Greedy Decentralized Auction-based Task Allocation for Multi-Agent Systems ⋆

Martin Braquet * **Efstathios Bakolas** *

* *Department of Aerospace Engineering and Engineering Mechanics, The University of Texas at Austin, Austin, Texas 78712-1221, USA (e-mail: braquet@utexas.edu, bakolas@austin.utexas.edu).*

Abstract: We propose a decentralized auction-based algorithm for the solution of dynamic task allocation problems for spatially distributed multi-agent systems. In our approach, each member of the multi-agent team is assigned to at most one task from a set of spatially distributed tasks, while several agents can be allocated to the same task. The task assignment is dynamic since it is updated at discrete time stages (iterations) to account for the current states of the agents as the latter move towards the tasks assigned to them at the previous stage. Our proposed methods can find applications in problems of resource allocation by intelligent machines such as the delivery of packages by a fleet of unmanned or semi-autonomous aerial vehicles. In our approach, the task allocation accounts for both the cost incurred by the agents for the completion of their assigned tasks (e.g., energy or fuel consumption) and the rewards earned for their completion (which may reflect, for instance, the agents' satisfaction). We propose a Greedy Coalition Auction Algorithm (GCAA) in which the agents possess bid vectors representing their best evaluations of the task utilities. The agents propose bids, deduce an allocation based on their bid vectors and update them after each iteration. The solution estimate of the proposed task allocation algorithm converges after a finite number of iterations which cannot exceed the number of agents. Finally, we use numerical simulations to illustrate the effectiveness of the proposed task allocation algorithm (in terms of performance and computation time) in several scenarios involving multiple agents and tasks distributed over a spatial 2D domain.

Keywords: Multi-agent and Networked Systems, Auction Algorithms, Decentralized Systems, Trajectory Planning.

1. INTRODUCTION

We present a decentralized auction-based algorithm to address dynamic task allocation problems for multi-agent systems. In our problem, the agents have to complete a set of tasks which are distributed over a given spatial domain. We propose a decentralized solution for the computation of task assignment profiles based on auction-based negotiations between the agents. Our proposed methods can find applications in problems in which agents (e.g., autonomous vehicles, humans, robots, intelligent machines, etc.) have to share resources and distribute the workload among them in order to accomplish one or more tasks. Disaster response by a fleet of unmanned aerial vehicles (UAV) which have to assess the severity of the situation and discover where help is needed more as well as the delivery of packages by autonomous or semi-autonomous ground or aerial robots are two characteristic examples.

Literature review: There are several types of task allocation problems for multi-agent systems depending on the ability of each agent to handle multiple tasks (involving task scheduling) and on whether it is possible to have multiple agents assigned to the same task (thus, allowing for the formation of coalition of agents).

An important consideration when developing algorithms for multi-agent task allocation is the ability of these algorithms to be deployed in systems where there is no single entity that allocates tasks and workload among the agents. In this regard, centralized methods rely on a single point of operation in the sense that the agents negotiate with each other under the direction of a central entity (Gerkey and Mataric (2002)). Decentralized methods avoid this single point of failure by allowing each agent to consult directly with the other agents and compute their own task assignments. Decentralized execution, however, adds significant computation time (Choi et al. (2009); Nanjanath and Gini (2010); Capitan et al. (2013)).

Auction-based approaches are derived from market economy principles in which each agent tries to maximize his own profit, based on the total reward that will then be redistributed among them. These methods find many applications (e.g., satellites in Phillips and Parra (2021), drones in Hayat et al. (2020)) mainly because of certain key benefits such as the worst-case global utility that can be derived theoretically by using them (Qu et al. (2019)), their fast convergence, low complexity and high computational efficiency (Kim et al. (2019); Shin et al. (2019)). Auction-based methods have also been boosted by recent breakthroughs in reinforcement learning (Rahili et al. (2020)). The consensus-based bundle algorithm (Choi et al. (2009)) (CBBA) utilizes a market-based decision strategy as the mechanism for decentralized task selection and uses a consensus routine based on local communication as a conflict resolution mechanism to achieve agreement on the winning bid values.

