# A Semidistributed Approach for the Feasible Min-Max Fair Agent-Assignment Problem With Privacy Guarantees

Yuzhe Xu, *Member, IEEE*, Elisabetta Alfonsetti, Pradeep Chathuranga Weeraddana, *Member, IEEE*, and Carlo Fischione, *Member, IEEE*

***Abstract*—In cyberphysical systems, a relevant problem is assigning agents to slots by distributed decisions capable of preserving an agent's privacy. For example, in future intelligent transportation systems, city-level coordinators may optimally assign cars (the agents) to parking slots depending on the cars' distance to final destinations in order to ensure social fairness and without disclosing or even using the car's destination information. Unfortunately, these assignment problems are combinatorial, whereas traditional solvers are exponentially complex, are not scalable, and do not ensure privacy of the agents' intended destinations. Moreover, no emphasis is placed to optimize the agents' social benefit. In this paper, the aggregate social benefit of the agents is considered by an agent-slot assignment optimization problem whose objective function is the fairness among the agents. Due to the problem's complexity, the problem is solved by an approximate approach based on Lagrange duality theory that enables the development of an iterative semidistributed algorithm. It is shown that the proposed algorithm is gracefully scalable compared to centralized methods, and that it preserves privacy in the sense that an eavesdropper will not be able to discover the destination of any agent during the algorithm iterations. Numerical results illustrate the performance and tradeoff of the proposed algorithm compared to the ideal optimal assignment and a greedy method.**

***Index Terms*—Algorithms, intelligent transportation systems, optimization methods, privacy.**

## I. Introduction

THE Problem of assigning agents to slots based on the intended destination of the agents appears in many domains. In logistics, agents could move goods to slots that facilitate the productions. Arguably, one of the most prominent cases of agents assignment is car parking (agents) to slots close to the intended destination of the drivers. In large cities, these problems are pronounced by hundreds or even thousands of drivers who are looking for slots during their daily activities. In [2] it is claimed that seeking for assignment slots (*cruising*) can account for more than 10% of the local circulations in central areas of large cities. In [3], it is reported that cruising for open assignment spaces accounts for 30% of the traffic, causing undesired congestion in big cities. In addition, cruising creates additional delays and drivers can even spend up to 10–20 minutes before they could find a proper slot. According to a recent British study, it is estimated that a person who owns an agent in big cities can take an average of 6 to 20 minutes to search for an empty slot, which accounts for monetary losses (e.g., deterioration, unnecessary fuel wastage), as well as for nonmonetary expenses (e.g., frustration, psychophysical stresses). In [3], it was noted that designing efficient parking car assignment mechanisms is instrumental in directly reducing the cruising traffic, and is just as important as other related methodologies to minimize undesired traffic conditions, especially in big cities.

Several research attempts have been made in the field of ITS, which support drivers to locate a free slot, see [4]–[12]. In general, these existing methods employ a central authority (CA), who is responsible for providing the underlying infrastructure. For example, thousands of sensors have to be deployed to detect the availability of free slots [4]–[9], [12]. The authors in [10] and [11] have considered vehicular ad-hoc networks, where advanced roadside units are assumed to be widely deployed and every vehicle is equipped with sophisticated onboard units. Real-time agent assignment mechanisms have been implemented by smart phones in central areas of large cities, see Steeteline Parker [13], VehicleSense Street assignment Information Network (SPIN) [14], VehicleSense SmartLot [14], and SFMTA SFpark [15]. Like another research work proposed in [4]–[12], these methods rely on a central authority for providing the underlying infrastructure, such as wireless sensors, database-management systems, etc.

However, in almost all existing methods mentioned before, no emphasis has been placed on optimizing an aggregate social benefit of the users during the slot assignment. For example, in [16], an assignment problem has been investigated, with particular emphasis on distributed solution methods. However, the cost function of the problem did not consider fairness. The presence of a fairness cost function demands a solution approach completely different from [16]. Arguably, the existing mechanisms can be interpreted as greedy methods, where each user selects the closest free slot to its destination. Such greedy methods can

Manuscript received May 4, 2016; revised August 9, 2016; accepted August 25, 2016. Date of publication September 13, 2016; date of current version March 16, 2018. This work was supported by EU projects Hycon2 and VR project PRESIDIUM. A preliminary version of this work appeared in [1]. Recommended by Associate Editor Shun-Ichi Azuma.

Y. Xu and C. Fischione are with Electrical Engineering, KTH Royal Institute of Technology, Stockholm, Sweden (e-mail: yuzhe@kth.se; agentlofi@kth.se).

E. Alfonsetti was with TerraSwarm Lab, Electrical Engineering Department, UC Berkeley, California, USA (e-mail: e.alfonsetti@berkeley.edu).

P. C. Weeraddana is with the Sri Lanka Institute of Information Technology, Malabe, Sri lanka (e-mail: chathuranga.we@sliit.lk).

Digital Object Identifier 10.1109/TCNS.2016.2609151

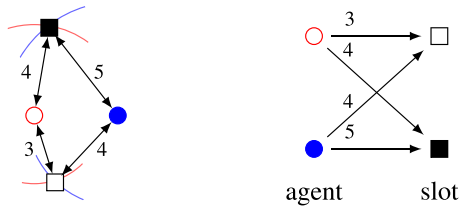


Fig. 1. An illustration of agent-slot assignment: (a) Physical locations of free slots (squares) and agents' destinations (circles). (b) Graph, where the weights denote the costs for assigning the slots to the agents.

easily account for substantial imbalances among the distances between users' slots and their destinations, which are not fair. In other words, some users can be assigned to slots that are very close to their destinations while others can be assigned to slots that are far from their destinations. Fig. 1 shows a case where two agents are to be assigned to two slots. A greedy approach yields the agent-slot assignments $(1, 1)$ and $(2, 2)$, which accounts for a total cost of 8 and a cost imbalance of 2. However, a fair approach yields the assignments $(1, 2)$ and $(2, 1)$, which accounts for a cost imbalance of 0. Although greedy methods are the most intuitive for agents such as drivers, more fair algorithms can be enforced by parking pricing mechanisms that discourage greedy parking.

Therefore, it is worth seeking efficient as well as advanced algorithms that are capable of optimizing some aggregate social benefit for the users of the system. Since the involvement of a central authority is instrumental in coordinating the agent assignment mechanisms, optimization criteria can be integrated into the assignment methods, where some aggregate social metric (utility) is considered during the assignment process. One such simple appealing utility is users' fairness [17]. In addition to fairness, ensuring privacy of the agent assignment is also important, see [11], [18], and [19]. Naturally, users would not like to publish information, such as their destinations to prevent a third party from predicting private traveling patterns. For example, government agencies could probe such information during investigations, and business entities might be interested in exploiting such information to promote their products and services. Therefore, exposure of private information raises serious concerns of personal privacy.

The main contributions of this paper are as follows.

1) We consider the *min-max fairness* as a metric for modeling the aggregate social benefit of the users [17]. In particular, we consider the distance between slot and the destination that corresponds to every user. We refer to this distance, associated with any user, as the *assignment distance*. Then, we design an algorithm to minimize the maximum assignment distance among all users. The proposed algorithm is based on duality theory [20, Sec 5]. Our formulation and the corresponding algorithm can be applied directly or with minor modifications in fair agent-target assignment problems in other application domains as well and, therefore, is not restricted to the car parking problem.
2) We capitalize on dual decomposition techniques and the subgradient methods [20]–[22] to accomplish distributed implementation (among users) of the proposed algorithm with a little coordination of the central authority. Therefore, the proposed algorithm holds scalable properties, which is indeed favorable in practice.
3) The proposed agent assignment mechanism is privacy preserving in the sense that any agent involved in the algorithm will not be able to find out the destination of any other agent during the algorithm's iterations. This privacy is accomplished as a result of the inherent decomposition structure of the problem together with randomization of the step size of the subgradient method.
4) Numerical examples are provided to evaluate the performance of the algorithm. In addition, the proposed algorithm is compared with the optimal assignment method and with a greedy assignment method.

Thus, our solution approach for the agent assignment problem is fair, distributed, and is easy to deploy with the coordination of a central entity. In addition, it has appealing privacy properties.

The rest of this paper is organized as follows. A description of the system model and the assignment problem formulation is presented in Section II. In Section III, we provide the solution method to the agent assignment problem by using duality theory and subgradient method. Section IV presents our proposed algorithm for the distributed agent assignment problem. The convergence and optimality properties of the proposed algorithm are given in Sections V. In Section VI, we describe privacy properties of the algorithm. In Section VII, numerical results are provided. Finally, Section VIII concludes this paper.

### I. Notations

Boldface lowercase and uppercase letters represent vectors and matrices, respectively, and calligraphy letters represent sets. The set of real $n$-vectors is denoted by $\mathbb{R}^n$ and the set of real $m \times n$ matrices is denoted as $\mathbb{R}^{m\times n}$. We use parentheses to construct matrices from comma-separated submatrices of agreed dimensions, e.g., $(\mathbf{A}, \mathbf{B}, \mathbf{C}) = [\mathbf{A}^{\mathrm{T}}\ \mathbf{B}^{\mathrm{T}}\ \mathbf{C}^{\mathrm{T}}]^{\mathrm{T}}$. We denote by $(\mathbf{A}_i)_{i=1,2,\ldots,N}$ the matrix $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N)$. The cardinality of a set $\mathcal{A}$ is denoted by $\mathrm{agentd}\,\mathcal{A}$.

## II. System Model and Problem Formulation

We consider a system consisting of $M$ slots and a number of destinations. We denote by $\mathcal{M} = \{1, \ldots, M\}$ the set of slots. Destinations can include *any* geographical locations, such as shops, bars, banks, cinemas, houses, parks, and hotels, among others. The slots and the destinations can be geographically dispersed and need not necessarily be concentrated. Few free slots could be relatively close to some agents' destinations, and these could be preferred by many agents. Once these slots are occupied, some agents have to continue cruising for other free slots leading to unexpected time and fuel costs. Therefore, it is not trivial for agents to find free parking slots. Knowledge of geographical location of each slot is assumed to be available to anyone in the system. A *trustworthy central controller* (CC) is responsible for coordinating the slot assignment mechanism,

namely, it is the central authority. The coordinations are carried out through secure channels.

