# A distributed joint-learning and auction algorithm for target assignment

Teymur Sadikhov Minghui Zhu Sonia Martínez

***Abstract*— We consider an agent-target assignment problem in an unknown environment modeled as an undirected graph. Agents do not know this graph or the locations of the targets on it. However, they can obtain local information about these by local sensing and communicating with other agents within a limited range. To solve this problem, we come up with a new distributed algorithm that integrates Q-Learning and a distributed auctions. The Q-Learning part helps estimate the assignment benefits for each agent-target pair, while the auction part takes care of assigning agents to targets in a distributed and almost optimal fashion. The algorithms are shown to terminate with a near-optimal assignment in a finite time.**

## I. Introduction

Recent technological advances are making possible the deployment of large groups of autonomous agents to accomplish different missions. Examples include distributed search and rescue operations, surveillance, and exploration tasks. These missions could be risky, expensive, or just hard to accomplish with the use of single agents. On the other hand, a group of less sophisticated agents could provide not only much more flexibility but also robustness to failure. This has sparked much interest in multi-agent systems and cooperative control algorithms, where the agents aim to accomplish a global goal through local interactions.

The problem we consider here is mainly motivated by pick-up and delivery transportation problems or routing problems, where the information about the environment cannot be obtained by satellites or aerial reconnaissance due to bad coverage or weather conditions. In particular, we consider a target assignment problem, where the exact locations of the targets may be unknown *a priori* and weather conditions can dramatically affect the optimality of different routes. A solution will necessarily require the vehicles to explore the environment in order to be matched to the targets producing the greatest assignment benefit. Moreover, since we are looking for a distributed solution, the vehicles should communicate during the assignment phase so that maximization of the total assignment benefit is achieved.

In our approach, we represent the environment by an undirected graph but the number of vertices, edges of the graph and the locations of the targets on it are unknown to the agents. In order to learn its surroundings and establish a good assignment, agents will gain rewards traveling along the edges of the graph. The agents should then estimate target-assignment benefits from the rewards and target locations. To solve the target-assignment problem in a distributed way, we adapt the algorithm in [15], and for the exploration of the environment we use the Q-Learning algorithm introduced in [12]. We consider the cases where agents or targets can be added to or removed from the system, and the cases where future rewards can be undiscounted or discounted.

*Literature review.* Different techniques have been proposed to solve assignment problems due to its practical and theoretical significance [1], [3], [8]. One of the most widely-used assignment algorithms is the auction algorithm first introduced by Bertsekas in [4], which guarantees a near-optimal assignment provided that a feasible one exists. More recently, a distributed version was proposed in [15], with the requirement that the network of agents be jointly connected sufficiently often over time.

Distributed target assignment problems for multi-agent systems were recently studied in [11]. In this paper, the authors propose monotonic algorithms to solve target assignment problems. Under one set of assumptions, agents have limited sensing to detect unknown target locations. However, in this case the authors do not consider the cost or reward to travel different paths in the environment. In [14], the authors develop distributed multi-destination potential fields to dynamically assign destinations to agents where the positions of destinations are available *a priori* to all agents. In [7], the distributed sequential augmenting path algorithm is employed to solve assignment problems.

As mentioned earlier, Q-Learning algorithm was first introduced in [12], and its convergence properties can be found in [13], [6], [10]. More recently, a distributed Q-Learning algorithm is proposed in [9] to solve cooperative multi-agent-decision-processes where reward functions depend upon the strategies of all agents and identical for all agents.

*Statement of contributions.* In this paper, we address a problem of target assignment and vehicle routing in unknown environments by efficiently combining distributed auction strategies with a Q-learning process. In this way, we propose the distributed Joint Learning and Assignment (JOLEAS) algorithm, and its variation, the AJOLEAS algorithm. By introducing suitable leading roles among vehicles, the schemes can make assignments when the number of vehicles is less than or exceeds the number of targets. Moreover, the algorithms can deal with changing number of targets and agents including the leader. In order to keep the assignment running properly, in the first algorithm initially virtual targets are introduced and later deleted as the real targets are found, whereas, in the second one virtual agents are introduced by the leader as the real targets are found. As opposed to the algorithm given in [15], the algorithms do not require agents to have *a priori* information about the assignment benefits associated with given targets; moreover, these benefits may

The authors are with Department of Mechanical and Aerospace Engineering, University of California, San Diego, 9500 Gilman Dr., La Jolla CA, 92093, {tsadikho, mizhu, soniamd}@ucsd.edu

change with time. The algorithms are analyzed and proved to work under both discounted and undiscounted rewards cases.

*Organization of the paper.* The remainder of the paper is organized as follows. In Section II, we introduce notation that will be used throughout the paper and statement of the problem that we investigate. In Section III, we briefly present some preliminaries on Q-Learning and Auction algorithms. In Section IV, we present the main contribution of the paper, the JOLEAS and AJOLEAS algorithms. Section V includes convergence analysis for the algorithms for both undiscounted and discounted rewards cases. Finally, Section VI includes some concluding remarks.

## II. Problem Formulation and Notation

In this section, we introduce the notation that will be used throughout the paper and state the problem that we address.

Consider a 2-D bounded environment $Q \subseteq \mathbb{R}^2$ on which a group of $n$ agents is deployed. The environment is discretized into a grid that we identify with an undirected graph $\mathcal{G} = (\mathcal{S}, \mathcal{A})$. The finite set of vertices is enumerated as $\mathcal{S} = \{s_1, \ldots, s_M\}$, while the edge set $\mathcal{A}$ satisfies $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S} \setminus \mathrm{diag}(S)$. In what follows, we will refer to the vertices of the graph $\mathcal{G}$ as *states* that an agent can reach by moving along the graph edges.

Consider that there are $m$ static *targets*, labeled by $j \in \mathcal{T} = \{1, \ldots, m\}$, located at the special states $s_{t_j} \in \mathcal{S}_T \subseteq \mathcal{S}$. The targets are to be found by the $n$ agents, labeled by $i \in \mathcal{V} = \{1, \ldots, n\}$. Each agent $i$ is initially at the source state $s_{0i}$. Agents travel along the edges of the graph by choosing *actions* or *feasible edges* connecting their current state with another one. The set of feasible actions at state $s \in \mathcal{S}$ is denoted as $\mathcal{A}(s) = \{a \in \mathcal{S} \times \mathcal{S} | a = (s, s') \in \mathcal{A}\}$. Denote the set of possible paths from $s_{0i}$ to target $j$ as $\Pi_{ij}$. A path $\pi_{ij} \in \Pi_{ij}$ has length $|\pi_{ij}|$ if it can be expressed as the chain of states $\pi_{ij} = s_{0i} s_1 \ldots s_{|\pi_{ij}|-1} s_{t_j} \in \Pi_{ij}$, where $(s_d, s_{d+1}) \in \mathcal{A}$, for $d \in \{0, \ldots, |\pi_{ij}| - 1\}$.

The agents do not have *a priori* information about the number of graph states, the reward that is incurred while traveling from one state to another, or the location of targets. However, the agents have a limited sensing and communication range $R$. Therefore, they can obtain local information about the edges in the graph, and can communicate with others within range $R$. We define the set of neighbors of agent $i$ at time $t$, by $\mathcal{N}_i(t) = \{j \in \mathcal{V} | j \textit{ is within distance } R \textit{ of } i\}$.

