Distributed Algorithm with Resilience for Multi-Agent Task Allocation

Xuan Wang, Jeffrey Hudack, and Shaoshuai Mou

***Abstract*— This paper develops an Iterated Auction-Consensus Algorithm (IACA) for resilient task allocation on a distributed multi-agent platform. The IACA enables each agent in the network to bid for tasks of interest at the auction stage and then utilizes the idea of consensus to resolve conflicts among agents' bids. Such a combination of auction and consensus enables each agent to repeatedly add tasks one by one to its task bundle until reaching its maximum bundle length. The major advantages of the IACA are twofold. One is the simplicity of implementation without requiring complicate decision logic. The other one is resilience, when there are malicious agents trying to block the normal agents' bids by giving infinitely large bids to all tasks, IACA is able to recover immediately after the removal of malicious agents. We investigate and compare costs for communications and computations by providing quantitative measures for distributed task allocation algorithms. Simulations are given for studying the tradeoff between communication and computational costs and the resilience of IACA in the presence of malicious agents.**

I. Motivation

Distributed algorithms for task allocation has gained extensive applications including emergency building survey [1], road intersection traffic control [2], and so on. It enables a multi-agent network to autonomously assign tasks to each agent with the goal of achieving a maximum reward while resolving conflicts among task assignments. By *distributed* here is meant that each agent can only communicate with its nearby neighbors and there is no centralized coordinator as assumed in [3]–[5]. Different from many distributed optimization problems with respect to continuous variables [6]–[8], for which the gradient-descent method has played a significant role, the problem of distributed task allocation usually involves discrete variables (chosen from 0 and 1) and belongs to the category of combinatorial problems, which has been shown to be NP-hard [9]. One way to solve the problem of task allocations for multi-agent networks is by the Hungarian algorithm [10], which employs the idea of primal-dual to iteratively improve the cost function but requires the existence of an agent knowing costs of all tasks associated with all agents. Another nice idea with the capability for tackling distributed task allocation problems is by the *auction* [11], which allows each agent to bid for tasks through auctions and allow a centralized agent to resolve conflicts among these bids. To further apply the idea of auction in a fully distributed scenario, the authors of [12] have utilized a clever combination of auction and consensus for developing distributed task allocation algorithms, in which the auction allows each agent to bid for tasks of its interests and the conflicts among all agents' bids are resolved by the consensus. This **C**onsensus-**B**ased **B**undle **A**lgorithm (CBBA) has been shown to solve the task allocation problem with at least 50% optimality. Further improvement have also been achieved in [13] by removing the synchronization requirements, in [14] by considering a hierarchical network structure, in [9] by considering time-sensitive tasks, in [15] by introducing heterogeneity in agents' capability and cost of handling tasks, as well as in [16] by integrating the task allocation with local path planning algorithms. As nice as CBBA algorithms are, they all require the introduction of additional time-stamps and complicated decision logics for revolving conflicts among agents. In addition, their resilience is limited, in the sense that the infinitely large bids given by malicious agents can permanently block all other agents' from winning any task. Recognition of this has motivated us to develop an Iterated Auction-Consensus Algorithm (**IACA**), which is also based on the combination of auction [11] and distributed consensus [17], [18]. However, as we limit each agent to append at most one task to its task bundle in each round of auction-consensus, the IACA does not require time-stamps and only involve simple decision logics, require less costs for communications (for particular cases) and computations. Further more, compared with CBBA, it is more resilient in the presence of malicious agents.

This work was funded by Air Force Research Laboratory, Information Directorate, Rome, NY, USA. Xuan Wang is with the Mechanical and Aerospace Engineering, University of California, San Diego, San Diego, CA, USA (xuw010@ucsd.edu). Jeffrey Hudack is with the Air Force Research Laboratory, Rome, NY, USA, (jeffrey.hudack@us.af.mil). Shaoshuai Mou is with the School of Aeronautics and Astronautics, Purdue University, West Lafayette, IN, USA (mous@purdue.edu). His research is also partly funded by Northrop Grumman Corporation.

II. Distributed Task Allocation

Given a number of m agents labeled as $\mathcal{I}_A = \{1, 2, ..., m\}$, in which each agent i could receive information at time t from certain agents denoted by set $\mathcal{N}_i(t)$. Let $\mathbb{G}(t)$ denote a graph such that there is a directed edge (i, j) from i to j if and only if $j \in \mathcal{N}_i(t)$. In the following we always let $d_i(t) = |\mathcal{N}_i(t)|$, $\bar{m}(t)$ denote the number of edges of $\mathbb{G}(t)$ at time t, and $\Delta_{\mathbb{G}(t)}$ denote the graph diameter of $\mathbb{G}(t)$.