The slot assignment mechanism is assumed to operate over time frames, with the discrete time being given by integer values $t = 1, 2, 3, \ldots$. At the beginning of every time frame $t$, the set $\mathcal{M}_t \subseteq \mathcal{M}$ of *free* slots is known, and denote by $M_t$ the number of free slots at time frame $t$.[1] In addition, at the beginning of every time frame $t$, a set $\mathcal{N}_t = \{1, \ldots, N_t\}$ of agents is scheduled for slot assignment. Assume $N_t \leq M_t$, and the agent comes last is excluded if the number of agents is larger than that of the slots.[2] We denote by $\mathrm{des}(i)$ the destination locations of agent $i \in \mathcal{N}_t$. Denote by $u_i(d_{\mathrm{ij}})$ the cost for agent $i$ to park at free slot $j \in \mathcal{M}_t$, where $d_{\mathrm{ij}}$ is the distance from $\mathrm{des}(i)$ to free slot $j$, and $u_i : \mathbb{R} \to \mathbb{R}$ is monotonically increasing and strictly concave in $d_{\mathrm{ij}}$. We assume that each agent $i$ can compute $\{u_i(d_{\mathrm{ij}})\}_{j \in \mathcal{M}_t}$ simply by knowing the geographical location of $\mathrm{des}(i)$. Such computations can be easily performed by using the state-of-the-art global positioning system (GPS).

To formally express the problem, first consider *binary decision variables* $\mathbf{x} = (x_{\mathrm{ij}})_{i \in \mathcal{N}_t,\ j \in \mathcal{M}_t}$, and let $x_{\mathrm{ij}} = 1$, if agent $i$ is assigned to slot $j$, $x_{\mathrm{ij}} = 0$, otherwise. A *feasible assignment* is such that an agent is assigned to a free slot and no more than an agent is assigned to a free slot. Now, we can formally express the cost from agent $i$'s assigned slot to its destination $\mathrm{des}(i)$ as $\sum_{j \in \mathcal{M}_t} u_i(d_{\mathrm{ij}}) x_{\mathrm{ij}}$, that is, the *assignment cost* of agent $i$.

Min-max fairness is appealing in many application domains in the sense that it ensures equalization of the costs incurred by the users, see [17]. To ensure min-max fairness among the agents, the objective is to *minimize the maximum assignment cost*. The problem is formally expressed as

$$\underset{\mathbf{x}}{\text{minimize}} \quad f(\mathbf{x}) := \max_{i \in \mathcal{N}_t} \sum\nolimits_{j \in \mathcal{M}_t} u_i(d_{\mathrm{ij}})\, x_{\mathrm{ij}} \tag{1a}$$

$$\text{subject to} \quad \sum_{i \in \mathcal{N}_t} x_{\mathrm{ij}} \leq 1, \quad j \in \mathcal{M}_t \tag{1b}$$

$$\sum_{j \in \mathcal{M}_t} x_{\mathrm{ij}} = 1, \quad i \in \mathcal{N}_t \tag{1c}$$

$$x_{\mathrm{ij}} \in \{0, 1\}, \quad i \in \mathcal{N}_t,\ j \in \mathcal{M}_t\ , \tag{1d}$$

where constraint (1b) ensures that no more than an agent is assigned to a free slot. Constraint (1c) imposes that each agent is assigned to only one free slot. Finally, constraint (1d) ensures that the values of $x_{\mathrm{ij}}$ are either 0 or 1. We remark here that optimization problem (1) is a special case of linear bottleneck assignment problem; thus, the optimal assignment depends only on the relative order of $u_i(d_{\mathrm{ij}})$ and not on their numerical value [24]. Therefore, without loss of generality, in the following text, we use $d_{\mathrm{ij}}$ instead of $u_i(d_{\mathrm{ij}})$ in (1a) to simplify the problem formulation. An illustration is shown in Fig. 1.

Optimization problem (1) is combinatorial. We have to rely on global optimal methods [21], such as exhaustive search, or branch-and-bound methods to *solve* it. The main disadvantage of global methods is the prohibitive computational complexity, even in the case of small problems. Such methods are not scalable with the numbers of agents and slots and, therefore, can be impractical. Compared to the assignment problem in [16], the cost function is different, which prevents applying the method therein proposed. In the sequel, we provide a solution method based on duality theory. We show that the method provides a solution of bounded suboptimality, and it is efficient, fast, and allows distributed implementation with a little coordination from the CC.

## III. Preliminaries

In this section, we first equivalently formulate problem (1) in its epigraph form [20]. Then, we apply the duality theory to obtain the related dual problem, and show that the problem is split into subproblems and a master problem which can be solved efficiently. The equivalent problem is given by[3]

$$\underset{\mathbf{x},\, s}{\text{minimize}} \quad s \tag{2a}$$

$$\text{subject to} \quad \sum_{j \in \mathcal{M}} d_{\mathrm{ij}}\, x_{\mathrm{ij}} \leq s,\ i \in \mathcal{N} \tag{2b}$$

$$\sum_{i \in \mathcal{N}} x_{\mathrm{ij}} \leq 1,\ j \in \mathcal{M} \tag{2c}$$

$$\sum_{j \in \mathcal{M}} x_{\mathrm{ij}} = 1,\ i \in \mathcal{N} \tag{2d}$$

$$x_{\mathrm{ij}} \in \{0, 1\},\ i \in \mathcal{N},\ j \in \mathcal{M}\ , \tag{2e}$$

Note that (2) is still combinatorial. Now we seek to decouple the problem among the agents for the scalability of the agent assignment mechanism. In this context, we can clearly see that constraints (2d), (2e) are already decoupled, yet constraints (2b), (2c) are coupled among the agents, which is an obstacle to distributed solution methods.

*Lemma 1:* Consider optimization problem (2). Let $\boldsymbol{\lambda} = (\lambda_i)_{i \in \mathcal{N}}$ be the Lagrangian multiplier for constraint (2b), and let $\boldsymbol{\mu} = (\mu_j)_{j \in \mathcal{M}}$ be the Lagrangian multiplier for constraint (2c). Then, the dual function is

$$g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \begin{cases} \sum_{i \in \mathcal{N}} g_i(\boldsymbol{\lambda}, \boldsymbol{\mu}) - \sum_{j \in \mathcal{M}} \mu_j & \sum_{i \in \mathcal{N}} \lambda_i = 1 \\ -\infty & \text{otherwise} \end{cases}, \tag{3}$$

where $g_i(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \sum\limits_{j \in \mathcal{M}} (\lambda_i d_{\mathrm{ij}} + \mu_j) x^{\star}_{\mathrm{ij}}$, with

$$x^{\star}_{\mathrm{ij}} = \begin{cases} 1 & \text{if } j = \arg\min_{l \in \mathcal{M}} (\lambda_i d_{il} + \mu_l) \\ 0 & \text{otherwise} \end{cases}. \tag{4}$$

*Proof:* The Lagrangian associated with problem (2) is

$$L(s, \mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = s \left(1 - \sum_{i \in \mathcal{N}} \lambda_i\right) + \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{M}} (\lambda_i d_{\mathrm{ij}} + \mu_j) x_{\mathrm{ij}} - \sum_{j \in \mathcal{M}} \mu_j\ . \tag{5}$$

[1] Such information is retrieved by installing sensors at every slot.

[2] The first-in first-out mechanism is applied to ensure $N_t \leq M_t$, which is one of the most popular methods for queuing management [23].

[3] Without loss of generality, we drop the subindex $t$ for notational simplicity.

The dual function is therefore given by

$$g(\boldsymbol{\lambda},\boldsymbol{\mu}) = \inf_{\substack{s\in\mathbb{R},\\ \sum_{j\in\mathcal{M}} x_{\mathrm{ij}}=1,\ i\in\mathcal{N},\\ x_{\mathrm{ij}}\in\{0,1\},\ i\in\mathcal{N},\ j\in\mathcal{M}}} L(s,\mathbf{x},\boldsymbol{\lambda},\boldsymbol{\mu}) \tag{6a}$$

$$= \begin{cases} \sum_{i\in\mathcal{N}} \left( \inf_{\substack{\sum_{j\in\mathcal{M}} x_{\mathrm{ij}}=1,\\ x_{\mathrm{ij}}\in\{0,1\},\ j\in\mathcal{M}}} \sum_{j\in\mathcal{M}} (\lambda_i d_{\mathrm{ij}} + \mu_j) x_{\mathrm{ij}} \right) - \sum_{j\in\mathcal{M}} \mu_j \\ \qquad\qquad\qquad\qquad \text{if } \sum_{i\in\mathcal{N}} \lambda_i = 1 \\ -\infty \qquad \text{otherwise,} \end{cases} \tag{6b}$$

where equality (6b) follows from constraints (2d), (2e) being separable, and $g_i(\boldsymbol{\lambda},\boldsymbol{\mu})$ is the optimal value of the problem

$$\begin{aligned} \underset{\mathbf{x}_i}{\text{minimize}} \quad & \sum_{j\in\mathcal{M}} (\lambda_i d_{\mathrm{ij}} + \mu_j) x_{\mathrm{ij}} \\ \text{subject to} \quad & \sum_{j\in\mathcal{M}} x_{\mathrm{ij}} = 1 \\ & x_{\mathrm{ij}} \in \{0,1\},\ j\in\mathcal{M}, \end{aligned} \tag{7}$$

with the binary decision variable $\mathbf{x}_i(x_{\mathrm{ij}})_{j\in\mathcal{M}}$. Note that problem (7) is a combinatorial problem. Nevertheless, simple algebra shows that it has a closed-form solution given by (4), which concludes the proof. ■

From the previous lemma, we obtain the dual master problem of (2) as

$$\underset{\boldsymbol{\lambda},\boldsymbol{\mu}}{\text{maximize}} \quad g(\boldsymbol{\lambda},\boldsymbol{\mu}) = \sum_{i\in\mathcal{N}} g_i(\boldsymbol{\lambda},\boldsymbol{\mu}) - \sum_{j\in\mathcal{M}} \mu_j \tag{8a}$$

$$\text{subject to} \quad \sum_{i\in\mathcal{N}} \lambda_i = 1 \tag{8b}$$

$$\lambda_i \geq 0,\ i\in\mathcal{N} \tag{8c}$$

$$\mu_j \geq 0,\ j\in\mathcal{M}. \tag{8d}$$

Then, we have the following simple result:

*Proposition 1:* The solution to (8) is given by the limit of the following iterations:

$$\boldsymbol{\lambda}^{(k+1)} = P_s(\boldsymbol{\lambda}^{(k)} - \alpha_k \mathbf{u}^{(k)}) \tag{9}$$

$$\boldsymbol{\mu}^{(k+1)} = [\boldsymbol{\mu}^{(k)} - \alpha_k \mathbf{v}^{(k)}]^+, \tag{10}$$

where $P_s(\cdot)$ is the Euclidean projection onto the probability simplex [22],

$$\Pi = \left\{ \lambda \,\middle|\, \textstyle\sum_{i=1}^N \lambda_i = 1, \lambda_i \geq 0 \right\} \tag{11}$$

the term $[\,\cdot\,]^+$ is the Euclidean projection onto the non-negative orthant $\alpha_k = \alpha/k$, where $\alpha$ is a positive scalar, and

$$\mathbf{u}^{(k)} = \left(u_i^{(k)}\right)_{i\in\mathcal{N}} = \left( -\sum_{j\in\mathcal{M}} d_{\mathrm{ij}} x_{\mathrm{ij}}^\star \right)_{i\in\mathcal{N}} \tag{12}$$

$$\mathbf{v}^{(k)} = \left(v_j^{(k)}\right)_{j\in\mathcal{M}} = \left( 1 - \sum_{i\in\mathcal{N}} x_{\mathrm{ij}}^\star \right)_{j\in\mathcal{M}}, \tag{13}$$

where $x_{\mathrm{ij}}^\star$ is given in (4).