As a consequence of taking certain actions, agents will incur cost or reward. To quantify the possible cost/reward, we define a function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Each edge $(s_k, s_l) \in \mathcal{A}(s_k)$, with $s_l \in \mathcal{S} \setminus \mathcal{S}_T$, is associated with a bounded negative cost $-\infty < r(s_k, (s_k, s_l)) < 0$. The bounded cost can be thought of as traveling time in our routing problem. We would like to minimize the traveling time, which is equivalent to minimizing the total cost sum in the problem that we consider. On the other hand, each edge $(s_m, s_{t_j})$, $s_{t_j} \in \mathcal{S}_T$, satisfies $0 < r(s_m, (s_m, s_{t_j})) < +\infty$.

The benefit for agent $i$ to follow $\pi_{ij}$ and reach the target $j$ is given by $c(\pi_{ij}) = \sum_{l=0}^{|\pi_{ij}|-1} \gamma^l r(s_l, a_l)$, where $0 \leq \gamma \leq 1$ is a discount factor. We consider two cases: undiscounted and discounted rewards cases. In the first case, the rewards of the edges that will be traveled in the future have the same weight as the rewards from the present edges; i.e., $\gamma = 1$. In the second case, the future rewards matter less than the present ones; i.e., $\gamma < 1$.

Denote by $\beta_{ij} = \sup_{\pi_{ij} \in \Pi_{ij}} c(\pi_{ij})$ the benefit of assigning the destination $j$ to agent $i$. Observe that $\beta_{ij}$ is unknown and equal to certain $c(\pi_{ij})$ for a path $\pi_{ij}$ with no loops. Thus, $\beta_{ij}$ is bounded, since $r$ is bounded and there is a finite number of paths without loops from $i$ to $j$.

Consider extended lists $\bar{\mathcal{V}} = \bar{\mathcal{T}} = \{1, \ldots, \max\{n, m\}\}$, so that $\mathcal{V} \subset \bar{\mathcal{V}}$ and $\mathcal{T} \subset \bar{\mathcal{T}}$. The extended lists will serve to introduce labels for virtual agents and virtual targets that can help balance the agent/target lists. An assignment map at time $t \geq 0$ is defined as $\alpha_t : \bar{\mathcal{V}} \to \bar{\mathcal{T}}$ such that $\alpha_t(i) \equiv \alpha_i(t) = j$ if and only if target $j$ has been assigned to agent $i$ at time $t$. With a slight abuse of notation, we will drop the subindex $t$ from $\alpha_t$ and denote $\alpha = (\alpha_i)$ to represent a generic bijection between $\bar{\mathcal{V}}$ and $\bar{\mathcal{T}}$ at a given time. The problem that we would like to solve is how to match the $n$ agents to the $m$ targets in a distributed way. In other words, we would like to find a bijection $\alpha_\tau : \bar{\mathcal{V}} \to \bar{\mathcal{T}}$, $\alpha_\tau \equiv (\alpha_i(\tau))$ at some terminal time $\tau$, so that the total assignment benefit $\sum_{i=1}^{\min\{m,n\}} \beta_{i\alpha_i(\tau)}$ is as close as possible to the optimal benefit $A^* = \sup_{\{\alpha \,|\, \alpha \text{ bijection over } \bar{\mathcal{V}}\}} \sum_{i=1}^{\min\{m,n\}} \beta_{i\alpha_i}$, where $i \in \mathcal{V}$ and $\alpha_i \in \mathcal{T}$. To solve this problem we will design a distributed algorithm that, on the one hand, estimates $\beta_{ij}$, with $j \in \mathcal{T}$, and, on the other hand, assigns agents to targets in an estimated optimal way. Our algorithm will make use of Q-Learning and distributed auction algorithms, as we describe in the following sections.

## III. Preliminaries

This section summarizes the main elements of the Q-Learning method and auction algorithms. For additional information on Q-Learning see [10]. Our exposition on the auction algorithm follows [2] and [15].

### A. Q-Learning

The main ingredients of the Q-learning algorithm are described next. We define the so-called *value function*, $V_\gamma^\sigma : \mathcal{S} \to \mathbb{R}$, as $V_\gamma^\sigma(s_0) = \sum_{k=0}^{\infty} \gamma^k r(s_k, a_k)$, where $\gamma$ is a discount factor, $0 \leq \gamma \leq 1$, and $\sigma$ is the policy, or the sequence of actions $\{a_k\}_{k \geq 0}$, that defines the state sequence $\{s_k\}_{k \geq 1}$ starting from $s_0$. In other words, $V_\gamma^\sigma(s_0)$ is the sum of future rewards over the infinite future starting at state $s_0$ and following the policy $\sigma$. Then, the *optimal value function* is given by $V_\gamma^*(s_0) = \sup_\sigma V_\gamma^\sigma(s_0)$. Denote by $\delta : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, the *state transition function*, which defines the state that is reached if action $a$ is taken at state $s$. Now define the *evaluation function*, $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, as:

$$Q(s, a) = r(s, a) + \gamma V_\gamma^*(\delta(s, a)). \tag{1}$$

The second term is the value discounted by $\gamma$ if the optimal policy is pursued beginning at state $\delta(s, a)$.

Combining (1) with the following property (see [10]):

$$V_\gamma^*(s) = \max_{a\in\mathcal{A}(s)} Q(s,a),$$

one can obtain the following equation:

$$Q(s,a) = r(s,a) + \gamma \max_{a'\in\mathcal{A}(\delta(s,a))} Q(\delta(s,a),a').$$

This equation is used to define the Q-Learning recursion to update a Q-Table with entries $\hat{Q}(s,a)$ as follows:

$$\hat{Q}(s,a) \leftarrow \hat{r}(s,a) + \gamma \max_{a'\in\mathcal{A}(s')} \hat{Q}(s',a').$$

Here, $s'$ is the new state that is observed after executing action $a$ at $s$ and computing the immediate reward $\hat{r}(s,a)$.

For the discounted rewards case, it is shown that the Q-Table values; that is, $\{\hat{Q}(s,a)\,|\,(s,a)\in\mathcal{S}\times\mathcal{A}\}$, converge to the true optimal values $Q^*(s,a) = \sup_{a\in\mathcal{A}(s)} Q(s,a)$ for every $s$ and $a$ provided that all state-action pairs are visited infinitely often and that $\hat{r}(s,a) = r(s,a)$, $\forall\,(s,a)\in\mathcal{S}\times\mathcal{A}$.

For notational simplicity, in the rest of this paper we will identify $\hat{Q}(s,a)$ and $\hat{r}(s,a)$ with $Q(s,a)$ and $r(s,a)$, respectively, since we will just make use of these estimates.

### B. The auction algorithm

The auction algorithm aims to solve the classical assignment problem where $n$ agents are matched to $n$ objects to maximize the total assignment benefit. Initially, the algorithm requires the prices, assignments, if any, and benefits for each agent-object pair. Then, the algorithm regulates a pricing mechanism in which agents can bid for their desired targets.

