2021 60th IEEE Conference on Decision and Control (CDC)
December 13-15, 2021. Austin, Texas

A Metropolis-Hastings Algorithm for Task Allocation

Doha Hamza* , Sarah Toonsi†, Jeff S. Shamma*†

2021 60th IEEE Conference on Decision and Control (CDC) | 978-1-6654-3659-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CDC45484.2021.9683575

***Abstract*—We consider a robot-location assignment problem. The problem is confounded by a location network that restricts robots' motion to neighboring locations. The problem can be optimally solved using a centralized Hungarian algorithm. We propose a distributed game-theoretic algorithm, based on the Metropolis-Hastings mechanism, to eliminate the need for a central coordinator. Agents are activated at random. Agents compare their action, location, against a neighboring one based on the location network. If the new location represents an improvement in agent's utility, they move with probability one, otherwise, they move with a probability proportional to the difference in the two locations' utilities. Our algorithm converges to the optimal assignment of robots to locations. We provide extensive simulations, compare with previous work, and demonstrate the versatility of the proposed algorithm to various task allocation scenarios.**

Keywords*—*Task allocation, game-theoretic algorithm, distributed learning, Metropolis-Hastings algorithm.

I. Introduction

Assignment problems often form building blocks for more complex tasks [1]. We consider a distributed task assignment problem in multi-agent robotic systems. Our setup is in a bounded geographical area with a finite number of locations. The locations are interconnected by a network, called the location network so that agents can only move between neighboring locations, i.e. locations connected by an edge in the location graph. Such a setup mimics many realistic multi-agent systems such as formation assignment [2] where a robot's next action is constrained by its current location. Furthermore, the graphical relations among the nodes can be generalized to represent the physical proximity of locations or it may represent relations among tasks so that tasks connected by an edge may be accomplished together. Agents in our model are dynamic, and keep exploring better opportunities through their neighboring locations. This is a generalization of models where the agents are effectively static, e.g. remote sensing satellites observing locations on earth [3].

The assignment of a particular robot to a specific location yields a utility value that depends on both the agent and the location. Our goal is to find the optimal one-on-one assignment of robots to locations so that the global score, the sum of robots' utilities, is maximized. The problem can be solved using a centralized Hungarian algorithm [4]. In this case, the centralized solution is indifferent to the location figure, i.e. the neighborhoods of the locations, since it assumes a global coordinator with full information of the network and the utility matrix. The centralized solution converges in $\mathcal{O}(N^3)$ where N is the number of robots/locations.

Centralized solutions converge quickly but require heavy message exchanging and global computations which makes them generally infeasible and many times impractical [1]. Furthermore, distributed algorithms are scalable and robust as they can adjust to agent failures and a dynamic changing environment [5]. Many works in the literature recognize the need for distributed task assignment. We list a few next.

A. Related Literature

A distributed version of the Hungarian algorithm is presented in [1]. In that work, at every iteration, multiple autonomous agents perform one of two tasks: 1) Exchange information with their neighbors, and 2) perform identical computation routines. The goal is to gather enough information, from neighbors, so that each robot can perform the Hungarian algorithm by itself. We are, however, seeking a distributed algorithm with more limited information exchange. Agents need not gather information about the state of the whole network, they only exchange local information and compare only two actions at any given time. Furthermore, in our game-theoretic setup, agents are myopic, meaning they only seek to optimize their own benefit in the network. They are also assumed to be simple entities with limited computational and memory capabilities so that only basic calculations can be made. In the numerical results section, we compare the performance of our algorithm against the centralized Hungarian algorithm. Our algorithm is shown to reach the optimal global value despite limited information and coordination among the agents.

In [3], a multi-agent task assignment problem is considered where agents - satellites- are static, hence they have pre-defined admissible task sets. It is required to find the optimal assignment of agents to locations so that the sum of the individual task utilities is maximized. The authors exploit the submodular features of the objective function to solve the problem efficiently using a global greedy and a distributed greedy algorithm. In our setup, agents are dynamic, hence they potentially can be assigned to any location, which makes our formulation more general. Also, we do not make any

*CEMSE division, King Abdullah University of Science and Technology (KAUST), Thuwal, KSA.

†Industrial and Enterprise Systems Engineering (ISE), University of Illinois Urbana-Champaign, Illinois, USA.

Emails:doha.hamzamohamed@kaust.edu.sa, stoonsi2@illinois.edu, jshamma@illinois.edu.