*Proof:* We solve problem (8) based on the projected subgradient method [25]. Note that $g(\boldsymbol{\lambda},\boldsymbol{\mu})$ is a concave function; therefore, we need to find the subgradient $\mathbf{s}\in\mathbb{R}^{N+M}$ of $-g$ at a feasible $(\boldsymbol{\lambda},\boldsymbol{\mu})$. For clarity, we separate $\mathbf{s}$ into two vectors as $\mathbf{s}=(\mathbf{u},\mathbf{v})$. The negative of dual function is given by

$$-g(\boldsymbol{\lambda},\boldsymbol{\mu}) = \sum_{j\in\mathcal{M}} \mu_j - \sum_{j\in\mathcal{M}} \mu_j \sum_{i\in\mathcal{N}} x_{\mathrm{ij}}^\star - \sum_{i\in\mathcal{N}} \lambda_i \sum_{j\in\mathcal{M}} d_{\mathrm{ij}} x_{\mathrm{ij}}^\star,$$

and particular choices for $u_i$, $i\in\mathcal{N}$ and $v_j$, $j\in\mathcal{M}$ are given by (12) and (13). Thus, the projected subgradient method is

$$(\boldsymbol{\lambda}^{(k+1)},\boldsymbol{\mu}^{(k+1)}) = P((\boldsymbol{\lambda}^{(k)},\boldsymbol{\mu}^{(k)}) - \alpha_k(\mathbf{u}^{(k)},\mathbf{v}^{(k)})), \tag{14}$$

where $k$ is the current iteration index of the subgradient method, $P(\mathbf{z})$ is the Euclidean projection of $\mathbf{z}\in\mathbb{R}^{N+M}$ onto the *feasible set* of the dual problem (8), and $\alpha_k > 0$ is the $k$th step size, chosen to guarantee the asymptotic convergence of the subgradient method, e.g., $\alpha_k = \alpha/k$. Since the feasible set of dual problem is separable in $\lambda$ and $\mu$, the projection $P(\cdot)$ can be performed independently. Therefore, the iteration (14) is equivalently performed as (12) and (13). Note that the Euclidean projection onto the probability simplex is posed as a convex optimization problem that can be solved efficiently, which concludes the proof. ■

Denote by $\boldsymbol{\lambda}^\star$ and $\boldsymbol{\mu}^\star$ the optimal solutions for the dual master problem (8). Note that $x_{\mathrm{ij}}^\star(\boldsymbol{\lambda}^\star,\boldsymbol{\mu}^\star)$ given in (4) is not necessary to satisfy constraint (2c). Therefore $x_{\mathrm{ij}}^\star(\boldsymbol{\lambda}^\star,\boldsymbol{\mu}^\star)$ may be infeasible for primary problem (1). In the next section, we develop a distributed solution algorithm together with a simple subroutine to optimization problem (1), which could produce a near-to-optimal and feasible assignment.

## IV. Algorithm Development

In this section, we first present our distributed algorithm to address problem (1) via the dual problem (8). The resulting algorithm is a distributed agent assignment mechanism that can be coordinated by the CC or the central controller. We first give the algorithm and then we present the convergence and optimality properties in the following section.

### A. Distributed Algorithm Development

The algorithm capitalizes on the ability of the CC to construct the subgradient $(\mathbf{u},\mathbf{v})$ in a distributed fashion via the coordination of scheduled agents. The involvement of a CC (e.g., an authority who handles the slots) is essential for realizing the overall algorithm in practice. This involvement is mainly for coordinating parameters among the agents, and for constricting a feasible assignment in case the assignment from the dual problem is infeasible. We let the initial dual variable $\lambda_i^{(1)} = 1/N$, whereas let $\boldsymbol{\mu}^{(1)} = (\mu_j^{(1)})_{j\in\mathcal{M}} = 0$ for each agent $i\in\mathcal{N}$. The algorithm is given below.

### B. DAA Algorithm Description

In step 1, the algorithm starts by choosing initial feasible values for $\lambda_i^{(k)}$, $i\in\mathcal{N}$ and $\mu_j^{(k)}$, $j\in\mathcal{M}$. Step 2 corresponds to the local computations of $\mathbf{x}_i^{(k)}$ at each agent $i$. These computations

**Algorithm:** Distributed agent-assignment (DAA)

1) Given the distances $(d_{\mathrm{ij}})_{j\in\mathcal{M}}$ for each agent $i\in\mathcal{N}$. The CC sets $k=1$, sets current objective value $p^{\mathrm{cur}}(0)=\infty$, sets number of conflicting users $N^{\mathrm{conflict}}=N$, and broadcasts the initial(feasible) $\lambda_i^{(1)}=1/N$ and $\boldsymbol{\mu}^{(1)}=(\mu_j^{(1)})_{j\in\mathcal{M}}=0$ to each agent $i\in\mathcal{N}$.
2) Every agent $i$ sets $\lambda_i=\lambda_i^{(k)}$ and $\boldsymbol{\mu}=\boldsymbol{\mu}^{(k)}$ and locally computes $\mathbf{x}_i^{(k)}=(x_{\mathrm{ij}}^{\star})_{j\in\mathcal{M}}$ from (4). Let $j_i^k$ denote the index of the nonzero component of $\mathbf{x}_i^{(k)}$.
3) Local subgradients: Each agent $i$
   a. sets $u_i^{(k)}=-\sum_{j\in\mathcal{M}}d_{\mathrm{ij}}x_{\mathrm{ij}}^{(k)}=-d_{\mathrm{ij}_i^k}$, [(12) and (13)].
   b. transmits $(u_i^{(k)},j_i^k)$ to CC.
4) Current assignment and Subgradient iteration at CC
   a. find set $\mathcal{J}_j^{(k)}$ of users assigned to slot $j$, i.e., $\mathcal{J}_j^{(k)}=\{i|j_i^k=j\}$. Set $N_k^{\mathrm{conflict}}=\sum_{j|\mathrm{agentd}(\mathcal{J}_j^{(k)})\geq 2}\mathrm{agentd}(\mathcal{J}_j^{(k)})$.
   b. if no conflicting assignments (i.e., $N_k^{\mathrm{conflict}}=0$), set $N^{\mathrm{conflict}}=N_k^{\mathrm{conflict}}$ and go to step 4-c. Otherwise, go to step 4-d.
   c. if $p^{\mathrm{cur}}(k-1)>\max_{i\in\mathcal{N}}d_{\mathrm{ij}_i^k}$, set $p^{\mathrm{cur}}(k)=\max_{i\in\mathcal{N}}d_{\mathrm{ij}_i^k}$ and set $\mathbf{X}^{\mathrm{cur}}(k)=\left(\mathbf{e}_{j_i^k}^{\mathrm{T}}\right)_{i\in\mathcal{N}}\in\mathbb{R}^{N\times M}$. Go to step 4-e.
   d. if $N_k^{\mathrm{conflict}}<N^{\mathrm{conflict}}$, set $N^{\mathrm{conflict}}=N_k^{\mathrm{conflict}}$, $p^{\mathrm{cur}}(k)=\infty$, $\mathbf{X}^{\mathrm{cur}}(k)=\left(\mathbf{e}_{j_i^k}^{\mathrm{T}}\right)_{i\in\mathcal{N}}\in\mathbb{R}^{N\times M}$, and $\mathcal{J}_j^{\mathrm{cur}}=\mathcal{J}_j^{(k)}$, $j\in\mathcal{M}$. Go to step 4-e.
   e. form $\mathbf{u}^{(k)}=(u_i^{(k)})_{i\in\mathcal{N}}$ and perform (9) to find $\boldsymbol{\lambda}^{(k+1)}$.
   f. set $v_j=1-\mathrm{agentd}(\mathcal{J}_j)$, $\mathbf{v}^{(k)}=(v_j)_{j\in\mathcal{M}}$, [(12) and (13)].
   g. perform (10) to find $\boldsymbol{\mu}^{(k+1)}$.
5) Stopping criterion: If $k>K$, go to step 6.[4] Otherwise,
   b. CC broadcasts the new $\lambda_i^{(k+1)}$ and $\boldsymbol{\mu}^{(k+1)}$ to each agent $i\in\mathcal{N}$.
   c. increment $k$, i.e., set $k=k+1$, and go to step 2.
6) Output: If $N^{\mathrm{conflict}}=0$ (i.e., a feasible assignment is achieved), CC returns $\mathbf{X}^{\mathrm{final}}=\mathbf{X}^{\mathrm{cur}}(k)$ and terminates the algorithm. Otherwise, CC sets $\mathbf{X}^{\mathrm{infeasible}}=\mathbf{X}^{\mathrm{cur}}(k)$, performs a simple *subroutine* to construct a feasible assignment $\mathbf{X}^{\mathrm{final}}$ from $\mathbf{X}^{\mathrm{infeasible}}$, returns $\mathbf{X}^{\mathrm{final}}$, and terminates the algorithm.

involve simple comparisons [see (4)] and can be performed in parallel by the agents. Step 3 involves coordination of scheduled agents and CC. First, each agent $i$ constructs *scalar* parameter $u_i^{(k)}$. Then, it transmits $u_i^{(k)}$ together with the potential agent slot index $j_i^k$ to CC.

In step 4, CC keeps records of the *best* assignment so far. The assignment is best, in the following sense. First, suppose the algorithm yields at least one feasible assignment, i.e., $N^{\mathrm{conflict}}=0$. Then, the best assignment is the one that corresponds to the smallest objective value among all feasible assignments, see step 4-c. On the other hand, suppose algorithm does not yield any feasible assignment, i.e., $N^{\mathrm{conflict}}>0$. Then, the best assignment is the one that corresponds to the smallest $N_k^{\mathrm{conflict}}$ among all infeasible assignments, see step 4-d. Note that $N^{\mathrm{conflict}}$ is equal to the total conflicting agents and, thus, quantifies the degrees of infeasibility, see step 4-a. Moreover, by using the information received from the agents, CC constructs the global subgradient components $\mathbf{u}^{(k)}\in\mathbb{R}^N$ and $\mathbf{v}^{(k)}\in\mathbb{R}^M$ which, in turn, are used to perform the subgradient iterations (9)–(10), see steps 4-e, 4-f, 4-g.