One of the simplest approaches to solve decentralized auction-based problems is via greedy algorithms, which consider the optimal (in a myopic sense) choice that maximizes a global objective (Luo et al. (2012)). Auction-based techniques have been proven to produce suboptimal solutions (Gerkey and Matarić (2004)) with a guaranteed convergence to a conflict-free assignment. Auc-

⋆ This work was supported in part by a Fellowship of the Belgian American Educational Foundation.

tions are known to be scalable and robust to variations in the communication network topology (Whitbrook et al. (2019); Otte et al. (2020)).

Other types of task allocation methods include game theoretic approaches and in particular methods based on potential games for the computation of mutually agreeable task assignments. Although the negotiation protocols are proven to converge to mutually agreeable tasks (Arslan et al. (2007)), their convergence is only guaranteed for the case in which the game remains the same (task utilities are constant) which is not the case in a dynamic task allocation problem. Such algorithms aim to compute a mutually agreeable profile corresponding to a Nash equilibrium (game-theoretic formulation of task allocation problems) for all agents (Bakolas and Lee (2021)). Game theory is an important tool to extend task allocation problems to multiple agents but finding *efficient* Nash equilibria (task assignment profiles giving high global utility) is not guaranteed (solutions based on individual rationality may not automatically lead to high global utility) and computational cost can be significant. Recently, machine-learning algorithms have received a significant amount of attention in control theory and for task allocation problems mainly because they can process a lot of information such as the agents' state (by utilizing, for instance, neural networks) and handle unknown environments (e.g., task features) via reinforcement learning (especially Deep Q-Learning Gautier et al. (2020)).

Contributions: In this paper, we propose a dynamic auction-based task allocation algorithm. In our approach, the task utilities depend on both the rewards earned by the agents for accomplishing their assigned tasks as well as the costs they incur while doing so (the latter correspond to cost-to-go functions of relevant optimal control problems). The utilities are thus in general dependent on the state of the agents. In this context, the agents can only perform one task while several agents can be assigned the same task (if this is beneficial to them and their team). In contrast with game-theoretic algorithms which may not always achieve high global utility for the team (inefficient Nash equilibria), our proposed auction-based task allocation mechanism finds task assignments that aim to greedily reach the global utility of the system. A key advantage of our proposed approach is time efficiency, yet with reasonably high global utility.

We propose a Greedy Coalition Auction Algorithm (GCAA) where the agents negotiate while moving in their state space towards their assigned tasks. When an agent changes his assignment, he needs to recompute the cost estimate and thus his own state-dependent utility. In contrast to game-theoretic solutions (Bakolas and Lee (2021)) which aim for individual rationality but cannot guarantee good team performance, we do not seek a mutually agreeable task assignment but consider instead a broader set of solutions that allows for a higher global utility. Furthermore, in contrast with the CBBA algorithm which clusters and schedules a sequence of tasks for each agent, in this work the problem is composed of multiple agents making a coalition for a specific task (which is the only task for that agent) and thus allows for a faster task assignment. This work hence falls under the category of *Single-Task Multi-Robot Instantaneous Assignment* (ST-MR-IA) problem, also known as the coalition formation problem (Gerkey and Matarić (2004)).

Outline: The rest of the paper is presented as follows. We discuss the problem setup in Section 2. In Section 3, we identify the utilities for the tasks, the agents individually, and the team as a whole. The proposed dynamic auction-based task allocation algorithm and the theoretical analysis on its convergence are presented in Section 4. In Section 5, we present extensive numerical simulations. Finally, concluding remarks and directions for future work are provided in Section 6.

2. PROBLEM SETUP

We assume a multi-agent system (MAS) comprised of n agents. These agents, who can be assigned only one task, are called active agents when they are far from their target so that they can recompute their best task assignment while moving toward the target. Otherwise they are called passive agents when they are too close to the target to consider other targets (they are then permanently assigned to this final task). Let $x_i \in \mathcal{S}_i \subseteq \Sigma$ and $u_i \in U_i$, for $i \in [1, n]_d$ be, respectively, the state and input of the i-th agent of the MAS at time $t \geq 0$ ($\mathcal{S}_i$ being his state space and U_i his input space), and $\Sigma \subseteq \mathbb{R}^m$. We also define $\boldsymbol{x} \in \mathcal{S}$ the joint state of the MAS, in which $\boldsymbol{x} := (x_1, \dots, x_n)$ and $\mathcal{S} := \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ (joint state space). Let $\boldsymbol{u} \in U$ be the joint input of the MAS, where $\boldsymbol{u} := (u_1, \dots, u_n)$ and $U := U_1 \times \cdots \times U_n$ (joint input space).