The research reported in this publication was supported by funding from KAUST.

assumptions on the objective function. Our algorithm will work for any setting in which agents may not even have knowledge of the value of the global assignment but can track changes in the global objective due to their actions and those of their neighbors. The global greedy algorithm in [3], as the name implies, requires global information and is suboptimal to the centralized Hungarian method. The distributed greedy algorithm proposed by the authors is suitable for their setup but in our dynamic environment, will yield suboptimal assignments since the robot may get stuck in a local maximum or a neighborhood where no greedy improvements exist around it. This is not a challenge in our setup since agents can *err* and have a non-zero probability to make suboptimal decisions. Indeed, our simulations show that the proposed Metropolis-Hastings Algorithm (MHA) outperforms the distributed greedy algorithm.

There are many dimensions to the task allocation problem [6]. One such different dimension is the one considered in [7] where a distributed auction-like algorithm is presented to provide an almost optimal solution to a task allocation problem. The assignment problem presented in [7] is a one-to-many assignment of robots to tasks where 1) the tasks form a number of disjoint groups, and 2) each robot has an upper bound on the number of tasks it can perform within the whole mission as well as within a certain task group. While the original problem we pose is not under the same assumptions as [7], we do adapt our model to show its inherent flexibility and applicability to many task assignment problems as well as its ability to solve problems that traditionally required a lot of information exchange.

To summarize, we propose a distributed game-theoretic algorithm, based on the Metropolis-Hastings algorithm, to solve a one-on-one assignment problem for tasks and robots. The game-theoretic modeling necessitates that agents are self-centric and only aim to maximize their own utility. The learning rule used by the proposed MHA is very simple and only requires local information to compute the difference in utility between the agent's current and recommended actions. Despite severe information and computational limitations, the algorithm is shown to achieve the globally optimal assignment. This is in contrast with previous works in the literature where either a lot of information exchange is required or sophisticated computations limit the applicability of the proposed distributed algorithms.

In section II we discuss our model and formulate our optimization problem, in section III, we formulate the robot-task assignment problem as a game and present the MHA. Section IV shows some variations to speed up the MHA and also various applications for it. We do extensive numerical simulations in section V to validate our theoretical work. Finally, section VI concludes this paper.

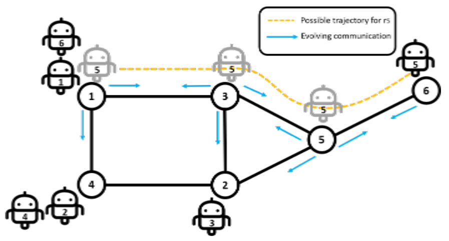

Fig. 1: A possible trajectory for robot r_5. The figure shows the evolving neighborhoods of r_5 as it moves through different locations. At location 5, r_5 can only communicate with locations $\{2, 3, 6\}$.

II. System Model

We consider a set of locations, $\mathcal{L}$ and a set of robots $\mathcal{R}$ within a bounded and known geographical area. The number of robots is not necessarily equal to the number of locations. However, for comparing with other algorithms that require an equal number of robots and locations, we assume an equal number of robots and locations, without loss of generality. Robot i receives a value u_{ij} when assigned to location j. The utilities are encoded in matrix $\mathcal{U} = < u_{ij} >$[1].

Locations are interconnected by a pre-defined location network, L. Once situated at a certain location, a robot can only communicate with robots at its location or at nearby locations.

Definition 1. *A **location matrix** L is a square matrix of dimension $|\mathcal{L}|$. We say that location j can communicate with location j' if $L_{jj'} = 1$, otherwise $L_{jj'} = 0$.*

We will also sometimes represent L as a graph with vertices indicating the different locations and edges between vertices connected locations. We use the terms location matrix/location graph interchangeably. We also make the following definition:

Definition 2. *A **neighborhood** of location j, $\mathcal{N}(j)$, is a collection of node/nodes that communicate with location j, i.e. the non-zero elements of row $L_{jj'}$ $\forall j' \in \mathcal{L}$.*

Fig. 1 shows a sample example of our model with 6 locations. Most robots start at locations 1 and 4, and then -through local interactions with neighbors- spread through the network exploring possible enhancements to their utilities. In this Fig., for example, $\mathcal{N}(6) = \{5\}$. Agents in our model are mobile, hence they have evolving action sets -locations[2]. For example, when r_5 is situated at l_1, it can explore locations in $\mathcal{N}(1)$, and at l_6, it can only explore locations in $\mathcal{N}(6)$. We compare this to the work in [3], where the agents are static and the action sets are fixed.

