcal{L}_{s_i}, k \in \mathcal{L}_{s_i'}\}$.

Then, we let F defined as follows:

$$F=\sum_{i\in\mathcal{U}}\frac{\beta_i}{\alpha_i}d(s_i')+\sum_{i\in\mathcal{U}}\frac{\gamma_i}{\alpha_i}b(s_i')-\sum_{i\in\mathcal{U}}\frac{\beta_i}{\alpha_i}d(s_i)-\sum_{i\in\mathcal{U}}\frac{\gamma_i}{\alpha_i}b(s_i). \tag{9}$$

Since $\mathcal{L}_{s_i} = \mathcal{L}_1 \cup \mathcal{L}_2$ and $\mathcal{L}_{s_i'} = \mathcal{L}_1 \cup \mathcal{L}_3$ we have:

$$\begin{aligned}
&\phi(\mathbf{s})-\phi(\mathbf{s}')\\
&=\sum\nolimits_{k\in\mathcal{L}}\sum\nolimits_{q=1}^{n_k(\mathbf{s})}\frac{w_k(q)}{q}-\sum\nolimits_{k\in\mathcal{L}}\sum\nolimits_{q=1}^{n_k(\mathbf{s}')}\frac{w_k(q)}{q}+F\\
&=\sum\nolimits_{k\in\mathcal{L}_2}(\sum\nolimits_{q=1}^{n_k(\mathbf{s})}\frac{w_k(q)}{q}-\sum\nolimits_{q=1}^{n_k(\mathbf{s}')}\frac{w_k(q)}{q})\\
&+\sum\nolimits_{k\in\mathcal{L}_3}(\sum\nolimits_{q=1}^{n_k(\mathbf{s})}\frac{w_k(q)}{q}-\sum\nolimits_{q=1}^{n_k(\mathbf{s}')}\frac{w_k(q)}{q})+F\\
&=\sum\nolimits_{k\in\mathcal{L}_1\cup\mathcal{L}_2}\frac{w_k(n_k(\mathbf{s}))}{n_k(\mathbf{s})}-\sum\nolimits_{k\in\mathcal{L}_1\cup\mathcal{L}_3}\frac{w_k(n_k(\mathbf{s}'))}{n_k(\mathbf{s}')}+F\\
&=\sum\nolimits_{k\in\mathcal{L}_{s_i}}\frac{w_k(n_k(\mathbf{s}))}{n_k(\mathbf{s})}-\sum\nolimits_{k\in\mathcal{L}_{s_i'}}\frac{w_k(n_k(\mathbf{s}'))}{n_k(\mathbf{s}')}+F.
\end{aligned} \tag{10}$$

Finally, according to Eq. (10), the following equation holds.

$$P_i(\mathbf{s}') - P_i(\mathbf{s}) = \alpha_i(\phi(\mathbf{s}') - \phi(\mathbf{s})). \tag{11}$$

Hence, Theorem 2 is proved. □

According to Theorem 2, the multi-user route navigation game is a weighted potential game, where the change of the profit of each user can be mapped into the potential function. The decision update of a user in each decision slot can lead to the increase of its profit function, which further increases the potential function value. When the value of potential function reaches the maximum value, a Nash equilibrium is obtained.

4 ALGORITHM DESIGN

In this section, to find a Nash equilibrium, we introduce the following two algorithms: Algorithm 1 is the distributed game-theoretical route navigation algorithm for the mobile users; Algorithm 2 is the information update algorithm for the platform. The distributed route navigation algorithm runs on each user's smart phone and helps mobile users autonomously select a route from the recommended route set to maximize their own profits. When the algorithm terminates, a Nash equilibrium will be obtained. The information update algorithm on the platform is proposed to update information and determine the users to update their decisions in each decision slot. Specifically, we propose two user update algorithms, Single User Update algorithm (SUU) and Parallel User Update algorithm (PUU, i.e., Algorithm 3), to select the users to update the decision.

Algorithm 1 Distributed Game-Theoretical Route Navigation Algorithm for user $i \in \mathcal{U}$.

1: Input α_i, β_i, λ_i, the initial location and the destination.
2: Receive the recommended routes R_i.
3: Initialize $s_i(0)=r$ by randomly selecting a route $r \in R_i$.
4: Report $s_i(0)$ to the platform.
5: Receive n_k for each task k that is covered by $s_i(0)$.
6: Calculate the profit P_i.
7: Receive $d(r)$ and $b(r)$ for each route r in R_i.
8: **repeat** for each decision slot t
9: Obtain n_k for each task k that is covered by R_i.
10: Compute the best route set $\triangle_i(t)$.
11: **if** $\triangle_i(t) \neq \varnothing$ **then**
12: Send the request to contend the opportunity for updating decision.
13: **if** Win the opportunity **then**
14: Update the route selection decision $s_i(t)$ by selecting a route $r \in \triangle_i(t)$.
15: Report $s_i(t)$ to the platform.
16: **else**
17: Choose the original decision $s_i(t) = s_i(t-1)$.
18: **until** The termination message is received.

4.1 Distributed Route Navigation Algorithm

Theorem 2 guarantees that the multi-user route navigation game converges to a Nash equilibrium within a finite number of decision slots. The main idea of the distributed route navigation algorithm is to utilize the finite improvement property and select a set of mobile users to improve their profits by updating their route selection decisions in each decision slot.

In the initialization phase (lines 1-7) of Algorithm 1, the mobile user first inputs the weight parameters of preferences α_i, β_i, λ_i, as well as the initial location and the destination (line 1). Then, the recommended routes will be sent to the mobile application (line 2). We use $s_i(t)$ to represent the route decision of user i in decision slot t. The algorithm initializes the route selection decision by randomly selecting a route from the recommended route set and calculates the profit (lines 3-6). Finally, the mobile application receives the detour distance $d(r)$ and congestion level $b(r)$ of each route r in the recommended route set (line 7).