The motion of the i-th agent is described by

$$\dot{x}_i = f_i(\boldsymbol{x}, \boldsymbol{u}), \qquad x_i(0) = x_i^0, \qquad i \in [1, n]_d, \tag{1}$$

where $x_i^0 \in \mathcal{S}_i$ is the initial state of the i-th agent and $f_i : \mathcal{S}_i \times U_i \to \mathcal{S}_i$ is his associated vector field. Consequently, $\boldsymbol{x}^0 = (x_1^0, \dots, x_n^0) \in \mathcal{S}$ denotes the joint initial state.

In general, task allocation aims to assign individual tasks for n agents and p tasks, $\mathcal{T} := \{\mathcal{T}_1, \dots, \mathcal{T}_p\}$. Let $X_\mathcal{T}$ be the set of states associated with the given tasks, where $X_\mathcal{T} := \{x_{\mathcal{T}_1}, \dots, x_{\mathcal{T}_p}\}$, and $\mathcal{A}_i := \{a_i^k : k \in [1, \mathrm{card}(\mathcal{A}_i)]_d\}$ the set of possible task assignments for the i-th agent given a set of tasks $\mathcal{T}$. While the agents have limited communication between each other, we suppose that they have complete information about all the tasks available. Each assignment $a_i^k \in \mathcal{A}_i$ is equal to either a task in $\mathcal{T}$, that is, $a_i^k = \mathcal{T}_\ell$ where $\mathcal{T}_\ell \in \mathcal{T}$, or the null assignment, that is, $a_i^k = \varnothing$.

Additionally, we denote the set of active agents as $\mathcal{N}_a \subseteq [1, n]_d$ and we fix the assignment a_i of agent i (thus switching his status from active to passive) for all $t > t_p$ if the agent lies inside the boundary of the target, that is, $\Phi_i(x_i(t_p), x_{\mathcal{T}_j}) < 0$ where $\Phi_i(x_i(t_p), x_{\mathcal{T}_j})$ is a boundary constraint; for instance, $\Phi_i(x_i(t_p), x_{\mathcal{T}_j}) := \|x_i(t_p) - x_{\mathcal{T}_j}\| - R_p$ where R_p is the minimum agent-to-target distance to make the task assignment permanent.

3. TASK UTILITIES

The task utility is characterized by a reward obtained for the completion of the task $\mathcal{T}_j \in \mathcal{T}$ and a state-dependent cost reflecting the cost to finish this task (for example, the transition cost due to the motion of the agent).

Static task utility: Given an assignment profile $\boldsymbol{a} = (a_1, \dots, a_n)$, we denote by $\mathcal{T}_j^{-1}(\boldsymbol{a})$ the index-set corresponding to the agents assigned to task $\mathcal{T}_j \in \mathcal{T}$ under the particular profile. Since a task is not necessarily accomplished when an agent is assigned to it, we let $p_{ij} \in [0, 1]$ be the probability of the task $\mathcal{T}_j$ to be completed successfully by the i-th agent. In this case, the probability that the task is successfully completed by at least one agent increases with the number of agents assigned to this task. The expected reward for completing task $\mathcal{T}_j$ is defined as Bakolas and Lee (2021):

$$r_{\mathcal{T}_j}(\boldsymbol{a}) = \bar{r}_{\mathcal{T}_j}\left[1 - \prod\nolimits_{i\in\mathcal{T}_j^{-1}(\boldsymbol{a})}(1-p_{ij})\right], \tag{2}$$

where $\bar{r}_{\mathcal{T}_j}$ is the nominal reward of $\mathcal{T}_j$. Indeed, the probability that at least one agent completes the task is equal to the complementary of the probability that no agent completes the task, i.e., $\prod_{i\in\mathcal{T}_j^{-1}(\boldsymbol{a})}(1-p_{ij})$. It is worth noting that the assignments (and their associated utility) of the passive agents are also taken into account when computing the total reward.