More precisely, in the distributed auction algorithm [15], suppose agent $i$ believes that matching with object $j$ has the price $p_{ij}(t)$. This is the price that agent $i$ thinks it needs to pay in order to be assigned to object $j$. The agent $i$ also receives pricing and bidding information for each object $j$ from its neighbors $k\in\mathcal{N}_i(t)$ at time $t$. Then the agent updates its own information for each object $j$ at every time step $t$ as follows:

$$p'_{ij}(t) := \max_{k\in\mathcal{N}_i(t)} \{p_{ij}(t), p_{kj}(t)\},$$
$$b'_{ij}(t) := \max_{k\in\operatorname{argmax}_{z\in\mathcal{N}_i(t)}\{p_{ij}(t),p_{zj}(t)\}} \{b_{kj}(t)\};$$

where $b_{ij}$ is the largest index bidder for object $j$ that is known to the agent $i$. If $\alpha_i(t)$ is the object that agent $i$ chooses at time $t$, ($\alpha_i(0)$ can take a random value), and $p_{i\alpha_i(t)}(t) \le p'_{i\alpha_i(t)}(t)$ or $b'_{i\alpha_i(t)}(t) \neq i$, then agent $i$ will update its assignment by $\alpha_i(t+1) \in \operatorname{argmax}_{k\in\{1,\ldots,m\}}\{\beta_{ik} - p'_{ik}(t)\}$, where $\beta_{ik}$ is the assignment benefit that agent $i$ would obtain by associating itself with object $k$. Then agent $i$ will set itself as the highest index bidder for object $\alpha_i(t+1)$, and increase the price of this object: $p_{i\alpha_i(t+1)}(t+1) = p'_{i\alpha_i(t+1)}(t) + \gamma_i(t)$. Here $\gamma_i(t)$ is given as follows:

$$\gamma_i(t) \triangleq v_i(t) - w_i(t) + \epsilon,$$
$$v_i(t) \triangleq \max_{j\in\{1,\ldots,m\}} \{\beta_{ij} - p'_{ij}(t)\},$$
$$w_i(t) \triangleq \max_{j\neq\alpha_i(t+1)} \{\beta_{ij} - p'_{ij}(t)\};$$

where $\epsilon$ is a small positive constant and $v_i(t)$ and $w_i(t)$ are the best and second best net values, respectively, available to agent $i$ at time $t$.

Since agent $i$ does not directly communicate with all other agents at each time step, the real price for agent $j$ could be outdated and actually much higher than $p_{ij}(t)$. However, under the assumption that the multi-agent network is jointly connected over time sufficiently often, the correct pricing information will be ultimately available to each agent. The net value of object $j$ for agent $i$ is $\beta_{ij} - p_{ij}(t)$. Each agent $i$ would like to be assigned to the object $j$ with maximal net value:

$$\beta_{ij} - p_{ij}(t) = \max_{k\in\{1,\ldots,m\}} \{\beta_{ik} - p_{ik}(t)\}. \tag{2}$$

If (2) is satisfied for all agents, then the assignment and the set of prices are at *equilibrium*. However, it can happen that several agents try to be assigned to a smaller number of equally desirable objects without raising the prices of the objects. This will lead to cycles in the algorithm preventing convergence to an equilibrium. Thus, we will consider an almost equilibrium assignment which occurs when

$$\beta_{ij} - p_{ij}(t) \ge \max_{k\in\{1,\ldots,m\}} \{\beta_{ik} - p_{ik}(t)\} - \epsilon,$$

for all $i\in\{1,\ldots,n\}$ and for a given small $\epsilon > 0$. This is known as $\epsilon$-*Complementary Slackness* ($\epsilon-$CS) condition.

## IV. Assignment algorithms

In this section, we present the JOLEAS (Joint Learning and Assignment) and AJOLEAS (Alternative Joint Learning and Assignment) algorithms.

### A. JOLEAS Algorithm

The JOLEAS algorithm is a synchronous algorithm executed by multiple agents, who communicate locally upon encountering each other in the environment.

The simplest case is that of $n$ agents and $m$ targets with $n = m$. Through the JOLEAS, every agent $i$ stores information about targets $j$ it finds and how to reach them in Q-Tables, $Q_{ij}$. The target-assignment decision making is based on a distributed auction process. Thus, when two agents find each other, they will interchange the information about the targets they found, the prices of known targets, highest bidder information, leader information, the lowest assignment benefit they have and available agents. This allows each agent to recompute the assignment benefits, update the target prices and establish an assignment.

Notice that, the real targets can be labeled by using their locations on the grid. Then, upon encountering a new real target, say $k$, agent $i$ will store the temporary Q-Table entries of the path leading to that target in table $Q_{ik}$. This can be accomplished by assigning zero rewards to all the edges that lead to the targets other than $k$. Observe that, if the agent has not found any real targets yet, it will remain assigned to the virtual target.

To deal with case of $m < n$, a virtual target idea is introduced. In this way, $n - m$ agents will be assigned

to virtual targets and the algorithm will terminate with a "virtual" one-to-one assignment. Each agent will need to store price and highest bidder information, updated using the same procedure used as for real targets. In fact, our algorithm starts by assuming that only virtual targets are available, and these are replaced when real targets are found. Each agent updates the virtual targets' assignment benefits by making them equal to a large negative multiple of the largest absolute value of assignment benefits among its neighbors. As a result, the replacement of virtual targets by real ones is guaranteed, since all the virtual target benefits will eventually be smaller than any real target benefit. Agents keep track of the replacements by updating lists of virtual (respectively real) targets, $\mathcal{T}_d^i(t)$ (respectively $\mathcal{F}^i(t)$).

To deal with the case $m > n$, a leader agent generates virtual $m - n$ agents and places bids on their behalf. These virtual agents are generated as extra targets are found. Similarly as before, the leader will adjust the virtual agents' assignment benefits by making them equal to a large negative multiple of the largest absolute value of assignment benefits among its neighbors. In this way, we can guarantee the algorithm will find a suboptimal assignment that favors the matching of real agents with real targets and the $\epsilon - \mathrm{CS}$ condition will be satisfied.

Finally, to make the algorithm robust with respect to the leader, a leader-election routine is introduced to check from time to time that the leader is active.

In what follows, we describe the algorithm in a more formal way by making use of a pseudocode language.

**Algorithm Requirements:**

1 Each agent should know the total number of agents in the group; i.e., $n$.
2 Agents possess a unique identifier UID $\in \{1, \dots, n\}$, which they know.
3 Agents can sense the edges adjacent to the state they are located at. They can communicate with others in $\mathcal{N}_i(t)$, and can locally move to adjacent graph states.
4 We assume there is a global time schedule that synchronizes the motion, communication and actions of the agents at every time $t \in \{0, 1, 2, \dots\}$.

