Incentive Facilitation for Peer Data Exchange in Crowdsensing

Xiang Yan, Fan Ye, Yuanyuan Yang, *Fellow, IEEE*, Dongge Wang, and Xiaotie Deng, *Fellow, IEEE*

Abstract—With mobile devices extensively used in daily life, there are ample opportunities to exchange sensing data through them, even without centralized management. In this paper, we design a peer based data exchanging model, where relay nodes move to certain locations to connect data providers and consumers to facilitate data delivery. Consumers are willing to pay for the data and these rewards are given to both relays and data providers. We first prove the NP-hardness of the problem on how to assign relay nodes to proper locations, and present a centralized optimal method with an approximation ratio. Then we define an autonomous compensation game for relays to make their individual decisions without any central authority. The sufficient and necessary condition for the existence of Nash equilibrium is derived, and an efficient reinforcement learning solver is designed to find the exact forms of equilibria. We analyze and compare this distributed game to the centralized social optimal solution, showing that the game incurs small bounded social costs, and is efficient under various network sizes, number of providers, number of consumers and device mobility.

Index Terms—Peer Data Exchange, Autonomous Compensation Game, Nash Equilibrium, Reinforcement Learning, Crowdsensing.

1 Introduction

The penetration of mobile devices with various sensors has made peer data exchange feasible and valuable in many daily life scenarios. In many places (e.g., roads, parks, airports) there are high densities of mobile devices like smartphones carried by users. Each device can collect sensing data of certain types around its location, and such data may carry important information needed by other users. For instance, a passenger on a bus passing by an accident scene can take a photo, which is important for drivers in nearby blocks so they can know what is causing the jam and how to change the route. Before taking her baby out for a walk, a mother wants to know the air quality distribution around a neighborhood, and pedestrians in the neighborhood could provide such kind of data. In these scenarios, data exchange among peer users provide valuable information, and users are willing to pay a reward to obtain desired data.

In such peer data exchange, the one that possessing certain data is called a *provider* and the one needing data from others is called a *consumer*. Usually, the devices have limited radio transmission range, and the density of mobile devices may not always be high enough to ensure direct connectivity among all consumers and providers. To facilitate data exchange, some *relay nodes*, motivated by the economic incentive, may move to certain *relay locations* to connect consumers and providers, and forward data possibly over multiple hops.

In this paper, we study the following problems: what is the optimal strategy to decide which relay node should go to which location? If consumers requesting the same data can pool their rewards, what is the optimal payment allocation algorithm among relays and providers? The strategy and algorithm must be efficient to incur small overheads in computation and node movements, and effective to incentivize relays and providers for peer data exchange.

There are several challenges to these questions. First, given multiple relay nodes and relay locations, there is a combinatorial thus exponential space of who goes where, leading to an NP-hard problem. We have to design an efficient algorithm. Secondly, the reward pool must be allocated and paid to relays and providers in a way to compensate them fairly based on their contributions. Third, although a centralized algorithm that computes and dictates which relay goes where may achieve global optimal efficiency, in reality no central authority exists and each relay may make its own decision. The payment must be allocated effectively among peers.

To address these challenges, we design an approximation algorithm to decide the best locations for relay nodes with polynomial complexity. Then we define concrete fairness goals and establish a payment mechanism to achieve these goals in centralized setting. To overcome practical obstacles in the centralized allocation, we devise an autonomous compensation game where individual relays make their own decisions without any central authority. We analyze the strategies that can achieve different types of Nash equilibrium, and extra cost when providers and consumers may move before relay nodes reach their respective locations. We summarize our contributions as follows:

- We formulate the problem of finding relay locations for relay nodes as an optimization problem in graph theory. We show its NP-hardness and propose a centralized approximation algorithm that assigns relay nodes to proper relay locations, and decides their payments based on the assignment.
- We design an autonomous compensation game for relay nodes to make decisions of where to go indi-

- *X. Yan is with Shanghai Jiao Tong University, China (email:xyansjtu@163.com).*
- *F. Ye and Y. Yang are with Stony Brook University, USA.*
- *D. Wang and X. Deng are with Peking University, China.*

vidually. We derive the condition for the existence of pure Nash equilibrium for the game, and design an efficient reinforcement learning solver to find the exact forms of equilibria.

- We show that the social cost, quantified by moving distances of relay nodes, is linear to network size for the distributed mechanism. Compared to the social optimal assignment by a central authority, the cost for distributed mechanism is bounded by two measures of Price of Anarchy or Price of Stability. We further analyze how practical factors such as numbers of consumers/providers, network sizes affect these bounds, and demonstrate our approach remains feasible under provider/consumer mobility.

A preliminary version of this work has been accepted by IWQoS-17 [21]. The rest of this paper is organized as follows: We first review the related studies in Section 2. In Section 3 we show a network structure modeling data exchange system and some basic assumptions. Then in Section 4 we introduce a centralized method to manage the system. After that, we give definition of the autonomous compensation game for relay nodes as well as the analysis for its pure Nash equilibrium in Section 5. To address the difficulties in solving Nash equilibrium for general cases, we further propose a reinforcement learning solver to it and show its performance in Section 6. Then, we provide evaluation for the game theory model and some numerical simulation results in Section 7. Finally, summarize our conclusions and discuss some future works in Section 8.

2 Related Work

Data exchange among peers has been studied for some time. Current work focuses on three main aspects: efficient and accurate method for managing the exchanging process, proper platform for peers to exchange data, and fair payment rules for attracting participants.

Helgeson et al. provide a bottom implementation approach for managing data exchange in a network from industrial point of view, and it could be the basis for all data exchange model [6]. Rahman et al. study how to assure security in the exchanging process [15]. More recent work designs data exchange models dealing with further constraints such as ability to recover document damage [13], and adaption to big data [1]. Compared with our approach, they all consider a centralized system to supervise the whole process.

As for the choice of platform for data exchange, some Internet platforms are studied. Jang et al. consider personal cloud for data exchange and study how it enhances users' experiences [8]. Recently crowdsensing network becomes more popular due to its feasibility to accommodate actual mobility for participants of data exchange. Some crowdsensing network has followed the approach of Named Data Networking (NDN), which proposes evolving current host-centric network architecture (IP) to a data-centric network architecture (NDN) [23]. With this idea, Xiao et al. provide offline and online algorithms to do multi-task assignment in crowdsensing network [20]. But their algorithms are designed without payment for participants, which is the main focus of our work.

Algorithmic game theory has been partly used to study participants' incentive to exchange data. Gao et al. propose an auction policy to attract more long-term user participation, thus more data in sensor selection problem [5]. Luo et al. also design mechanisms for participatory sensing systems to incentivize contribution from users [12]. Duan et al. include implicit incentives, such as gratification, social position and sense of honor, as rewards to motivate more participants [3]. Similar as our work, they are trying to attract more users to participate through providing rewards, but their designs are dealing with the payments for providers and consumers. In contrast, our autonomous compensation game focuses on the rewards for relay nodes, which is also an important part in peer data exchange.

At the same time, some common mechanisms of classical game scenarios have been introduced to crowdsensing networks. Feng et al. consider a reverse auction as an approach for assigning sensing tasks, where each participant competing for tasks by bidding the payment he requires and the one with lowest bid wins [4]. Li et al. design a randomized auction for assigning tasks in crowdsensing networks, specifically dealing with unbalanced load that some sensors got few tasks but some others got to many tasks [11]. A Stackelberg game was introduced as an incentive framework for sensing smartphones against crowdsourcing platforms [22]. Compared to our work, when the assignments are done through auctions, a centralized platform acting as auctioneer is needed to collect all the bids and make the decision. And a Stackelberg game consists of two separate stages, requiring more time as a practical approach.

On the other hand, data transmissions under mobility have attracted much interest in the past ten years. Zhao et al. start a frame work of controlling the mobilities for data transport ferries in a delay-tolerant network [24]. In their model, data is relayed by ferrying nodes and stationary nodes that work together, playing a similar role as relay nodes in our work. They focus on designing routes for ferrying nodes. After them, Kavitha and Altman study similar message ferry routes design in sensor networks using polling models [9]. Zhu et al. survey other approaches considers exploiting social behaviors of delay-tolerant network nodes to make better routing decision [25]. Our work considers a different form for delivering data, mainly taking advantages of peer mobile devices. We consider incentives to facilitate their individual participation as relays, which does not require centralized routing design. If the desired data requires huge storage, these data ferry schemes may also be good choice.

3 Network Model and Basic Assumptions for Data Exchange

In this section, we first present our crowdsensing model for peer data exchange and basic assumptions. Then we illustrate how it works within the whole process through an example.

3.1 A Crowdsensing Approach for Data Exchange

Our data exchange system consists of several kinds of participants. Those willing to pay for specific data are consumers, and those who have the data, either originally generated by themselves or just being cached by them, are providers. Each provider may have different data, and each consumer needs data from all providers. In general, providers may not transmit the desired data directly to consumers due to the distance. Some intermediate nodes, called relays, have to move to certain locations to connect consumers and providers. After data delivery is completed, all these nodes send relay information about who send what data to whom to some clearance nodes, which decide how to divide consumers' payments among relays and providers. Note that it is nature that consumers can also act as relay nodes for transmitting data to other consumers, since they have the incentive to relay the data to others after caching it and share the corresponding rewards. But for providers, on the other hand, it is possible that some of them do not have enough storage capacity for all the required data except those they have cached. Thus we do not include any provider as a relay node for the sake of convenience.

Before describing the whole process, we first list several important assumptions here. Firstly, providers, consumers and clearance nodes are uniformly distributed and their positions do not change much in a short time. Clearance nodes can communicate with each other, so each can acquire all the relay information. Each node u entering the system, a provider, a consumer, or a relay, has a unique l-bit account ID number, denoted by A_u and known by themselves and the clearance nodes. Secondly we do not consider transmission failures or cheating nodes. The relay information has much smaller size compared to data, thus their transmission costs are negligible and not considered. Lastly, a relay node transmits one unit of data one time and the amount of transmission is the number of data units.

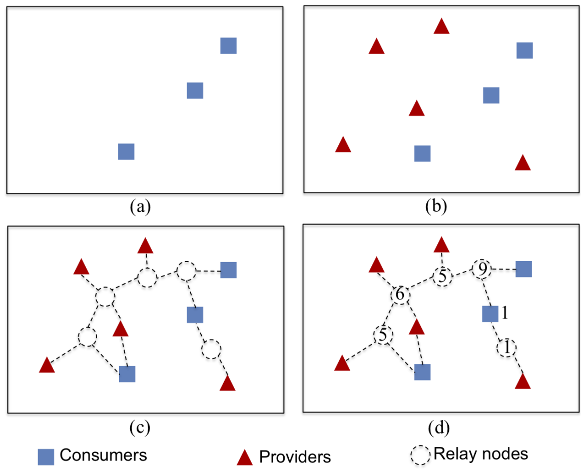


Fig. 1: A complete process for transmitting a group of data. The numbers beside nodes is the corresponding rewards if a relay nodes works there.