[1]Although no agent knows such a matrix, we use this matrix to solve for the optimal assignment of a given network using the Hungarian algorithm. This is to compare with our approach.

[2]We formulate the assignment problem as a game in which the action sets are the robots' location choice. Hence we use the terms actions and locations interchangeably.

We define an assignment matrix μ. Element $\mu_{ij} = 1$ if robot i is assigned to location j, $\mu_{ij} = 0$, otherwise. Sometimes we use the notation $\mu(i)$ to denote the location assigned to r_i and use $\mu(j)$ to denote the robot assigned to l_j. We assume the following:

Assumption 1. *The graph associated with locations is fully connected, i.e. robots move from one location to the other in finite time.*

For Fig. 1, the associated matrix $\mathcal{U}$ is as follows:

$$\mathcal{U} = \begin{bmatrix} 3 & 4 & 2 & \enclose{circle}{8} & 5 & 9 \\ 4 & 6 & \enclose{circle}{7} & 8 & 9 & 10 \\ 1 & 4 & 5 & 6 & \enclose{circle}{7} & 8 \\ \enclose{circle}{15} & 2 & 6 & 10 & 8 & 7 \\ 1 & 3 & 4 & 8 & 9 & \enclose{circle}{16} \\ 9 & \enclose{circle}{6} & 4 & 5 & 3 & 2 \end{bmatrix}$$

where the encircled numbers denote the optimal matching obtained using the Hungarian algorithm.

A. Optimization Problem

Given a location graph L, we want to optimize the following:

$$\max_{\mu_{ij}} \quad \phi = \sum_{i \in \mathcal{R}} \sum_{j \in \mathcal{L}} \mu_{ij} u_{ij} \tag{1}$$

$$\text{s.t.} \quad \sum_{j} \mu_{ij} = 1, \;\; \forall i \in \mathcal{R} \tag{2}$$

$$\sum_{i} \mu_{ij} = 1, \;\; \forall j \in \mathcal{L} \tag{3}$$

$$\mu_{ij} \in \{0, 1\}, \;\; \forall i \in \mathcal{R} \text{ and } \forall j \in \mathcal{L} \tag{4}$$

we are after the optimal assignment μ that maximizes the global score, ϕ. The constrained optimization problem in (1) can be solved using a centralized Hungarian Algorithm. However, the communication overhead can be prohibitive. Hence, we propose a distributed, low-information and low-complexity algorithm to solve this optimization problem. We explore this next.

B. About the Metropolis-Hastings Algorithm

We give a brief summary of the algorithm used to solve our problem. The Metropolis-Hastings Algorithm (MHA) was introduced in [8] and generalized in [9]. The algorithm is used in Monte Carlo Markov Chain (MCMC) simulations. The algorithm samples from a probability distribution, called a target distribution, from which direct sampling is difficult. Given a finite state space and a desired distribution π, the algorithm's sampling rule constructs a Markov chain with a transition matrix p with π as its stationary distribution. The transition probability between two states a and a', $p_{aa'}$, is assumed of the form:

$$p_{aa'} = q_{aa'} \alpha_{aa'}$$

where $q_{aa'}$ is the "acceptance probability" and $\alpha_{aa'}$ is the "recommended probability". One popular choice for acceptance probabilities $q_{aa'}$, called the Metropolis choice, is:

$$q_{aa'} = \min\{1, \frac{\pi_{a'} \alpha_{a'a}}{\pi_a \alpha_{aa'}}\},$$

where this choice of $q_{aa'}$ ensures that the transition probabilities satisfy the detailed balance equations and also that the Markov chain is irreducible. Hence, the MHA guarantees existence and uniqueness of the stationary distribution of its induced Markov chain, π.

The MHA is a staple in MCMC simulations and statistics [10]. It remains popular today but also found a new application in game theory since the work of [11]. This work introduced a simple, but clever, trick to use MHA to find the optimal global score of players' actions, $\max_a \phi(a)$, for some collective action of agents a, by letting the acceptance probabilities be:

$$q_{aa'} = \min\{1, e^{(\phi(a') - \phi(a))/T} \frac{\alpha_{a'a}}{\alpha_{aa'}}\}. \tag{5}$$