Consider a number of n tasks labeled as $\mathcal{I}_T = \{1, 2, ..., n\}$. If task j is assigned to agent i, the agent will receive a time-discounted reward $r_{ij}(t) \in \mathbb{R}$, depending on the time that agent i is able to finish task j. (A specific $r_{ij}(t)$ will be defined in the next section.) Let $x_{ij} \in \{0, 1\}$ be the state of agent i, for which $x_{ij} = 1$ if task j is assigned to agent i, and $x_{ij} = 0$, otherwise. The goal of *distributed task*

978-1-7281-6207-2/21/$31.00 ©2021 IEEE

2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS) | 978-1-7281-6207-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICPS49255.2021.9468154

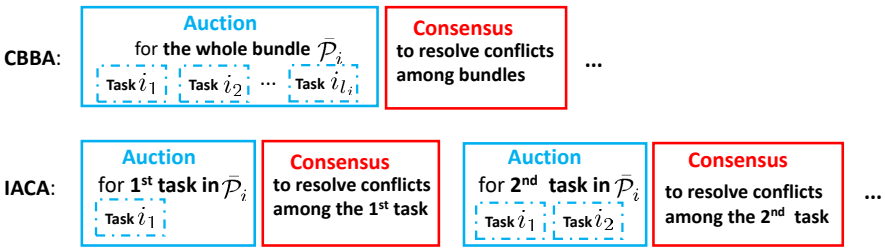

Fig. 1. Key ideas of the CBBA and IACA

allocation is to find a set of $x_{ij} \in \{0,1\}$ which

$$\text{maximize}: \quad R = \sum_{i=1}^{m}\left(\sum_{j=1}^{n} r_{ij}(t)x_{ij}\right) \tag{1}$$

$$\text{subject to}: \quad \sum_{i=1}^{m} x_{ij} \leq 1, \forall j \in \mathcal{I}_T \tag{2}$$

$$\text{and} \quad \sum_{j=1}^{n} x_{ij} \leq \ell_i, \ \forall i \in \mathcal{I}_A. \tag{3}$$

The constraint (2) needs to be satisfied since one task can only be assigned to at most one agents. The constraint (3) comes from each agent's limited capability of handling tasks, namely, each agent i can be assigned by a maximum number of ℓ_i tasks. Under distributed algorithms for task allocation, each agent i will generate a *task bundle* $\bar{\mathcal{P}}_i$, which is a list of tasks that are executed in a chronological order. All elements of $\bar{\mathcal{P}}_i$ are in $\mathcal{I}_T$ and $j \in \bar{\mathcal{P}}_i$ if and only if $x_{ij} = 1$. All task bundles needs to be *conflict-free*, namely, task bundles do not share any task in common.

To solve the above distributed task allocation problem, the authors of [12] have derived a consensus-based auction algorithm (CBAA) and its generalization consensus-based bundle algorithm (CBBA) based on a combination of auction and consensus. By first allowing each agent to bid for rewarded tasks through **auction**, the idea of **consensus** is introduced to avoid one task being assigned to more than one agents and each task is always assigned to an agent with the highest reward. As shown in Fig. 1, each agent i in CBBA first generates a candidate bundle of the size ℓ_i in the auction stage, which usually has conflicts with other bundles in terms that their bundles may share one common task called a *conflict task*. Each conflict task is then assigned to an agent with the highest reward in a later consensus stage. By repeatedly auction and consensus, all agents bundles become conflict free in a finite number of steps. Note that when agents' bundles have conflicts, which is usually the case when the bundle size is large, each conflict task can only be won by only one agent. Thus, all the other agents need to release the conflict task and all the tasks in their bundles that is added after this conflict one. Such task releasing require not only time-stamps to each information in the network but also introduction of complicated logics. Recognition of this has motivated **the problem of interest** in this paper: developing a distributed algorithm for each agent i to generate a task bundle $\bar{\mathcal{P}}_i$ for a given group of tasks $\mathcal{I}_T$, which does not involve any time-stamps or complicated logics for resolving conflicts.

III. Iterated Auction-Consensus Algorithm (IACA)

In this section, we present an Iterated Auction-Consensus Algorithm (IACA) for each agent i to achieve a conflict-free task bundle $\bar{\mathcal{P}}_i$. The key idea of IACA as shown in Fig. 1, is by a different combination of auction and consensus, in which each agent aims to augment its bundle by at most one task at one iteration of auction-consents. The IACA to be developed consists of three updates: the auction update, the consensus update and the bundle update.