The new parameters $\boldsymbol{\lambda}^{(k+1)}$ and $\boldsymbol{\mu}^{(k+1)}$ are broadcasted to every agent, and the algorithm is iterated until a stopping criterion is satisfied, see step 5. In this algorithm, we use a natural and simple-stated stopping criterion, where a fixed number of iterations are executed for the subgradient method.

Recall that the solution for primal problem (2) by considering its dual problem (8) does not always guarantee the primal feasibility, because the original problem (2) is *nonconvex*. Therefore, if a feasible assignment is not achieved, a subroutine call is required to construct a feasible one after the stopping criterion is satisfied. Step 6 is essential due to addressing this infeasibility issue. In particular, once the stopping criterion is satisfied(step 5), CC checks whether the current assignment $\mathbf{X}^{\mathrm{cur}}(k)$ obtained is feasible. If it is feasible, the algorithm terminates by returning $\mathbf{X}^{\mathrm{final}}=\mathbf{X}^{\mathrm{cur}}(k)$, where CC informs each agent $i$, its slot. Otherwise, CC performs a simple subroutine to construct a feasible assignment $\mathbf{X}^{\mathrm{final}}$ by using the current infeasible assignment $\mathbf{X}^{\mathrm{infeasible}}$, before the algorithm terminates. We outline a subroutine that can be implemented at CC for constructing a feasible assignment in the Appendix.

## V. DAA Convergence and Optimality Properties

In this section, we present the convergence and optimality properties of the proposed DAA algorithm for agent assignment. In particular, in Section V-A, we show that for a sufficiently large number of subgradient iterations, the DAA algorithm converges to the dual optimal value of problem (8). In Section V-B, we analyze the optimality of the DAA algorithm with respect to primary problem (1).

### A. Convergence Properties

The convergence is established by the following proposition:

*Proposition 2:* Denote by $(\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*)$ the optimal solution of dual problem (8), and let $g_{\mathrm{best}}^{(k)}=\max\{g(\boldsymbol{\lambda}^{(1)},\boldsymbol{\mu}^{(1)}),\ldots,g(\boldsymbol{\lambda}^{(k)},\boldsymbol{\mu}^{(k)})\}$ denote the dual objective value found after $k$ subgradient iterations. Suppose $\|(\boldsymbol{\lambda}^{(1)},\boldsymbol{\mu}^{(1)})-(\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*)\|$ is bounded from above. Then $\forall\varepsilon>0,\ \exists n\geq 1$ such that $\forall k\ \ k\geq n\Rightarrow(d^\star-g_{\mathrm{best}}^{(k)})<\varepsilon$, where $d^*$ is the optimal value of the dual problem (8).

*Proof:* The proof is based on the approach of [22]. We have

$$\begin{aligned}&\left\|(\boldsymbol{\lambda}^{(k+1)},\boldsymbol{\mu}^{(k+1)})-(\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*)\right\|_2^2\\&=\left\|P\big((\boldsymbol{\lambda}^{(k)}-\alpha_k\mathbf{u}^{(k)})-\boldsymbol{\lambda}^*,(\boldsymbol{\mu}^{(k)}-\alpha_k\mathbf{v}^{(k)})-\boldsymbol{\mu}^*\big)\right\|_2^2\end{aligned}\tag{15}$$

[4] $K$ is the number of the subgradient iterations executed in DAA.

$$\leq \left\|((\boldsymbol{\lambda}^{(k)} - \alpha_k \mathbf{u}^{(k)}) - \boldsymbol{\lambda}^*, (\boldsymbol{\mu}^{(k)} - \alpha_k \mathbf{v}^{(k)}) - \boldsymbol{\mu}^*)\right\|_2^2 \quad (16)$$

$$= \|(\boldsymbol{\lambda}^{(k)}, \boldsymbol{\mu}^{(k)}) - (\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\|_2^2 - 2\alpha_k \mathbf{u}^{(k)T}(\boldsymbol{\lambda}^{(k)} - \boldsymbol{\lambda}^*) - 2\alpha_k \mathbf{v}^{(k)T}(\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu}^*) + \alpha_k^2 \|\mathbf{u}^{(k)}\|_2^2 + \alpha_k^2 \|\mathbf{v}^{(k)}\|_2^2 \quad (17)$$

$$\leq \|(\boldsymbol{\lambda}^{(k)}, \boldsymbol{\mu}^{(k)}) - (\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\|_2^2 - 2\alpha_k \big(g(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) - g(\boldsymbol{\lambda}^{(k)}, \boldsymbol{\mu}^{(k)})\big) + \alpha_k^2 \|\mathbf{u}^{(k)}\|_2^2 + \alpha_k^2 \|\mathbf{v}^{(k)}\|_2^2 , \quad (18)$$

where (15) follows from (14), and (16) follows from that the Euclidean projection $P(\mathbf{z})$ of any $\mathbf{z} \in \mathbb{R}^{N+M}$ onto the *feasible set* of the dual problem (8) always decreases the distance of $P(\mathbf{z})$ to every point in the *feasible set* and, in particular, to the optimal point $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$, and (18) follows from the definition of the subgradient. Recursively applying (18) and rearranging the terms, we obtain

$$2\sum_{l=1:k} \alpha_l (d^* - g(\boldsymbol{\lambda}^{(l)}, \boldsymbol{\mu}^{(l)}) \leq - \left\|(\boldsymbol{\lambda}^{(k+1)}, \boldsymbol{\mu}^{(k+1)}) - (\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\right\|_2^2 + \|(\boldsymbol{\lambda}^{(1)}, \boldsymbol{\mu}^{(1)}) - (\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\|_2^2 + \sum_{l=1:k} \alpha_l^2 \|\mathbf{u}^{(l)}\|_2^2 + \sum_{l=1:k} \alpha_l^2 \|\mathbf{v}^{(l)}\|_2^2$$

$$\leq \|(\boldsymbol{\lambda}^{(1)}, \boldsymbol{\mu}^{(1)}) - (\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\|_2^2 + \sum_{l=1:k} \alpha_l^2 \big(\|\mathbf{u}^{(l)}\|_2^2 + \|\mathbf{v}^{(l)}\|_2^2\big)$$

$$\leq R^2 + (G_1^2 + G_2^2) \sum_{l=1:k} \alpha_l^2 , \quad (19)$$

where the second inequality follows from that: $\|(\boldsymbol{\lambda}^{(k+1)}, \boldsymbol{\mu}^{(k+1)}) - (\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\|_2^2 \geq 0$, and the last inequality follows from that: $\|(\boldsymbol{\lambda}^{(1)}, \boldsymbol{\mu}^{(1)}) - (\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\|$ is bounded from above, i.e., $\exists\, R < \infty$ such that $\|(\boldsymbol{\lambda}^{(1)}, \boldsymbol{\mu}^{(1)}) - (\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\| < R$ and the norm of the subgradient $(\mathbf{u}, \mathbf{v})$ is bounded as

$$\|\mathbf{u}\|_2 \leq G_1 = \sqrt{\textstyle\sum_{i\in\mathcal{N}} (\max_{j\in\mathcal{M}} d_{\mathrm{ij}})^2} \quad (20)$$

$$\|\mathbf{v}\|_2 \leq G_2 = \sqrt{(N-1)^2 + (M-1)}\,. \quad (21)$$

The bound (20) is obtained by noting that there exists, at most, a nonzero element in $(x_{\mathrm{ij}})_{j\in\mathcal{M}}$, see (4) and (12) and (13). Moreover, (21) follows when all agents are assigned to one slot, see (12) and (13). By using the trivial relation

$$d^* - g_{\mathrm{best}}^{(k)} \leq d^* - g(\boldsymbol{\lambda}^{(l)}, \boldsymbol{\mu}^{(l)}),\ \ l = 1, \ldots, k\,, \quad (22)$$

and (19), we obtain an upper bound on $d^* - g_{\mathrm{best}}^{(k)}$ as

$$d^* - g_{\mathrm{best}}^{(k)} \ \leq\ \left(R^2 + (G_1^2 + G_2^2)\textstyle\sum_{l=1}^k \alpha_l^2\right) \Big/ \left(2\textstyle\sum_{l=1}^k \alpha_l\right) \quad (23)$$

Noting that step size $\alpha_l = \alpha/l,\ \ 0 < \alpha < \infty$ is *square summable*, i.e., $\sum_{l=1}^{\infty} \alpha_l^2 = \alpha^2\pi/6$. Moreover, $\sum_{l=1}^k \alpha_l$ is strictly monotonically increasing in $k$ (it grows without bound as $k \to \infty$). Therefore, for any $\epsilon > 0$, we can always find an integer $n \geq 1$ such that $\sum_{l=1}^k \alpha_l > \epsilon\,(R^2/2 + a^2(G_1^2 + G_2^2)\pi/12)$ if $k \geq n$, which concludes the proof. ■

The bound derived in (23), together with (20)–(21), allows us to predict some key behaviors of the convergence of the proposed algorithm. For example, the larger the $d_{\mathrm{ij}}$ values, the larger the $G_1$ value and, therefore, the larger the number of iterations to achieve a given accuracy. Nevertheless, the influence of $G_1$ can be made negligible by arbitrarily scaling down the objective function of problem (1). From (21), we note that the number of scheduled agents (i.e., $N$) and the number of free slots (i.e., $M$) directly influence the convergence.

We remark here that Proposition 2 proves that the DAA algorithm converges to the dual optimal of problem (8), not the primal optimal of problem (2). Furthermore, since problem (2) is nonconvex; thus, the strong duality theorem does not hold. As a consequence, there is no guarantee that the recovered primal solution is feasible and optimal by letting $k \to \infty$ in the DAA algorithm. In the next subsection, we develop an upper bound on the duality gap for the proposed algorithm with respect to primal problem (1).

### B. Optimality Properties

In this section, we characterize the suboptimality of the DAA algorithm. We start by introducing an asymmetric assignment problem related to (1). Then, based on such an assignment problem, we derive an upper bound on the duality gap of optimization problem (2), and, hence, to (1). Moreover, at the end of the subsection, we discuss how to apply the DAA algorithm to compute the optimal solution of the asymmetric assignment problem that eventually gives the solution to (1), but at the cost of a larger computational complexity than the simple application of the DAA algorithm to the dual of (1).

Consider the following asymmetric assignment problem with the same variable $\mathbf{x}$ and constraints (1b)$\sim$(1c) of (1).

$$\begin{aligned} \text{minimize} \quad & \tilde{f}^{\eta}(\mathbf{x}) := \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} d_{\mathrm{ij}}^{\eta} x_{\mathrm{ij}} \\ \text{subject to} \quad & x_{\mathrm{ij}} \in [0,1]\,, \text{and } (1\mathrm{b}) \sim (1\mathrm{c})\,, \end{aligned} \quad (24)$$

where constant parameter $\eta \geq 1$ is given. Build on assignment problem (24), the following lemma characterizes the duality gap of optimization problem (2).