**Initialization:**

- Introduce $n$ virtual targets labeled by $d_j \in \mathcal{T}_d^i(0) = \{d_1, \dots, d_n\}$. Here, $\mathcal{T}_d^i(t)$ is a set of virtual targets that agent $i$ updates at every $t$. Set $\beta_{id_j}(0)$ to arbitrary negative number for all $i \in \{1, \dots, n\}$ and $d_j \in \mathcal{T}_d^i(0)$. The labels of the virtual targets should be known to all agents. Moreover, denote by $\mathcal{F}^i(t)$ the set of the real targets found by agent $i$ by the time $t$. We also define the list of targets for each agent $i$ as $\mathcal{T}^i(t) := \mathcal{T}_d^i(t) \cup \mathcal{F}^i(t)$.
- Introduce a list $\mathcal{L}_i(0) = \{1, \dots, n\}$ for all $i \in \{1, \dots, n\}$. Here, $\mathcal{L}_i(t)$ is the list of the available agents known to each agent $i$ at time $t$.
- Arbitrarily assign each agent to one of the virtual targets; e.g., $\alpha_i(0) = d_i$ for all $i \in \{1, \dots, n\}$.
- Set the temporary Q-Table for each agent, $Q_i(s, a)(0)$, equal to zero.
- Set $\mathrm{lead}_i(0) = 1$, $\forall i \in \{1, \dots, n\}$. Here, $\mathrm{lead}_i(t)$ is the UID of the agent that $i$ considers to be the leader at $t$. The set $\Lambda_i(t)$ is generated by the leader agent $i$ at time $t$ and consists of labels for virtual agents.

**Execution for Agent $i$ at time $t + 1 \geq 1$:**

1 Process messages $\mathcal{T}^k(t)$, $\mathcal{L}_k(t)$, $p_{kj}(t)$, $b_{kj}(t)$, $B_k(t) \triangleq \min_{j \in \mathcal{T}^k(t)} \beta_{kj}(t)$ and $\mathrm{lead}_k(t)$, with $j \in \mathcal{T}^k(t)$, received from $k \in \mathcal{N}_i(t)$.
2 Delete from $\mathcal{T}^i(t)$ and $\mathcal{L}_i(t)$, the virtual targets which are not in $\{\mathcal{T}^k(t)\}_{k \in \mathcal{N}_i(t)}$, and the real agents which are not in $\{\mathcal{L}_k(t)\}_{k \in \mathcal{N}_i(t)}$, respectively.
3 Update the virtual target assignment benefits by:

$$\beta_{id_r}(t+1) = -C \max_{k \in \mathcal{N}_i(t)} \{|B_k(t)|, |B_i(t)|\}|,$$

with a constant $C \gg 1$, for all $d_r \in \mathcal{T}_d^i(t)$.
4 **if** $\mathrm{lead}_i(t) = i$, **then:**
5 Update virtual agent assignment benefits by:

$$\beta_{\vartheta_l j}(t+1) = \beta_{\vartheta_l d_r}(t+1) = -C \max_{k \in \mathcal{N}_i(t)} \{|B_k(t)|, |B_i(t)|\},$$

for all $\vartheta_l \in \Lambda_i(t)$, $j \in \mathcal{F}^i(t)$ and $d_r \in \mathcal{T}_d^i(t)$.
6 **if** $|\mathcal{F}^i(t)| - |\mathcal{L}_i(t)| - |\Lambda_i(t)| > 0$, **then:**
7 Generate new virtual agents $\vartheta_l \in \Lambda_i(t+1)$, with $\beta_{\vartheta_l j}(t+1) = \beta_{\vartheta_l d_r}(t+1) = -C \max_{k \in \mathcal{N}_i(t)} \{|B_k(t)|, |B_i(t)|\}$, for all $l \in \{|\Lambda_i(t)| + 1, \dots, |\mathcal{F}^i(t)| - |\mathcal{L}_i(t)|\}$, $j \in \mathcal{F}^i(t)$ and $d_r \in \mathcal{T}_d^i(t)$, and carry out the bidding process on their behalf.
8 **end if.**
9 **end if.**
10 Choose $a \in \mathcal{A}(s)$ and move to $s' = \delta(s, a)$.
11 **if** the real target $k$ is newly found, **then:**
12 update $\mathcal{T}^i(t)$ to $\mathcal{T}^i(t+1)$ by deleting the virtual target $d_k$ from $\mathcal{T}_d^i(t)$ and adding the real target to $\mathcal{F}^i(t+1)$.
13 reset $r(s, a) = 0$ if $s' \in \mathcal{S}_{\mathcal{F}^i(t)}$, where $\mathcal{S}_{\mathcal{F}^i(t)} := \bigcup_j s_{t_j}$ with $j \in \mathcal{F}^i(t)$.
14 Recalculate the $Q_i(s, a)(t)$ table as

$$Q_i(s, a)(t) = r(s, a) + \gamma \max_{a' \in \mathcal{A}(s')} Q_i(s', a')(t). \quad (3)$$

and store it in $Q_{ik}(s, a)(t) = Q_i(s, a)(t)$.
15 Calculate the assignment benefit:

$$\beta_{ik}(t) = \max_{a \in \mathcal{A}(s_{i0})} Q_{ik}(s_{i0}, a)(t). \quad (4)$$

16 **else:**
17 for each target $l \in \mathcal{F}^i(t)$, reset $r(s, a) = 0$ if $s' \in \mathcal{S}_{\mathcal{F}^i(t)} \setminus \{s_{t_l}\}$.
18 Recalculate the $Q_i(s, a)(t)$ table as in (3), and store it in $Q_{il}(s, a)(t) = Q_i(s, a)(t)$.
19 Calculate the assignment benefit:

$$\beta_{il}(t) = \max_{a \in \mathcal{A}(s_{i0})} Q_{il}(s_{i0}, a)(t).$$

20 **end if.**
21 **if** $\alpha_i(t) \notin \mathcal{T}^i(t)$, **then:**
22 $\alpha_i(t) = \emptyset$[1].

[1]In other words, remain unassigned

23 **end if**.

24 Update the prices of all targets $j \in \mathcal{T}^i(t)$ and the corresponding highest bidders by:

$$p'_{ij}(t) := \max_{k \in \mathcal{N}_i(t)} \{p_{ij}(t), p_{kj}(t)\},$$
$$b'_{ij}(t) := \max_{k \in \text{argmax}_{z \in \mathcal{N}_i(t)} \{p_{ij}(t), p_{zj}(t)\}} \{b_{kj}(t)\}.$$

25 **if** $\alpha_i(t) = \emptyset$, or $p_{i\alpha_i(t)}(t) \leq p'_{i\alpha_i(t)}(t)$, or $b'_{i\alpha_i(t)}(t) \neq i$ or agent $i$ does not satisfy the $\epsilon-$CS condition:

$$\beta_{ij}(t) - p_{ij}(t) \geq \max_{k \in \mathcal{T}^i(t)} \{\beta_{ik}(t) - p_{ik}(t)\} - \epsilon,$$

**then:**

26 Update the assignment by

$$\alpha_i(t+1) \in \text{argmax}_{k \in \mathcal{T}^i(t)} \{\beta_{ik}(t) - p'_{ik}(t)\}.$$

27 Set $b_{i\alpha_i(t+1)}(t+1) := i$ and raise the price for target $\alpha_i(t+1)$ to $p_{i\alpha_i(t+1)}(t+1) := p'_{i\alpha_i(t+1)}(t) + \gamma_i(t)$, where

$$\gamma_i(t) \triangleq v_i(t) - w_i(t) + \epsilon_1, \quad (5)$$
$$v_i(t) \triangleq \max_{j \in \mathcal{T}^i(t)} \{\beta_{ij}(t) - p'_{ij}(t)\}, \quad (6)$$
$$w_i(t) \triangleq \max_{j \in \mathcal{T}^i(t) \backslash \{\alpha_i(t+1)\}} \{\beta_{ij}(t) - p'_{ij}(t)\}. \quad (7)$$

$v_i$ and $w_i$ are the best and the second best net values available to agent $i$, respectively, and $\epsilon > \epsilon_1 > 0$ is a minimum bid increment.