† **The Auction Update.** At each iteration t, each agent i performs the auction update aiming to increase its bundle by one task. We adopt a time-discounted reward $r_{ij}(t)$ for agent i to task j, as

$$r_{ij}(t) = c_j \lambda_j^{\delta_{i,0j}(t)} \tag{4}$$

where $0 < \lambda_j < 1$ is a constant and c_j is an initial profit for task j. Here, $\delta_{i,kj}(t)$ denotes agent i's *embark time*, which is the time difference between it leaves task k and finishes task j, specially, when $k = 0$, it indicates the time when agent i leaves its initial position. Let a list $\bar{\mathcal{P}}_{i,j}(t)$ denote the *sub-bundle* of $\bar{\mathcal{P}}_i(t)$ indexed by j, which is the bundle before task j. Specially if $j \notin \bar{\mathcal{P}}_i(t)$, then $\bar{\mathcal{P}}_{i,j}(t) = \bar{\mathcal{P}}_i(t)$. Obviously, $\delta_{i,0j}(t)$ is dependent on $\bar{\mathcal{P}}_i(t)$ since agent i will not perform task j until it finishes all the tasks in $\bar{\mathcal{P}}_{i,j}(t)$. Suppose $\bar{\mathcal{P}}_{i,j}(t) = \langle j_1, j_2, ..., j_r \rangle$, then one has

$$\delta_{i,0j}(t) = \delta_{i,0j_1} + \delta_{i,j_1j_2} + \cdots + \delta_{i,j_rj} \tag{5}$$

where $\delta_{i,0j_1}$ denotes the time for agent i to finish task j_1, δ_{i,j_1j_2} denotes the time taken for agent i to finish task j_2 after task j_1, by adding up the traveling time and execution time. An example of $\bar{\mathcal{P}}_{i,j}(t)$ and $\delta_{i,0j}(t)$ is given in Fig. 2.

With the definition of $\delta_{i,0j}(t)$ and the reward function $r_{ij}(t)$ in (4), the IACA allows each agent i to simply perform the auction update by initializing two vectors $y_i, z_i \in \mathbb{R}^n$ for all j in the task set $\mathcal{I}_T$. In the following, $[y_i]_j$ is the jth entry of y_i, corresponding to the price that can be placed by agent i to task j; $[z_i]_j$ is the jth entry of z_i, indicating that this price is placed by agent i.

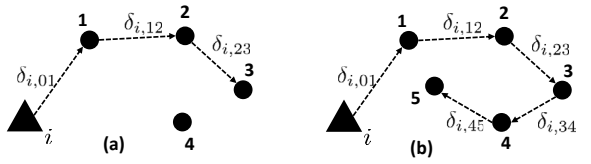

Fig. 2. An example of the sub-bundle $\mathcal{P}_{i,j}(t)$ and the embark time $\delta_{i,0j}(t)$ where $\bar{\mathcal{P}}_i(t) = \{1, 2, 3, 4\}$: (a.) If $j = 3 \in \bar{\mathcal{P}}_i(t)$, then $\bar{\mathcal{P}}_{i,j}(t) = \{1, 2\}$ and $\delta_{i,0j}(t) = \delta_{i,01} + \delta_{i,12} + \delta_{i,23}$; (b.) If $j = 5 \notin \bar{\mathcal{P}}_i(t)$, then $\bar{\mathcal{P}}_{i,j}(t) = \{1, 2, 3, 4\}$ and $\delta_{i,0j}(t) = \delta_{i,01} + \delta_{i,12} + \delta_{i,23} + \delta_{i,34} + \delta_{i,45}$.

Algorithm 1: Auction Update for agent i at time t

Input $\bar{\mathcal{P}}_i(t), \mathcal{I}_T$.
for $j \in \mathcal{I}_T$ **do**
  if $j \notin \bar{\mathcal{P}}_i(t)$ *and* $|\bar{\mathcal{P}}_i(t)| = \ell_i$ **then**
    $[y_i]_j(t) = 0, \quad [z_i]_j(t) = 0.$
  else
    $[y_i]_j(t) = c_j \lambda_j^{\delta_{i,0j}(t)}, \quad [z_i]_j(t) = i.$; // Reward
  end
end
Output $y_i(t), z_i(t)$.

Remark 1: IACA aims to enable each agent to increase its bundle by **at most one task at one iteration**. That is why the embark time $\delta_{i,0j}(t)$ is defined as the time from i to j after finishing all tasks in the sub-bundle $\bar{\mathcal{P}}_i(t)$. By doing this, IACA treats each task $j \notin \bar{\mathcal{P}}_i(t)$ equally as a candidate to be added at the end of $\bar{\mathcal{P}}_i(t)$. For each $j \in \bar{\mathcal{P}}_i(t)$, agent i still updates $\delta_{i,0j}(t)$ and $r_{ij}(t)$ for the purpose of checking whether the task bundle $\bar{\mathcal{P}}_i(t)$ is still a nice one through later the consensus stage. Usually $\bar{\mathcal{P}}_i(t)$ will still be good until agent i finish these tasks since they are increased one by one and each one is resulted from a consensus among all agents in the network. We however still updates the bids for these tasks in the bundle in case that if there is a large delay for agent i to perform task j in $\bar{\mathcal{P}}_i(t)$ for some unknown reason, the tasks after j are automatically released without any time-stamps and can still be won by other agents.