*Lemma 2:* Denote by $p^*$ and $d^*$ the optimal values of the primal and dual problem of the mixed-integer linear problem (2), respectively. Then, the duality gap of (2) is bounded as follows:

$$p^* - d^* \leq f(\tilde{\mathbf{x}}^{\eta *}) - \sqrt[\eta]{\frac{1}{N}\tilde{f}^{\eta}(\tilde{\mathbf{x}}^{\eta *})}\,, \quad (25)$$

where $\tilde{\mathbf{x}}^{\eta *}$ is the optimal solution of (24).

*Proof:* We remark here that if the asymmetric assignment problem (24) is feasible, then there exists a binary optimal solution $\tilde{\mathbf{x}}^{\eta *}$ [26]. Since optimization problem (24) is feasible by assumption, thus $\tilde{\mathbf{x}}^{\eta *}$ is feasible for optimization problems (1) and (2). Moreover, recall that (2) is a mixed-integer linear problem. Consider the relaxation of (2) where $x_{\mathrm{ij}}$ can be any real value between 0 and 1. Then, strong duality holds for the dual problem (8) and the relaxation of (2). Therefore, we have $Nd^{*\eta} \geq \tilde{f}^{\eta}(\mathbf{x}^*) \geq \tilde{f}^{\eta}(\tilde{\mathbf{x}}^{\eta *})$, where $\mathbf{x}^*$ is the optimal solution for the relaxation of (2). Furthermore, we have $p^* \leq f(\tilde{\mathbf{x}}^{\eta *})$. It completes the proof. ■

We will show in the numerical results section that the bound (25) is tight in many relevant situations. In the following, based

on (24), we give an alterative solution method that can provide the optimal solution of optimization problem (1), but at the cost of a large number of iterations. First, note that $\sqrt[\eta]{\tilde{f}^{\eta}(\mathbf{x})}$ approaches $f(\mathbf{x})$, when $\eta \to \infty$. Therefore, by solving optimization problem (24) with large $\eta$, we could obtain the optimal association $\mathbf{x}^*$. Interestingly, the DDA algorithm could be used for this purpose to solve (24). Let $\boldsymbol{\mu} = (\mu_j)_{j\in\mathcal{M}}$ be the Lagrangian multipliers for constraint (1b). Then, the resulting Lagrangian function is

$$L(\mathbf{x}, \boldsymbol{\mu}) = \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} (d_{\mathrm{ij}}^{\eta} + \mu_j) x_{\mathrm{ij}} - \sum_{j\in\mathcal{M}} \mu_j \, .$$

The dual function is therefore given by

$$\tilde{g}(\boldsymbol{\mu}) = \sum_{i\in\mathcal{N}} \tilde{g}_i(\boldsymbol{\mu}) - \sum_{j\in\mathcal{M}} \mu_j$$

where $\tilde{g}_i(\boldsymbol{\mu}) = \sum_{j\in\mathcal{M}} (d_{\mathrm{ij}}^{\eta} + \mu_j)\tilde{x}_{\mathrm{ij}}^*$, with

$$\tilde{x}_{\mathrm{ij}}^* = \begin{cases} 1 & \text{if } j = \mathrm{argmin}_{l\in\mathcal{M}}(d_{il}^{\eta} + \mu_l), \\ 0 & \text{otherwise.} \end{cases}$$

Compared to (8), it is clear that the DAA algorithm could solve (24). We remark here that as stated in Section V-A, the larger the $d_{\mathrm{ij}}^{\eta}$ values, the larger the subgradient norm; therefore, the larger the number of iterations to achieve a given accuracy [compared to (23)]. In other words, the application of the DAA algorithm to (24) with very large $\eta$ is able to find the optimal solution for (1) but at the cost of a slow convergence rate. In the next section, we highlight appealing privacy preserving properties of the DAA algorithm.

## VI. Privacy Properties

We see that the proposed agent assignment mechanism DAA is preserving privacy in the sense that any agent $n \neq i$ will not be able to find out the destination $\mathrm{des}(i)$ of the $i$th agent while using the DAA algorithm. We refer to an attempt of an arbitrary agent $n$ to discover the destination of any other agent $i$, as a passive attack [27, § 5.1–5.3], where agent $n$ keeps records of possibly all of the information that it exchanges with CC and by using those, it tries to discover private data $\mathrm{des}(i)$. In what follows, we first present sufficient information that an arbitrary agent $n$ can use to discover $\mathrm{des}(i)$. Then, we show how the DAA algorithm hides such sufficient information and ensures privacy.

### A. Sufficient Information to Discover the Destination

Let us first fix the adversary to be agent $n$ and assume that agent $n$ wants to discover $\mathrm{des}(1)$, that is, the destination of agent 1. Now suppose agent $n$ knows the set $\mathcal{C}_1 = \{(j_1^k, d_{1j_1^k})\}_{k=1,2,\ldots,K}$ of data associated with agent 1, where $K$ is the total iterations of the DAA algorithm. Provided there exists at least three distinct $j_1^k$ s, agent $n$ can simply locate $\mathrm{des}(1)$ as illustrated in Fig. 2. Even if agent $n$ knows only the set $\mathcal{D}_1 = \{d_{1j_1^k}\}_{k=1,2,\ldots,K}$ of data associated with agent 1, it turns out that agent $n$ can locate $\mathrm{des}(1)$ exhaustively. In particular, in every iteration $k$, agent $n$ draws $M-1$ circles with radius

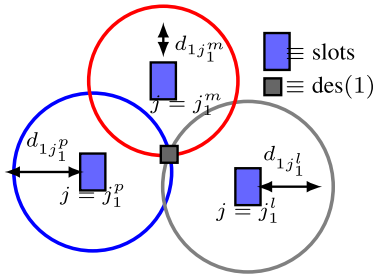


Fig. 2. Given $(j_1^p, d_{1j_1^p})$, $(j_1^l, d_{1j_1^l})$, $(j_1^m, d_{\mathrm{ij}_1^m})$ pairs known to the adversary (agent $n$), $j_1^p \neq j_1^l$, $j_1^p \neq j_1^m$, and $j_1^l \neq j_1^m$, discovering the location of $\mathrm{des}(1)$.

$d_{1j_1^k}$ centered at slots $\mathcal{M} \setminus \{j_n^k\}$. Let $\mathcal{S}^k$ denote the aforementioned set of circles. Provided there are at least three distinct $j_1^k$ s, which correspond to some iteration indices $l, m$, and $p$, one can see that there exists at least one point at which a circle in $\mathcal{S}^l$, a circle in $\mathcal{S}^m$, and a circle in $\mathcal{S}^p$ intersect. If this point is unique, then it corresponds to $\mathrm{des}(i)$.[5] The discussion above indicates that *if* the adversary (agent $n$) knows $\mathcal{C}_1$ or even $\mathcal{D}_1$, under mild conditions, it can locate $\mathrm{des}(i)$. In the sequel, we show how DAA precludes such situations. In particular, we show how $\{d_{1j_1^k}\}_{k=1,2,\ldots,K}$ is kept hidden from the adversary agent $n$.

### B. How to Preserve Privacy

Note that the only means by which agent $n$ gets access to some functions of $\{d_{1j_1^k}\}_{k=1,2,\ldots,K}$ is via $\{\lambda_n^{(k)}\}_{k=1,2,\ldots,K}$, see step 5 of DAA algorithm. In other words, the involvement of agent $n$ during the DAA algorithm is restricted so that in every iteration $k$, it has access to only some *interface* variables $\lambda_n^{(k)}$ and $\boldsymbol{\mu}^{(k)}$.[6] This restriction is indeed achieved by the decomposition structure of problem (1). Moreover, we consider the situation that CC uses the step size $\alpha_k$ of DAA as

$$\alpha_k = \alpha / k \, , \tag{26}$$

where $\alpha$ is arbitrarily chosen on $[\alpha^{\min}, \alpha^{\max}]$, $\alpha^{\min}$ and $\alpha^{\max}$ are positive numbers known only to CC such that $\alpha^{\min} < \alpha^{\max}$. Note that the aforementioned choice of $\alpha_k$ still preserves the convergence properties established in Proposition 1 (see Section V-A) [compared with (23)]: The arbitrary step size (26) essentially introduces more protection to the problem data $\{d_{\mathrm{ij}_1^k}\}_{k=1,2,\ldots,K}$.

*Proposition 3:* Consider algorithm DAA using (26). Suppose $n$ is an adversary agent of agent $i$. It is impossible for agent $n$ to record $\{d_{\mathrm{ij}_1^k}\}_{k=1,2,\ldots,K}$ and, thus, to locate $\mathrm{des}(i)$.

*Proof:* In the following, we show that if agent $n$ can document the connections among the unknown parameters including $\{d_{\mathrm{ij}_1^k}\}_{k=1,2,\ldots,K}$, among others, it can only come up with an under determined set of nonlinear equations. Thus $\{d_{\mathrm{ij}_1^k}\}_{k=1,2,\ldots,K}$ cannot be computed as we will see next.

[5] There can be more than one intersection point, which will create uncertainties in correctly locating $\mathrm{des}(i)$.

[6] The knowledge of $\boldsymbol{\mu}^{(k)}$ is irrelevant here because it does not agentry any information of $d_{1j_1^k}$.

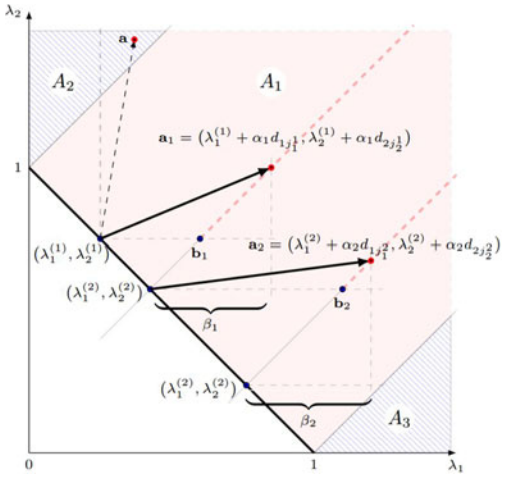


Fig. 3. Evolution of $\boldsymbol{\lambda}^{(k)} = (\lambda_1^{(k)}, \lambda_2^{(k)})$ in a case of 2-agents.