State-dependent task completion cost: The cost to finish the task $\mathcal{T}_j$ associated with the state $x_{\mathcal{T}_j}$ at time $t = t_{\mathsf{f},\mathcal{T}_j}$ by the i-th agent is defined as the optimal cost of the following the optimal control problem.

Problem 1. Let $a_i = \mathcal{T}_j$, where $\mathcal{T}_j \in \mathcal{T}$ and $i \in [1,n]_d$. Furthermore,let $x_{\mathcal{T}_j} \in \mathcal{S}_i$ denote the state linked to $\mathcal{T}_j$ and $t_{\mathsf{f},\mathcal{T}_j} > 0$ the related completion time for $\mathcal{T}_j$. The goal is to obtain an optimal control input $u_i^\star(\cdot) : [0,t_\mathsf{f}] \to U_i$ which is a piece-wise continuous function that minimizes the following functional:

$$\mathcal{J}_i(u_i(\cdot); x_i^0, x_{\mathcal{T}_j}) := \int_0^{t_\mathsf{f}} \mathcal{L}_i(x_i(t), u_i(t))\mathrm{d}t, \tag{3}$$

such that the dynamic constraints (1) and the following terminal constraint $\Psi_i(x_i(t_\mathsf{f}), x_{\mathcal{T}_j}) = 0$, where $\Psi_i(\cdot; x_{\mathcal{T}_j})$ is a given C^1 function, are respected. Finally, the minimum cost is given by $\rho_i(x_i^0; x_{\mathcal{T}_j}) := \mathcal{J}_i(u_i^\star(\cdot); x_i^0, x_{\mathcal{T}_j})$.

Total Task Utility: The total completion cost of task $\mathcal{T}_j$ given the assignment profile $\boldsymbol{a} = (a_1, \ldots, a_n)$ is given by

$$\mathfrak{R}_{\mathcal{T}_j}(\boldsymbol{a}; \boldsymbol{x}^0, x_{\mathcal{T}_j}) := \sum\nolimits_{i\in\mathcal{T}_j^{-1}(\boldsymbol{a})} \rho_i(x_i^0; x_{\mathcal{T}_j}), \tag{4}$$

which leads to the definition of the total task utility associated with task $\mathcal{T}_j$ for a given $\boldsymbol{x}_0$

$$\mathcal{U}_{\mathcal{T}_j}(\boldsymbol{a}; \boldsymbol{x}^0) := r_{\mathcal{T}_j}(\boldsymbol{a}) - \lambda_{\mathcal{T}_j}\mathfrak{R}_{\mathcal{T}_j}(\boldsymbol{a}, \boldsymbol{x}^0; x_{\mathcal{T}_j}) \tag{5}$$

where $\lambda_{\mathcal{T}_j}$ is a constant which is used to convert the cost-to-go to the same units as the reward (e.g., from a loss of energy to a loss of money).

Individual, Team Utilities: Let us denote the global utility as $\mathcal{U}(\boldsymbol{a}; \boldsymbol{x}^0) := \sum_{\mathcal{T}_j\in\mathcal{T}} \mathcal{U}_{\mathcal{T}_j}(\boldsymbol{a}; \boldsymbol{x}^0)$. The goal is to set this team's utility equal to the sum of each individual utility in order to maximize each individual utility separately. In this regard, based on the task assignment $\boldsymbol{a}$, we set the individual utility of agent i equal to his marginal contribution to the global utility $\mathcal{U}(\boldsymbol{a}; \boldsymbol{x}^0)$:

$$\begin{aligned}\mathcal{U}_i(\boldsymbol{a}; \boldsymbol{x}^0) &:= \mathcal{U}(\boldsymbol{a}; \boldsymbol{x}^0) - \mathcal{U}((a_\varnothing, a_{-i}); \boldsymbol{x}^0)\\ &= \mathcal{U}_{\mathcal{T}_j}(\boldsymbol{a}; \boldsymbol{x}^0) - \mathcal{U}_{\mathcal{T}_j}((a_\varnothing, a_{-i}); \boldsymbol{x}^0).\end{aligned} \tag{6}$$

4. DYNAMIC TASK ALLOCATION

4.1 Problem formulation

The task allocation is called dynamic since the utilities of the agents change along their path towards their target (state-dependent cost and agents obtain new information by communicating with other agents in the surrounding). In this case, a new assignment profile $\boldsymbol{a}^\star(t)$ has to be selected at each time step $t \in [0, t_\mathsf{f}]$ as the agents evolve in their state space.