In the calculation phase (lines 8-18), each user obtains the information on the number of users performing each task covered by R_i (line 9). The algorithm then calculates the best route set $\triangle_i(t)$ (line 10). The best route set is defined as the set of route decisions that maximize the profit of the user and can improve the profit compared to the previous decision slot.

If the best route set $\triangle_i(t) \neq \varnothing$, which means that the user can improve the profit by altering the route selection decision, the user sends a request to the platform for updating the decision. If the user wins the opportunity to update the decision, it selects a route from the best route set to update the current decision. Otherwise, the user will maintain the decision consistent with the previous decision slot (lines 11-17). The calculation process repeats until the termination message is received from the platform (line 18). It is worth noting that when all the users receive the termination

Algorithm 2 Information Update Algorithm for the platform.

1: Send the recommended route set R_i to the user $i \in \mathcal{U}$.
2: Receive $s_i(0)$ from each user $i \in \mathcal{U}$.
3: Calculate n_k for each task $k \in \mathcal{L}$.
4: Send n_k, $d(r)$ and $b(r)$ to the corresponding user.
5: **repeat** for each decision slot t
6: Receive the request from the users and let $\mathcal{U}'$ denote the set of users that send the request.
7: **if** $\mathcal{U}' \neq \varnothing$ **then**
8: Select a set of users μ by SUU or PUU algorithm.
9: Inform the users in μ to update the decisions.
10: Receive $s_i(t)$ from user $i \in \mu$ and update n_k for each task $k \in \mathcal{L}$.
11: **until** No request is received from the user.
12: Send the termination message to all users.

message, the algorithm converges to a Nash equilibrium and the mobile users achieve mutually satisfactory decisions.

4.2 Information Update Algorithm

The information update algorithm updates the number of users that perform each task and interacts with the mobile users, such as exchanging the information and selecting a set of users to update the decision in each decision slot.

As shown in Algorithm 2, in the initialization phase (lines 1-4), the information update algorithm first sends the recommended routes to each user (line 1). After receiving the initial decisions from all the users (line 2), the algorithm calculates the number of users (n_k) that perform each task $k \in \mathcal{L}$ (line 3). Finally, the information that is required to calculate the profit is sent to the user (line 4).

Next, in each decision slot, the platform receives the requests from the users (line 6). Then, the platform utilizes Single User Update algorithm (SUU) or Parallel User Update algorithm (PUU) to determine the set of users to update their decisions and informs the selected users to update the decision (lines 8-9). When receiving the updated decision from the user, the algorithm updates the number of the participants of each task $k \in \mathcal{L}$ (line 10). When no request is received from the users in a decision slot, the algorithm sends the termination messages to all users and the information update algorithm terminates (lines 11-12).

Furthermore, we introduce the above two user update algorithms, SUU and PUU. SUU algorithm randomly selects only one user from the set of users that send the requests to the platform and allows the user to update the decision in each decision slot. To decrease the convergence time, we further propose PUU algorithm, which is inspired by the idea that some users whose selected routes cover no overlapping tasks could concurrently update their route selections in the same decision slot. This leads to a larger increase in the potential function value in each decision slot and reduce the convergence time.

The detailed description is as follows. As shown in Algorithm 3, the inputs are $\mathcal{U}'$, $\boldsymbol{\tau}$ and $\mathbf{B}$. Specifically, $\mathcal{U}'$ is the set of users sending the requests to update the decisions, $\boldsymbol{\tau} = \{\tau_i, \forall i \in \mathcal{U}'\}$ and $\mathbf{B} = \{B_i, \forall i \in \mathcal{U}'\}$. Let s_i denote the original strategy of user i and s_i' denote a new strategy which maximizes the profit in the best route set. We use B_i to denote the set of tasks jointly covered by s_i and s_i', and $\tau_i = (P_i(s_i', s_{-i}) - P_i(s_i, s_{-i}))/\alpha_i$. For each user $i \in \mathcal{U}'$, it sends B_i and τ_i to the platform. PUU algorithm first calculates δ_i for each user $i \in \mathcal{U}'$ (line 2) and sorts the users in $\mathcal{U}'$ in a non-ascending order of δ_i (line 3). Then, the algorithm greedily chooses the set of users μ, which maximizes the sum of τ_i and satisfies the constraint that the covered task set B_i of $i \in \mu$ does not intersect with each other (lines 4-5). After inspecting all the users in $\mathcal{U}'$, the algorithm returns the user set μ and informs the users in μ to update decisions concurrently.

Algorithm 3 Parallel User Update Algorithm.

Input: $\mathcal{U}'$, $\boldsymbol{\tau}$, $\mathbf{B}$.
1: Initialize $\mu = \varnothing$, $\sigma = \varnothing$.
2: Calculate $\delta_i = \frac{\tau_i}{|B_i|}, \forall i \in \mathcal{U}'$.
3: Sort $i \in \mathcal{U}'$ in a non-ascending order of δ_i.
4: **for all** $i \in \mathcal{U}'$ **do**
5: **if** $\sigma \cap B_i = \varnothing$ **then** $\mu \leftarrow \mu \cup i$, $\sigma \leftarrow \sigma \cup B_i$.
return μ.

We then analyze the performance of PUU algorithm compared to the optimal solution, which maximizes the value of $\sum_{i \in \mu} \tau_i$. Let $\hat{\mu}$ denote the set of selected users to update the route decisions of the optimal solution. Let $\tau = \sum_{i \in \mu} \tau_i$ and $\hat{\tau} = \sum_{i \in \hat{\mu}} \tau_i$. The following theorem holds.