TABLE I
RELATIONS OF UNKNOWN PARAMETERS AS SEEN BY THE ADVERSARY AGENT 2, WHERE $k$ IS THE ALGORITHM'S ITERATION $U_K$ AND $E_K$ ARE THE NUMBERS OF THE UNKNOWN VARIABLES AND AVAILABLE RELATIONS, RESPECTIVELY

| $k$ | relations | | unknowns | $U_k$ | $E_k$ |
|---|---|---|---|---|---|
| 1 | $\lambda_1^{(1)} + \lambda_2^{(1)} = 1$ | (1.1) | $\lambda_1^{(1)}$ | 1 | 1 |
| 2 | | (1.1) | $\lambda_1^{(1)}, \lambda_1^{(2)}$ | | |
| | $\lambda_1^{(2)} + \lambda_2^{(2)} = 1$ | (2.1) | $\beta_1$ | 5 | 4 |
| | $\lambda_1^{(2)} + \beta_1 = \lambda_1^{(1)} + \alpha_1 d_{1j_1^1}$ | (2.2) | $\alpha_1$ | | |
| | $\lambda_2^{(2)} + \beta_1 = \lambda_2^{(1)} + \alpha_1 d_{2j_2^1}$ | (2.3) | $d_{1j_1^1}$ | | |
| 3 | (1.1), (2.1), (2.2), | (2.3) | $\lambda_1^{(1)}, \lambda_1^{(2)}, \lambda_1^{(3)}$ | | |
| | $\lambda_1^{(3)} + \lambda_2^{(3)} = 1$ | (3.1) | $\beta_1, \beta_2$ | 9 | 7 |
| | $\lambda_1^{(3)} + \beta_2 = \lambda_1^{(2)} + \alpha_2 d_{1j_1^2}$ | (3.2) | $\alpha_1, \alpha_2$ | | |
| | $\lambda_2^{(3)} + \beta_2 = \lambda_2^{(2)} + \alpha_2 d_{2j_2^2}$ | (3.3) | $d_{1j_1^1}, d_{1j_1^2}$ | | |
| $\vdots$ | | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |

For notational simplicity, we consider only the case with $n = 2$ and suppose that agent 2 is the adversary that wants to discover the destination of agent 1, that is, $\mathrm{des}(1)$. The discussion can be generalized to scenarios with $n > 2$, in a straightforward manner.

First, note that CC performs the projection of $\boldsymbol{\lambda}^{(k)} - \alpha_k \mathbf{u}^{(k)}$ onto the probability simplex to yield $\boldsymbol{\lambda}^{(k+1)}$ [see step 4-c of DAA algorithm]. In the considered 2-agent case, the probability simplex is the line segment form $(0, 1)$ to $(1, 0)$, see Fig. 3. Once agent 2 is given $\lambda_2^{(1)}$, it can locate $(\lambda_1^{(1)}, \lambda_2^{(1)})$, because $\lambda_1^{(1)} = 1 - \lambda_2^{(1)}$. Yet, agent 2 cannot locate $\mathbf{a}_1 = \boldsymbol{\lambda}^{(1)} - \alpha_1 \mathbf{u}^{(1)}$. After receiving $\lambda_2^{(2)}$, agent 2 can locate $(\lambda_1^{(2)}, \lambda_2^{(2)})$. It can also locate $\mathbf{a}_1$ up to the *ray* originating at $\mathbf{b}_1$, see Fig. 3. However, agent 2 cannot exactly locate $\mathbf{a}_1$. The algorithm continues in a similar manner. For example, the evolution of $\boldsymbol{\lambda}^{(k)} = (\lambda_1^{(k)}, \lambda_2^{(k)})$ is illustrated in Fig. 3 for $k = 1$, 2, and 3. With this knowledge of $\boldsymbol{\lambda}^{(k)}$ evolution, agent 2 can write a set of equations in every iteration $k$ as given in Table I. Note that the set of equations for $k > 1$ is nonlinear, because there are products of unknowns, for example, (2.2), (2.3), (3.2), and (3.3).

When documenting the relations of unknowns in Table I, we assume $\{\mathbf{a}_k\}_{k=1,2,3\ldots}$ lies only in the shaded area $A_1$. In contrast, suppose $\mathbf{a}_k$ lies either in the hatched area $A_2$ or $A_3$ for some iterations. One such point is depicted in Fig. 3, where $\mathbf{a}_1 = \mathbf{a}$, see the hatched area $A_2$. In this case, $(\lambda_1^{(2)}, \lambda_2^{(2)})$ will be $(0, 1)$. Consequently, agent 2 can locate $\mathbf{a}_1$ only up to a *cone* instead of a *ray* which, in turn, accounts for more uncertainties in determining $\mathbf{a}_1$. Such situations can only increase the difference between the number of unknowns $(U_k)$ and the number of equations $(E_k)$. For example, in this case, we will have $U_2 - E_2 > 1$, instead of $U_2 - E_2 = 1$ as in Table I.

Thus, we conclude that the total unknowns are always greater than the total equations. This results an under determined set of nonlinear equations, see Table I. Therefore, agent 2 cannot make records of unknowns $\{d_{1j_1^k}\}_{k=1,2,\ldots,K}$ and, consequently, it cannot discover $\mathrm{des}(1)$ as discussed in Section VI-A. This concludes the proof. ■

Moreover, we assume that there is another layer between the CC and the agents. We assume that the locations and the index of the destinations are only known by this layer. Therefore, hacking only the CC or the agent is insufficient to find the real destinations of the agent, which informally increases the security.

## VII. NUMERICAL RESULTS

In this section, we present the numerical evaluation of our proposed algorithm DAA to optimization problem (1), and compare it to the following benchmarks:

a) Greedy assignment policy: In this case, each agent selects the closest slot to its destination.
b) Optimal assignment policy: A solution of the optimization problem (2) is found by using the general solver, the IBM CPLEX optimizer.

In each time slot, the proposed algorithm is carried out for $K$ subgradient iterations. In addition, the greedy policy and the optimal policy have also been performed at every time frame. We average the results over $T$ time frames to demonstrate the average performances of the DAA algorithm.[7] Specifically, at the beginning of every time frame, the total number of agents $N$ and the total number of free slots $M$ are considered to be fixed, and the random locations of destinations of the agents and the free slots are uniformly distributed over a $1000 \times 1000$ area in each time frame. Thus, we could compute the distance $\{d_{\mathrm{ij}}\}_{i\in\mathcal{N}, j\in\mathcal{M}}$ for each agent $i$ with respect to each free slot $j$. Note that the assignment distances $\{d_{\mathrm{ij}}\}_{i\in\mathcal{N}, j\in\mathcal{M}}$ change from frame to frame.

To simplify the presentation, we denote by $p^{\mathrm{cur}}(t, k)$ the best objective value achieved at time frame $t$ after $k$ subgradient iterations [compare to $p^{\mathrm{cur}}(k)$ in step 4-c,d of the DAA algorithm]. In particular

$$p^{\mathrm{cur}}(t, k) = \underset{l=1,\ldots,k}{\mathrm{argmin}}\ p(t, l), \tag{27}$$

[7]Note that each time frame can be regarded as one experiment. In this paper, we investigate the performance of the proposed algorithms among 1000 numerical experiments by setting $T = 1000$.

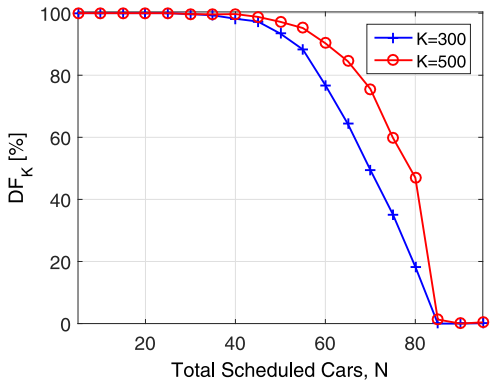


Fig. 4. Degree of feasibility $\mathrm{DF}_K$ versus total agents $N$ with $M = 100$.

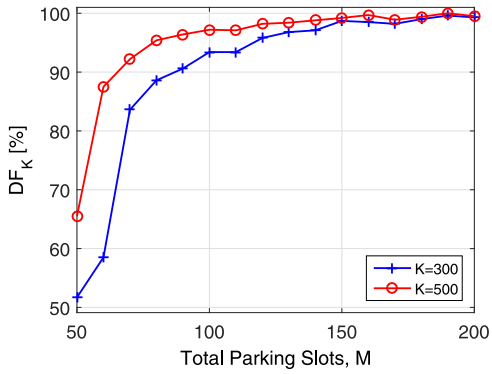


Fig. 5. Degree of feasibility $\mathrm{DF}_K$ versus total slots $M$ with $N = 50$.

where $p(t, l)$ is the objective value at time frame $t$ and at subgradient iteration $l$. Note that $p^{\mathrm{cur}}(t, k)$ is similar to $p^{\mathrm{cur}}(k)$ of the DAA algorithm with an additional index $t$ to indicate the time frame. Moreover, we denote by $\mathbf{X}^{\mathrm{cur}}(t, k)$ the *best* feasible or infeasible solution, which is identical to $\mathbf{X}^{\mathrm{cur}}(k)$ of the DAA algorithm with an additional index $t$ to indicate the time frame.

In all considered simulations, we use $T = 1000$. Moreover, $K$ is chosen to be 300 or 500. We first define a performance metric called *the degree of feasibility* of $\mathbf{X}^{\mathrm{cur}}(t, k)$. Note that $\mathbf{X}^{\mathrm{cur}}(t, k)$ is, in fact, the assignment at the beginning of step 6 of DAA algorithm, which can be either feasible or infeasible. If $\mathbf{X}^{\mathrm{cur}}(t, k)$ is feasible, we have $N^{\mathrm{conflict}} = 0$ or equivalently $p^{\mathrm{cur}}(t, K) < \infty$ (compared with step 6, 4-c, and 4-d). On the other hand, if $\mathbf{X}^{\mathrm{cur}}(t, k)$ is infeasible, we have $N^{\mathrm{conflict}} > 0$ or equivalently $p^{\mathrm{cur}}(t, K) = \infty$ (compared with step 6, 4-c, and 4-d). This motivates defining the degree of feasibility (DF) of $\mathbf{X}^{\mathrm{cur}}(t, k)$ as

$$\mathrm{DF}_K = \frac{1}{T}\sum_{t=1}^{T} I\big(p^{\mathrm{cur}}(t, K) < \infty\big) \times 100\% \,, \qquad (28)$$

where $I(E)$ is the indicator function of event $E$, that is $I(E) = 1$ if $E$ is true or $I(E) = 0$ otherwise.

Fig. 4 shows $\mathrm{DF}_K$ versus $N$ for fixed $M = 100$. The results show that when $N$ is significantly smaller than $M$, the degree of feasibility is almost 100%. The results further show that as $N$ becomes closer to $M$, the degree of feasibility starts deteriorating. Not surprisingly, running the DAA algorithm for a larger number of subgradient iterations (e.g., $K = 500$) yields better feasibility results compared to a smaller number of subgradient iterations (e.g., $K = 300$).

Fig. 5 shows $\mathrm{DF}_K$ versus $M$ for fixed $N = 50$. The results resemble the observations of Fig. 4, where desirable feasibility is achieved when $M$ is significantly larger than $N$ and the performances are pronounced for smaller dimensional problems.