Problem 2. (DTA: Dynamic Task Allocation). Given $t_\mathsf{f} > 0$ and $\boldsymbol{x}^0 \in \mathcal{S}$, find a time-varying task assignment profile $\boldsymbol{a}^\star(\cdot) : [0, t_\mathsf{f}] \to \mathcal{A}$ for all the remaining active agents $i \in \mathcal{N}_a$, that maximizes the global utility in a decentralized way (in the presence of communication constraints) according to the permanent assignment a_{i_p} of the passive agents $i_p \in \mathcal{N}_p = [1,n]_d \backslash \mathcal{N}_a$ and the terminal constraints $\Psi_i(x_i(t_\mathsf{f}), x_{\mathcal{T}_j}) = 0$.

4.2 Auction protocols for decentralized task allocation

The main principle of auctions consists in the computation of agents' individual utility for some tasks (called bids). Based on these proposed bids, the agents communicate between each other in order to deduce the best allocation for each of them. A key point is that for their realization, an agent does not have to know the utilities of his teammates (decentralized implementation). Next we propose an algorithm solving Problem 2, its main idea is to find the best task coalition for the multi-agent network by allocating the tasks to the agents obtaining the highest utility (greedy approach).

4.3 Greedy Coalition Auction Algorithm

The Greedy Coalition Auction Algorithm (GCAA) is an auction-based algorithm that leverages the simplicity of greedy approaches to provide a solution with fast convergence. The main idea is to iterate between an auction phase and a consensus phase such that it converges to a winning bids list (Choi et al. (2009)).

Each agent has three vectors that are constantly updated at each iteration step t. The first vector $\boldsymbol{z}_i \in [0,p]_d^n$ is the list of selected tasks among $\mathcal{T}$, meaning that agent i possesses a vector $\boldsymbol{z}_i$ of length n where the k-th element of the vector is the expected task assignment of agent k to the best knowledge of agent i. The second vector $\boldsymbol{y}_i \in \mathbb{R}_{>0}^n$ is the list of winning bids (agent's utilities), that is, the k-th element of $\boldsymbol{y}_i$ is the expected individual utility of agent k by selecting the task $z_{i,k}$ (k-th element of the vector of selected tasks $\boldsymbol{z}_i$). The third vector $\boldsymbol{c}_i \in [0,1]_d^n$ is the list of finalized (or completed) allocations and informs agent i about the status of the allocation for the other agents. In particular, the k-th element of $\boldsymbol{c}_i$ is set to 1 if the agent k does not plan to change his target anymore, and 0 otherwise. This way, the agents for which the assignment is completed are not taken into account for the auction process in subsequent steps. Based on these three vectors that are first updated, each agent will decide and propose the best assignment for themselves (i.e., maximizing their own utility). The main algorithm is presented in Main Algorithm and the two associated phases are explained next.

Main Algorithm: Greedy Coalition Auction Algorithm
Input: $\boldsymbol{x}^0$
Output: $\boldsymbol{z}(t)$
1: $t = 0$
2: $\boldsymbol{y}(0) = \mathbf{0}$
3: $\boldsymbol{z}(0) = \mathbf{0}$
4: $\boldsymbol{c}(0) = \mathbf{0}$
5: **while** $\exists i : c_{i,i}(t) = 0$ **and** $c_{i,i}(t-1) = 0$ **do**
6:     SelectBestTask()
7:     ShareStateVectors()
8:     UpdateStateVectors()
9:     $t = t + 1$

Auction process: The first phase of the algorithm is the auction process. Here, each agent aims to select his best task based on his own utility. At lines 2–4 of Algorithm 1, the previous bid vectors are copied into the current ones.