Adopting this acceptance probability guarantees that the unique stationary distribution of the underlying Markov chain is a Gibbs distribution of the form:

$$\pi_a = \frac{e^{\phi(a)/T}}{Z} \tag{6}$$

for some constant $Z = \sum_{a' \in \mathcal{A}} \phi(a')$ and T, a parameter called the temperature. T induces a regular perturbed Markov process p^T. According to [12], for sufficiently large iterations, one could interpret π_a to equal the probability that the current action of the players $a^t = a$. As $T \to 0$, all the weight of the stationary distribution is on the joint actions that maximize the global score. So although the MHA assigns a non-zero probability to every action a' from any neighboring action in $\mathcal{N}(a)$, as the temperature decreases, only the state(s) which maximizes the global score will occur often, a so-called stochastically stable state(s). The following definition is due to Young [13]:

Definition 3. *A state $a \in \mathcal{A}$ is stochastically stable relative to a regular perturbed Markov process p^T if $\lim_{T \to 0} \pi_a^T > 0$.*

Rather than sampling from a *target distribution*, the game-theoretic approach focuses on a *target state*: The maximizer of the global action score. This solution is facilitated through local interaction (see Remark 1).

III. A Game Theoretic Formulation

We formulate the robot assignment problem as a game $\mathcal{G} = (\mathcal{R}, L, \mathcal{A}, \mathcal{U})$, where $\mathcal{R}$ represents the agents or players set, L is the underlying location graph. At any instance t, r_i selects a feasible location, action a_i^t from its action set $\mathcal{A}_i$[3]. $\mathcal{A} = \prod_i \mathcal{A}_i$ is the resulting

[3]Because of the underlying graph L, the agent is generally making a selection from a smaller set $\mathcal{A}_i^t = \mathcal{N}(a_i^t)$ which is changing according to the evolving play. However, to simplify the notation, we will only use $\mathcal{A}_i$ since $\mathcal{A}_i^t \subseteq \mathcal{A}_i$. We will also sometimes drop the time index t when we simply want to focus on the value attained for a certain action.

action set of the agents. Note that L together with $\mathcal{A}$ determine the current neighborhoods of the agents, i.e. $a_i^{t+1} \in \mathcal{N}(a_i^t)$. It is convenient to use the notation a_{-i}^t to denote the actions of players, other than i, at time t.

The utility the agents receive from selecting a particular location depends also on the choice of other agents, hence the game formulation. Here is an example of how this might happen: Consider the current placement of robots shown in the example in Fig. 1, r_2 and r_4 both select l_4 and $u_{24} = 8$ while $u_{44} = 10$. However, since we stipulate that only one robot is to be assigned to a given location, the *winner* in this case is r_4 since this selection improves the global utility.

We now explicitly use $\phi(a)$ to denote the global utility attained for the collective action $a \in \mathcal{A}$ of agents. We are after the action which maximizes $\phi(.)$. The following dynamic, based on the Metropolis-Hastings algorithm, converges to the optimal assignment of robots to locations and lingers there most of the time:

1) Initialization: All players are randomly dispersed on the available locations.
2) Activate an agent i from the set $\mathcal{R}$ -at random.
3) Activated agent selects an action -at random- from its current neighboring set, i.e. $a_i^{\text{temp}} \in \mathcal{N}(a_i^t)$, while $a_{-i}^{t+1} = a_{-i}^t$, i.e. all other players repeat previous action.
4) Let $a^t = (a_i^t, a_{-i}^t)$, and $a^{\text{temp}} = (a_i^{\text{temp}}, a_{-i}^t)$. Player i calculates the following acceptance probability:
$$q(a^t, a^{\text{temp}}) = \min\{1, e^{\frac{(\phi(a^{\text{temp}})-\phi(a^t))}{T}} \frac{|\mathcal{N}(a_i^{\text{temp}})|}{|\mathcal{N}(a_i^t)|}\}. \tag{7}$$
5) Agent i flips a coin according to the probability $q(a^t, a^{\text{temp}})$. If $q(a^t, a^{\text{temp}}) = 1$, this means a^{temp} represents a clear improvement for i, otherwise i will transition to state a^{t+1} with probability $q(a^t, a^{\text{temp}})$.
6) If i transitions to a_i^{temp}, then the action is updated so that $a_i^{t+1} = a_i^{\text{temp}}$.
7) If i has the highest utility at a_i^{t+1}, then it is assigned. A re-assignment of the *winner* occurs at a_i^t.
8) Repeat.