† **The Consensus Update.** Within the iteration t, each agent i needs to perform consensus update indexed by τ. This is for the purpose of resolving conflicts between agents' bids generated in the auction update. A classical idea is by employing a distributed update for achieving the MAX-consensus [19]–[22], which can guarantee a consensus in a finite number τ^* sub-iterations. Here, $\tau^* \geq \Delta_{\mathbb{G}}$ if the underlying graph is static and strongly connected. At the end of the consensus update, all agents reach the MAX-consensus for which all $y_i(t, \tau^*)$ and all $z_i(t, \tau^*)$ converge to the same $y^*(t)$ and $z^*(t)$, with each of their entries correspond to the highest reward of the task and the agent that offers this reward, respectively. Based on this $y^*(t)$ and $z^*(t)$, all agents actually achieve a consensus for the winning list.

† **The Bundle Update.** In this stage, each agent i updates its bundle $\bar{\mathcal{P}}_i(t)$ to be $\bar{\mathcal{P}}_i(t+1)$ based on $y^*(t)$ and $z^*(t)$

Algorithm 2: Consensus Update for agent i

Input $y_i(t), z_i(t)$.
Initialization
  $y_i(t, 0) \leftarrow y_i(t), \quad z_i(t, 0) \leftarrow z_i(t), \quad \tau \leftarrow 0.$
for $\tau \leq \tau^*$ **do**
  for $j \in \mathcal{I}_T$ **do**
    $[y_i]_j(t, \tau+1) \leftarrow \max_{k \in \mathcal{N}_i(t) \cup \{i\}} \{[y_k]_j(t, \tau)\};$
    $[z_i]_j(t, \tau+1) \leftarrow \arg\max_{k \in \mathcal{N}_i(t) \cup \{i\}} \{[y_k]_j(t, \tau)\}.$
  end
  $\tau \leftarrow \tau + 1.$
end
$y^*(t) \leftarrow y_i(t, \tau^*), \quad z^*(t) \leftarrow z_i(t, \tau^*).$
Output $y^*(t), z^*(t)$.

resulted from the consensus update. Let $\mathcal{H}_i(t)$ denote the *valid consensus bundle*, which is the longest sub-bundles $\bar{\mathcal{P}}_{i,j}(t)$ of $\bar{\mathcal{P}}_i(t)$, validated by the condition that $[z]_j^* = i$, for $\forall j \in \mathcal{H}_i(t)$. Let $h_i(t)$ denote the *winning task* such that

$$h_i(t) = \arg \max_{j \in \mathcal{J}_i(t)} [y]_j^*(t)$$

where $\mathcal{J}_i(t) = \{k : z_{ik} = i, \forall k \in \mathcal{I}_T \setminus \bar{\mathcal{P}}_i(t)\}$ denotes the set of agent i's winning tasks not in $\bar{\mathcal{P}}_i(t)$. Then the bundle update is as follows, where $\oplus$ is an operation that appends $h_i(t)$ to the end of list $\bar{\mathcal{P}}_i(t)$.

Algorithm 3: Bundle Update for agent i at time t.

Input $\mathcal{P}_i(t), y^*(t), z^*(t)$.
Compute the valid consensus bundle $\mathcal{H}_i(t)$ **and the winning task** $h_i(t)$.
if $\mathcal{H}_i(t) = \bar{\mathcal{P}}_i(t)$ *and* $|\bar{\mathcal{P}}_i(t)| < \ell_i$ **then**
  $\bar{\mathcal{P}}_i(t+1) = \bar{\mathcal{P}}_i(t) \oplus h_i(t)$; // Bundle update
else
  $\bar{\mathcal{P}}_i(t+1) = \mathcal{H}_i(t)$; // Resolve conflicts
end
Output $\bar{\mathcal{P}}_i(t+1)$

Remark 2: In this step, by $\mathcal{H}_i(t)$, agent i first checks whether $\bar{\mathcal{P}}_i(t)$ is still valid. If a task j in $\bar{\mathcal{P}}_i(t)$ is outbided by another agent, it needs to release all tasks after j to resolve conflicts. If no conflict shows up, agent i generates a task $h_i(t) \notin \bar{\mathcal{P}}_i(t)$ which has the highest reward among all tasks won by agent i. By appending this to the end of $\bar{\mathcal{P}}_i(t)$, one gets $\bar{\mathcal{P}}_i(t+1)$. Actually, $\bar{\mathcal{P}}_i(t)$ is usually valid unless there are some unpredictable reasons which causes dramatic changes to $\delta_{i,0j}(t)$ for a task $j \in \mathcal{P}_i(t)$.