If the task selected by one agent i is not finalized (line 5), agent i picks the task J_i that maximizes his expected utility (lines 6–7). Agent i then updates his bid vector with the selected task (line 8) and the associated utility (line 9).

Algorithm 1 Select the best task for agent i

Function SelectBestTask
Input: $\boldsymbol{y}(t-1), \boldsymbol{z}(t-1), \boldsymbol{c}(t-1), \boldsymbol{x}^0$
Output: $\boldsymbol{y}(t), \boldsymbol{z}(t)$

```
1: for i ∈ [1, n]_d do
2:     y_i(t) = y_i(t − 1)
3:     z_i(t) = z_i(t − 1)
4:     c_i(t) = c_i(t − 1)
5:     if c_{i,i}(t) = 0 then
6:         a = z_i(t)
7:         J_i = argmax_j U_i((z_{i,j}(t), a*_{−i}); x_i^0)
8:         z_{i,i}(t) = J_i
9:         y_{i,i}(t) = U_i(z_i(t); x_i^0)
```

Consensus process: In Algorithm 2, the consensus process first aims to share the bid vectors $\boldsymbol{y}_i$, $\boldsymbol{z}_i$, $\boldsymbol{c}_i$ with the other agents within the communication range of agent i. For each agent i, the agents k within the communication range of agent i (satisfying $g_{ik}(t) = 1$ at lines 1–2) send their bid vectors $y_{k,k}(t), z_{k,k}(t)$ and $c_{k,k}(t)$ (lines 3–5). Then in Algorithm 3, based on his winner bids vector, agent i determines the set of agents $\tilde{\mathcal{A}}_i(t)$ allocated to the same selected task (line 2) and extracts the winner based on their utility (line 3). He adds the winner to the list of finalized allocations $\boldsymbol{c}_i$ (line 4) and resets the values of the losers in the bids $\boldsymbol{y}_i$ and tasks $\boldsymbol{z}_i$ (lines 5–8).

Algorithm 2 Share the bid vectors to agent i

Function ShareStateVectors
Input: $\boldsymbol{y}(t), \boldsymbol{z}(t), \boldsymbol{c}(t)$
Output: $\boldsymbol{y}(t), \boldsymbol{z}(t), \boldsymbol{c}(t)$

```
1: for i ∈ [1, n]_d do
2:     for k ∈ {k | g_ik(t) = 1} do
3:         z_{i,k}(t) = z_{k,k}(t)
4:         y_{i,k}(t) = y_{k,k}(t)
5:         c_{i,k}(t) = c_{k,k}(t)
```

Algorithm 3 Update the bid vectors of agent i according to the winners/losers

---

Function UpdateStateVectors
Input: $\boldsymbol{y}(t), \boldsymbol{z}(t), \boldsymbol{c}(t)$
Output: $\boldsymbol{y}(t), \boldsymbol{z}(t), \boldsymbol{c}(t)$

```
1: for i ∈ [1, n]_d do
2:     Ã_i(t) = {k | z_{i,k}(t) = z_{i,i}(t), f_{i,k}(t) = 0}
3:     K_i = argmax_{k∈Ã_i(t)} y_{i,k}(t)
4:     c_{i,K_i}(t) = 1
5:     for k ∈ Ã_i(t) \ K_i do
6:         z_{i,k}(t) = 0
7:         y_{i,k}(t) = 0
```

Then the time is updated ($t \leftarrow t+1$) and the main algorithm loops to Algorithm 1. Finally, the algorithm has converged when the finalized choices are validated for some agents ($c_{i,i} = 1$) and the other agents not assigned to a task ($c_{i,i} = 0$) have not changed since the past iteration (meaning that the cost to reach each task is higher than the marginal reward they can obtain). Once the task allocation is completed, the agents move according to the solution of Problem 1.

4.4 Application example

To illustrate the main steps of the algorithm through a simple example with 2 tasks and 4 agents, Fig. 1 shows a task allocation along with their bid vectors. The communication links are shown with red dashed lines and the final task allocation is given with green dashed lines.