Remark 1. *We note how robots can calculate the difference in global utility $\phi(a^{\text{temp}}) - \phi(a^t)$ in (7) to evaluate $q(a^t, a^{\text{temp}})$. Since agents only move one at a time, then the difference in global utility is due to the difference in action between a_i^{temp} and a_i^t and its impact on global utility. Without loss of generality, suppose $a_i^{\text{temp}} = l_2$ and $a_i^t = l_3$, then:*

$$\phi(a^{\text{temp}})-\phi(a^t) = \max_{a_{i'}=2} u_{i'2} + \max_{\substack{a_{i'}=3 \\ i'\neq i}} u_{i'3} - \max_{\substack{a_{i'}=2 \\ i'\neq i}} u_{i'2} - \max_{a_{i'}=3} u_{i'3} \tag{8}$$

where the first term in (8) is the possible utility if r_i moves to l_2 under the action a_i^{temp}. The second term represents the utility at l_3 if r_i moves. The third term is the existing utility at l_2, without r_i. The fourth term is the existing utility at l_3, with r_i being there. Clearly, only local information is needed to evaluate (8).

Algorithm 1 Pseudo code for the MHA

```
 1: Initialize with arbitrary placement of robots
 2: loop
 3:     Activate $r_i$-at random
 4:     $r_i$ selects any action $a_i^{\text{temp}} \in \mathcal{N}(a_i^t)$
 5:     $a_{-i}^{t+1} = a_{-i}^t$
 6:     $r_i$ calculates $q(a^t, a^{\text{temp}})$ according to (7)
 7:     if $\text{RAND}[0,1] \leq q(a^t, a^{\text{temp}})$ then
 8:         $a_i^{t+1} \leftarrow a_i^{\text{temp}}$
 9:         $\mu(a_i^{t+1}) = \arg\max_{a_{i'}=a_i^{t+1}} u_{i' a_i^{t+1}}$
10:         $\mu(a_i^t) = \arg\max_{a_{i'}=a_i^t} u_{i' a_i^t}$
11:     end if
12: end loop
```

According to the MHA dynamic, there is a non-zero probability for agents to make a suboptimal decision, for example by moving to a state a_i^{t+1} with a lower utility than their current utility. However, these decisions help the agents explore the whole terrain of choices. They eventually will settle into the optimal assignment.

Theorem 1. *Given two agents' actions $a = (a_i, a_{-i})$ and $a' = (a_i', a_{-i})$ and $T > 0$, the transition probabilities:*

$$p_{aa'} = \begin{cases} \frac{1}{|\mathcal{N}(a_i')|} e^{(\phi(a')-\phi(a))/T} \textit{if } e^{(\phi(a')-\phi(a))/T} \frac{|\mathcal{N}(a_i)|}{|\mathcal{N}(a_i')|} \leq 1 \\ \frac{1}{|\mathcal{N}(a_i)|} \quad \textit{o.w.} \end{cases} \tag{9}$$

guarantee the unique stationary distribution of the underlying Markov chain is a Gibbs distribution of the form in (6).

Proof: The choice of $p_{aa'}$ as in (9) guarantees that the detailed balance property applies to the Markov chain induced by MHA, this guarantees the existence of a stationary distribution as in (6). Irreducibility of the Markov chain follows from the fact that the sampler assigns a non-zero probability to any action in the neighboring action set. Irreducibility guarantees the uniqueness of the stationary distribution. [8], [9]. ■

Lemma 1. *Using the learning rule of Theorem 1, the stochastically stable state of MHA is the optimal state, a^*, the solution to the constrained problem in (1).*

Proof: The Gibbs distribution in (6) is centered at the state of the maximum global score. As $T \to 0$, $\pi_a \to 0 \; \forall \, a \neq a^*$. According to [13], over the long run, states that are not stochastically stable will be observed infrequently compared to states that are, provided that T is small. The unique stochastically stable state will be observed almost all of the time when T is small. ■

Fig. 2 shows a run of MHA on a 4-node network which is connected as a line. The optimal is calculated using the centralized Hungarian method. The results

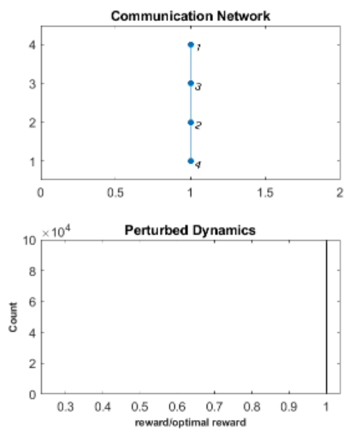

Fig. 2: A run of MHA on a line topology network. The robots spend most of their time at the optimal state.

clearly validate lemma 1.

IV. MHA: A Versatile Learning Algorithm

We show here various variations to our original setup to demonstrate the adaptability of the MHA.