IV. Analysis

A. Costs Analysis for IACA

In distributed algorithms based on the combination of auction and consensus, costs for computation and communi-

cation are naturally decomposed.

Computation Costs for Auction. Note that at the auction stage, each agent needs to compute the reward for each task, so there are no communications but repeated computations for rewards. Let the cost for computing one reward function one time be the unit cost for computations. Let c_P denote the total computation cost. For IACA, recall that at each iteration for auction, each agent needs to compute rewards for all n tasks. Let T_A denote the number of auctions involved in IACA. Then one has the total computation cost as

$$c_P = n \cdot m \cdot T_A \quad \text{for IACA} \tag{6}$$

where m denotes the number of agents and n denotes the number of total tasks. For CBBA, at each iteration of auction, each agent i needs to build a bundle consisting of ℓ_i tasks. To determine the jth task i_j in i's bundle, agent i needs to compute the reward inserted in different positions of the current bundle for each task not in the bundle, which requires $j(n+1-j)$. Let ℓ_i denote the bundle length of agent i and T_{iA} denote the number of iterations of auctions for agent i in CBBA. If all agents are not synchronized, T_{iA} may be different for different agents. Thus the computation cost for the auction is

$$c_P = \sum_{i=1}^{m} \left(T_{iA} \sum_{j=1}^{\ell_i} j(n+1-j) \right) \tag{7}$$

When all agents have the same bundle length ℓ and are synchronized with the same auctions $\bar{T}_A$, one has

$$c_P = \frac{1}{6}\ell(\ell+1)\big(3n+2-2\ell)\big)m\bar{T}_A \quad \text{for CBBA.} \tag{8}$$

By general assumptions that $\bar{T}_A = T_A$ and $\ell \geq \lceil \frac{m}{n} \rceil$, it can be observed that the computational cost for CBBA is much larger than that for IACA in (6).

Communication Costs for Consensus. In consensus state, by each agent simply choosing maximum values from its nearby neighbors, agents could achieve the global maximum in a finite number of steps. Suppose the unit cost for communication is that for one pair of neighbor agents to exchange information for one time (Here, for simplicity, we ignore the size of messages). Let c_M denote the total communication cost. If the network is undirected, fixed and connected with the graph diameter Δ, a number of Δ steps of consensus are enough to achieve the MAX-consensus and resolves conflicts produced at each auction stage. Then for the number of T_A auctions in IACA, the total number of consensus steps involved is $T_A \cdot \Delta$ and thus

$$c_M = \bar{m} \cdot \Delta \cdot T_A \quad \text{for IACA} \tag{9}$$

with $\bar{m}$ the number of edges of the underlying network.

If all agents are synchronized to reach consensus after each auction in the CBBA, we note that each agent also needs to communicate with each of its neighbors for a number of Δ steps. Thus

$$c_M = \bar{m} \cdot \Delta \cdot \bar{T}_A \quad \text{for CBBA} \tag{10}$$

with $\bar{T}_A$ the number of auction steps.

B. Resilience for IACA towards Cyber-attacks.

Reliable information exchange among agents is the basis for distributed algorithms. However, for large scale systems, the presence of malicious agents are sometimes unavoidable. Because of this, resilience is a critical criterion for an effective task allocation algorithm. Note that in CBBA, each agent i only bids for a task j when its local expected reward $r_{ij}(t)$ is greater than the public consensused price $[y]_j^*(t)$. If a malicious agent gives very high bids to all tasks and then leaves the network. These high bids from the malicious agents will remain in the network forever and block other agents to win any task. However, in the auction update, Algorithm 1 of IACA, we allow each agent to bid for all tasks. That is, each agent repeatedly initializes y_i by assuming itself wins all the tasks. As a consequence, the bids placed by malicious agent will be cleared in each round of auction. Thus, the attacker cannot prohibit the normal task allocation process of the multi-agent system unless it stays in the network and always broadcasting faulty information, which will expose itself.

V. SIMULATIONS

In this section, we will perform simulations to validate the proposed IACA, and also provide comparison with the CBBA developed in [12]. Note that compared with CBBA, its asynchronized version in [13] has advantage on behalf of the practical implementation. However, in terms of computation or communication cost, there's no assertion that one algorithm outperforms the other one. Then, since the proposed IACA relay on synchronized updates, we only compare its performance with the synchronized CBBA.