With these assumptions, we give a brief description of the whole process (Fig. 1.). Firstly, several consumers send requests for some data, each naming a payment. Then, providers with required data are found. After that, relay locations to connect consumers and providers are found, as well as the delivery paths. Then a central entity calculates the reward for each location, and assigns relay nodes to go to those locations to finish data delivery. Alternatively, if each relay knows the information about those locations and corresponding rewards, it can make its own decision to go to specific location, without the central entity. Once the delivery is done, all involved nodes send messages to clearance nodes, which decide the payments for all relays and providers. Table 1 provides all the notations used in the rest of our paper.

TABLE 1: Notations

N_1	Number of providers
N_2	Number of consumers
k	Number of necessary relay nodes
M	Number of potential relay nodes
D	Length of one side of network
d	Transmission range
ρ_i	i-th provider or its coordinates
σ_i	i-th consumer or its coordinates
ξ_i	i-th potential relay nodes
w_i	The i-th location or its coordinates
r_i	Reward associated with i-th relay location
$\mathcal{X}(w_i)$	Set of potential relay nodes going to i-th relay location
$U(\xi_i)$	Utility function of i-th potential relay nodes
d_{ij}	Distance between i-th relay location and j-th potential relay node
c	Constant factor between moving cost and moving distance
M_0	Smallest number of potential relay nodes Nash equilibrium exists
d_{max}	Largest distance between relay location and potential relay node
d_{min}	Shortest distance between relay location and potential relay node

3.2 Payment Decision for Each Participant

Two key factors decide payments for relays and providers. First, the system should get information for all the actual transmissions. Second, the payments should be "fair" to all participants. We will introduce a transmission graph structure as a record of how a specific piece of data is delivered from providers to consumers, then show how payments are decided to achieve fairness.

To construct the transmission graph, providers and relays send short messages to their nearest clearance nodes each time they receive a piece of data from others (i.e., inflow message, denoted by $T^{(i)}$), or send one to the next node (i.e., outflow message, denoted by $T^{(o)}$). Specifically, when data with label L is transmitted from node u to node v, both u and v will send a $3l+|L|$-bit tuple (L, A_ρ, A_u, A_v), where node ρ is the original provider of the data and $|L|$ is the length of the label. u may be a provider ρ or relay ξ, and v may be a relay ξ or consumer σ.

After data delivery, all clearance nodes communicate with each other so that all messages about the same data are handled by the same clearance node. For each group of providers and consumers (labeled by $\hat{L}$), such messages are used to build the whole transmission graph $TG(V, E, \hat{L}, \beta)$ by Algorithm 1 (Table 2). The algorithm analyzes all messages to obtain participating peers as vertices and corresponding transmissions as edges. $d_{in}(u)$ and $d_{out}(u)$ denote the in/out degrees for vertex u, and they are further used to decide which role the vertex plays in the whole transmission process.

The time/space complexities of Algorithm 1 are analyzed here. Firstly, the input size is $O(\Lambda)$ where Λ is the number of transmissions for $\hat{L}$. For time complexity, the initial setup requires a traversal of all input, which is $O(\Lambda)$. The main loop needs at most $\Lambda(\Lambda-1)/2$ times traverse of all input, which is at most $O(\Lambda^3)$. The final step to mark each vertex needs to traverse all the vertices and determine the number of their neighbors, which is at most $O(\Lambda^2)$. Thus the time complexity for the whole algorithm is $O(\Lambda^3)$. For space complexity, while running this algorithm, all variables that

TABLE 2: Algorithm 1: Rebuilding Transmission Graph

```
Input: $\{T_j^{(*)}\}_{j=1}^{\Lambda} = \{(\hat{L}, A_{\hat{\rho}}, A_{u_j}, A_{v_j})\}_{j=1}^{\Lambda}$
Output: $TG(\hat{L}) = (V, E, \beta, \hat{\Delta})$
1: Set $V = \bigcup_{j=1}^{\Lambda} \{u_j, v_j\}$, $\hat{\Delta} = \bigcup_{j=1}^{\Lambda} \{\rho_j\}$ and $E = \emptyset$
2: For $(\hat{u}, \hat{v} \in V)$
     If $\exists j_1, j_2$ such that $u_{j_1} = u_{j_2} = \hat{u}, v_{j_1} = v_{j_2} = \hat{v}$
       Add directed edge $(\hat{u}, \hat{v})$ into $E$
3: For $(\widetilde{u} \in V)$
     If $d_{in}(\widetilde{u}) > d_{out}(\widetilde{u})$
       $\beta(\widetilde{u}) = 1$
     Else If $d_{in}(\widetilde{u}) = d_{out}(\widetilde{u})$
       $\beta(\widetilde{u}) = 2$
     Else
       $\beta(\widetilde{u}) = 3$
```

need to be saved is the adjacent matrix of the graph, which has a size of at most $O(\Lambda^2)$. The type of each vertex has size of at most $O(\Lambda)$. Thus the space complexity is $O(\Lambda^2)$.

Participants are classified by function $\beta : V \to \{1, 2, 3\}$ into 3 types for payment decision. To be precise, nodes of type-1 are consumers or those who act like consumers, since they receive at least one piece of data but do not propagate it. Meanwhile, nodes of type-2 and type-3 are relay nodes. The difference between them is that type-2 nodes transmit a piece of data to one peer after receiving it, but type-3 ones transmit it to multiple peers. These type-3 nodes may be located at important positions for transmitting data, or store the data for a long time so that even after the provider leaves, the new coming consumers can still access the data. In this sense, they make extra efforts in transmitting data.

With previous characterization, we can decide the payment rule.

- Type-1 nodes pay a price of data (decided in advance). All payments form the whole reward pool.
- P_1 percent of the whole payment is shared equally by the providers $\rho \in \hat{\Delta}$.
- P_2 percent of the whole payment should be given to all the vertices of type-2 and 3, and how much each of them (say u) receives is proportional to the ratio $\frac{d_{out}(u)}{\sum\limits_{\beta(v)\neq 1} d_{out}(v)}$. This awards each relay node proportionally to the transmissions it does.
- The rest of the payment will be given to all the vertices of type-3. How much a peer u receives is proportional to the ratio $\frac{d_{out}(u)-d_{in}(u)}{\sum\limits_{\beta(v)=3} d_{out}(v)-d_{in}(v)}$. We use the number of transmissions for sending out data minus the number of transmissions for receiving data to measure a node's extra effort in transmitting data, and this rule is to award it according to the proportion of its extra efforts in all such efforts.

Note that the provider here may be the generator or a previous relay node who stored the data. In either case, it will be rewarded as nodes of type-3 for its effort in transmitting data, and further rewarded according to the second role if it is the generator.

4 Centralized Method to Assign Relay Nodes

In this section, we provide a centralized method to assign relay nodes to relay locations. Here we suppose that the central entity knows all nodes' positions. Providers and consumers do not move before data delivery completes, and relay nodes, supposed to be enough, will follow the assignments. Meanwhile, the energy cost of transmitting each data unit is the same. Then the assignment has two steps. The system first finds necessary locations requiring relay nodes to deliver data. Then it assigns proper potential relay nodes to these locations.

Step 1: Find relay locations
Given N_1 providers $\{\rho_j\}_{j=1}^{N_1}$, N_2 consumers $\{\sigma_j\}_{j=1}^{N_2}$ in some bounded 2-dimensional field, say a $D \times D$ square, and a transmission range d, the system seeks the least k locations $\{w_j\}_{j=1}^{k}$ and corresponding edges, such that $\forall i, j$ there exists a path from ρ_i to σ_j via some $w_l's$ or $\sigma_{j'}'s$, with all edges shorter than d. We call this *relay location decision problem*. Here we want to minimize the number of relay locations, i.e. the number of necessary relay nodes, so that a centralized system can assign the least number of relay nodes to relay locations.

To show the complexity of the relay location decision problem, we refer to a NP-hard problem, namely *Steiner tree problem with minimum number of Steiner points* (STP-MSP in short), which is to find the least number of extra points and corresponding edges within bounded length to connect a set of terminal points $\{p_1, p_2, \ldots, p_n\}$ in two-dimensional Euclidean plane [2]. To be specific, relay location decision problem requires a similar solution with respect to two sets of terminal points, and one of them must be leaf-points, i.e. having degree one.

We reduce the STP-MSP problem to relay location decision problem to show the complexity. The solution to STP-MSP problem must be a tree and at least one of terminal points is leaf-point (because if an extra point is leaf-point, it could be deleted to decrease the number). So solving a STP-MSP problem for terminal points $\{p_1, p_2, \ldots, p_n\}$ is exactly solving the relay location decision problem for a p_j being provider ρ_1 and the rest of points being consumers $\{\sigma_1, \sigma_2, \ldots, \sigma_{n-1}\}$. Thus if we have an algorithm to solve relay location decision problem, we could solve STP-MSP problem by running it n times, selecting a p_j each time as ρ_1, and find the best result among them. This shows the relay location decision problem is also NP-hard.

Despite its NP-hardness, we provide an approximation algorithm to solve it in Table 3. And here we only give a rough estimate for such algorithm's approximation ratio. In fact Algorithm 2 without the last loop (line 8) is a ratio-3 approximation algorithm solving STP-MSP problem, and the last loop will increase the number of vertices in the solution by at most N_1. At the same time, the number of relay locations in a solution to relay location decision problem (denoted by k) must not exceed the number of Steiner points in STP-MSP problem, when both N_1 providers and N_2 consumers are terminal points. To sum up, Algorithm 2 outputs a solution that minimizes the number of relay locations at no more than $3k + N_1$. Meanwhile, k is a non-negative integer. When $k = 0$, all providers can transmit data directly to consumers, and line 3 in Algorithm 2 makes sure that in this case the algorithm also outputs a solution that no relay location is needed. And when $k > 0$, $3k + N_1 \leq (3 + N_1) \cdot k$. In conclude, the Algorithm 2's approximation ratio, which is the ratio between the number of locations the algorithm find in the worst case and the one in the optimal solution, is $N_1 + 3$.

And to see the time complexity of such algorithm, we only need to look at its five loops and an operation of sorting. In the third loop (line 6), there are at most $O(N_1 + N_2)$ connected components at beginning and each of them has at most $O(N_1 + N_2)$ vertices. The number of connected components will decrease by two each round, or remain unchanged at some round and then the loop ends. So the whole loop consists of at most $O(N_1 + N_2)$ rounds and each round requires at most $O(N_1 + N_2)^3$, and the time complexity is $O((N_1 + N_2)^4)$. For the rest loops (line 3, 5, 7, 8), the time complexity for each of them is at most $O(N_1 + N_2)^2$, and the sorting operation (line 4) is $O((N_1 + N_2)^2 Log(N_1 + N_2))$, since there are at most $O((N_1 + N_2)^2)$ edges. To sum up, the time complexity of Algorithm 2 is $O((N_1 + N_2)^4)$.