A. Convergence Rate and Accelerating MHA

The convergence rate of the MHA depends on the target distribution and is in general difficult to characterize [14]. In our case, this target distribution is not known apriori as it is the subject of our optimization. Nevertheless, there are various methods by which we can accelerate the convergence of MHA.

We note in Algorithm 1 that only one agent is active at a time. However, as highlighted in Remark 1 and equation (8), the difference in global utility is only due to changes in utility based on the player's recommended action and its current action. This hints at the possibility of activating more than one agent at a given time, provided that the agents' considered actions a_i^t and a_i^{temp} do not overlap. Fig. 4 represents a sample on the example we presented in Fig. 1. In the numerical results section, we also show one extreme case of activating all agents at the same time, whether actions are overlapping or not. The MHA is quite robust to such uncoordinated play.

The work in [15] highlights that the convergence rate of MHA depends also on the temperature T. Fixed, large values of T lead to rapid exploration of the state space. However, such agent behavior typically doesn't settle into an optimal configuration since inferior actions tend to be accepted and agents' motions are volatile. Small values of T, on the other hand, lead to greedy exploitation. However, the MHA may get *stuck* because the probability of accepting inferior actions is small.

Hence, in [15], one possibility to speed up the convergence is to make the learning rate variable by

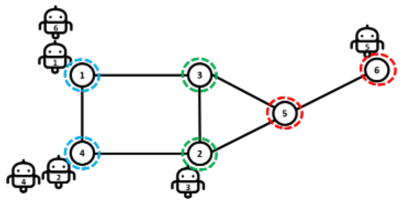

Fig. 3: Using the same network as Fig. 1, it is possible for three agents to be activated at the same time. Robots' current and temporary actions are highlighted using the same colored circles.

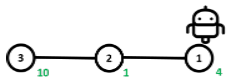

Fig. 4: Highlighting the difficulty the robot faces to reach a better assignment if the temperature is small. The utility values are in green. The robot could improve its utility drastically from 4 to 10 if it managed to get to location 3. However, this is extremely unlikely to happen with a small T value due to the bottle neck location l_2.

starting with a small nominal value for T and then increasing the value, and hence the exploration, if no improvement is encountered for N times, where N is a pre-defined variable. To maintain cohesion among the nodes, the learning rate then undergoes an exponential decay to its nominal value. In the numerical results section, we explore the impact of a variable temperature on the convergence of MHA.

The work in [16] takes a different approach by studying the transient behavior of the Markov chain resulting from the MHA learning rule and other stochastic learning algorithms. Starting from an initial condition, that work identifies the subset of state space, called cycles, that have a small hitting time and long exit times. This approach is useful when the lifetime of the network is smaller than the convergence time of the learning mechanism. We cite this work here to emphasize that techniques exist to analyze the transient behavior of the MHA and other learning algorithms.

B. Extension to Multi-task and Multi-robot Assignments

We examine here the possible extension to the multi-task (A robot can be assigned to many tasks) or multi-robot (many robots simultaneously assigned to a single task) scenarios. This occurs often in the task assignment literature. For example in [17], a multi-task scenario is considered. Each robot r_i can perform a maximum of m_i tasks. Tasks also form disjoint groups, so that r_i can perform at most m_{ik} tasks of the k-th task group, T_k, where there are n_t task groups. So the optimization problem becomes:

$$
\begin{aligned}
\max_{\mu_{ij}} \quad & \phi = \sum_{i\in\mathcal{R}} \sum_{j\in\mathcal{L}} \mu_{ij} u_{ij} \\
\text{s.t.} \quad & \sum_{i} \mu_{ij} = 1, \ \ \forall j \in \mathcal{L} \\
& \sum_{j} \mu_{ij} \le m_i, \ \ \forall i \in \mathcal{J} \\
& \sum_{j\in T_k} \mu_{ij} \le m_{ik}, \forall i \in \mathcal{R} \text{ and}, \forall k = 1, ..., n_t \\
& \mu_{ij} \in \{0,1\}, \ \ \forall i \in \mathcal{R} \text{ and } \forall j \in \mathcal{L}
\end{aligned}
$$

We can adapt the MHA to this setup. We start with a feasible assignment of robots to tasks. When a robot gets activated with a task(s): The agent calculates the marginal value of adding the task following the total and task group constraints. This process eventually leads the system to the global optimal assignment as long as we can track the change in global utility due to any agent's current and temporal action.