Theorem 3. *The performance of the PUU algorithm and the optimal solution maximizing the value of $\sum_{i \in \mu} \tau_i$ satisfy the following equation:*

$$\tau/\hat{\tau} \geq |B_{i'}|/(|\hat{\mu}| \cdot B_{max}), \tag{12}$$

where $i' = \arg\max_{i \in \mu} \delta_i$, $B_{max} = \max_{i \in \hat{\mu}} |B_i|$, and $|\hat{\mu}|$ represents the number of selected users needed to update the decisions in the optimal solution.

Proof. The PUU algorithm first selects the user that has the maximum value of δ_i in $\mathcal{U}'$, $\delta_{i'} \geq \tau_i/|B_i| \ \forall i \in \hat{\mu}$. Hence, the following equation holds.

$$|\hat{\mu}|\tau_{i'}/|B_{i'}| \geq \sum\nolimits_{i \in \hat{\mu}} \tau_i/|B_i|. \tag{13}$$

Since $B_{max} = \max_{i \in \hat{\mu}} |B_i|$, we get Eq. (14).

$$\sum\nolimits_{i \in \hat{\mu}} \tau_i/|B_i| \geq \hat{\tau}/B_{max}. \tag{14}$$

There exists $\tau \geq \tau_{i'}$. According to Eq. (13) and Eq. (14), we get the following equation by rearranging the items.

$$\tau/\hat{\tau} \geq |B_{i'}|/(|\hat{\mu}| \cdot B_{max}). \tag{15}$$

Hence, Theorem 3 is proved. □

4.3 Convergence Analysis

According to Theorem 2, the proposed distributed route navigation algorithm will converge to a Nash equilibrium within a finite number of update iterations. We then analyze the upper bound of the iteration number for convergence. Let $\mathcal{S}$ denote the strategy space of all the users. For $\forall k \in \mathcal{L}, \forall \mathbf{s} \in \mathcal{S}, 1 \leq q \leq n_k(\mathbf{s})$, let $g_{min} = \min\{w_k(q)/q\}$, $g_{max} = \max\{w_k(q)/q\}$. There exists $0 \leq d(s_i) \leq d_{max}, \forall i \in \mathcal{U}$ and $0 \leq b(s_i) \leq b_{max}, \forall i \in \mathcal{U}$. We denote the minimum change value of the users' profit when the

user updates the decision as ΔP_{min}. For the number of decision slots for convergence, the following theorem holds.

Theorem 4. *The number of decision slots C for convergence of the distributed route navigation algorithm satisfies the following equation.*

$$C < \frac{e_{max}}{\Delta P_{min}} |\mathcal{U}|(|\mathcal{L}|(g_{max}-g_{min}) + \frac{e_{max}}{e_{min}} d_{max} + \frac{e_{max}}{e_{min}} b_{max}). \quad (16)$$

Proof. During a decision slot, consider the worst case where there is only a user $i \in \mathcal{U}$ who alters the current strategy s_i to s_i', which leads to an increase in its profit function, i.e., $P(s_i, s_{-i}) < P(s_i', s_{-i})$. According to Eq. (8), we have:

$$\phi(\mathbf{s}) > |\mathcal{L}||\mathcal{U}|g_{min} - |\mathcal{U}|e_{max}d_{max}/e_{min} - |\mathcal{U}|e_{max}b_{max}/e_{min}. \quad (17)$$

$$\phi(\mathbf{s}) < |\mathcal{L}||\mathcal{U}|g_{max}. \quad (18)$$

Furthermore, we have:

$$\phi(\mathbf{s}') - \phi(\mathbf{s}) < |\mathcal{U}|(|\mathcal{L}|(g_{max}-g_{min}) + \frac{e_{max}}{e_{min}} d_{max} + \frac{e_{max}}{e_{min}} b_{max}). \quad (19)$$

According to Eq. (11), the change of the value of the potential function is equivalent to the change of a user's profit divided by α_i. Therefore, for the number of decision slots C for convergence, we have the following results:

$$C < \frac{e_{max}}{\Delta P_{min}} |\mathcal{U}|(|\mathcal{L}|(g_{max}-g_{min}) + \frac{e_{max}}{e_{min}} d_{max} + \frac{e_{max}}{e_{min}} b_{max}). \quad (20)$$

Hence, Theorem 4 is proved. □

From Theorem 4, to accelerate the convergence time, we can select a set of users with no overlapping tasks to concurrently update their route decisions in each decision slot as shown in Algorithm 3. Since each selected user can increase the profit by updating the decision in parallel and leads to an increase in the value of the potential function, the potential function reaches the maximum value quickly. Hence, the convergence time is decreased.

4.4 Theoretical Analysis

We then analyze the performance of the proposed distributed route navigation algorithm by analyzing Price of Anarchy (PoA) [11]. PoA is a metric measured by the ratio of the total profit of all users in the worst case of Nash equilibrium to the maximum total profit of the optimal strategy, as defined in Eq. (21). By analyzing PoA, we quantify the efficiency of the worst-case Nash equilibrium over the centralized optimal solution in terms of the total profit. Let S' be the set of strategy profiles that can achieve Nash equilibrium and $\mathbf{s}^*$ denote the centralized optimal strategy.

$$PoA = \min_{\mathbf{s} \in S'} \sum_{i \in \mathcal{U}} P_i(\mathbf{s}) / \sum_{i \in \mathcal{U}} P_i(\mathbf{s}^*). \quad (21)$$

We consider a special case of the multi-user route navigation game with the following restrictions. First, each route only covers one task and a task may be covered by several routes. Second, the recommended route set R_i for each user $i \in \mathcal{U}$ contains two parts: r_i' and R, where the task on route r_i' is only covered by r_i' and R is a set of routes that covers the same set of tasks for all users. The common task set covered by R is denoted as $\mathcal{L}'$. In other words, the task on r_i' is only covered by user i and the tasks in $\mathcal{L}'$ can be covered by all users. Third, the reward of a task $k \in \mathcal{L}'$ is defined as $w_k = a + \ln x$, $a > 0$ and we do not make any restrictions for the reward of other task $k \notin \mathcal{L}'$. Let k_i denote the task covered by