Simulation Setup. Given a fixed number of m agents with positions randomly initialized in a bounded 3D cube. Suppose all agents' task bundles' sizes are the same ℓ^*, which is upper bounded by a fixed integer $\ell_{\max}$. The total number of tasks n is tunable by

$$n = \alpha \cdot m \cdot \ell^*.$$

Here, α is the density of the tasks, which measures the proportion of the exact number of tasks n, over the task handling capability of the overall multi-agent network which is ml^*.

Simulations on Optimality Gap. This simulation aims to validate the performance of the proposed IACA algorithm based on the total reward R defined in (1). Note that finding optimum solution for large-scale task allocation problem is NP-hard, [9]. Thus, in stead of providing the exact optimality gap $O_G \triangleq \frac{R_{\max} - R_{IACA}}{R_{\max}} \times 100\%$, we define a relative optimality

$$O_R \triangleq \frac{R_{IACA}}{R_{CBBA}} \times 100\%,$$

by evaluating the total reward obtained by IACA based on the one obtained by CBBA. For the CBBA, it has been validated that the exact optimality gap is 3% [12]. Let $m = 20$ and $\ell_{\max} = 20$. We do the simulation under the task density $\alpha \in \{0.1,\ 0.2, \cdots,\ 1\}$, and vary the

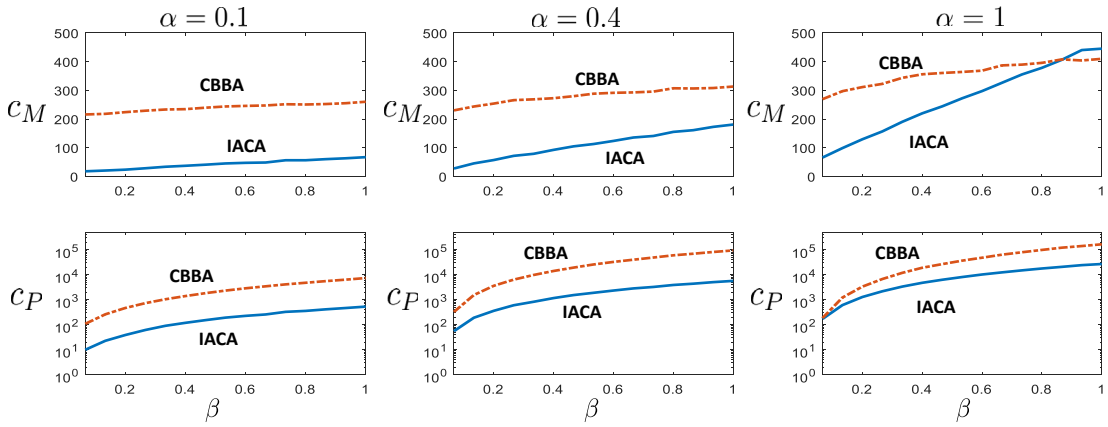

Fig. 3. Simulation results for IACA and CBBA in star graphs.

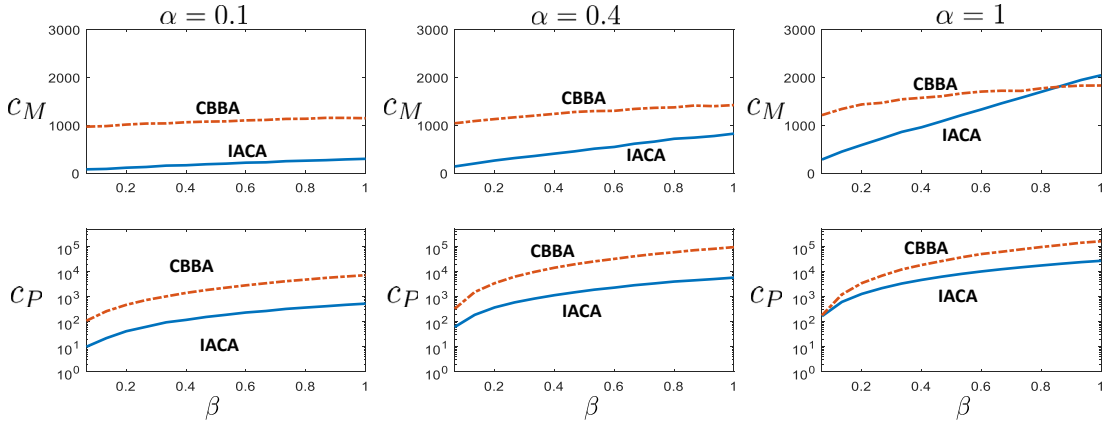

Fig. 4. Simulation results for IACA and CBBA in path graphs.