TABLE 3: Algorithm 2: Deciding relay locations

```
Input: $\rho_1, \rho_2, \ldots, \rho_{N_1}, \sigma_1, \sigma_2, \ldots, \sigma_{N_2}, d$
Output: Graph $T = (V, E)$
1: Set $V = \{\rho_1, \rho_2, \ldots, \rho_{N_1}, \sigma_1, \sigma_2, \ldots, \sigma_{N_2}\}$, $E = \emptyset$
2: Let $e_{uv}$ be the edge between $u \neq v \in V$
3: For ($u \in \{\rho_1, \rho_2, \ldots, \rho_{N_1}\}, v \in \{\sigma_1, \sigma_2, \ldots, \sigma_{N_2}\}$)
     If $e_{uv} \leq d$
       Add $e_{uv}$ into E
4: Sort all $e_{uv}$'s to $\{e_i\}$ in length increasing order
5: For ($i \wedge e_i \leq d$)
     If $e_i$ connects two different connected components of $T$
       Add $e_i$ into E
6: While ($\exists$ more than two connected components) Do
     For ($a, b, c \in V$ in three connected components of $T$)
       If $\exists s$, s.t. edge $e_{sa}, e_{sb}, e_{sc}$ shorter than $d$
         Add $s$ into $V$ and $e_{sa}, e_{sb}, e_{sc}$ into $E$
   End while
7: For ($i$)
     If $e_i$ connects two different connected components of $T$
       Divide $e_i$ into $\lceil \frac{|e_i|}{d} \rceil$ parts and add into $T$
8: For ($1 \leq i \leq N_1$)
     If $\rho_i$ has degree larger than one
       Replace $\rho_i$ by a new vertex $\rho_i'$ and add $(\rho_i, \rho_i')$ into $E$
```

In the output graph, vertices who are not from input are relay locations, in each of which at least one relay node is needed to help transmit data. The reward $r_i, i = 1, 2, \ldots, k$ associated with each of them can be calculated as follows. Firstly, find a path from each provider to each consumer. Then change these undirected edges into directed ones according to the direction of paths they are in to get the transmission graph. Finally use the payment rule introduced in Section 3.

Step 2: Assign relay nodes

Given relay locations, the system chooses relay nodes closest to these locations from potential ones, so that the summation of distances all relay nodes should move is minimized.

To obtain this, we can first construct a complete weighted bipartite graph $G(V_1, V_2, E, w)$ where the relay locations belong to V_1 and the original positions of potential relay nodes belong to V_2. There is an edge between each vertex $u \in V_1$ and each one $v \in V_2$ weighted by their distance. Here we call it a bipartite matching that a set of edges where no two edges share a common vertex and all vertices in V_1 is included. Here we only consider the situation that the numbers of vertices in V_1 and V_2 are the same, since in practice the system could either choose a proper number of vertices for V_2 according to their appearance time, or add virtual vertices to V_1 and set the weights of edges between virtual vertices and vertices from V_2 to be 0, meaning relay nodes going no where. Thus a feasible assignment for relay nodes is corresponding to a bipartite matching, and the bipartite matching where the sum of the weights of the edges has a minimum value, called the minimum weighted matching, is the optimal assignment in term of the summation of distances all relay nodes should move.

With this construction, we use a modified version of *Kuhn-Munkres* algorithm [14] to finds the minimum weighted bipartite matching for $G(V_1, V_2, E, w)$. The algorithm is summarized in Table 4. In the defined function $Match(G, M)$, the alternating path means a path that begins with an unmatched vertex and is a path in which the edges belong alternatively to the matching and not to the matching. Thus in the algorithm, if the path T meets another unmatched vertex, simply taking the symmetric difference of T and M yields another matching M' which matches one more vertex than M, and such a path is often referred as an augmenting path.

TABLE 4: Algorithm 3: Assigning relay nodes

```
Input: Graph $G(V_1, V_2, E, w)$
Output: Optimal assignment for relay nodes
1: Def $Match(G, M)$ \\ Graph $G$, Matching $M$ in $G$
2:     If $M$ matches all vertices in $V_1$
         Return $M$
3:     Else
         Let $u$ be a vertex in $V_1$ unmatched in $M$
         Search an alternating path $T$ from $u$
         If an unmatched vertex $v \in V_2$ is found in $T$
           Use the path from $u$ to $v$ to augment $M$, yielding $M'$
           Return $Match(G, M')$
         Else
           Return $T$
4: Set $w(u, v) = -w(u, v), \forall u \in V_1, v \in V_2$
5: Set Vertex labels $l(u) = \max_{v \in V_2}\{w(u, v)\}$ for all $u \in V_1$
       and $l(u) = 0$ for all $v \in V_2$
6: Let $G_=(V_1, V_2, E_=, w)$ be the equality subgraph where
       $E_= = \{(u, v) \in E | l(u) + l(v) = w(u, v)\}$
7: Let $M$ be any matching in $G_=$
8: If $Match(G_=, M)$ returns a match $M$
     Return $M$
9: Else \\ $Match(G_=, M)$ returns $T$
     Let $\alpha = \min_{u \in T \cap V_1, v \notin T \cap V_2}\{l(u) + l(v) - w(u, v)\}$
     Update $l(v) = \begin{cases} l(v) - \alpha & v \in T \cap V_1 \\ l(v) + \alpha & v \in T \cap V_2 \\ l(v) & otherwise \end{cases}$
10:      Goto Line 6
11: For each relay location $u \in V_1$
      Assign relay node $v = M(u)$ to $u$
```

The main time complexity of algorithm 4 comes from repeatedly searching for desired matching in the equality subgraph. In precise, each searching consists of at most $|V_1|$ modification of augmenting paths for each vertex in V_1, resulting in a complexity of $O(|V_1|^2)$. And such a searching is done after updating vertex labels $l(u)$ according to a factor α calculated at line 9. The updating is needed for at most $O(|V_1|)$ times, meaning a total time complexity of $O(|V_1|^3)$. As for the space complexity, the algorithm only requires storing the graph and intermediate variables such the alternating paths and matchings. Thus its space complexity is $O(|V_1|^2)$.

5 Autonomous Compensation Game for Individual Relay Node Decisions

The centralized method does not consider the preferences of relay nodes. As shown in Fig. 1. (d), the reward associated with each relay location varies a lot. A centralized system is able to minimize the relay nodes' moving distance so data transmission can begin quickly. However, if relays can make their own decisions, they may choose locations with high rewards, not closest.

We want to design a game where relay nodes make individual decisions of destinations, and we hope there exists Nash equilibrium. Intuitively, a Nash equilibrium is a group of players' strategies where no one could strictly benefit by changing only his own strategy. Specific to our problem, the equilibrium means each relay node goes to a location where he can gain the most reward if others do not change their locations. With the assumption that every player is rational to choose the best strategy for himself, relay nodes will choose locations corresponding to the Nash equilibrium individually.

Now we introduce the autonomous compensation game. We suppose that the optimal positions for relay nodes, as well as the corresponding rewards, are known to all potential relay nodes, either calculated by themselves or broadcast by the system. In this game, the reward of each location will be equally shared by all players moving there. So players need to find a strategy to decide where to move to gain higher utility. Generally speaking, the utility of each player should be the reward he receives by acting as relay minus the energy cost of doing so and other cost for moving to the specific position. The energy cost is proportional to the number of transmission one does, thus proportional to the reward. While the relationship between the moving cost and the reward is hard to estimate. When the reward is large enough, the moving cost can be ignored. Otherwise, the moving cost will influence the equilibria of the game. So we analyze both cases.

5.1 Utilities without Moving Cost

Suppose there are M players ($\{\xi_j\}$) and k locations ($\{w_i\}$) with value (r_i). Define $\mathcal{X} : \{w_i\} \to 2^{\{\xi_j\}}, \mathcal{X}(w_i) = \{j | \text{node } \xi_j \text{ goes to location } w_i\}$. The utility of each player is $U(\xi_j) = \frac{r_i}{|\mathcal{X}(w_i)|}$ where $j \in \mathcal{X}(w_i)$. Here we consider players choose pure strategy, that is each one decides to go to only one location to do transmission in a short period. Then the corresponding pure Nash equilibrium requires

$$\mathcal{X}(w_i) \neq \emptyset, \qquad \forall i \tag{1}$$

$$\frac{r_i}{|\mathcal{X}(w_i)|} \geq \frac{r_j}{|\mathcal{X}(w_j)+1|}, \qquad \forall i \neq j \tag{2}$$

Constraint (1) guarantees that there is at least one relay node at each relay location, so the network is connected. Constraint (2) ensures that when everyone chooses the strategy associated to the equilibrium, each player cannot gain more utility by changing his strategy when others keep their strategies. Here players are homogeneous since they choose the strategies to maximize their utilities without considering the cost generated during their moving to specific locations. As a result, to find an equilibrium, we only need to study the number of players going to each location.

Before we show Theorem 1 about the condition for the existence of Nash equilibrium, we first give an intuition how it is derived. From constraint (1), at least one peer occupies location w_k. From his point of view, if there exists a location w_t with reward r_t satisfying: $r_t > m \cdot r_k$ but $|\mathcal{X}(w_t)| < m$, then he will go to w_t to improve his utility. Thus, the reward of location w_i should be divided by at least $\lceil \frac{r_i}{r_k} \rceil - 1$ agents once $\frac{r_i}{r_k} > 2$. This inspires us that if the number of peers $M = M_0 = \sum_{i=1}^{k} \max\{1, \lceil \frac{r_i}{r_k} \rceil - 1\}$, there exists an equilibrium. Furthermore, if $M > M_0$, we can first construct an equilibrium allocation for M_0 agents, and prove there is an equilibrium for each $M > M_0$ by induction. Finally the uniqueness can be proved by contradiction.

Theorem 1. *A sufficient and necessary condition for the existence of pure Nash equilibrium is:*

$$M \geq \sum_{i=1}^{k} \max\{1, \lceil \frac{r_i}{r_k} \rceil - 1\} \tag{3}$$

where r_i is in decreasing order, and the equilibrium is unique if the equality holds.

Proof. Firstly, we prove that $M \geq \sum_{i=1}^{k} \max\{1, \lceil \frac{r_i}{r_k} \rceil - 1\}$ is a necessary condition. It suffices to prove that if $M < \sum_{i=1}^{k} \max\{1, \lceil \frac{r_i}{r_k} \rceil - 1\}$, the equilibrium does not exist. Suppose otherwise, i.e., there is an equilibrium, under which the allocation is denoted by $\mathcal{X}$. In this case, there must be at least one i such that $|\mathcal{X}(w_i)| < \max\{1, \lceil \frac{r_i}{r_k} \rceil - 1\}$, and let t be the smallest index i satisfying $|\mathcal{X}(w_i)| < \max\{1, \lceil \frac{r_i}{r_k} \rceil - 1\}$. Since it is an equilibrium, $\mathcal{X}(w_t) \neq \emptyset$, i.e., $\mathcal{X}(w_t) \geq 1$, then $\max\{1, \lceil \frac{r_i}{r_k} \rceil - 1\} = \lceil \frac{r_i}{r_k} \rceil - 1$, and further $|\mathcal{X}(w_t)| + 1 < \frac{r_t}{r_k}$. So for any $\xi_j \in \mathcal{X}(w_k)$, its reward is at most r_k in the equilibrium, but will increase to $\frac{r_t}{|\mathcal{X}(w_t)|+1} > r_k \geq \frac{r_k}{\mathcal{X}(w_k)}$ if it goes to location w_t, which is a contradiction.