V. Numerical Results

We start with a comparison with the work in [3] which considers a static task assignment problem similar to ours. By static, we mean that the agents - satellites- are immobile and have constrained action sets. This means certain tasks are not compatible with the agents but those sets are fixed. The authors propose greedy solutions to solve their problem using first a global greedy approach, which enumerates the maximum element in matrix $\mathcal{U}$, assigns the robot-task pair which attains this value, and then keeps assigning pairs in decreasing order of the utility until it is done. This is suboptimal to the centralized Hungarian algorithm but converges very quickly. The authors also consider a distributed version of the greedy algorithm in which agents calculate their marginal contribution to a task, i.e. the added value they bring when assigned to a given task. Agents then propose to the target with the maximum marginal contribution. Conflicting proposals are resolved arbitrarily among the robots. Fig. 5 reports our results. The location graph is shown on top and the achieved relative reward using all approaches is shown on the bottom. Both greedy algorithms converge fast but the performance of the distributed greedy algorithm is always suboptimal to our approach. By insisting that robots only propose to locations with maximum marginal contribution, the algorithm gets stuck in suboptimal assignments because there is no improvement in the immediate neighborhood. While the global greedy converges fast, it is suboptimal and centralized.

In Fig. 6, we explore the impact of varying the temperature on the performance of MHA. As outlined in [15], a varying temperature provides the best performance. In the variable rate scheme, we start with a large temperature to allow the agents to explore the network, and then the temperature is lowered so that only the optimal payoff is maintained. A constant large value for the temperature results in a noisy performance while a small value may make the algorithm stuck in a local optimal for a long time.

In Fig. 7, we compare MHA with the algorithm proposed in [17]. The algorithm in [17] uses a consensus-like distributed algorithm to reach an almost optimal assignment. In the bidding process of each agent, the latest price of each task is propagated to each agent so

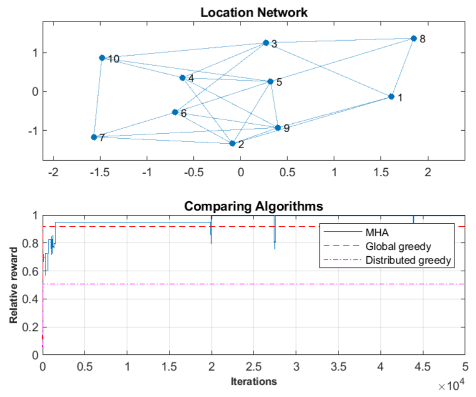

Fig. 5: Comparing MHA with the global and distributed greedy algorithms of [3].

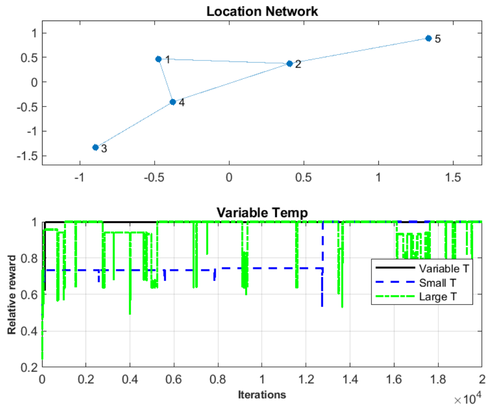

Fig. 6: Impact of varying the learning rate on the MHA.

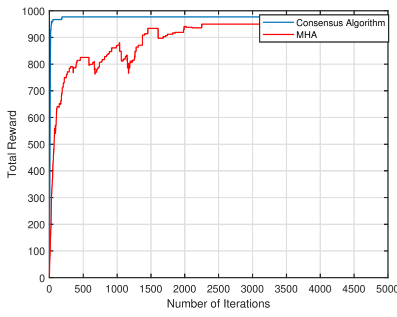

Fig. 7: A comparison with the multi-task scenario of [17] where the number of robots = 20, the number of tasks = 60, the number of task subsets = 20 and $m_{ik} = 3$.

that the agent can decide whether to bid on the task. One version of the algorithm considers a sequential activation of agents, but it is assumed that agents always have access to the latest network-wide price of all tasks. This is one reason why the algorithm in [17] converges very fast versus our algorithm, as shown in Fig. 7. However, despite the strong information disadvantage, MHA is well on track to achieve the performance of the consensus algorithm. In fact, in less than 100 iterations, MHA achieves more than 50% of the performance achieved by the consensus algorithm. It is also possible to modify MHA using the techniques discussed previously to achieve better convergence.