28 **else** $\alpha_i(t+1) := \alpha_i(t)$.

29 **end if**.

30 **if** received the information that the agent with UID $= b$ is no longer active, **then:**

31 Update the list of the available agents by:

$$\mathcal{L}_i(t+1) = \mathcal{L}_i(t) \backslash \{b\}.$$

32 **end if**.

33 **if** received the information that the leader (suppose with UID $= a$) is no longer active, **then:**

34 Update the leader information by:

$$\text{lead}_i(t) = a + 1.$$

35 **end if**.

36 Update the leader information by:

$$\text{lead}_i(t+1) = \max_{k \in \{i\} \cup \mathcal{N}_i(t)} \text{lead}_k(t).$$

37 **if** $\text{lead}_i(t+1) = i$, **then:**

38 Announce itself as the leader.

39 **end if**.

40 Communicate $\mathcal{T}^i(t+1)$, $\mathcal{L}_i(t+1)$, $p_{ij}(t+1)$, $b_{ij}(t+1)$, $B_i(t)$ and $\text{lead}_i(t+1)$, with $j \in \mathcal{T}^i(t+1)$, to all agents $k$ such that $k \in \mathcal{N}_i(t)$.

*Remark 1:* The main difference between the auction algorithm part in the description above (Steps 24-29) and the distributed auction algorithm presented in [15] is the following. Here, we need to introduce $\epsilon \neq \epsilon_1$ to deal with the time-varying $\beta_{ij}(t)$ due to the learning process. In this way the $\epsilon-$CS condition can be satisfied. Another consequence of the time-varying bids is that, in addition to checking for the highest-bidder and price increase of the assigned target, every agent should also check whether the current assignment still satisfies the $\epsilon-$CS condition. Moreover, the real agents that become unassigned due to the deletion of the virtual targets should also perform assignment update given at step 26. Furthermore, the JOLEAS algorithm can deal with the problem of $n \neq m$. All in all, our algorithm extends the distributed auction algorithm presented in [15] in that we make it robust to agent failure, and capable of dealing with unknown environments. •

As it will be clear in the next section, we need the following conditions to guarantee that the algorithm terminates with a near optimal assignment:

*Assumption 1:* The agents must communicate with each other over infinitely many time indices $t$.

*Assumption 2:* All agents will visit each state-action pair infinitely often.

*Remark 2:* Notice that, due to the introduction of the virtual targets and agents, we can guarantee that there exists at least one feasible (one-to-one) assignment. •

Observe that, due to the nature of the Q-Learning iteration, the JOLEAS algorithm is greedy; i.e. only the actions that resulted in maximizing $Q(s,a)$ values in early stages will be favored. Therefore, it may not be possible to satisfy Assumption 2. This can be resolved by executing a random action with probability $\mu$ and executing the greedy action with probability $1-\mu$, where $\mu$ is a small positive parameter [6].

### *B. Alternative JOLEAS (AJOLEAS) Algorithm*

As an alternative algorithm we can make the following changes to the JOLEAS algorithm. Steps 6-8 can be replaced by the following steps:

1 **if** $|\mathcal{F}^i(t)| > 0$ and $|\Lambda_i(t)| < |\mathcal{F}^i(t)| < |\mathcal{L}_i(t)|$, **then:**

2 Generate virtual agents $\vartheta_l \in \Lambda_i(t+1)$ with $\beta_{\vartheta_l j}(t+1) = \beta_{\vartheta_l d_r}(t+1) = -C \max_{k \in \mathcal{N}_i(t)} \{|B_k(t)|, |B_i(t)|\}$, for all $l \in \{|\Lambda_i(t)|+1, \ldots, |\mathcal{F}^i(t)|\}$, $j \in \mathcal{F}^i(t)$, $d_r \in \mathcal{T}^i_d(t)$, and carry out the following bidding process on their behalf.

3 **else if** $|\mathcal{F}^i(t)| - |\mathcal{L}_i(t)| - |\Lambda_i(t)| > 0$, **then:**

4 Generate virtual agents $\vartheta_l \in \Lambda_i(t+1)$ with $\beta_{\vartheta_l j}(t+1) = \beta_{\vartheta_l d_r}(t+1) = -C \max_{k \in \mathcal{N}_i(t)} \{|B_k(t)|, |B_i(t)|\}$, for all $l \in \{|\Lambda_i(t)|+1, \ldots, |\mathcal{F}^i(t)| - |\mathcal{L}_i(t)|\}$, $j \in \mathcal{F}^i(t)$, $d_r \in \mathcal{T}^i_d(t)$, and carry out the following bidding process on their behalf.

5 **end if**.

In this way, the leader generates virtual agents as soon as it finds a new real target. Therefore, there is no need for the deletion of the virtual targets at the next steps of the JOLEAS algorithm. Using this algorithm, upon termination the number of both targets and agents, as well as the number of assignments will be $m + \min_{k \in \{1,\ldots,n\}, t} \{|\mathcal{L}_k(t)|\}$.

Let $q_1 = \max\{m,n\}$, $q_2 = m+n$ and $q = \min\{m,n\}$. Then, we have:

*Proposition 3:* (*Convergence Results for the JOLEAS algorithm*). Suppose that Assumptions 1 and 2 hold. Then there

exists a finite time $T$ such that $\alpha_i(t) = \bar{\alpha}_i$ and $\beta_{i\alpha_i}(t) = \bar{\beta}_{i\bar{\alpha}_i}$ for all $i \in \{1, \ldots, q_1\}$ and for all $t \geq T$. Furthermore, the final assignment obtained from the Joint Learning and Assignment algorithm is

(i) within $q_1\epsilon_1$ of the optimal assignment; i.e. $A^* - q_1\epsilon_1 \leq \sum_{i=1}^{q} \bar{\beta}_{i\bar{\alpha}_i} \leq A^*$, for undiscounted rewards case;
(ii) within $q_1\epsilon_2$ of the optimal assignment; i.e. $A^* - q_1\epsilon_2 \leq \sum_{i=1}^{q} \bar{\beta}_{i\bar{\alpha}_i} \leq A^*$, for discounted rewards case provided that $\epsilon_2 > \epsilon_1$.