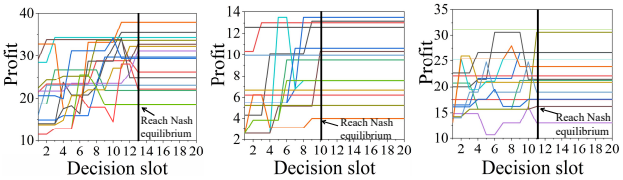


Figure 3: User profit vs. decision slot.

the route decision s_i for user i. Since a route only covers one task, the profit for each user i is calculated as $P_i(\mathbf{s}) = w_{k_i}(n_{k_i}(\mathbf{s}))/n_{k_i}(\mathbf{s})$. Specifically, the profit that user i achieves by selecting route r_i' is denoted as P_i.

Theorem 5. *For the multi-user route navigation game, the PoA metric of the overall profits satisfies that*

$$\frac{\sum_{i \in \mathcal{U}} \max\{P_i, P_i^{min}\}}{\sum_{i \in \mathcal{U}} \max\{P_i, P_i^{max}\}} \leq PoA \leq 1, \quad (22)$$

where $P_i^{min} = \frac{a + \ln p}{p}$, $p = \frac{|\mathcal{U}| + |\mathcal{L}'| - 1}{|\mathcal{L}'|}$, $P_i^{max} = a$.

Proof. There exists $n_{k_i}(\mathbf{s}) = 1 + \sum_{j \in \mathcal{U} \setminus \{i\}} I\{k_j = k_i\}$. $I\{E\}$ is an indicator function, where $I\{E\} = 1$ if the event E is true and $I\{E\} = 0$ otherwise. In the formulated special case, since a route only covers one task, the profit of a route decreases with the growth of the number of users performing the task on that route. Since no user can increase the profit by changing the decision unilaterally in Nash equilibrium, considering a strategy profile $\mathbf{s}$ that achieves a Nash equilibrium and user i selects a route in R, the following equation holds:

$$\sum_{j \in \mathcal{U} \setminus \{i\}} I\{k_j = k_i\} \leq \sum_{j \in \mathcal{U} \setminus \{i\}} I\{k_j = k\} \ \forall k \in \mathcal{L}'. \quad (23)$$

Furthermore, we have:

$$|\mathcal{L}'|(\sum_{j \in \mathcal{U} \setminus \{i\}} I\{k_j = k_i\}) \leq \sum_{k \in \mathcal{L}'} \sum_{j \in \mathcal{U} \setminus \{i\}} I\{k_j = k\}. \quad (24)$$

According to Eq. (24), we substitute and rearrange the corresponding terms. The following equation holds:

$$\sum_{j \in \mathcal{U} \setminus \{i\}} I\{k_j = k_i\} \leq (|\mathcal{U}| - 1)/|\mathcal{L}'|. \quad (25)$$

Based on Eq. (25), there exists $n_{k_i}(\mathbf{s}) \leq (|\mathcal{U}| + |\mathcal{L}'| - 1)/|\mathcal{L}'|$. Furthermore, the following equation holds.

$$P_i(\mathbf{s}) \geq \frac{a + \ln((|\mathcal{U}| + |\mathcal{L}'| - 1)/|\mathcal{L}'|)}{(|\mathcal{U}| + |\mathcal{L}'| - 1)/|\mathcal{L}'|)}. \quad (26)$$

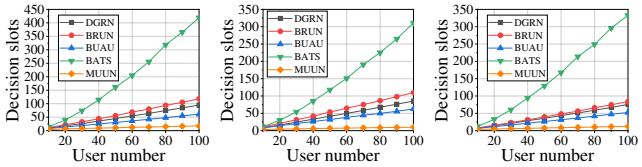
We use p to denote $\frac{|\mathcal{U}| + |\mathcal{L}'| - 1}{|\mathcal{L}'|}$ and P_i^{min} to denote $\frac{a + \ln p}{p}$. As mentioned above, if user i selects r_i', the profit is P_i. Hence, $P_i(\mathbf{s}) \geq \max\{P_i, P_i^{min}\}$.

On the other hand, due to the fact that $n_{k_i}(\mathbf{s}) \geq 1$, there exists $P_i(\mathbf{s}) \leq a$. Let $P_i^{max} = a$. Furthermore, we can conclude that $P_i(\mathbf{s}^*) \leq \max\{P_i, P_i^{max}\}$.

In conclusion, according to the above description, the following equation holds:

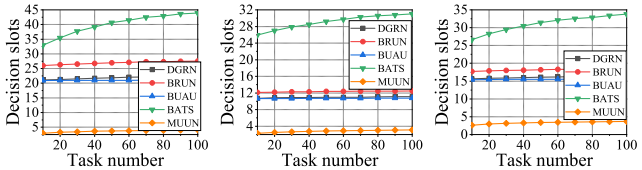
$$\frac{\sum_{i \in \mathcal{U}} \max\{P_i, P_i^{min}\}}{\sum_{i \in \mathcal{U}} \max\{P_i, P_i^{max}\}} \leq PoA \leq 1. \quad (27)$$

□



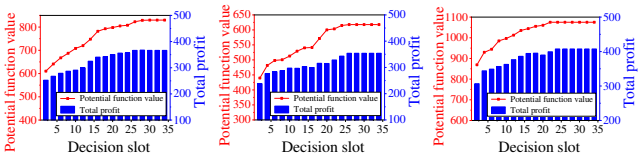
(a) Shanghai (b) Roma (c) Epfl

Figure 4: Decision slot vs. user number.