Secondly, we prove that $M \geq \sum_{i=1}^{k} \max\{1, \lceil \frac{r_i}{r_k} \rceil - 1\}$ is a sufficient condition. Here we only need to construct a feasible allocation under such condition and prove that it is an equilibrium. Let $M_0 = \sum_{i=1}^{k} \max\{1, \lceil \frac{r_i}{r_k} \rceil - 1\}$ and $M_1 = M - M_0$. To construct the allocation, we do it in two steps. If $M_1 = 0$, step one is enough.

In the first step, we let $\mathcal{X}^0(w_1) = \{\xi_j, j = 1, 2, \ldots, \max\{1, \lceil \frac{r_1}{r_k} \rceil - 1\}\}$, and $\mathcal{X}^0(w_i) = \{\xi_j, j = \sum_{l=1}^{i-1} \max\{1, \lceil \frac{r_l}{r_k} \rceil - 1\} + 1, \ldots, \sum_{l=1}^{i} \max\{1, \lceil \frac{r_l}{r_k} \rceil - 1\}\}$ for $i = 2, \ldots, k$. Correspondingly, we let $r_i^0 = \frac{r_i}{|\mathcal{X}^0(w_i)|}, i = 1, 2, \ldots, k$.

In the second step, for $t = 1, \ldots, M_1$:

1) find s such that $r_s^{t-1} = \max_i \{r_i^{t-1}\}$ (if there are more than one s's, choose the first one)
2) update $\mathcal{X}^{t-1}$ to $\mathcal{X}^t$ by adding ξ_{M_0+t} into $\mathcal{X}^{t-1}(w_s)$
3) update r^{t-1} to r^t correspondingly.

Now we prove that by the previous two steps we construct an equilibrium. On the one hand, we show that the first step constructs an equilibrium for $M = M_0$. It is obvious that the first condition of an equilibrium, $\forall i, \mathcal{X}^0(w_i) \neq \emptyset$, holds. For the second condition, we prove by contradiction. Suppose $\exists i_1, i_2, i_1 \neq i_2$ such that

$$\frac{r_{i_1}}{|\mathcal{X}^0(w_{i_1})|} < \frac{r_{i_2}}{|\mathcal{X}^0(w_{i_2})| + 1}.$$

Since $|\mathcal{X}^0(w_{i_1})| = \max\{1, \lceil \frac{r_{i_1}}{r_k} \rceil - 1\}$, there are three possible cases:

1) $|\mathcal{X}^0(w_{i_1})| = 1$ and $i_1 = k$
2) $|\mathcal{X}^0(w_{i_1})| = 1$ and $i_1 < k$
3) $|\mathcal{X}^0(w_{i_1})| = \lceil \frac{r_{i_1}}{r_k} \rceil - 1 > 1$

and all these cases lead to $\frac{r_{i_1}}{|\mathcal{X}^0(w_{i_1})|} \geq r_k$.

Similarly, there are three possible cases for i_2:

1) $|\mathcal{X}^0(w_{i_2})| = 1$ and $i_2 = k$
2) $|\mathcal{X}^0(w_{i_2})| = 1$ and $i_2 < k$
3) $|\mathcal{X}^0(w_{i_2})| = \lceil \frac{r_{i_2}}{r_k} \rceil - 1 > 1$

In case 1), $\frac{r_{i_2}}{|\mathcal{X}'(w_{i_2})|+1} = \frac{r_k}{2} < r_k$. In Case 2), we derive that $\lceil \frac{r_{i_2}}{r_k} \rceil - 1 \leq 1$, that is $\frac{r_{i_2}}{r_k} \leq 2$, and further we get $\frac{r_{i_2}}{|\mathcal{X}'(w_{i_2})|+1} = \frac{r_{i_2}}{2} \leq r_k$. Finally, in case 3), $\frac{r_{i_2}}{|\mathcal{X}'(w_{i_2})|+1} = \frac{r_{i_2}}{\lceil \frac{r_{i_2}}{r_k} \rceil} < r_k$. To sum up, $\frac{r_{i_2}}{|\mathcal{X}'(w_{i_2})|+1} \leq r_k$, and

$$\frac{r_{i_2}}{|\mathcal{X}^0(w_{i_2})|+1} \leq r_k \leq \frac{r_{i_1}}{|\mathcal{X}^0(w_{i_1})|}$$

, which leads to a contradiction.

One the other hand, we show that the second step constructs an equilibrium for M agents by induction. It suffices to prove that after the first loop, we actually construct an equilibrium for $M_0 + 1$ agents. Let us consider $\mathcal{X}^1$, and it is obvious that the first condition of an equilibrium, $\forall i, \mathcal{X}^1(w_i) \neq \emptyset$, holds. For the second condition, we prove by contradiction. Suppose $\exists i_1, i_2, i_1 \neq i_2$ such that

$$\frac{r_{i_1}}{|\mathcal{X}^1(w_{i_1})|} < \frac{r_{i_2}}{|\mathcal{X}^1(w_{i_2})|+1}.$$

Since $\mathcal{X}^0$ is an equilibrium of M_0 agents, and the only difference between $\mathcal{X}^0$ and $\mathcal{X}^1$ is $\mathcal{X}^1(w_s) = \mathcal{X}^0(w_s) \cup \{\xi_{M_0+1}\}$, i_1 must be s. But according to the definition of s, $\frac{r_{i_1}}{|\mathcal{X}^0(w_{i_1})|} \geq \frac{r_{i_2}}{|\mathcal{X}^0(w_{i_2})|}$, and further $\frac{r_{i_1}}{|\mathcal{X}^0(w_{i_1})|+1} \geq \frac{r_{i_2}}{|\mathcal{X}^0(w_{i_2})|+1}$. This means

$$\frac{r_{i_1}}{|\mathcal{X}^1(w_{i_1})|} \geq \frac{r_{i_2}}{|\mathcal{X}^1(w_{i_2})|+1}$$

, which is a contradiction.

Finally, we consider the uniqueness of the equilibrium. From the previous analysis, if $M = M_0$, we have constructed an equilibrium, where the number of players going to each location is determined by the formula $|\mathcal{X}(w_i)| = \max\{1, \lceil \frac{r_i}{r_k} \rceil - 1\}$ for $i = 1, 2, \ldots, k$. Suppose there is another equilibrium, that means there exists at least one $l < k$ such that $|\mathcal{X}(w_l)| < \max\{1, \lceil \frac{r_l}{r_k} \rceil - 1\}$. With similar analysis we know for any $\xi_j \in \mathcal{X}(w_k)$, its reward will increase if it goes to location w_l, which is a contradiction. □

5.2 Utilities with Moving Cost

In practice, relay nodes moving to specific positions also incur costs, which depend on the moving distances. We define d_{ij} as the distance between w_i and ξ_j, and it is common to assume the cost for ξ_j moving to w_i is $c \cdot d_{ij}$ with a constant factor c. Then the utility of each agent become $U(\xi_j) = \frac{r_i}{|x(w_i)|} - c \cdot d_{ij}$ where $j \in \mathcal{X}(w_i)$. The Nash equilibrium requires

$$\mathcal{X}(w_i) \neq \emptyset, \tag{4}$$

$$\frac{r_i}{|\mathcal{X}(w_i)|} - c \cdot d_{ij} > 0, \tag{5}$$

$$\frac{r_i}{|\mathcal{X}(w_i)|} - c \cdot d_{ij} \geq \frac{r_{i'}}{|\mathcal{X}(w_{i'})+1|} - c \cdot d_{i'j}, \quad \forall i' \neq i \tag{6}$$

for $\forall i$ and $\forall j \in \mathcal{X}(w_i)$. Constraint (4) is the same as constraint (1) and constraint (6) plays the same role as constraint (2). Constraint (5) assures all players have positive utilities, because only then they have incentive to participate.

Since the players are no more homogeneous, the equilibrium is determined by not only the number of players going to each location, but also who those players are. So we use a group of vectors $\vec{x}_j = \{x_{1j}, x_{2j}, \ldots, x_{kj}\}$ for $j = 1, 2, \ldots, M$ to represent players' strategies. Here $x_{ij} \in \{0, 1\}$ representing whether the player j goes to location w_i. And a simple constraint is that $\sum\limits_{i=1}^{k} x_{ij} \leq 1$ for $\forall j$, due to the fact each player is only able to go to at most one location. And further constraint (4)-(6) is derived into integer inequations:

$$\sum_{j=1}^{M} x_{ij} - 1 \geq 0, \forall i \tag{7}$$

$$\sum_{i=1}^{k} x_{ij} \left(\frac{r_i}{\sum\limits_{j'=1}^{M} x_{ij'}} - c \cdot d_{ij} \right) > 0, \forall j \tag{8}$$

$$\sum_{i=1}^{k} x_{ij} \left(\frac{r_i}{\sum\limits_{j'=1}^{M} x_{ij'}} - c \cdot d_{ij} \right) - \left(\frac{r_{i'}}{1 + \sum\limits_{j'=1}^{M} x_{i'j'}} - c \cdot d_{i'j} \right) \geq 0 \quad \forall i' \neq i, \forall j \tag{9}$$

Generally speaking, solving a system of integer equations is an NP-hard problem and there is no common method for approximate solution. It is possible that there are multiple solutions to it. As a result, we seek the necessary and sufficient condition for the existence of pure Nash equilibrium in analytic form. Here we provide an analysis similarly as the one before. We suppose r_i is in descending order. When the equilibrium exists, we could still consider the utility of a player going to the location with the lowest reward, that is $\xi_j \in \mathcal{X}(w_k)$. Constraint (6) for him becomes:

$$\frac{r_i}{|\mathcal{X}(w_i)|+1} - c \cdot d_{ij} \leq r_k - c \cdot d_{kj}$$

for each location w_i. Then a necessary condition for the existence of a solution can be derived as follows. We define $d_{max} = \max\{d_{ij}, 1 \leq i \leq k, 1 \leq j \leq M\}$ and $d_{min} = \min\{d_{ij}, 1 \leq i \leq k, 1 \leq j \leq M\}$. And consider for each i, in the above inequality $d_{ij} < d_{max}$ and $d_{kj} > d_{min}$, then $|\mathcal{X}(w_i)| \geq \frac{r_i}{r_n + c \cdot (d_{max} - d_{min})}$. Combined with constraint (4), which is $|\mathcal{X}(w_i)| > 0$, and the fact $|\mathcal{X}(w_i)|$ is an integer, we have the following necessary condition

$$M \geq \sum_{i=1}^{k} \max\{1, \lceil \frac{r_i}{r_k + c \cdot (d_{max} - d_{min})} \rceil - 1\} \tag{10}$$

However, the necessary condition above is not sufficient, as Fig. 2. illustrates. In the example, condition (10) is satisfied, but any possible choice of these two players could not satisfy all constraints (4)-(6).