*Proposition 4:* (*Convergence Results for the AJOLEAS algorithm*). Suppose that Assumptions 1 and 2 hold. Then there exists a finite time $T$ such that $\alpha_i(t) = \bar{\alpha}_i$ and $\beta_{i\alpha_i}(t) = \bar{\beta}_{i\bar{\alpha}_i}$ for all $i \in \{1, \ldots, q_2\}$ and for all $t \geq T$. Furthermore, the final assignment obtained from the Alternative Joint Learning and Assignment algorithm is

(i) within $q_2\epsilon_1$ of the optimal assignment; i.e. $A^* - q_2\epsilon_1 \leq \sum_{i=1}^{q} \bar{\beta}_{i\bar{\alpha}_i} \leq A^*$, for undiscounted rewards case;
(ii) within $q_2\epsilon_2$ of the optimal assignment; i.e. $A^* - q_2\epsilon_2 \leq \sum_{i=1}^{q} \bar{\beta}_{i\bar{\alpha}_i} \leq A^*$, for discounted rewards case provided that $\epsilon_2 > \epsilon_1$.

## V. Convergence analysis

We include here the convergence proofs of Proposition 3 and 4 of the previous section.

### A. Undiscounted Rewards Case

In this subsection, we present the claims and corresponding proofs for the undiscounted rewards case. In order to prove the convergence of the entire algorithm, first we need to show the convergence of the assignment benefits, since due to the learning action they are changing with time. We will follow the ideas of [2] to prove some of the claims.

*Claim 1.1:* There exists a finite time $T_1$ such that $\beta_{ij}(t) = \bar{\beta}_{ij}$ for all $i, j \in \{1, \ldots, q_1\}$ and $\forall t \geq T_1$, for some $\bar{\beta}_{ij} \in \mathbb{R}$.

*Proof:* Suppose first that $m = n$. Define the error of $Q(s,a)(t)$ update by $e(s,a)(t) = Q^*(s,a) - Q(s,a)(t)$, where $Q^*(s,a)$ is the true value associated with $(s,a)$, and $Q(s,a)(t)$ is the Q-Table updated at time $t$ by any agent using the Q-Learning recursion. For simplicity, assume that we initialize all Q-Table entries to zero. We also consider that $Q_i(s_{t_j}, a)(t) = 0$, $\forall a \in \mathcal{A}(s_{t_j})$ and $\forall t \geq 0$, since $s_{t_j}$ are absorbing states for all $j$.

We will make use of induction over the state-action pairs that lead to the target to prove the claim. First, suppose that action $a_l \equiv a_1$ executed at state $s_l \equiv s_1$ leads to state $s_{t_j}$. Then, we obtain for all time $t \geq 0$:

$$Q(s_1, a_1)(t) = r(s_1, a_1) + \max_{a' \in \mathcal{A}(s_{t_j})} Q(s_{t_j}, a')(t)$$
$$= r(s_1, a_1) = Q^*(s_1, a_1) \quad \Longrightarrow \quad e(s_1, a_1)(t) = 0.$$

Now by induction, suppose $e(s_k, a_k)(t_a) = 0$ for some finite time $t_a$. In particular, this implies that $Q(s_k, a_k)(t) = Q^*(s_k, a_k)$ for all $t \geq t_a$. By Assumption 2 we know that there exists a time $t_b > t_a$ such that at time $t_b$ action $a_{k+1}$ executed at state $s_{k+1}$ will lead to state $s_k$. Suppose also that $Q(s_k, a_k)(t_b) = \max_{a' \in \mathcal{A}(s_k)} Q(s_k, a')(t_b)$. Then:

$$Q(s_{k+1}, a_{k+1})(t_b)$$
$$= r(s_{k+1}, a_{k+1}) + \max_{a' \in \mathcal{A}(s_k)} Q(s_k, a')(t_b)$$
$$= r(s_{k+1}, a_{k+1}) + Q^*(s_k, a_k) = Q^*(s_{k+1}, a_{k+1})(t_b)$$
$$\Longrightarrow e(s_{k+1}, a_{k+1})(t_b) = 0.$$

Thus, by induction on $(s_k, a_k)$ after finite number of iterations, $e(s_k, a_k)(t) = 0$ for all $k$ and $\forall t > T_1$, where $T_1$ is some finite time. Since $\beta_{ij}(t)$ is calculated by (4), after finite number of iterations it will converge to $\bar{\beta}_{ij}$, where $\bar{\beta}_{ij} = \max_{a \in \mathcal{A}(s_{i0})} Q^*_{ij}(s_{i0}, a)$, for all $i$ and $j$.

Now suppose that $m \neq n$ (virtual targets and/or agents). Convergence of the virtual agents' and virtual targets' benefits also holds since 1) they are set to be proportional to some real agent benefits and 2) these values will reach every agent in the network because of Assumption 1. Also notice that, all the benefits associated with the virtual entities will converge to the same value, say $\tilde{\beta}$. ■

*Claim 1.2:* The JOLEAS algorithm will terminate with a feasible assignment.

*Proof:* The proof for the (centralized) auction algorithm in [5], can be used in this context. We reproduce it here for the sake of completeness. Suppose that the algorithm does not terminate. Then, there are targets (resp. agents) that receive (resp. make) an infinite number of bids. Define by $\mathcal{T}^\infty \subseteq \mathcal{T}$ (resp. $\mathcal{V}^\infty \subseteq \mathcal{V}$) the set of the targets that receive (resp. that make) an infinite number of bids. Since each bid increases the price of a target by $\epsilon_1$, the prices of targets $j \in \mathcal{T}^\infty$ are increased to infinity. It is clear from (**??**) that the best net value available to agent $i \in \mathcal{V}^\infty$; i.e., $v_i$, will decrease to negative infinity.

For the $\epsilon - \mathrm{CS}$ condition to hold it is obvious that target $j \in \mathcal{T}^\infty$ can only be assigned to agent $i \in \mathcal{V}^\infty$. However, at least one agent in $\mathcal{V}^\infty$ will be unassigned at the beginning of each auction iteration after finite time, since the algorithm does not terminate. Thus, since the number of agents in $\mathcal{V}^\infty$ will be more than the number of targets in $\mathcal{T}^\infty$, there does not exist a feasible assignment. This contradicts the statement in Remark 2. Thus, auction iteration part of the JOLEAS algorithm must terminate in finite time. Combining this result with Claim 1.1, we conclude that the JOLEAS algorithm will terminate with a feasible assignment. ■

Note that, since the algorithm terminates in finite time, and communication with connected graphs is enabled, each agent belief of the target prices will eventually converge to the same fixed values; i.e., $p_{ij}(t) \to p_j$ for all $i, j \in \{1, \ldots, q_1\}$ for all $t > T_2 > T_1$, for some time $T_2 < +\infty$. Now we prove that under Claim 1.1, after time $T_1$ the algorithm will eventually converge to a feasible assignment.

*Claim 1.3:* There exists a finite time $T \geq T_1$ such that $\alpha_i(t) = \bar{\alpha}_i$ for all $i \in \{1, \ldots, q_1\}$ and $t \geq T$.