(a) Shanghai (b) Roma (c) Epfl

Figure 5: Decision slot vs. task number.



(a) Shanghai (b) Roma (c) Epfl

Figure 6: Potential function and total profit vs. decision slot.

Table 2: Simulation Parameters

Parameters	Value
Route number recommended to a user	1~5
Original reward of a task a_k	10~20
Parameter measuring reward increment μ_k	0~1
User weight parameters α_i, β_i and γ_i	0.1~0.9
System weight parameters φ, θ	0.1~0.8
Number of repeated simulations	500

Table 3: The selected user number vs. overlap ratio.

Total task #	50	60	70	80	90
Overlap ratio	0.526	0.531	0.532	0.536	0.538
Selected user #	2.013	1.978	1.842	1.751	1.701

5 PERFORMANCE EVALUATION

5.1 Trace-Based Sets

We use three widely-used real-world data sets to evaluate the proposed algorithm. *Shanghai* [32] contains the GPS trace data of taxis collected from August 2006 to October 2006 in Shanghai, China. We select 200 traces and the traces are collected from one day. *Roma* [1] contains GPS information about 320 taxis collected over 30 days in Rome, Italy. We select 150 traces in the center of the city. *Epfl* [21] is a trace set of mobility data about 500 taxi cabs collected over 30 days in the San Francisco Bay Area, USA. We select 200 traces and each of which was collected from the same period of one day.

We extract the origin and the destination from the traces and utilize Google Maps API to generate the recommended route set for each origin-destination pair. The tasks are randomly generated with the reward and each recommended route may cover some tasks. The detour distance for each route is calculated as the extra distance compared with the shortest route and the congestion level is calculated by the velocity of the vehicles on the route. The simulation parameters are shown in Table 2.

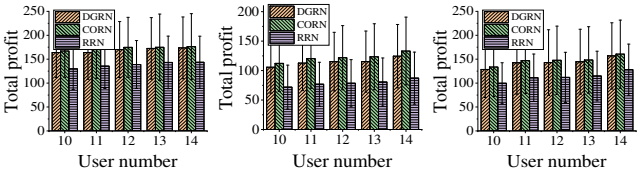
5.2 Comparison Algorithms

We use the following algorithms in the simulations. (1) Distributed Game-theoretical Route Navigation(DGRN): The proposed algorithm utilizes SUU algorithm to randomly select a user from the users that send the update requests and allows the user to select the best route to maximize its profit. (2) Multi-User Update Navigation (MUUN): The proposed algorithm utilizes PUU algorithm to select a set of users from the users that send the update requests and allows the selected users to parallel update the route decisions by selecting the best route to maximize their profits. (3) Better Response Update Navigation (BRUN): BRUN randomly selects a user from the users that send the update requests and allows the user to randomly select a route that is better than the current route in each decision slot. (4) Best Update of All Users (BUAU): BUAU inspects all users and selects the user that maximizes the value of the potential function to update the decision in each decision slot. (5) Bayesian Asynchronous Task Selection (BATS): We modify the task selection approach in the existing research [5] to apply to our scenario. (6) Centralized Optimal Route Navigation (CORN): The centralized optimal approach to maximize the total profit of all users. (7) Random Route Navigation (RRN): Each mobile user randomly selects a route from the recommended route set.

5.3 Numerical Results

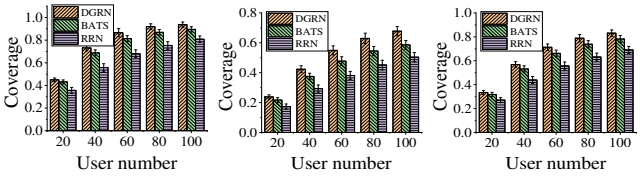
5.3.1 Convergence for Nash equilibrium. We first verify the convergence for the proposed distributed algorithm in Fig. 3. Specifically we randomly select 15 users in each data set and observe the dynamics of the profits in 20 decision slots. It is obvious that the profit of the user changes with the decision updates in the beginning and can converge to a stable point (i.e., Nash equilibrium of the multi-user game). It is worth noting that some users' profits may decrease as a result of other users' decision updates. Since the reward of a task is equally shared by the participants, if a user selects a route, the profits of all users performing the task on that route decrease.

In Fig. 4, we investigate the number of decision slots for convergence with the change of the number of users. The simulation results show that the number of decision slots ranks as follows: MUUN<BUAU<DGRN<BRUN<BATS. The reason is that MUUN selects multiple users and the selected users update their decisions in parallel, while BUAU selects only one user who maximizes the potential function in each decision slot. Hence, MUUN reaches the maximum value as quickly as possible. DGRN and BRUN randomly select a user to update the decision with the best and better response update manner respectively. Hence, DGRN converges to the equilibrium a little faster than BRUN. For BATS, the user updates the decision in sequence to maximize the profit in each decision slot. In some decision slots, some users cannot increase the profits but



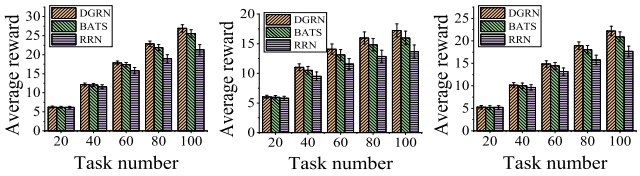
(a) Shanghai (b) Roma (c) Epfl

Figure 7: Total profit vs. user number.



(a) Shanghai (b) Roma (c) Epfl

Figure 8: Coverage vs. user number.