5.3 Dynamic Approach in Practice

According to Theorem 1 in Section 5.1, if there are enough players, the pure Nash equilibrium exists for the case that moving cost is ignored. However, it is very important that such equilibrium can be approached in practice. This is because in most circumstance, the peers join the game, meaning deciding to relay data, in a series of time. In this part, we maintain all the notations in previous analysis and

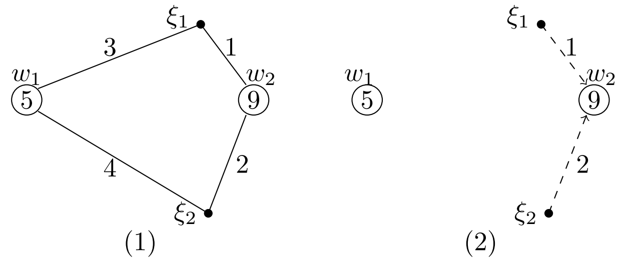


Fig. 2: An example showing the Nash equilibrium may not exist. Each circle stands for a relay locations with the number inside showing the corresponding reward. Each solid point stands for the original location of a potential relay and the number on each edge is the distance between two locations. We use solid lines to describe the settings and directed dashed lines for positive choice of relay nodes. (1) The setting of the example. Here the necessary condition (Eqn. 10) is satisfied. (2) Players ξ_1 and ξ_2 may choose strategies to maximize their utility, but leaving location w_1 empty, and further failing the whole transmission.

suppose that there is at most one player joining the game at a time. Without lose of generality, we suppose player ξ_1 is the first one, followed by ξ_2, and so on. Now we show how the equilibrium is approached when the moving cost is relative low with respect to the rewards:

(1) Before the M_0-th ($M_0 = \sum_{i=1}^{k} \max\{1, \lceil \frac{r_i}{r_k} \rceil - 1\}$ defined as before) player joins, each player chooses the location with highest actual reward, that is the reward for each peer going there, at the moment he decides to participate. If there are more than two such kind of locations, choose the one with lowest total reward. This makes sure that when there are M_0 players, their choice is exactly the equilibrium we construct in the proof of Theorem 1.

(2) After that, each player still chooses the location with highest actual reward when he comes. But if there are more than two such kind of locations, choose the one with highest total reward. According to the proof of Theorem 1, we know the equilibrium always holds after any players join in this way.

However, if the moving cost cannot be ignored, the previous strategy is not always suitable. Peers may regret their choices after their followers appear, as the example in Fig. 3 shows. In practice, it is possible that a peer is willing to turn around to go to another location after knowing others' choices, but this may incur more moving cost which is hard to measure. Thus in our analysis, we assume that players are not able to change their destinations after they have made decisions.

6 A Reinforcement Learning Solver for Autonomous Compensation Game

To address the difficulties in solving Nash equilibria when moving costs cannot be ignored, we propose a new approach through reinforcement learning [17] in this section.

As we introduced in Section 5.2, solving the integer inequations corresponding the equilibria is an *NP-hard* problem, and a direct approach, searching through all possible values for variables and check the validity, requires exponential time complexity ($O(k^M)$). For such kind of games with complicated strategy spaces, reinforcement learning methods have been commonly applied to find approximated optimal strategies for players or Nash equilibria for the game [7], [18], especially after its great success in the Go game [16]. We adopt reinforcement learning (RL) as a solver framework for general autonomous compensation game, illustrated in Fig. 4a.

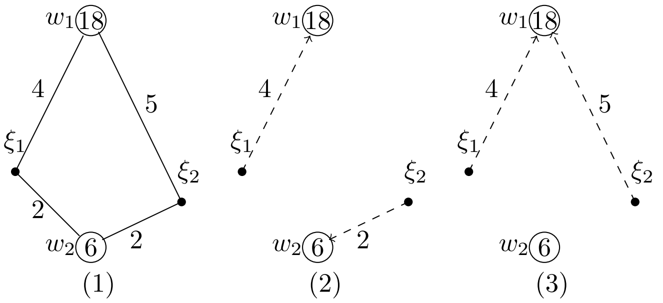


Fig. 3: An example showing whether the Nash equilibrium can be approached depends on the order of players joining the game. We use the same representation as Fig. 2. (1) The setting of the example. (2) If ξ_1 joins first, the equilibrium is achieved. (3) If ξ_2 joins first, the equilibrium cannot be achieved.

6.1 Markov Decision Process in Autonomous Compensation Game

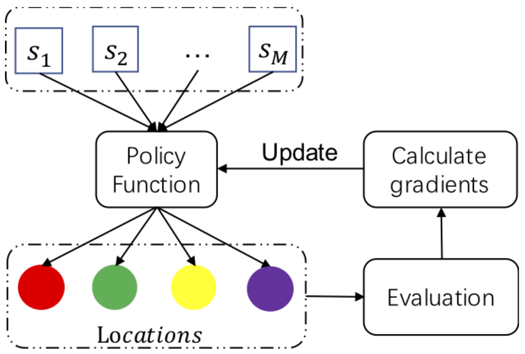
A Markov decision process (MDP) is a 4-tuple, (S, A, T, R), where S is the set of states, and A is the set of actions. $T^a_{s_t} = s_{t+1}$ is the transition function indicating that if we choose action a at time t in state s_t, the process will lead to state s_{t+1} at time $t+1$. R^a_s is the reward at the time t, transiting from state s to state s_{t+1}, due to action a.

We can model the process of an agent solving a Nash equilibrium, i.e. solving inequations (7)-(9), as an MDP. Here we consider the input of an autonomous compensation game, M players ($\{\xi_j\}$), k locations ($\{w_i\}$) with value (r_i), as well as the distances between them $\{d_{ij}\}$ and the constant c, are given. At the beginning, all the locations are empty and the players come in the order from ξ_1 to ξ_M and decide which locations to go. Thus, for each player ξ_j, the agent observes a state s_j (the situation of the game when player ξ_j comes), and its policy chooses an action a_j (i.e one location ξ_j goes to). Then the reward R^a_s is partially obtained (described later), and the next player ξ_{j+1} comes and the state of the game changes to s_{j+1}. Specifically, after the last player ξ_M goes to location a_M, the process ends. In other words, state s_M is called terminal state. The process is shown in Fig. 4b. It is named "Markov" because the transition from state s_j to s_{j+1} can be decided only by s_j and a_j, which is also known as Markov property.

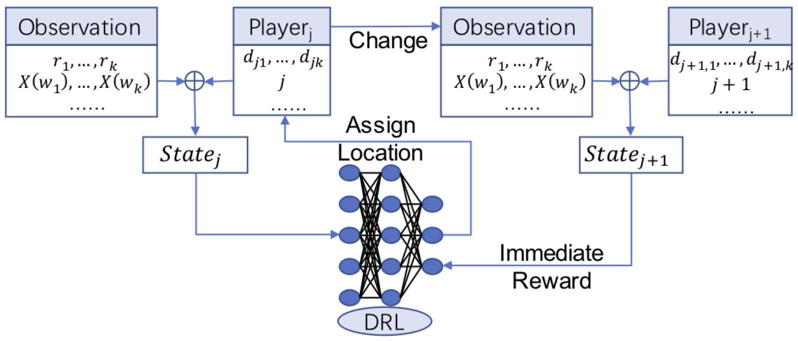
State: Basically, each state s_j in MDP consists of two parts, the feature of player ξ_j (including his ordinality and the distance between him and each location) and the observation of the game (including values of locations and the number of players in each location).

Action: Actions in MDP include locations ($\{1, 2, \ldots, k\}$) and a special action (0) representing going nowhere. The policy $\pi(\cdot)$ decides which action to choose to maximize the reward. Specifically, if action $a_j \neq 0$ is taken for player ξ_j, it means $x_{a_j,j} = 1$ and $x_{ij} = 0$ for $i \neq a_j$.

Transition: Corresponding to states, transitions in MDP also



(a) Overall framework.



(b) The Markov Decision Process

Fig. 4: (a) The overall framework. The squares represent states in a trajectory. The circles are corresponding actions the policy decides. The results are evaluated after the whole trajectory. Then the policy is updated accordingly. (b) Part of the Markov Decision Process for an autonomous compensation game.

include two parts. For each state s_j, the feature of player ξ_j transmits to the one of the next player ξ_{j+1} and the observation of the game changes according to the fact that ξ_j goes to location a_j.

Reward: Generally speaking, the state-action reward $R_{s_j}^{a_j}$ corresponding to each player cannot be decided accurately until all players' actions are decided. This is because each player's utility in the game depends on all other players' decisions. Thus we calculated the reward for the whole trajectory ($R_{s_j}^{a_j}$ for $j = 1, 2, \dots, M$) at the terminal state, and back-propagate the reward signal through the trajectory, which is commonly called delayed reward. To be precise, we sum the values of the left-hand-side of all the inequations (7)-(9) with proper normalization as the delayed reward. The summation is naturally higher if an equilibrium is obtained. On the other hand, we can observe that for each player ξ_j, once the action $a_j \neq 0$, the inequation $\sum_{j=1}^{M} x_{a_j,j} - 1 \geq 0$ is guaranteed. Similarly, if for some player ξ_j, $\sum_{i=1}^{k} x_{ij}(\frac{r_i}{\sum_{j'=1}^{j} x_{ij'}} - c \cdot d_{ij}) < 0$ after the first j actions are decided, then the corresponding inequation in (8) is invalid no matter what the rest actions are. This means we can decide an immediate reward for each state-action pair by checking these validations. As a result, we calculate the state-action reward $R_{s_j}^{a_j}$ by the sum of the immediate reward and delayed reward.

6.2 Policy Gradient Method

An RL agent learns to maximize its expected future rewards in MDP. Each time the agent chooses an action a when the current state is s according a policy $\pi(\theta)$, $a \sim \pi(s, \theta)$, where θ is a parameter of the neural network. We can evaluate the policy $\pi(\theta)$ according to their expected reward,

$$J(\theta) = \mathbf{E}_{\pi(\theta)}[\sum_{j=1}^{M} R_{s_j}^{\pi(s_j,\theta)}] \tag{11}$$

Therefore, we can directly optimize the parameter θ to maximize J. According to the policy gradient theorem [17], we can calculate the gradient by:

$$\nabla_\theta J(\theta) = \mathbf{E}_{\pi(\theta)}[\sum_{j=1}^{M} \nabla_\theta \log(\pi(a_j|s_j, \theta)) \sum_{j=1}^{M} R_{s_j}^{\pi(s_j,\theta)}], \tag{12}$$

Then the learning process consists of a group of batches $\mathbf{\Gamma} = \{\Gamma_i\}$. In each batch t, we sample $|\Gamma_t|$ trajectories to estimate the expected reward. The order of players in each trajectory is randomly selected. And we optimize θ for each batch Γ_t by the following update formula,

$$\theta \leftarrow \theta + \alpha \frac{1}{|\Gamma_i|} \sum_{t=1}^{|\Gamma_i|} \sum_{j=1}^{M} \nabla_\theta \log(\pi(a_{j,t}|s_{j,t}, \theta)) \sum_{j=1}^{M} R_{s_{j,t}}^{\pi(s_{j,t},\theta)}, \tag{13}$$

where α is the learning rate.