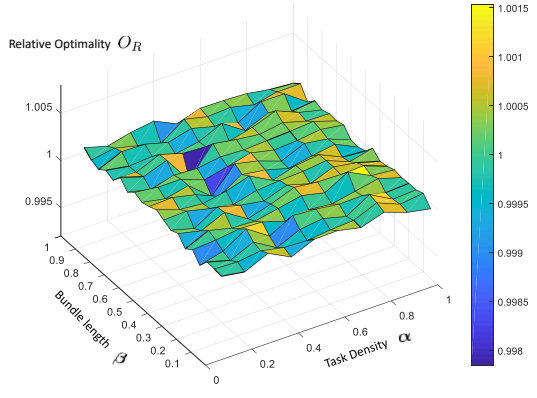

Fig. 5. Simulation that compares the total reward.

bundle size ℓ^* from 1 to $\ell_{\max}$, or equivalently normalized as $\beta = \frac{\ell^*}{\ell_{\max}} \in \{\frac{1}{20}, \frac{2}{20}, \cdots, 1\}$, for the convenience of comparing the simulation results with different bundle sizes. Suppose the underlying graph $\mathbb{G}$ is a complete graph. With random agent and task locations, the simulation is performed 100 times and the result in Fig. 5 shows that IACA reaches a similar optimality with CBBA ($0.998 < O_R < 1.0015$).

Simulations on Computation and Communication Costs. We compare the IACA and CBBA with respect to the communication cost c_M and the computation cost c_P. Let $m = 20$ and $\ell_{\max} = 20$. Specifically, task density is chosen from $\alpha \in \{0.1, 0.4, 1\}$. Suppose the underlying graph $\mathbb{G}$ is fixed and strongly connected. For simplicity we consider $\mathbb{G}$ is a star graph, which corresponds to the fastest tree for MAX-consensus, and the case when $\mathbb{G}$ is a path graph, which corresponds to the slowest tree for MAX-consensus. For each case, we use random agent locations and task locations to initiate the problem and run both algorithm IACA and CBBA for 100 times. Simulation results on path networks and star networks suggest that the total reward achieved by IACA and CBBA are very close to each other. As shown in Fig. 3 and Fig. 4, the proposed IACA outperforms CBBA for small α (i.e. the number of tasks is a small portion of the network capability.) by requiring much smaller costs for computation and communication. As the number of tasks increases (in terms of $\alpha \to 1$), the IACA still requires smaller computational costs c_P than CBBA; but for communication costs c_M, the CBBA only shows slight advantages when the bundle size is large (i.e. β is large).

Simulations on Resilience. We choose $n = 4$, $\alpha = 0.75$ and $\ell^* = 20$ and perform simulations in the presence of malicious agents. As shown in the left of Fig. 6, both the proposed

IACA and CBBA converge to achieve the same total rewards denoted by R when there is no cyber-attacks. As time evolves, the total rewards under IACA gradually increases to reach its maximum while the CBBA increases very fast at the beginning then decreases because of resolving conflicts in agents' task bundles. When there is a malicious agent at time $T_A = 3$ joining the multi-agent network and gives large bids to all tasks, the CBBA fails to work as shown in the right of Fig 6 since these large bids always exist in network and block all agents from biding any tasks, which in turn leads to zero total reward R. Even after the malicious agent leaves the network, the CBBA still can not recover from such attacks as indicated by $R = 0$ even for $T_A \geq 8$. When the same type of attacks appear in the proposed IACA algorithm, the total reward remain unchanged when the malicious agent is in the network, and immediately recover after the malicious agent leaves the network indicated by the increase of total rewards. Thus, to cause other agents in IACA to bid incorrectly, there must be a malicious agent staying in the multi-agent network for infinitely long time while attacks existing for a finite-time steps (for example one iteration) could easily cause the CBBA to malfunction. Thus, we say the proposed IACA is more resilient than CBBA in the sense that influence of cyber-attacks to CBBA is permanent while to IACA is just temporal.

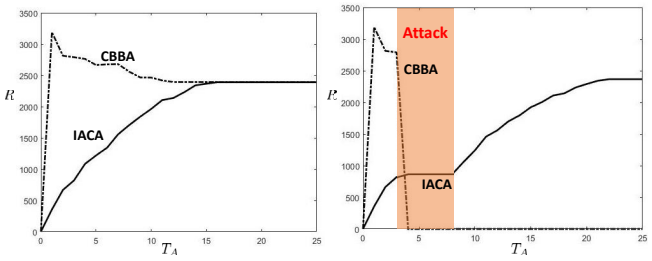

Fig. 6. Simulation results for IACA and CBBA under cyber-attack. On the left figure, no attack is performed. On the right figure, a malicious agent joins the network at $T_A = 3$ and leaves at $T_A = 8$.