(a) Shanghai (b) Roma (c) Epfl

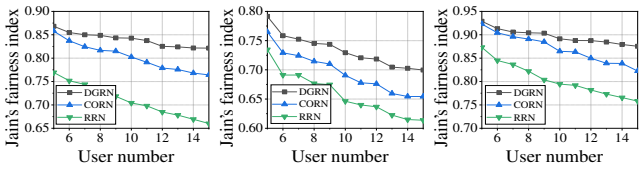
Figure 9: Average reward vs. task number.

still update the decisions, which increases the number of decision slots for convergence.

In Fig. 5, we investigate the number of decision slots for convergence with the change of the number of tasks. The simulation results show that the number of decision slots ranks as follows: MUUN<BUAU<DGRN<BRUN<BATS. The reason for this is the same as the description in Fig. 4. With the growth of the number of tasks, the number of decision slots slightly increases. This is because when the number of tasks increases, the users are more likely to cover the same tasks with others. Hence, the route decisions of users are more likely to be coupled with each other and it will take more decision slots to reach a Nash equilibrium state.

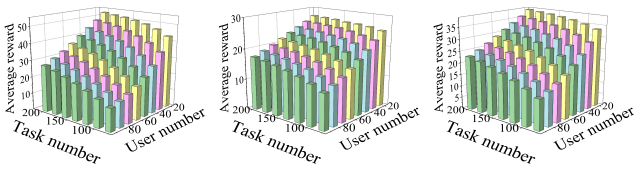
In Fig. 6, we observe the dynamics of the value of the potential function and the total profit of all users with the change of the decision slot. We can find that the value of the potential function increases in the beginning and finally converges to a stable state (i.e., Nash equilibrium), which matches the theoretical analysis. The total profit increases with the growth of decision slots on the whole with some fluctuations, because each user maximizes its own profit in the multi-user game rather than the total profit and a user's decision update may decrease other users' profits. Hence, the total profit may decrease sometimes.

In Table. 3, we observe the selected number of users to update the decision with the change of the overlap ratio in MUUN. The overlap ratio is defined as the ratio between the number of tasks that have more than one participant and the total number of tasks. Specifically, we change the overlap ratio by varying the total number of tasks from 50 to 90, and observe the average number of selected user



(a) Shanghai (b) Roma (c) Epfl

Figure 10: Jain's fairness index vs. user number.



(a) Shanghai (b) Roma (c) Epfl

Figure 11: Average reward vs. task number and user number.

Table 4: Comparison between DGRN and CORN.

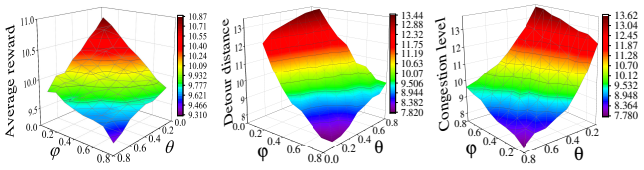
User #	DGRN	CORN	Ratio	Bound
9	65	65	1	0.717
10	78	81	0.963	0.688
11	87	89	0.977	0.753
12	94	97	0.969	0.809
13	109	110	0.990	0.785
14	115	116	0.991	0.876

of all decision slots. We conduct the simulation on *Shanghai* data set and repeat the simulation 500 times. With the growth of the number of tasks, more routes intersect with other routes on some task locations. Since MUUN selects users whose selected route does not intersect with others' on the task locations, the average selected number of users decreases with the increase of the overlap ratio.

5.3.2 Profit, coverage, and reward. As shown in Fig. 7, we investigate the trend of total profit with the growth of the number of users and we repeat the simulations 500 times. The total profit is the sum of the profit function values of all the users. The total profit ranks as follows: RRN<DGRN<CORN. Since DGRN does not maximize the total profit but maximize each user's profit in an equilibrium state, the total profit of DGRN is a little less than that of CORN.

As shown in Fig. 8, we investigate the coverage with the growth of the number of users. The simulation is repeated 500 times. The coverage is calculated as the ratio between the number of covered tasks and the total number of tasks. The coverage ranks as follows: RRN<BATS<DGRN, as DGRN can adjust the settings to increase the coverage of tasks.

As shown in Fig. 9, we investigate the trend of average reward with the growth of the number of tasks. The average reward is defined as the total reward of all the users divided by the number of users. We repeat the simulations 500 times. The simulation results rank as follows: RRN<BATS<DGRN. It is easy to find out that the average reward increases with the growth of the number of tasks, as



(a) Average reward (b) Detour distance (c) Congestion level

Figure 12: The influence of system parameters.

(a) Shanghai (b) Roma (c) Epfl

Figure 13: The presentation on real data sets.

Table 5: The influence of the user parameters.

α_i	reward	β_i	detour	γ_i	congestion
0.1	7.74	0.1	12.24	0.1	12.03
0.2	7.85	0.2	10.97	0.2	10.48
0.3	7.94	0.3	9.88	0.3	9.52
0.4	7.96	0.4	9.38	0.4	8.75
0.5	7.98	0.5	8.84	0.5	8.48
0.6	8.08	0.6	8.38	0.6	8.20
0.7	8.10	0.7	8.07	0.7	8.05
0.8	8.16	0.8	7.99	0.8	7.97

a user may perform more tasks when the number of tasks increases. The error bar guarantees that the simulation results are accurate.

Fig. 10 shows the dynamics of Jain's fairness index with the growth of the number of users. We repeat the simulations 500 times. Jain's fairness index [14] is used to measure the fairness of the user's profit, which is defined as $\frac{(\sum_{i\in\mathcal{U}} P_i(\mathbf{s}))^2}{|\mathcal{U}|\sum_{i\in\mathcal{U}} P_i(\mathbf{s})^2}$. It is worth noting that the fairness depends on how evenly distributed the profit of each user is. The simulation results show that the proposed DGRN achieves the highest Jain's fairness index among CORN and RRN, as DGRN can reach a Nash equilibrium of multi-user game.