The whole algorithm is shown in Tab. 5.

TABLE 5: Algorithm 4: Reinforcement Learning Solver

```
Input: Batch number $numBatch$, batch size $bs$, $\alpha$, $\{r_i\}$,$\{d_{ij}\}$, $c$
Output: The policy network $\theta$
1: Initialized the policy network $\theta$
2: For $batch \in \{1, 2, \dots, numBatch\}$
3:     For $t \in \{1, 2, \dots, bs\}$
4:         Shuffle the order of players
5:         Generate a trajectory $(s_{1,t}, a_{1,t}, \dots, s_{M,t}, a_{M,t})$ by $\theta$
           and calculate the corresponding rewards
6:     Update $\theta$ according to Eqn.13
```

Finally, if there exists no Nash equilibrium for a specific input, it corresponds to case where values of locations are not high enough to facilitate peer data exchange. Thus, we should increase the values, or equivalently decrease the constant c when solving the equilibria, and repeat.

7 Performance Evaluation for Data Exchange System

In this section, we evaluate our design in four aspects. We first show that the transmission overhead is roughly a linear function of the network size, or numbers of providers/consumers. Then we show how practical provider/consumer mobility impact the performance. After that, we analyze Nash equilibrium and find the upper/lower bounds of the extra costs incurred when potential relay nodes make individual decisions. Finally, we present an evaluation for the novel RL solver, in the senses of both precision and efficiency when it is used to find Nash equilibrium.

7.1 Transmission Overhead as Functions of Network Size

We present the relationship between transmission overhead and the numbers of providers and consumers in the network. Since there is no closed form solution for finding proper positions for relay nodes, we use approximation algorithms. Fig. 5 shows the total transmission overhead as functions to the numbers of providers/consumers, and network diameter. The results are averaged over 1000 runs, where all consumers, providers and potential relay nodes are independently and uniformly distributed in a $D \times D$ square. We can see that when the transmission overhead

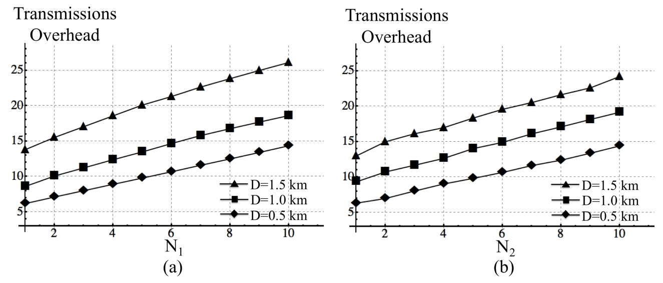


Fig. 5: Total transmission overhead as functions of numbers of providers/consumers (N_1/N_2) with different D's.

is almost proportional to the number of providers or consumers when one of them remain fixed. This is mainly because providers and consumers contribute similarly to transmissions. The only difference is that consumers may also act as relay nodes so they can earn back part of what they pay for the data. Another observation is that as network diameter increases, the transmission overhead increases linearly. This is because a larger space of the same numbers of consumers and providers need more relay nodes, thus more transmissions to delivery data.

7.2 Impact of Practical Mobility

In reality, providers and consumers are not always static. To analyze how devices' mobility influences data exchange, we follow the model in [10] that finds the nodes' speeds and pause times each follow a log-normal distribution.

We assume providers and consumers are walking around slowly in the $D \times D$ square, and following Log-Normal distributions denoted by $ln\mathcal{N}(\mu, \sigma^2)$. In precise, for each provider or consumer, its speed can be factorized into two speeds on both of the two-dimensional coordinates. The absolute values of these speeds follow Log-Normal distribution, and their directions on each dimension are randomly chosen. Potential relay nodes can move at $10km/h$ on average to assigned locations, and providers/consumers will stop once all relay nodes reach their destinations and data delivery starts. Here we define a concept of *second move*. When relay nodes reach their destinations, some of these locations may no longer be suitable for data delivery because providers/consumers have moved. Thus a fraction of relay nodes may need to move a second time to some new locations.

Fig. 6 shows how the average speed of providers/consumers impact second move overhead. It varies from static, normal walking (3km/h) to slow running (5km/h). The probability that second move happens increases almost linearly as providers/consumers move faster (Fig. 6.(a)). This is intuitive because the faster they move, the more likely some original relay locations become obsolete. The fraction of relay nodes that need a second move, however, fluctuate but remains at a low percentage (8 $\sim$ 12% in Fig. 6.(b)). Thus when second move is needed, only one in ten relay node is affected. Also the total second move distance as a ratio of previous move's total distance, is also small (7 $\sim$ 15% in Fig. 6.(b)). These show that the mobility of providers/consumers has small chance (5 $\sim$ 25%) of incurring a very small additional overhead.

7.3 Metric for Nash Equilibrium

In autonomous compensation game, each potential relay node chooses strategy of going to some relay location individually. It is nature that they would like to choose locations with high rewards. Thus more relay nodes go to those relatively "richer" locations. When the Nash equilibrium is achieved, the utility of each relay node is nearly equal, so that no one is willing to change to another location. Locations with high reward have more relay nodes. They work together to relay data, even though one is enough. This can be regarded as a "inefficiency" due to individual choices. We can use a welfare function of cost form to measure the inefficiency, and compare it to the social optimal assignment. Meanwhile, more than one Nash equilibrium may exist. The best or the worst equilibrium as quantified by the welfare correspond to the two metrics of Price of Anarchy (*PoA* in short) and Price of Stability (*PoS*).

7.3.1 Price of Anarchy

Consider a game $G = (N, S, U)$, defined by a set of players N, strategy sets S_i for each player and utilities $U_i : S \to \mathbb{R}$. The cost form welfare function, $C : S \to \mathbb{R}$, is defined as a measure of cost of each outcome that is to be minimized. Let $E \subseteq S$ be the set of strategies in equilibrium. The Price of Anarchy is defined as the ratio between the optimal "centralized" solution and the "worst equilibrium":

$$PoA = \frac{\max_{s \in E} C(s)}{\min_{s \in S} C(s)}$$

In this game, the numbers of relay nodes, relay locations, the range of rewards of all locations, the range of moving costs are possible factors influencing *PoA*. Among them, the range of rewards affects only the existence of Nash equilibrium.

Now we analyze how the other three factors affect *PoA*. The total moving cost is considered as social welfare which should be minimized, and the optimal solution can be found according to the analyze in Section 4. In the simplest model, all relay nodes ignore the moving cost when choosing strategies. Assume there are k locations and M agents ($M \geq M_0 = \sum_{i=1}^{k} \max\{1, \lceil \frac{r_i}{r_k} \rceil - 1\}$ as analyzed before), and denote d_{max} and d_{min} the longest and shortest distance between any agent and any location. A lower bound for optimal social welfare is $k \cdot d_{min}$ and an upper bound for

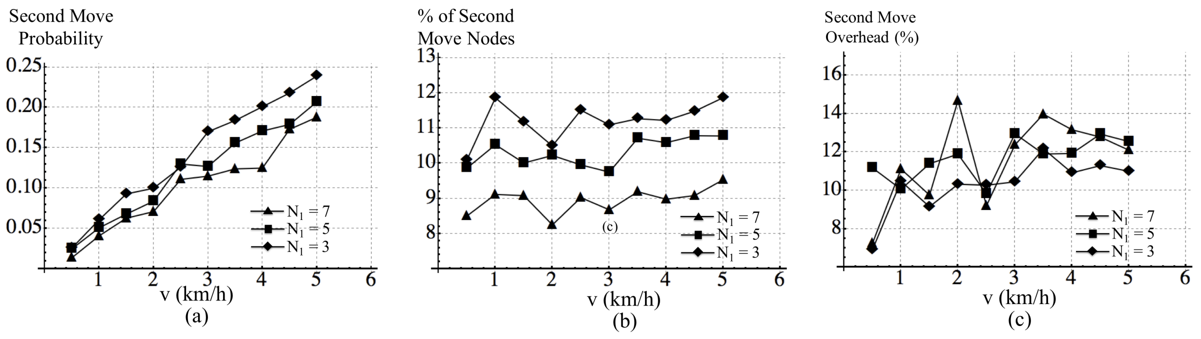


Fig. 6: Performance of our approach as a function of average speed v (km/h) of providers and consumers with impact of N_1 (N_2 fixed). (a) Second move probability vs v. (b) Fraction of second move nodes vs v. (c) Second move overhead vs v.

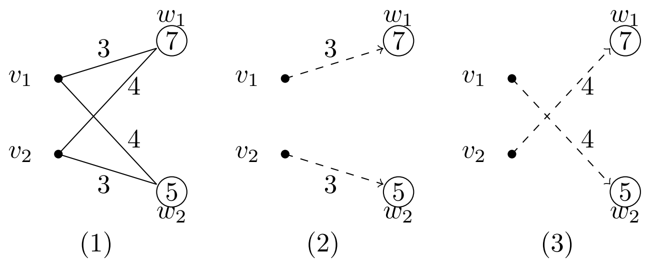


Fig. 7: An example showing a Nash equilibrium is not always social optimal. We use the same representation as Fig. 2. (1) The setting of the example. (2) The optimal assignment for relay nodes. (3) A Nash equilibrium but not optimal.

the worst equilibrium is $M \cdot d_{max}$. So the correspondingly upper bound for Price of Anarchy is

$$\frac{M \cdot d_{max}}{k \cdot d_{min}} = \frac{M\delta}{k} \tag{14}$$

For fixed numbers of locations and distance range, i.e., $\frac{d_{max}}{d_{min}} = \delta$ for some constant δ, the more participants relaying data, the larger *PoA* is. This is intuitive since when more people are doing a fixed amount of work, there will be more unnecessary cost.

Meanwhile, for fixed number of potential relay nodes (sufficiently large such that $M > M_0$) and the distance range, i.e. $\frac{d_{max}}{d_{min}} = \delta$ for some constant δ, the more relay locations, the smaller *PoA* is. Intuitively when more amount of work are given to a fixed number of people, there will be less waste due to competition.