To see the average behavior of the DAA algorithm, now we consider the following performance metric, which is a measure of the average objective value at subgradient iteration k

$$p^{\mathrm{ave}}(k) = \frac{1}{T}\sum_{t=1}^{T} p^{\mathrm{cur}}(t, k) \,, \qquad k = 1, \ldots, K \,. \qquad (29)$$

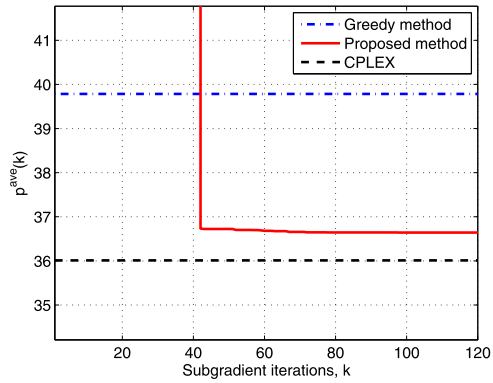


Fig. 6. Average objective $p^{\mathrm{ave}}(k)$ versus subgradient iterations $k$ with $N = 20$ and $M = 100$.

Fig. 6 shows $p^{\mathrm{ave}}(k)$ versus subgradient iterations $k$. In particular, we consider the cases $M = 100$ and $N = 20$. Note that the vertical drops of the curves associated with our proposed method correspond to the subgradient iteration, before which a feasible assignment is found during any time frames $t = \{1, \ldots, T\}$. Not surprisingly, the optimal CPLEX method gives the best average objective, which is achieved at the expense of high computational complexity. However, our proposed method trades off an increase in average objective value for a low complexity in the algorithm, which is gracefully scalable. Still, the performance degradation of the proposed method is not critical. For example, the performance loss of the DAA method compared with the optimal is $1.72\%$ and that of the greedy method is $10.48\%$. The results thus show that even in large networks, our proposed DAA method can outperform the greedy approach substantially. Moreover, it requires only around 50 iterations by the DAA algorithm to converge to a near to optimal solution, which indicates that even $K = 300$ is large enough for being the stopping criterion.

Recall that if $p^{\mathrm{cur}}(t, K) = \infty$, then the corresponding assignment $\mathbf{X}^{\mathrm{cur}}(t, K)$ is infeasible. In such situations, our proposed algorithm invokes its subroutine (see Appendix) to construct a feasible assignment by using the best infeasible solution achieved so far, see step 6 of the DAA algorithm. On the other

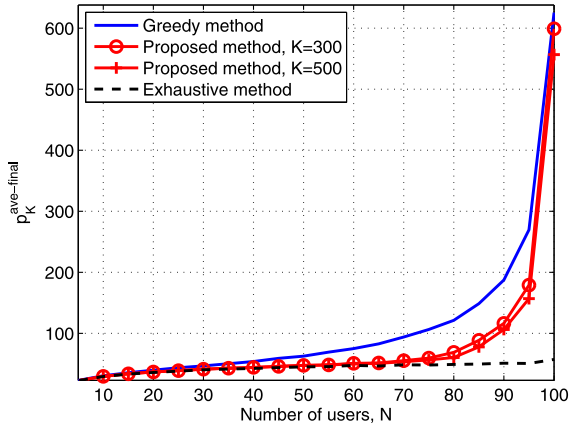


Fig. 7. Average objective value after the termination of DAA $p_K^{\text{ave}-\text{final}}$ versus total agents $N$ with $M = 100$.

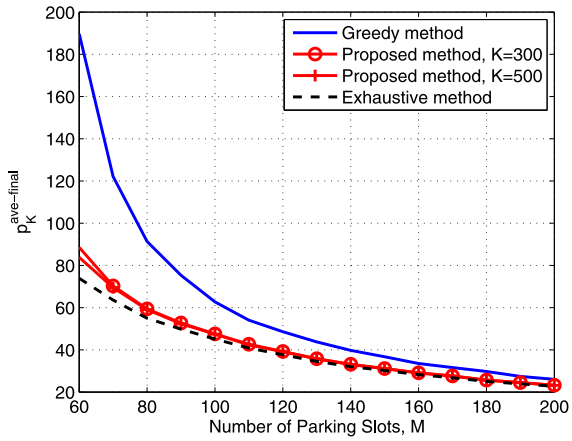


Fig. 8. Average objective value after the termination of DAA $p_K^{\text{ave}-\text{final}}$ versus total slots $M$ with $N = 50$.

hand, if $p^{\text{cur}}(t, K) < \infty$, then the corresponding assignment $\mathbf{X}^{\text{cur}}(t, K)$ is already feasible. In either case, the DAA algorithm returns a feasible point as given in step 6. We denote by $\mathbf{X}^{\text{final}}(t)$ this feasible assignment returned by our proposed DAA algorithm at time frame $t$ and by $p^{\text{final}}(t, K)$ the corresponding objective value. Finally, we denote by $p_K^{\text{ave}-\text{final}}$ the average objective value achieved after the termination of the DAA algorithm. In particular, $p_K^{\text{ave}-\text{final}}$ is given by

$$p_K^{\text{ave}-\text{final}} = \frac{1}{T}\sum_{t=1}^{T} p^{\text{final}}(t, K)\,. \tag{30}$$

Note that when $X^{\text{cur}}(t, K)$ is feasible for all $t \in \{1, \ldots, T\}$, then $p_K^{\text{ave}-\text{final}} = p^{\text{ave}}(K)$ [compare with (29)].

Fig. 7 shows the average objective value $p_K^{\text{ave}-\text{final}}$ versus $N$ for the cases $M = 100$. The results show that DAA algorithm always outperforms the greedy method. Using subgradient iterations $K = 500$ accounts for an increase in the performance gain compared with $K = 300$, though the gains are not substantial. When the setup is lightly loaded, the proposed DAA performs very close to the optimal approach. For fixed $M$, increasing $N$ results increasing of the performance gap. For example, in the case of $M = 100$ and $N = 50$ with $K = 300$ corresponds to a $5.40\%$ performance deviation of DAA compared to the optimal CPLEX and $N = 95$ corresponds to a $259.20\%$ performance deviation. Such reductions in the performance gains are certainly expected, because there is a tradeoff between the complexity of the algorithms and the performance loss. The results further show that the larger the slots $M$, the larger the performance deviation, especially in heavily loaded cases.

Fig. 8 shows the average objective value $p_K^{\text{ave}-\text{final}}$ versus $M$ for the cases $N = 50$. The observations are similar to those in Fig. 7. The results confirm that our DAA algorithm performs very close to the optimal CPLEX method in lightly loaded cases, where $N/M < 0.5$, see Fig. 8 curves with $M > 100$. However, there is a noticeable performance degradation in heavily loaded cases, see Fig. 8 curves with $M < 100$. Moreover, the proposed method substantially outperforms the greedy method in all considered scenarios.

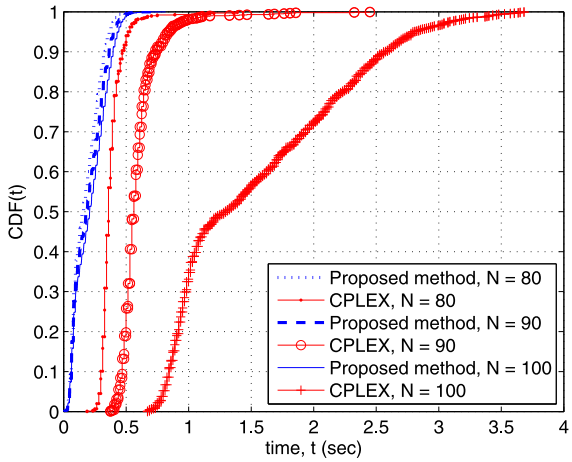


Fig. 9. CDF of time with $M = 100$.

In order to provide a statistical description of the speed of the proposed algorithm, we consider empirically the cumulative distribution function (CDF) plots. Specifically, for each time frame $t \in \{1, \ldots, T\}$, we store the total CPU time required for the DAA algorithm to find $\mathbf{X}^{\text{final}}(t, K)$, where we use $K = 300$. Similarly, the total CPU time required to find the *optimal* value by using CPLEX is recorded. Fig. 9 shows the empirical CDF plots of the time for $N = 80, 90, 100$ with $M = 100$. In the case of the DAA algorithm, the effects of changing the problem size by increasing $N$ on the CDF plots are almost indistinguishable. However, in the case of optimal CPLEX method, there is a prominent increase in the time required to compute the optimal value. It should be emphasized that CPLEX finds the optimal assignment with a high penalty to do it on time, whereas the DAA algorithm finds the feasible assignment efficiently with a penalty on the optimality.

Fig. 10 depicts the average time required by DAA algorithm, the optimal method, and the greedy method versus $N$ for $M = 100$. The results show that the average time required by DAA to possibly find a suboptimal solution is not sensitive to the variation of $N$ and is in the range $0 - 1$ seconds. Moreover, they are comparable to the average time of simple greedy

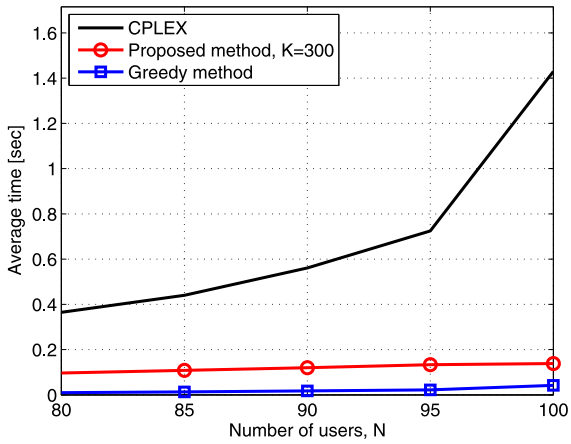


Fig. 10. Average CPU time with $M = 100$.

method. However, the average time required by the optimal method to find the optimal solution grows approximately exponentially with $N$. This is certainly expected because problem (2) is combinatorial, and therefore the worst-case complexity of the optimal method grows exponentially with the problem size [20, § 1.4.2]. Thus, there is naturally a tradeoff between the optimality and the efficiency of the algorithms. The result suggests that our proposed DAA algorithm yields a good tradeoff between the optimality and the efficiency, especially in lightly loaded cases. These properties are favorable for practical implementation.