*Proof:* Suppose that target $j_i \equiv j$ is assigned to agent $i$ after the assignment phase at time $\tau_{ij_i} \equiv \tau \geq T_1$; i.e.,

$\alpha_i(\tau) = j^*$. Then:

$$\begin{aligned} p_{ij^*}(\tau+1) &= p'_{ij^*}(\tau) + v_i(\tau) - w_i(\tau) + \epsilon_1 \\ &= \beta_{ij^*}(\tau) - w_i(\tau) + \epsilon_1. \end{aligned} \tag{8}$$

Using this relationship, Claim 1.1 and (7), we obtain:

$$\begin{aligned} &\beta_{ij^*}(\tau+1) - p_{ij^*}(\tau+1) = \beta_{ij^*}(\tau) - p_{ij^*}(\tau+1) \\ &= \max_{j \neq j^*} \{\beta_{ij}(\tau) - p_{ij}(\tau)\} - \epsilon_1. \end{aligned}$$

Now considering the fact that the prices are nondecreasing, we get the following relationship:

$$\begin{aligned} &\beta_{ij^*}(\tau+1) - p_{ij^*}(\tau+1) \\ &\geq \max_{k \in \{1,\ldots,q_1\}} \{\beta_{ik}(\tau+1) - p_{ik}(\tau+1)\} - \epsilon_1. \end{aligned}$$

Thus, we have shown that $\epsilon - \text{CS}$ condition is satisfied with $\epsilon = \epsilon_1$ for agent $i$ and target $j_i$ for all $t \geq \tau_{ij_i}$. Consequently, there exists finite time $\tau^* = \max_i \tau_{ij_i}$ such that all agent-target pairs will satisfy $\epsilon - \text{CS}$ condition $\forall t \geq \tau^*$.

Moreover, it is clear from the algorithm that if all the agents and targets satisfy $\epsilon - \text{CS}$ condition $\forall t \geq \tau^*$, then agents have no incentive to bid for other targets different from the ones they are assigned to. Thus, choosing $T = \tau^*$ completes the proof. ■

Now we present the result on the optimality of the final assignment; i.e., the assignment obtained at time $T$.

*Claim 1.4:* The final total benefit of the assignment obtained from the JOLEAS algorithm is within $q_1\epsilon_1$ of the optimal one.

*Proof:* Recall that the final assignment to agent $i$ is denoted by $\bar{\alpha}_i$ and $\bar{\beta}_{ij}$ denotes the assignment benefit of $i$ if it was matched with $j$ at the termination time. The total assignment benefit (including the benefits of the virtual entities) is given as:

$$\begin{aligned} \sum_{i=1}^{q_1} \bar{\beta}_{i\bar{\alpha}_i} &= \sum_{i=1}^{q_1} \bar{\beta}_{i\bar{\alpha}_i} + \sum_{j=1}^{q_1} p_j - \sum_{j=1}^{q_1} p_j \\ &\leq \sum_{j=1}^{q_1} p_j + \sum_{i=1}^{q_1} \max_{k \in \{1,\ldots,q_1\}} \{\bar{\beta}_{ik} - p_k\}. \end{aligned} \tag{9}$$

Recall that the optimal assignment benefit is defined by:

$$A^* = \max_{(\alpha_i) \,|\, (\alpha_i) \text{ bijection}} \sum_{i=1}^{q} \bar{\beta}_{i\alpha_i},$$

where $i \in \mathcal{V}$ and $\alpha_i \in \mathcal{T}$. Now define the optimal assignment benefit which includes the benefits of the virtual entities:

$$\tilde{A} = \max_{(\alpha_i) \,|\, (\alpha_i) \text{ bijection}} \sum_{i=1}^{q_1} \bar{\beta}_{i\alpha_i}.$$

Also define the optimal dual cost (see [2]):

$$\tilde{D} \triangleq \min_{p_1,\ldots,p_{q_1}} \{\sum_{j=1}^{q_1} p_j + \sum_{i=1}^{q_1} \max_{j \in \{1,\ldots,q_1\}} \{\bar{\beta}_{ij} - p_j\}\}.$$

Then, from equation (9) one can see that $\tilde{A} \leq \tilde{D}$. From the $\epsilon - \text{CS}$ condition and the result of Claim 1.3, we have $\bar{\beta}_{i\bar{\alpha}_i} - p_{\bar{\alpha}_i} \geq \max_{j \in \{1,\ldots,q_1\}} \{\bar{\beta}_{ij} - p_j\} - \epsilon_1$. Using this together with the previous relationships we obtain:

$$\begin{aligned} \tilde{D} &\leq \sum_{i=1}^{q_1} (p_{\bar{\alpha}_i} + \max_{j \in \{1,\ldots,q_1\}} \{\bar{\beta}_{ij} - p_j\}) \\ &\leq \sum_{i=1}^{q_1} \bar{\beta}_{i\bar{\alpha}_i} + q_1\epsilon_1 \leq \tilde{A} + q_1\epsilon_1. \end{aligned}$$

Since $\tilde{A} \leq \tilde{D}$, we obtain:

$$\tilde{A} - q_1\epsilon_1 \leq \sum_{i=1}^{q_1} \bar{\beta}_{i\bar{\alpha}_i} \leq \tilde{A}.$$

Now recall that the benefits of all the virtual entities will have the same value, $\tilde{\beta}$, eventually. Subtracting $(q_1 - q)\tilde{\beta}$ from the above inequality we get:

$$A^* - q_1\epsilon_1 \leq \sum_{i=1}^{q} \bar{\beta}_{i\bar{\alpha}_i} \leq A^*.$$

Thus, the total assignment benefit obtained upon termination of the algorithm is within $q_1\epsilon_1$ of the optimal one. ■

*Remark 5:* Notice that in the cases $m < n$ (virtual targets) (resp. $n < m$ (virtual agents)) the virtual target-real agent benefits are much lower than the real target-real agent benefits (resp. real target-virtual agent benefit will be lower than the real target-real agent benefit) because of the constraints imposed on them. In this way, the algorithm will favor real target/real agent matchings. At termination time, any real agent or target assigned to a virtual entity will be actually unassigned. •

Combining proofs for Claims 1.1- 1.4, we obtain the proof of Proposition 3 for the undiscounted rewards case. The proof of Proposition 4 for the undiscounted rewards case is analogous. The only modification to the claims is to change $q_1$ to $q_2$, since the total number of final assignments are different in two algorithms. As a result, upon termination of the AJOLEAS algorithm, the total assignment benefit is within $q_2\epsilon_1$ of the optimal one.

### *B. Discounted Rewards*

Here we include the proofs for discounted rewards cases.

*Claim 2.1:* Let $\Delta$ to be any positive real number. Then there exists a time $T_\Delta < +\infty$ such that $|\beta_{ij}(\tau+1) - \beta_{ij}(\tau)| \leq \Delta$ for all $i, j \in \{1, \ldots, q_1\}$ and $\forall \tau \geq T_\Delta$.

*Proof:* The convergence of Q-Learning algorithm for the discounted rewards case is guaranteed under Assumption 2 and can be found as the proof of Theorem 13.1 in [10]. Consequently, since $\bar{\beta}_{ij} = \max_{a \in \mathcal{A}(s_{i0})} Q^*_{ij}(s_{i0}, a)$, the proof of the claim is trivial. ■

*Remark 6:* Notice that Claim 1.1 is satisfied for virtual agents, since their assignment benefits will converge to the fixed value given that Assumption 1 holds. •

*Claim 2.2:* The JOLEAS algorithm will terminate with a feasible assignment.