According to Eqn. 14, when the numbers of potential relay nodes and locations are fixed, PoA may increase as the distance range δ increases. This reflects an important feature of Nash equilibrium - players will not deviate the equilibrium point unless others' strategies change, even when this equilibrium is not the social optimal. Note that this also means the solution of Nash equilibrium does not satisfy *envy-free* condition (i.e., no pair of participants want to exchange their locations with each other). Figure 7 illustrates a simple example. Consider two locations w_1 and w_2 with correspondingly rewards $r_1 = 7$ and $r_2 = 5$, and two potential relay nodes v_1 and v_2. Moving cost is also shown in the figure. The assignment that v_1 goes to w_1 while v_2 goes to w_2 is the optimal solution, for both social welfare and Nash equilibrium. However, another assignment that v_1 goes to w_2 while v_2 goes to w_1 is also a Nash equilibrium, since v_1 does not want to change to w_1 if v_2 does not change, and vise versa.

7.3.2 Price of Stability

With the same setting as *PoA*, *PoS* measures the ratio between the "best equilibrium" and the optimal "centralized" solution:

$$PoS = \frac{\min_{s\in E} C(s)}{\min_{s\in S} C(s)}$$

By definition, $1 \leq PoS \leq PoA$. The closer they are to 1, the less inefficiency in the equilibrium. The metric of *PoA* is an upper bound, and the metric of *PoS* is the lower bound for the inefficiency in an equilibrium. The impact factors of *PoS* is the same as those of *PoA*. The optimal solution is exactly the same as before. For the best equilibrium, the upper bound corresponds to the assignment where k nodes go to the same locations as in the optimal solution, and others go to the furthest locations. Then we deduce an upper bound for *PoS*:

$$1 + \frac{(M-k) \cdot d_{max}}{k \cdot d_{min}} = 1 + \frac{(M-k)\delta}{k} = \frac{M\delta}{k} - (\delta - 1) \tag{15}$$

For fixed number of locations and distance range, i.e. $\frac{d_{max}}{d_{min}} = \delta$ for some constant δ, the more participants relaying data, the larger *PoS* is, the same intuition that more unnecessary cost for more people doing a fixed amount of work.

Meanwhile, for fixed number of potential relay nodes (sufficient large such that $M > M_0$) and the distance range, i.e. $\frac{d_{max}}{d_{min}} = \delta$ for some constant δ, *PoS* decreases similarly as *PoA* when the number of relay locations increases, with same intuition that less waste for fixed number of people competing more amount of work.

7.4 Numeric Evaluation of PoA/PoS

We evaluate the impact of different factors in PoA/PoS following similar settings with simulation.

Fig. 8. shows that PoA and PoS are monotonously decreasing as the number of providers (consumers) increases while number of consumers (providers respectively) fixed. The curves are similar to inverse proportional functions, consistent with our earlier theoretical analysis.

Fig. 9 shows PoA and PoS are both positive related to the number of potential relay nodes (M), similar as previous analysis. We also find that the gap between *PoS* and *PoA* becomes narrower when more providers/consumers exist. The gap is caused by multiple Nash equilibria. Thus less equilibrium and narrower gap exist when there are more providers and consumers.

Fig. 10 shows that *PoA* and *PoS* follow earlier analysis at when network diameter (D) is small but becomes unrea-

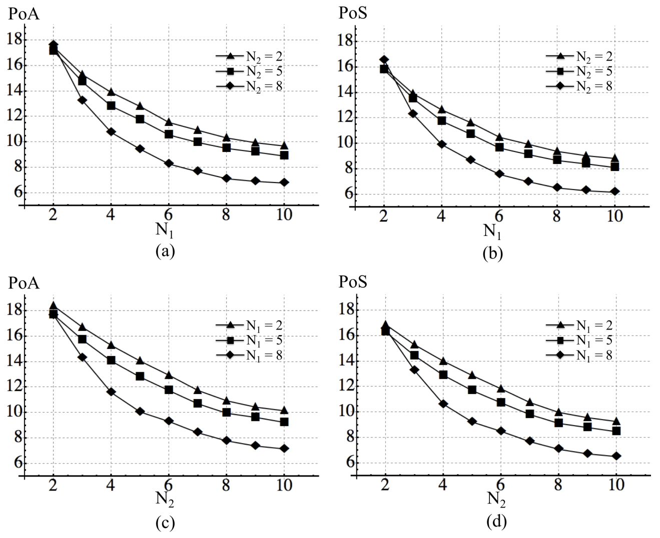


Fig. 8: *PoA*/*PoS* as a function of N_1 (N_2) with impact of N_2 ($N_1 respectively$). $M = 50$, $D = 1km$.

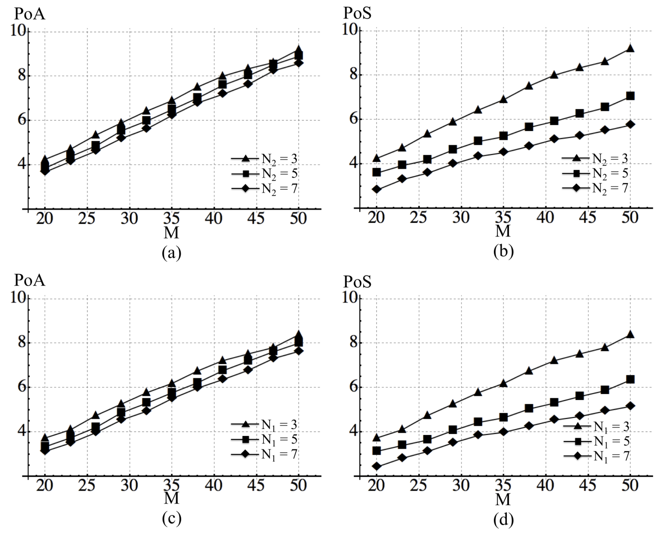


Fig. 9: *PoA*/*PoS* as a function of M with impact of N_2 (N_1 fixed) or N_1 (N_2 fixed respectively). $D = 1km$. (a) *PoA* vs M when $N_1 = 3$. (b) *PoS* vs M when $N_1 = 3$. (c) *PoA* vs M when $N_2 = 3$. (d) *PoS* vs M when $N_2 = 3$.

sonable when D is larger than $1.5km$. This may be caused by the fixed price for each piece of data, regardless to the distribution of providers and consumers. When providers and consumers are in a larger square, more relay nodes are needed, resulting in a decrease of reward to each of them. This shows increasing the price for data for larger network diameter is necessary.

7.5 Evaluation for the RL Solver

The framework of RL solver, based on neural networks, provides approximation solutions to our problem. It begins from searching some feasible solutions randomly, followed by evaluation, and then justifies its searching direction accordingly and repeats. In this learning process, multiple Nash equilibria, if existing, can be found efficiently, but it is possible that there are other equilibria ignored. In comparison, one may consider exhaustively searching all feasible solutions (traversal search), or randomly selecting

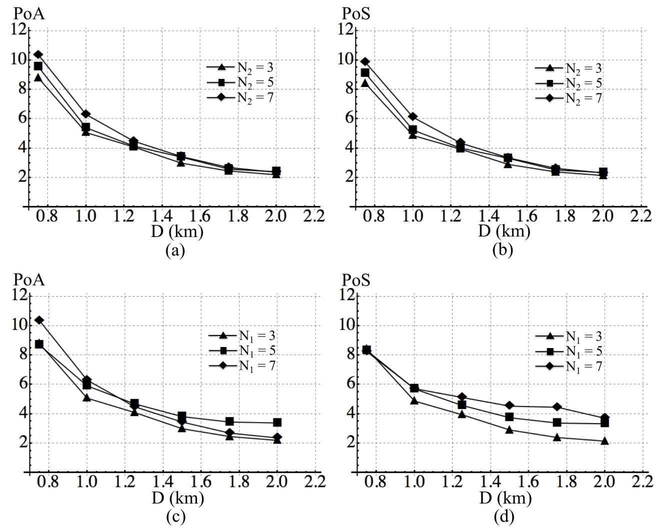
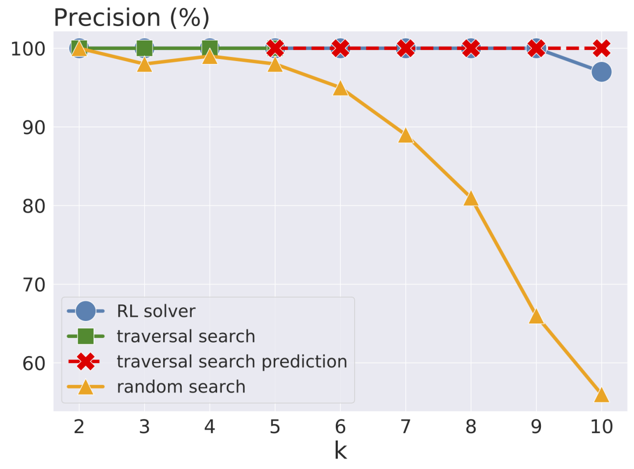


Fig. 10: *PoA*/*PoS* as a function of D with impact of N_2 (N_1 fixed) or N_1 (N_2 fixed respectively). $M = 100$. (a) *PoA* vs D when $N_1 = 3$. (b) *PoS* vs D when $N_1 = 3$. (c) *PoA* vs D when $N_2 = 7$. (d) *PoS* vs D when $N_2 = 7$.

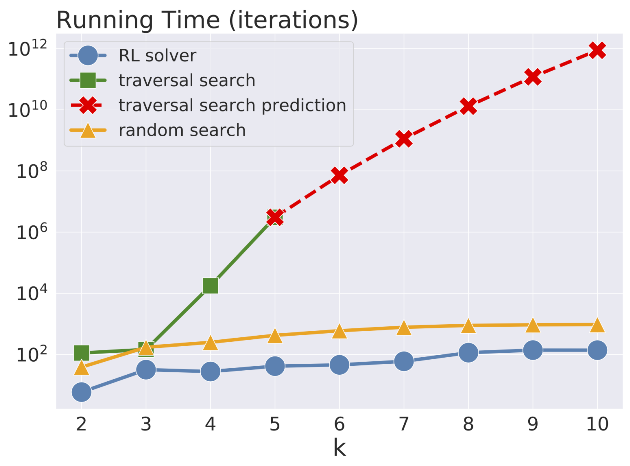
some feasible solutions (random search) to check which ones satisfy Nash equilibrium conditions. Basically, a traversal search takes exponential time complexity, but it can find all Nash equilibria if existing, or validate the non-existence for sure. And random search costs much less time in searching, but there is no guarantee for finally finding a Nash equilibrium. Thus, we compare the RL solver with both methods through numerical experiments and evaluate them in two aspects, precision and efficiency. The precision refers to the proportion of repeated experiments that at least one Nash equilibrium is found. The efficiency can be reflected by the running time, the number of iterations in precise, when the first Nash equilibrium is found.