## VIII. Conclusions

In this paper, we considered the problem of agent assignment. Unlike the existing greedy approaches, our problem formulation considered fairness among the scheduled agents in the sense that the global objective was to minimize the maximum distance from the slots to the intended destinations of the agents. A method based on Lagrange duality theory was proposed to address the nonconvex and combinatorial assignment problem. Our formulation and the corresponding algorithm generally apply to fair agent-target assignment problems in other application domains beyond intelligent transport systems. We showed that the proposed method is privacy preserving in the sense that any agent involved in the algorithm will not be able to discover the destination of any other agent during the algorithm iterations. Unlike the optimal exponentially complex approaches, our proposed method is scalable. Numerical results showed that for all considered cases, where the number of free slots are equal or higher than twice the scheduled agents, our proposed algorithm's performance is similar to that of the optimal method. In all considered cases, the proposed algorithm outperformed the simple greedy approach. Therefore, the proposed algorithm yields a good tradeoff between the implementation-level simplicity and the optimality.

## Appendix

Here, we show how to construct a feasible assignment. The key idea of the subroutine is summarized as follows: 1) select the set of agents that are assigned to the *same* slot; 2) find the set of free slots; and 3) assign the conflicting agents found in the first stage to the free slots found in the second stage in an iterative manner. We start by introducing some useful notations for clarity. We denote by $\mathcal{M}^{\text{over}-\text{assigned}}$ the set of slots, where two or more than two agents are assigned. Moreover, we denote by $\mathcal{M}^{\text{free}}$ the set of free slots. Let $\boldsymbol{\sigma} = (\sigma_l)_{l=1,\ldots,\text{agentd}(\mathcal{M}^{\text{over}-\text{assigned}})}$ denote the slot indices $j \in \mathcal{M}^{\text{over}-\text{assigned}}$ arranged in an increasing order. Moreover, we denote by $n_j$ the total agents assigned to the $j$th slot. The subroutine can be formally expressed as follows:

**Algorithm:** Construct a feasible assignment from $\mathbf{X}^{\text{infeasible}}$.

1) Given the infeasible assignment $\mathbf{X}^{\text{infeasible}}$; $\mathcal{M}^{\text{over}-\text{assigned}}$; $\mathcal{M}^{\text{free}}$, $\boldsymbol{\sigma}$, and $n_j \ \forall j \in \mathcal{M}^{\text{over}-\text{assigned}}$. Set $\mathbf{X}^{\text{final}} = \mathbf{X}^{\text{infeasible}}$, $k = 1$, and $l = 1$.
2) CC sets $\boldsymbol{\pi} = (\pi_n)_{n=1,\ldots,n_{\sigma_l}}$ to be the agent indices $i \in \mathcal{J}^{\text{cur}}_{\sigma_l}$ arranged in an increasing order.
3) For $n = 2 : n_{\sigma_l}$
   a. CC sends $\mathcal{M}^{\text{free}}$ to agent $\pi_n$.
   b. agent $\pi_n$ chooses slot $j$, where $j = \arg\min_j d_{\pi_n j}$ and sends $j$ to CC.
   c. CC updates $\mathcal{M}^{\text{free}} \leftarrow \mathcal{M}^{\text{free}} \setminus \{j\}$ and sets $[\mathbf{X}^{\text{final}}]_{\pi_n j} = 1$.
4) If $l = \text{agentd}(\mathcal{M}^{\text{over}-\text{assigned}})$, return $\mathbf{X}^{\text{final}}$ and STOP. Otherwise, set $l = l + 1$ and go to step 2.

## References

[1] E. Alfonsetti, P. C. Weeraddana, and C. Fischione, "Min-max fair car-parking slot assignment," in *Proc. IEEE SmartVehicles Workshop*, 2015.
[2] J.-P. Rodrigue, C. Comtois, and B. Slack, *The Geography of Transport Systems*, New York, USA: Routledge, 2nd ed., 2013. [Online]. Available: http://people.hofstra.edu/geotrans.
[3] D. C. Shoup, "Cruising for parking," *Transport Policy: Special Issue on Parking*, vol. 13, no. 6, pp. 479–486, 2006.
[4] M. Ergen, S. Coleri, Y. Shon, M. Ozalp, and A. Long, EzPARK. [Online]. Available: http://wow.eecs.berkeley.edu/ergen/EZPARK/index.htm, 2003.
[5] M. Piorkowski, M. Grossglauser, and A. Papaioannou, "Mobile user navigation supported by WSAN: Full-fledge demo of the smartpark system," in *Mobile Ad Hoc Netw. Comput.*, Florence, Italy, May 22–25 2006.
[6] M. Piorkowski, M. Grossglauser, and A. Papaioannou, "SmartPark: High-mobility application supported by wireless sensor and actuator network," in *Proc. Mobile Inf. Commun. Syst.*, Zurich, Switzerland, Oct. 17–19 2006.
[7] V. W. S. Tang, Y. Zheng, and J. Cao, "An intelligent car park management system based on wireless sensor networks," in *Proc. 1st Int. Symp. Pervasive Comput. Appl.*, 2006, pp. 65–70.
[8] Y.-Z. Bi, L.-M. Sun, H.-S. Zhu, T.-X. Yan, and Z.-J. Luo, "A parking management system based on wireless sensor network," in *ACTA Automatica SINICA*, 2011, vol. 32, pp. 38–45.
[9] J. Chinrungrueng, U. Sunantachaikul, and S. Triamlumlerd, "Smart parking: An application of optical wireless sensor network," in *Proc. Int. Symp. Appl. Internet Workshops*, 2007, pp. 66–66.
[10] L. Rongxing, X. Lin, H. Zhu, and X. Shen, "Spark: A new vanet-based smart parking scheme for large parking lots," in *Proc. IEEE INFOCOM*, 2009, pp. 1413–1421.
[11] L. Rongxing, X. Lin, H. Zhu, and X. Shen, "An intelligent secure and privacy-preserving parking scheme through vehicular communications," *IEEE Trans. Veh. Technol.*, vol. 59, no. 6, pp. 2772–2785, 2010.
[12] R. Souissi, O. Cheikhrouhou, I. Kammoun, and M. Abid, "A parking management system using wireless sensor networks," in *Proc. Int. Conf. Microelectron.*, 2011, pp. 1–7.

[13] Streetline Networks Inc., Parker mobile. [Online]. Available: http://www.streetline.com/find-parking/.
[14] VehicleSense Inc., Street parking information network (SPIN). [Online]. Available: http://www.vehiclesense.com/vs_solutions.html.
[15] San Francisco Municipal Transportation Agency (SFMTA), SFpark. [Online]. Available: http://sfpark.org/.
[16] M. Bürger, G. Notarstefano, F. Bullo, and F. Allgöwer, "A distributed simplex algorithm for degenerate linear programs and multi-agent assignments," *Automatica*, vol. 48, no. 9, pp. 2298–2304, 2012.
[17] B. Radunović and J.-Y. Le Boudec, "A unified framework for max-min and min-max fairness with applications," *IEEE/ACM Trans. Netw.*, vol. 15, no. 5, pp. 1073–1083, Oct. 2007.
[18] F. Farokhi, I. Shames, M. G. Rabbat, and M. Johansson, "On reconstructability of quadratic utility functions from the iterations in gradient methods," *submitted to Automatica*, 2015.
[19] P. C. Weeraddana, G. Athanasiou, C. Fischione, and J. S. Baras, "Per-se privacy preserving solution methods based on optimization," in *Proc. 52nd IEEE Conf. Dec. Control*, 2013, pp. 206–211.
[20] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, U.K., 2004.
[21] R. Horst, P. Pardolos, and N. Thoai, "Introduction to global optimization," vol. 48, 2000.
[22] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, MA, USA, 2nd ed., 1999.
[23] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris, *Fundamentals of Queueing Theory*, Wiley-Interscience, New York, USA, 4th ed., 2008.
[24] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problem*, SIAM, 2012.
[25] S. Boyd, Subgradient methods. [Online]. Available: http://www.stanford.edu/class/ee364b/lectures/subgrad_method_slides.pdf, 2007.
[26] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Method*, Athena Scientific, Belmont, MA, USA, 2nd ed., 1998.
[27] O. Goldreich, *The Foundations of Cryptography*, vol. 2, Cambridge University Press, Cambridge, U.K., 2004.

**Yuzhe Xu** (S'11–M'16) received the M.Sc. degree in system control and robotics program and the Ph.D. degree in electrical engineering from KTH Royal Institute of Technology, Stockholm, Sweden, in 2011 and 2016, respectively.

His research interests include wireless communications, distributed optimization, combinational optimization, and signal processing.

**Elisabetta Alfonsetti** received the M.Sc. degree in computer science from the University of L'aquila, Italy, in 2012.

She was then a Research Engineer with the Automatic Control Lab, Electrical Engineering Department, KTH Royal Institute of Technology, Stockholm, Sweden, focusing on optimization techniques and the privacy-preserving issue. She is currently with the TerraSwarm Lab, Electrical Engineering Department, University of California at Berkeley, Berkeley, CA, USA.

Her research interests include the application of the contract-based design paradigm for the design of complex systems.

**Pradeep Chathuranga Weeraddana** (S'08–M'11) received the M.Eng. degree in telecommunication from the School of Engineering and Technology, Asian Institute of Technology, Thailand, in 2007 and the Ph.D. degree from the University of Oulu, Finland, in 2011.

He was a Postdoctoral Researcher in the Department of Automatic Control, School of Electrical Engineering and ACCSS Linnaeus Center, KTH Royal Institute of Technology, Stockholm, Sweden, from 2012 to 2014. He is currently a Senior Lecturer at the Sri Lanka Institute of Information Technology, Malabe, Sri Lanka. His research interests include the application of optimization techniques in various application domains, such as signal processing, wireless communications, and smart grids. He is also interested in application domains of computational topology for data analysis.

**Carlo Fischione** (M'05) received the Laurea (Hons.) degree in electronic engineering and the Ph.D. degree in electrical and information engineering from the University of L'Aquila, Italy, in 2001 and 2005, respectively.

Currently, he is a tenured Associate Professor at KTH Royal Institute of Technology, Electrical Engineering, Stockholm, Sweden. He has held research positions at the Massachusetts Institute of Technology, Cambridge, MA, USA (2015, Visiting Professor); Harvard University, Cambridge (2015, Associate); and the University of California at Berkeley, Berkeley, CA, USA (2004–2005, Visiting Scholar, and 2007–2008, Research Associate). His research interests include optimization with applications to wireless-sensor networks and IoT, networked control systems, wireless networks, as well as security and privacy.

Prof. Fischione received or co-received a number of awards, including the best paper award from the IEEE Transactions on Industrial Informatics (2007), the best paper awards at the IEEE International Conference on Mobile Ad-hoc and Sensor System 05 and 09 (IEEE MASS 2005 and IEEE MASS 2009), and the Best Paper Award of the IEEE Sweden VT-COM-IT Chapter (2014). He is Associate Editor of Elsevier Automatica and, as a technical member, has chaired program committees of several international conferences. He is co-funder and CSO of the company MIND (ancient and modern musical instruments networked). He is an Ordinary Member of DASP (the academy of history Deputazione Abruzzese di Storia Patria).