VI. Conclusions

In this paper, we have devised an Iterated Auction-Consensus Algorithm, which only involves simple decision logics but has good resilience towards cyber-attack. To compare the performance of the proposed algorithm with existing algorithms, we defined quantitative measures with respect to the cost for computations and communications. It has been shown by simulations that the proposed IACA reaches similar optimality but requires smaller cost for communication and computations and is more resilient in the presence of malicious agents. Our future work include validations of the proposed IACA in large networks and also investigate ways for building the network with minimum computation and communication costs.

References

[1] L. B. Johnson, H.-L. Choi, and J. P. How, "The role of information assumptions in decentralized task allocation: A tutorial," *IEEE Control Systems*, vol. 36, no. 4, pp. 45–58, 2016.

[2] F. Molinari and J. Raisch, "Automation of road intersections using consensus-based auction algorithms," *arXiv preprint arXiv:1804.10411*, 2018.

[3] S. Mou, M. A. Belabbas, A. S.Morse, Z. Sun, and B. D. O. Anderson, "Undirected rigid formations are problematic," *IEEE Transactions on Automatic Control*, vol. 61, no. 10, pp. 2821–2836, 2016.

[4] C. Yan and H. Fang, "Observer-based distributed leader-follower tracking control: a new perspective and results," *International Journal of Control*, pp. 1–10, 2019.

[5] Z. Sun, S. Mou, B. D. O. Anderson, and M. Cao, "Exponential stability for formation control systems with generalized controllers: A unified approach," *Systems & Control Letters*, vol. 93, pp. 50–57, 2016.

[6] X. Chen, M.-A. Belabbas, and T. Bacsar, "Distributed averaging with linear objective maps," *Automatica*, vol. 70, pp. 179–188, 2016.

[7] X. Wang, S. Mou, and B. D. Anderson, "Scalable, distributed algorithms for solving linear equations via double-layered networks," *IEEE Transactions on Automatic Control*, vol. 65, no. 3, pp. 1132–1143, 2020.

[8] X. Wang, J. Zhou, S. Mou, and M. J. Corless, "A distributed algorithm for least squares solutions," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 4217–4222, 2019.

[9] L. Luo, N. Chakraborty, and K. Sycara, "Distributed algorithms for multirobot task assignment with task deadline constraints," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 876–888, 2015.

[10] G. A. Korsah, A. T. Stentz, and M. B. Dias, "The dynamic hungarian algorithm for the assignment problem with changing costs," Carnegie Mellon University, Tech. Rep., July 2007.

[11] D. P. Bertsekas and D. A. Castanon, "Parallel synchronous and asynchronous implementations of the auction algorithm," *Parallel Computing*, vol. 17, no. 6-7, pp. 707–732, 1991.

[12] H. L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust allocation," *IEEE Transactions on Automatic Control*, vol. 25, no. 4, pp. 912–926, 2009.

[13] L. Johnson, S. Ponda, H.-L. Choi, and J. How, "Asynchronous decentralized task allocation for dynamic environments," in *Infotech@ Aerospace 2011*. AIAA, 2011, pp. 1441–1452.

[14] M. Argyle, D. W. Casbeer, and R. Beard, "A multi-team extension of the consensus-based bundle algorithm," in *American Control Conference (ACC), 2011*. IEEE, 2011, pp. 5376–5381.

[15] L. Luo, N. Chakraborty, and K. Sycara, "Provably-good distributed algorithm for constrained multi-robot task assignment for grouped tasks," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 19–30, 2015.

[16] A. Prasad, H.-L. Choi, and S. Sundaram, "Min-max tours and paths for task allocation to heterogeneous agents," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 3, pp. 1511–1522, 2020.

[17] X. Chen, M.-A. Belabbas, and T. Bacsar, "Consensus with linear objective maps," in *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 2847–2852.

[18] C. Lageman and Z. Sun, "Consensus on spheres: Convergence analysis and perturbation theory," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 19–24.

[19] M. Abdelrahim, J. M. Hendrickx, and W. Heemels, "Max-consensus in open multi-agent systems with gossip interactions," *arXiv preprint arXiv:1709.05793*, 2017.

[20] F. Iutzeler, P. Ciblat, and J. Jakubowicz, "Analysis of max-consensus algorithms in wireless channels," *IEEE Transactions on Signal Processing*, vol. 60, no. 11, pp. 6103–6107, 2012.

[21] B. M. Nejad, S. A. Attia, and J. Raisch, "Max-consensus in a max-plus algebraic setting: The case of fixed communication topologies," in *Information, Communication and Automation Technologies, 2009. ICAT 2009. XXII International Symposium on*. IEEE, 2009, pp. 1–7.

[22] S. Zhang, C. Tepedelenlioğlu, M. K. Banavar, and A. Spanias, "Max consensus in sensor networks: Non-linear bounded transmission and additive noise," *IEEE Sensors Journal*, vol. 16, no. 24, pp. 9089–9098, 2016.