Fig. 11 shows the average reward of the proposed algorithm with the change of the number of tasks and number of users. We repeat the simulation 500 times. From Fig. 11, we find that the average reward increases with the growth of the number of tasks and decreases with the growth of the number of users. The reason is that the user may perform more tasks when the number of tasks increases and the reward of a task may be shared with more participants when the number of users increases. The simulation results match the theoretical analysis.

In Table. 4, we further study the gap between DGRN and CORN. With the change of the number of users, the ratio between the total profit of DGRN and CORN is always larger than the lower bound of PoA, which matches the theoretical analysis.

5.3.3 The influence of algorithm parameters and the real-world example. In Fig. 12, we evaluate the influence of the system parameters φ and θ on the *Shanghai* data set. We repeat the simulations 500 times. It is interesting to find out that the average reward increases with the decrease of φ and θ, because the decrease of φ and θ suggests that the platform emphasizes the importance of letting users receive more rewards. Correspondingly, the detour distance decreases with the growth of φ and the congestion level decreases with the growth of θ respectively.

In Table. 5, we randomly select a user from the user set and vary the parameters α_i, β_i and γ_i from 0.1 to 0.8 respectively. We repeat the simulation for 500 times. When changing α_i, we observe the value of the reward obtained by user i. It is easy to find out that the reward increases with the growth of α_i, because α_i is the weight parameter concerning how much the user emphasizes getting the task reward. When α_i increases, user i prefers to select the route with a high reward to increase the profit. Similarly, the detour distance and the congestion level decrease with the growth of β_i and γ_i respectively. Hence, the user can adjust the values of α_i, β_i and γ_i to achieve the different individual preferences.

Finally, to better demonstrate the schemes, we introduce three examples based on three data sets by utilizing Google Maps. As shown in Fig. 13 (a), the sensing tasks are distributed in the city. We consider two users and utilize Google Maps API to generate the recommended routes between the initiation and destination. The platform recommends the user 2 or 3 routes. The user will choose one route (marked with a green color) and complete the tasks on that route. Since the situation of Fig. 13 (b) and (c) is similar to that in Fig. 13 (a), we do not give the additional description.

6 CONCLUSION

In this paper, motivated by the widely-used map navigation systems, we propose to utilize the route navigation system to perform distributed vehicular crowdsensing task allocation. We first prove that the centralized optimization problem is NP-hard and formulate the problem as a multi-user potential game. Then, we propose a distributed route navigation algorithm. Users can modify the parameters of the profit function to satisfy their individual preferences, and the platform can also do the same to achieve different task allocation purposes. Furthermore, we analyze the performance of the algorithm theoretically. Finally, we conduct extensive simulations based on three real-world datasets. The simulation results show that the proposed approach achieves a Nash equilibrium while achieving a total user profit close to that of the optimal solution.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundations of China under Grant No. 61772230, No. 61972450, No. 62072209, and No. 61806121, Natural Science Foundations of Jilin Province No. 20190201022JC, Innovation Capacity Building Project of Jilin Province Development and Reform Commission No. 2020C017-2, Changchun Science and Technology Development Project under Grant No.18DY005, Jilin Province Young Talents Lifting Project No. 3D4196993421, China Postdoctoral Science Foundation No. 2021T140261, and in part by NSF grants CNS 1824440, CNS 1828363, CNS 1757533, CNS 1629746, CNS 1651947, and CNS 1564128.

REFERENCES

[1] Lorenzo Bracciale, Marco Bonola, Pierpaolo Loreti, Giuseppe Bianchi, Raul Amici, and Antonello Rabuffi. 2014. CRAWDAD dataset roma/taxi (v. 2014-07-17). Downloaded from https://crawdad.org/roma/taxi/20140717.

[2] Z. Cai, Z. Duan, and W. Li. 2020. Exploiting Multi-Dimensional Task Diversity in Distributed Auctions for Mobile Crowdsensing. *IEEE Transactions on Mobile Computing* (2020), 1–1.

[3] B. Cao, S. Xia, J. Han, and Y. Li. 2020. A Distributed Game Methodology for Crowdsensing in Uncertain Wireless Scenario. *IEEE Transactions on Mobile Computing* 19, 1 (2020), 15–28.

[4] Mihaela Cardei, My T. Thai, Yingshu Li, and Weili Wu. [n.d.]. Energy-efficient target coverage in wireless sensor networks. In *IEEE INFOCOM 2005.*

[5] M. H. Cheung, F. Hou, J. Huang, and R. Southwell. 2020. Distributed Time-Sensitive Task Selection in Mobile Crowdsensing. *IEEE Transactions on Mobile Computing* (2020), 1–1.

[6] Man Hon Cheung, Richard Southwell, Fen Hou, and Jianwei Huang. [n.d.]. Distributed Time-Sensitive Task Selection in Mobile Crowdsensing. In *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2015.* ACM, 157–166.

[7] C. Dai, X. Wang, K. Liu, D. Qi, W. Lin, and P. Zhou. 2020. Stable Task Assignment for Mobile Crowdsensing with Budget Constraint. *IEEE Transactions on Mobile Computing* (2020), 1–1.

[8] F. Fabiani and S. Grammatico. 2020. Multi-Vehicle Automated Driving as a Generalized Mixed-Integer Potential Game. *IEEE Transactions on Intelligent Transportation Systems* 21, 3 (2020), 1064–1073.

[9] G. Fan, H. Jin, Q. Liu, W. Qin, X. Gan, H. Long, L. Fu, and X. Wang. 2021. Joint Scheduling and Incentive Mechanism for Spatio-Temporal Vehicular Crowd Sensing. *IEEE Transactions on Mobile Computing* 20, 4 (2021), 1449–1464. https://doi.org/10.1109/TMC.2019.2960328

[10] Bin Guo, Yan Liu, Wenle Wu, Zhiwen Yu, and Qi Han. 2016. ActiveCrowd: A Framework for Optimized Multi-Task Allocation in Mobile Crowdsensing Systems. *IEEE Transactions on Human-Machine Systems* (08 2016).