Also, we used PX4 SITL in Gazebo to simulate six drones (agents) on a location graph similar to Fig. 1. Drones fly at different altitudes to avoid collision. A demonstration video of the simulation is available at https://youtu.be/G5DSbMg6EtA. We assume the nodes form a wireless network where each node can only communicate with its neighbors. An agent chooses a neighbor at random and connects to that neighbor channel. Any given node channel can provide agents with the value of the currently winning score.

Using local clocks along with a predefined waiting time and order, algorithm 1 can be easily implemented in any multi-agent setup. To demonstrate the versatility of the algorithm, we simulated some runs while activating more than one agent, starting with non-overlapping actions and then moving to the extreme case where all agents move at the same time. The results of the real-time implementation are shown in Fig. 8. A single agent activation provides the best performance with minimal noisy performance. The reasonably good performance with all agents' activation is a testament to the robustness of MHA to lack of synchronization.

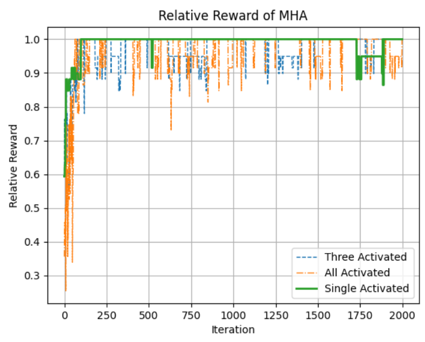

Fig. 8: A real time implementation of MHA using a single agent activation, non-overlapping simultaneous agent activation, and all-agent-activation versions of MHA

VI. Conclusion

We proposed a game-theoretic learning algorithm for robotic task assignment with an underlying location graph restricting robot motion. The algorithm is based on the Metropolis-Hastings mechanism used in MCMC simulations. We showed that our proposed low-information distributed algorithm will reach the globally optimal state. Extensive simulations verified our approach. We also showed various ways to improve the performance of the proposed algorithm and to widen its applicability to various task assignment problems.

References

[1] Smriti Chopra, Giuseppe Notarstefano, Matthew Rice, and Magnus Egerstedt, "A distributed version of the hungarian method for multirobot assignment," *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 932–947, 2017.

[2] H. Jaleel and J. S. Shamma, "Distributed optimization for robot networks: From real-time convex optimization to game-theoretic self-organization," *Proceedings of the IEEE*, vol. 108, no. 11, pp. 1953–1967, 2020.

[3] Guannan Qu, Dave Brown, and Na Li, "Distributed greedy algorithm for multi-agent task assignment problem with submodular utility functions," *Automatica*, vol. 105, pp. 206–215, 2019.

[4] Rainer E Burkard and Eranda Cela, "Linear assignment problems and extensions," in *Handbook of combinatorial optimization*, pp. 75–149. Springer, 1999.

[5] Nathan Michael, Michael M Zavlanos, Vijay Kumar, and George J Pappas, "Distributed multi-robot task assignment and formation control," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 128–133.

[6] Brian P Gerkey and Maja J Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International journal of robotics research*, vol. 23, no. 9, pp. 939–954, 2004.

[7] G Ayorkor Korsah, Anthony Stentz, and M Bernardine Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[8] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[9] W Keith Hastings, "Monte carlo sampling methods using markov chains and their applications," 1970.

[10] David B Hitchcock, "A history of the metropolis–hastings algorithm," *The American Statistician*, vol. 57, no. 4, pp. 254–257, 2003.

[11] Daniel Pickem, Magnus Egerstedt, and Jeff S Shamma, "A game-theoretic formulation of the homogeneous self-reconfiguration problem," in *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 2829–2834.

[12] Jason R. Marden and Jeff S. Shamma, "Revisiting log-linear learning: Asynchrony, completeness and payoff-based implementation," *Games and Economic Behavior*, vol. 75, no. 2, pp. 788 – 808, 2012.

[13] H Peyton Young, "The evolution of conventions," *Econometrica: Journal of the Econometric Society*, pp. 57–84, 1993.

[14] Heikki Haario, Eero Saksman, and Johanna Tamminen, "An adaptive metropolis algorithm," *Bernoulli*, vol. 7, no. 2, pp. 223–242, 04 2001.

[15] D. Pickem, *Self-reconfigurable Multi-Robot Systems*, Ph.D. thesis, Georgia Institute of Technology, Atlanta, 2016.

[16] Hassan Jaleel and Jeff S Shamma, "Transient response analysis of metropolis learning in games," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 9661–9667, 2017.

[17] Lingzhi Luo, Nilanjan Chakraborty, and Katia 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, 2014.