The results shown in Fig. 11 are precision and average running time over 100 repeated experiments, where necessary relay nodes and potential relay nodes are independently and uniformly distributed in a $D \times D$ square. In precise, for each number of necessary relay locations k, we randomly generate these locations first, then the potential relay nodes one by one, until the necessary condition in Eqn. 10 is satisfied. In each repeated experiment, we set $D = 1km$ and choose r_i for $\forall i$ randomly from $\{1, 2, \ldots, 10\}$. The neural network of our RL solver is three-layers fully connected neural network with the hidden layer being a size of 128. If our RL solver does not find any Nash Equilibrium within 200 iterations (1000 iterations for random search), we say it cannot find the NE, and the the running time is recorded as 200 (1000 for random search).

As we can see, the precision and efficiency of RL solver is stable as the number of necessary relay nodes k increases. The solver costs much less iterations compared to random and traversal search, and also guarantees on almost 100% precision for finding the Nash equilibria. Traversal search always finds all equilibria as it should do, but it has much lower efficiency and cannot finish in reasonable time when the search space becomes larger. For the same reason, we show the predicted precision and efficiency for it when k get larger as dotted lines in figures. On the other hand, the precision of random search descends sharply with k



(a)



(b)

Fig. 11: Evaluation in precision and efficiency among RL solver, random search and traversal search, with respect to the number of necessary relay nodes k in autonomous compensation games. (a) The precision, i.e. the proportion of repeated experiments that at least one Nash equilibrium is found, with the increase of k. (b) The running time, i.e. the number of iterations when the first Nash equilibrium is found, with the increase of k.

increases. This is because the search space expands exponentially where random search becomes less likely to find a solution. In the meantime, RL solver holds a constantly updated policy through the learning process which leads to the ability to adapt to the environment.

8 Conclusion and Future Work

We design a crowd sensing incentive framework for peer data exchange where consumers need data from providers, and relay node facilitate the exchange by going to relay locations to connect them. Relays and providers gain utilities by relaying or generating desired data. We first propose a centralized method to assign relay nodes to locations, then introduce a new autonomous compensation game model for them to make decisions individually. We analyze the condition for the existence of the Nash equilibrium, evaluate how different factors impact the overhead. We also compare the inefficiency in individual decision to that of social optimal assignment, and find that it does not increase too much when participants have more freedom choosing their strategies. This is obtained due to the bounds in the increasing rate of the Price of Anarchy and the Price of Stability.

In consideration to more practical settings, there are some extension for future work, not only to the basic model to facilitate peer data exchange, but also to better understand the autonomous compensation game. In this section, we list two of them.

In the current system, the price for each piece of data is the same. From previous analysis to Fig. 10, it will make the system more robust if it uses dynamic pricing for data according to the network size. At the same time, providers may be willing to post prices depending on the value of their data, which should also be heterogeneous. The corresponding theoretical issue is how heterogeneous prices influence the Nash equilibrium of the compensation game.

As Fig. 3 shows, when players in a compensation game join the game in a series of time, some of them may be willing to change their mind after knowing others' choices. Similarly, it is possible that some consumers regret their original requests and want to pay for only part of data. It is an interesting theoretical problem to calculate the condition under which these behaviors happen. Meanwhile, estimating how many peers doing so is quite important for the real system.

Acknowledgments

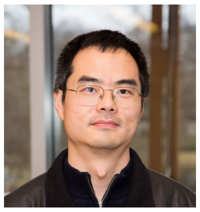
This work is supported in part by US NSF CSR-1513719, and the National Natural Science Foundation of China (Grant No. 61632017, 61761146005).

References

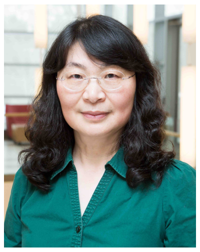
[1] Asad Z, Chaudhry M A R, Malone D. Greener Data Exchange in the Cloud: A Coding-Based Optimization for Big Data Processing[J]. IEEE Journal on Selected Areas in Communications, 2016, 34(5): 1360-1377.
[2] Du D, Wang L, Xu B. The Euclidean bottleneck Steiner tree and Steiner tree with minimum number of Steiner points[M]//Computing and Combinatorics. Springer Berlin Heidelberg, 2001: 509-518.
[3] Duan L, Kubo T, Sugiyama K, et al. Motivating smartphone collaboration in data acquisition and distributed computing[J]. IEEE Transactions on Mobile Computing, 2014, 13(10): 2320-2333.
[4] Feng Z, Zhu Y, Zhang Q, et al. TRAC: Truthful auction for location-aware collaborative sensing in mobile crowdsourcing[C]//INFOCOM, 2014 Proceedings IEEE. IEEE, 2014: 1231-1239.
[5] Gao L, Hou F, Huang J. Providing long-term participation incentive in participatory sensing[C]//Computer Communications (INFOCOM), 2015 IEEE Conference on. IEEE, 2015: 2803-2811.
[6] Helgeson C S, Lipkin D S, Larson R S, et al. Method and apparatus for managing data exchange among systems in a network: U.S. Patent 6,643,652[P]. 2003-11-4.
[7] Hu J, Wellman M P. Nash Q-learning for general-sum stochastic games[J]. Journal of machine learning research, 2003, 4(Nov): 1039-1069.
[8] Jang M, Schwan K, Bhardwaj K, et al. Personal clouds: Sharing and integrating networked resources to enhance end user experiences[C]//IEEE INFOCOM 2014-IEEE Conference on Computer Communications. IEEE, 2014: 2220-2228.
[9] Kavitha V, Altman E. Analysis and design of message ferry routes in sensor networks using polling models[C]//WiOpt'10: Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks. 2010: 62-70.

[10] Kim M, Kotz D, Kim S. Extracting a Mobility Model from Real User Traces[C]//INFOCOM. 2006, 6: 1-13.
[11] Li J, Zhu Y, Hua Y, et al. Crowdsourcing sensing to smartphones: A randomized auction approach[J]. IEEE Transactions on Mobile Computing, 2017.
[12] Luo T, Tan H P, Xia L. Profit-maximizing incentive for participatory sensing[C]//IEEE INFOCOM 2014-IEEE Conference on Computer Communications. IEEE, 2014: 127-135.
[13] Milosavljevic N, Pawar S, El Rouayheb S, et al. Efficient algorithms for the data exchange problem[J]. IEEE Transactions on Information Theory, 2016, 62(4): 1878-1896.
[14] Munkres J. Algorithms for the assignment and transportation problems[J]. Journal of the society for industrial and applied mathematics, 1957, 5(1): 32-38.
[15] Rahman S M M, Masud M M, Hossain M A, et al. Privacy preserving secure data exchange in mobile P2P cloud healthcare environment[J]. Peer-to-Peer Networking and Applications, 2015: 1-16.
[16] Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of Go without human knowledge[J]. Nature, 2017, 550(7676): 354.
[17] Sutton R S, Barto A G. Reinforcement learning: An introduction[M]. MIT press, 2018.
[18] Wang X, Sandholm T. Reinforcement learning to play an optimal Nash equilibrium in team Markov games[C]//Advances in neural information processing systems. 2003: 1603-1610.
[19] Wennerström H, Smith D B. A game theoretic approach to sensor data communications in an opportunistic network[C]//2015 IEEE International Conference on Communications (ICC). IEEE, 2015: 6306-6311.
[20] Xiao M, Wu J, Huang L, et al. Multi-task assignment for crowdsensing in mobile social networks[C]//2015 IEEE Conference on Computer Communications (INFOCOM). IEEE, 2015: 2227-2235.
[21] Yan X, Ye F, Yang Y, et al. An autonomous compensation game to facilitate peer data exchange in crowdsensing[C]//2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS). IEEE, 2017: 1-6.
[22] Yang D, Xue G, Fang X, et al. Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing[C]//Proceedings of the 18th annual international conference on Mobile computing and networking. ACM, 2012: 173-184.
[23] Zhang L, Afanasyev A, Burke J, et al. Named data networking[J]. ACM SIGCOMM Computer Communication Review, 2014, 44(3): 66-73.
[24] Zhao W, Ammar M, Zegura E. Controlling the mobility of multiple data transport ferries in a delay-tolerant network[C]//Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE, 2005, 2: 1407-1418.
[25] Zhu Y, Xu B, Shi X, et al. A survey of social-based routing in delay tolerant networks: positive and negative social effects[J]. IEEE Communications Surveys and Tutorials, 2013, 15(1): 387-401.
[26] Zhang X, Xue G, Yu R, et al. Truthful incentive mechanisms for crowdsourcing[C]//2015 IEEE Conference on Computer Communications (INFOCOM). IEEE, 2015: 2830-2838.

Xiang Yan got his BSc from Shanghai Jiao Tong University. He is currently a PhD student of Department of computer science in Shanghai Jiao Tong University. Yan's current research focuses on algorithmic game theory and machine learning, with their applications to Internet Economics.



Fan Ye is an Associate Professor in the ECE department of Stony Brook University, before that he was a Research Staff Member at IBM T. J. Watson Research after getting his Ph.D. from UCLA CS department in 2004. His research interests include mobile sensing platforms, systems and applications, Internet-of-Things, edge computing, wireless and sensor networks. He has published over 90 papers with 10,000+ citations according to Google Scholar, and 26 granted/pending patents/applications. He has received NSF CAREER award, Google Faculty Research Award, IBM Research Division Award, 5 Invention Achievement Plateau awards, Best Paper Award for IEEE ICCP 2008. He has been a panelist for NSF and Canada, Hong Kong government funding agencies, on program/organizing committees for conferences including ACM Mobicom, ACM Sensys and IEEE Infocom, IEEE ICDCS.



Yuanyuan Yang received the BEng and MS degrees in computer science and engineering from Tsinghua University, Beijing, China, and the MSE and PhD degrees in computer science from Johns Hopkins University , Baltimore, Maryland. Dr. Yang is a SUNY Distinguished Professor in the Department of Electrical & Computer Engineering and Department of Computer Science, and Associate Dean for Diversity and Academic Affairs, College of Engineering and Applied Sciences, at Stony Brook University. Prior to joining Stony Brook in 1999, she had held a tenured faculty position at University of Vermont. Dr. Yang is internationally recognized for her contributions in networking and parallel & distributed computing systems areas. She was elected as an IEEE Fellow in 2009 "for contributions to parallel and distributed computing systems." Her current research interests include cloud computing, data center networks, edge computing, and wireless/mobile networks.

Dongge Wang got her BSc from Shaanxi Normal University. She is currently a MSc student of School of Electronics Engineering and Computer Science in Peking University. Wang's current research focuses on algorithmic game theory and reinforcement learning.

Xiaotie Deng got his BSc from Tsinghua University, MSc from Chinese Academy of Sciences, and PhD from Stanford University. He is currently a Zhiyuan Chair Professor of Shanghai Jiao Tong University. He taught in the past at University of Liverpool, City Universtiy of Hong Kong, and York University. Before that, he was an NSERC international fellow at Simon Fraser University. Deng's current research focuses on algorithmic game theory, with applications to Internet Economics. His works cover online algorithms, parallel algorithms, and combinatorial optimization. He is an ACM fellow for his contribution to the interface of algorithms and game theory.