*Proof:* The proof follows along the same lines of the proof of Claim 1.2. ■

Under Claim 2.1 and as proven in the following, the algorithm will converge to a feasible assignment in finite time.

*Claim 2.3:* There exists a finite time $T \geq T_\Delta$ such that $\alpha_i(t) = \bar{\alpha}_i$ for all $i \in \{1, \ldots, q_1\}$ and $\forall t \geq T$.

*Proof:* Suppose that $\alpha_i(\tau) = j^*$ after the assignment phase at time $\tau \geq T_\Delta$. Then using (8) together with (7) we obtain:

$$\begin{aligned}
&\beta_{ij^*}(\tau+1) - p_{ij^*}(\tau+1) \\
&= \beta_{ij^*}(\tau) - p_{ij^*}(\tau+1) + (\beta_{ij^*}(\tau+1) - \beta_{ij^*}(\tau)) \\
&= \max_{j \neq j^*}\{\beta_{ij}(\tau) - p_{ij}(\tau)\} - \epsilon_1 + (\beta_{ij^*}(\tau+1) - \beta_{ij^*}(\tau)).
\end{aligned}$$

By Claim 2.1, and because prices are nondecreasing:

$$\beta_{ij}(\tau) - p_{ij}(\tau) \geq \beta_{ij}(\tau+1) - \Delta - p_{ij}(\tau+1),$$

which implies

$$\max_{j \neq j^*}\{\beta_{ij}(\tau) - p_{ij}(\tau)\} \geq \beta_{ij}(\tau+1) - p_{ij}(\tau+1) - \Delta,$$

for all $j \in \{1, \ldots, q_1\} \setminus \{j^*\}$. Putting this together with the above equation for $\beta_{ij^*}(\tau+1) - p_{ij^*}(\tau+1)$, we obtain:

$$\begin{aligned}
&\beta_{ij^*}(\tau+1) - p_{ij^*}(\tau+1) \geq \\
&\beta_{ij}(\tau+1) - p_{ij}(\tau+1) - \epsilon_1 - 2\Delta, \quad \forall j \in \{1, \ldots, q_1\} \setminus \{j^*\}.
\end{aligned}$$

Since the above inequality holds trivially for $j^*$, and taking $\Delta \leq \frac{\epsilon_1 - \epsilon_2}{2}$, we have that:

$$\begin{aligned}
&\beta_{ij^*}(\tau+1) - p_{ij^*}(\tau+1) \\
&\geq \max_{k \in \{1,\ldots,q_1\}}\{\beta_{ik}(\tau+1) - p_{ik}(\tau+1)\} - \epsilon_1 - 2\Delta \\
&\geq \max_{k \in \{1,\ldots,q_1\}}\{\beta_{ik}(\tau+1) - p_{ik}(\tau+1)\} - \epsilon_2.
\end{aligned}$$

Thus, we have shown that the $\epsilon - \mathrm{CS}$ condition is satisfied with $\epsilon = \epsilon_2$ for agent $i$ and target $j^*$ for all $t \geq \tau$. Repeating this for each agent, one can clearly find some finite time $\tau^*$ such that all the agent-target pairs will satisfy $\epsilon - \mathrm{CS}$ condition $\forall t \geq \tau^*$. Following the same reasoning as in Claim 1.3, all the agents will remain assigned to their targets $\forall t \geq \tau^*$. Choosing $T = \tau^*$ completes the proof. ■

*Claim 2.4:* The final total benefit of the assignment obtained from the JOLEAS algorithm is within $q_1\epsilon_2$ of the optimal total assignment benefit.

*Proof:* By using that the $\epsilon - \mathrm{CS}$ condition holds with $\epsilon = \epsilon_2$ we can follow along the same lines of the proof of Claim 1.4 to conclude the result. ■

The proofs of Claims 1.1-1.4 provide the proof of Proposition 3 (ii) for the discounted rewards case. The proof of Proposition 4 (ii) parallels the proof for Proposition 3 (ii). The only difference is that, upon termination of the AJOLEAS algorithm, the total assignment benefit is within $q_2\epsilon_2$ of the optimal one.

We can restate an observation made in [2] as a remark for both Propositions 3 and 4.

*Remark 7:* The total assignment benefit will be integer if all the benefits are integers. Consequently, if all $\bar{\beta}_{ij}$ are integers and any one of the following conditions is true:

(i) $\epsilon_1 < 1/q_1$ for the undiscounted rewards case of the JOLEAS algorithm,

(ii) $\epsilon_2 < 1/q_1$ for the discounted rewards case of the JOLEAS algorithm,

(iii) $\epsilon_1 < 1/q_2$ for the undiscounted rewards case of the AJOLEAS algorithm,

(iv) $\epsilon_2 < 1/q_2$ for the discounted rewards case of the AJOLEAS algorithm,

then the agent-target assignment upon termination of the algorithm is optimal. •

## VI. Conclusion

In this paper, we considered agent-target assignment in an unknown environment which is represented as an undirected graph and the locations of the targets are unknown to the agents. Furthermore, the number of agents and targets can be arbitrary. To deal with this challenge we proposed the Joint Learning and Assignment algorithm, as well as its alternative version, and analyzed their convergence properties. The algorithm was designed as an integration of the Q-Learning algorithm and the distributed auction algorithm presented in [15]. The algorithms can also accommodate to the addition and deletion of targets and agents including the leader.

## References


[1] M. L. Balinski. Signature methods for the assignment problem. *Journal on Operations Research*, 33:527–537, 1985.

[2] D. Bertsekas. *Linear Network Optimization: Algorithms and Codes*. The MIT Press, 1991.

[3] D. P. Bertsekas. A new algorithm for assignment problem. *Mathematical Programming*, 21(1):152–171, 1981.

[4] D. P. Bertsekas. The auction algorithm: a distributed relaxation method for the assignment problem. *Annals of Operations Research*, 14:105–123, 1988.

[5] D. P. Bertsekas. Auction algorithms for network flow problems: A tutorial introduction. *Computational Optimization and Applications*, 1:7–66, 1992.

[6] D. P. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[7] D. A. Castañón and C. Wu. Distributed algorithms for dynamic reassignment. In *IEEE Conf. on Decision and Control*, pages 13–18, Maui, HI, December 2003.

[8] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics*, 2:83–97, 1955.

[9] M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 535–542, 2000.

[10] T. M. Mitchell. *Machine learning*. McGraw-Hill, 1997.

[11] S. L. Smith and F. Bullo. Monotonic target assignment for robotic networks. *IEEE Transactions on Automatic Control*, 54(10), 2009. To appear.

[12] C. J. C. H. Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge, 1989.

[13] C. J. C. H. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8(3-4):279–292, May 1992.

[14] M. M. Zavlanos and G. J. Pappas. Dynamic assignment in distributed motion planning with local coordination. *IEEE Transactions on Robotics*, 24(1):232–242, 2008.

[15] M. M. Zavlanos, L. Spesivtsev, and G. J. Pappas. A distributed auction algorithm for the assignment problem. In *Proceedings of 47th IEEE Conference on Decision and Control*, pages 1212–1217, December 2008.