[11] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, and Y. Yang. 2020. A Game-Theoretical Approach for User Allocation in Edge Computing Environment. *IEEE Transactions on Parallel and Distributed Systems* 31, 3 (2020), 515–529.

[12] S. He, D. Shin, J. Zhang, and J. Chen. 2014. Toward optimal allocation of location dependent tasks in crowdsensing. In *IEEE INFOCOM 2014.* 745–753.

[13] Zicong Hong, Wuhui Chen, Huawei Huang, Song Guo, and Zibin Zheng. 2019. Multi-hop Cooperative Computation Offloading for Industrial IoT-Edge-Cloud Computing Environments. *IEEE Transactions on Parallel and Distributed Systems* 99 (2019), 1–1.

[14] Raj Jain, Dah Ming Chiu, and Hawe WR. 1984. A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems. *ACM Transactions on Computer Systems* (1984).

[15] Haiming Jin, Hongpeng Guo, Lu Su, Klara Nahrstedt, and Xinbing Wang. 2019. Dynamic Task Pricing in Multi-Requester Mobile Crowd Sensing with Markov Correlated Equilibrium. In *IEEE INFOCOM 2019.* 1063–1071.

[16] H. Li and L. Zhijian. 2010. The study and implementation of mobile GPS navigation system based on Google Maps. In *2010 International Conference on Computer and Information Application.* 87–90.

[17] C. H. Liu, Z. Dai, H. Yang, and J. Tang. 2020. Multi-Task-Oriented Vehicular Crowdsensing: A Deep Learning Approach. In *IEEE INFOCOM 2020.* 1123–1132.

[18] Tong Liu, Yanmin Zhu, and Liqun Huang. 2019. TGBA: A two-phase group buying based auction mechanism for recruiting workers in mobile crowd sensing. *Computer networks* 149, FEB.11 (2019), 56–75.

[19] W. Liu, Y. Yang, E. Wang, and J. Wu. 2020. Dynamic User Recruitment with Truthful Pricing for Mobile CrowdSensing. In *IEEE INFOCOM 2020* (27-30).

[20] J. Ni, K. Zhang, Q. Xia, X. Lin, and X. S. Shen. 2020. Enabling Strong Privacy Preservation and Accurate Task Allocation for Mobile Crowdsensing. *IEEE Transactions on Mobile Computing* 19, 6 (2020), 1317–1331.

[21] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. 2009. CRAWDAD dataset epfl/mobility (v. 2009-02-24). Downloaded from https://crawdad.org/epfl/mobility/20090224.

[22] Alessandro Raschellà, Faycal Bouhafs, Michael Mackay, Qi Shi, and María Canales. [n.d.]. AP selection algorithm based on a potential game for large IEEE 802.11 WLANs. In *IEEE/IFIP NOMS 2018.*

[23] E. Wang, Y. Yang, J. Wu, W. Liu, and X. Wang. 2018. An Efficient Prediction-Based User Recruitment for Mobile Crowdsensing. *IEEE Transactions on Mobile Computing* 17, 1 (2018), 16–28.

[24] H. Wang, D. Zhao, H. Ma, and L. Ding. 2020. MB-GVNS: Memetic Based Bidirectional General Variable Neighborhood Search for Time-Sensitive Task Allocation in Mobile Crowd Sensing. *IEEE Transactions on Vehicular Technology* 69, 2 (2020), 2219–2229.

[25] J. Wang, F. Wang, Y. Wang, L. Wang, Z. Qiu, D. Zhang, B. Guo, and Q. Lv. 2020. HyTasker: Hybrid Task Allocation in Mobile Crowd Sensing. *IEEE Transactions on Mobile Computing* 19, 3 (2020), 598–611.

[26] Jiangtao Wang, Leye Wang, Yasha Wang, Daqing Zhang, and Linghe Kong. 2018. Task Allocation in Mobile Crowd Sensing: State-of-the-Art and Future Opportunities. *IEEE Internet of Things Journal.* 5, 5 (2018), 3747–3757.

[27] L. Wang, Z. Yu, D. Zhang, B. Guo, and C. H. Liu. 2019. Heterogeneous Multi-Task Assignment in Mobile Crowdsensing Using Spatiotemporal Correlation. *IEEE Transactions on Mobile Computing* 18, 1 (2019), 84–97.

[28] Mingjun Xiao, Jie Wu, Liusheng Huang, Ruhong Cheng, and Yunsheng Wang. 2017. Online Task Assignment for Crowdsensing in Predictable Mobile Social Networks. *IEEE Transactions on Mobile Computing* (2017), 1–1.

[29] Chen Xu, Jiao Lei, Wenzhong Li, and Xiaoming Fu. 2016. Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing. *IEEE/ACM Transactions on Networking* 24, 5 (2016), 2795–2808.

[30] Y. Yang, W. Liu, E. Wang, and J. Wu. 2019. A Prediction-Based User Selection Framework for Heterogeneous Mobile CrowdSensing. *IEEE Transactions on Mobile Computing* 18, 11 (2019), 2460–2473.

[31] B. Zhao, S. Tang, X. Liu, X. Zhang, and W. Chen. 2020. iTAM: Bilateral Privacy-Preserving Task Assignment for Mobile Crowdsensing. *IEEE Transactions on Mobile Computing* (2020), 1–1.

[32] H Zhu, Y Zhu, M Li, and L. M Ni. 2009. HERO: Online real-time vehicle tracking in Shanghai. *IEEE Transactions on Parallel and Distributed Systems* 20, 5 (2009), 740–752.