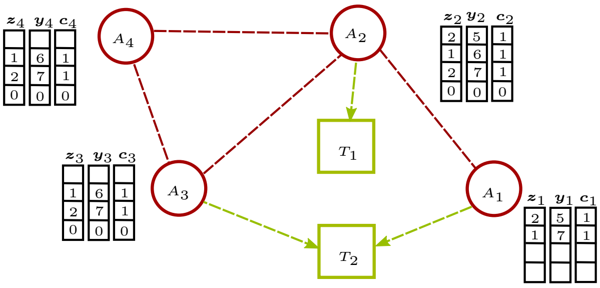


Fig. 1. Graphical illustration of the auction-based greedy algorithm.

Since agents A_1 and A_3 cannot communicate directly with each other, they assume that they will obtain the entire reward by completing their selected task T_2 while they will actually need to split it. More precisely, each agent i fills in his bid vector (associated to his i-th row) depending on his best assignment during the first iteration. For example, agent A_1 chooses task T_2 with a utility of 5 while agent A_2 chooses task T_1 with a utility of 6. Then, they share their bid vector (i.e., fill in their rows) with their neighbors only, so that A_1 does not have information about A_3, and reciprocally. Each agent finally updates his bid vector by selecting the task with the highest utility and setting the associated assignment status $\boldsymbol{c}$ to 1 (e.g., at the first iteration, all agents finalize the assignment of A_2 because he proposes a utility of 7). At the next iterations, the assignment of A_2 is no longer computed and the other agents take into account the permanent assignment of A_2 for the computation of their own utility (e.g., A_4 no longer proposes a bid for T_2 because the reduction of the marginal reward associated to the coalition with A_2 dropped his marginal utility below zero, it is thus preferable for A_4 not to select any task by securing a null utility). At the second iteration, A_1 and A_3 propose and finalize their assignment for T_2 since they think that they are completing T_2 individually (no communication between them) and A_4 does not propose any assignment. This example thus shows that communication constraints can lead to suboptimal solutions because the actual utility that A_1 and A_3 will receive by completing T_2 is lower than their prediction.

4.5 Theoretical convergence

Theorem 1. Consider the auction-based task allocation process solved by the GCAA algorithm (Main Algorithm) where the communication range can be limited. Let n be the number of agents, then GCAA converges to an assignment within at most n steps.

Proof: The proof is derived from the definition of greedy algorithms. In particular at each time iteration t and for all agents $i \in [1, n]_d$, one element (index K_i as presented in Algorithm 3) of $\boldsymbol{c}_i$ is set to 1 as the task of agent i is set to be finalized. As a consequence at time t, there are

t elements of $\boldsymbol{c}_i$ set to 1 and $n-t$ elements still initialized to 0. Hence at time $t=n$, all the elements of $\boldsymbol{c}_i$ are set to 1 for each agent i which means that the stopping criteria in Main Algorithm ($c_{i,i}=1$ for all agents i) is necessarily verified. The algorithm is thus proven to converge after at most n steps (the number of agents). $\square$

Remark 1 This convergence theorem guarantees that the computation time is growing linearly with the number of agents.

5. NUMERICAL SIMULATIONS

In this section, we present numerical simulations [1] to illustrate the main ideas of the methods proposed so far. We consider a team of agents with double integrator dynamics, that is, $\ddot{p}_i = u_i$, with $p_i(0) = p_i^0$ and $\dot{p}_i(0) = v_i^0$, where $p_i \in \mathbb{R}^2$ ($p_i^0 \in \mathbb{R}^2$) and $\dot{p}_i \in \mathbb{R}^2$ ($v_i^0 \in \mathbb{R}^2$) denote, respectively, the position and velocity of the i-th agent at time t ($t_0 = 0$), $i \in [1,n]_d$. The performance index is given by the control effort $\mathcal{J}(u_i(\cdot)) := (1/2)\int_0^{t_\mathsf{f}} |u_i(t)|^2 \mathrm{d}t$ and the conversion constant is $\lambda_{\mathcal{T}_j} = 1$ ($j \in [1,p]_d$). By setting $x_i := (p_i,\ \dot{p}_i) \in \mathbb{R}^4$ and $x_{\mathcal{T}_j} := (p_{\mathcal{T}_j}, 0) \in \mathbb{R}^4$, the terminal constraint function is chosen randomly for the different between:

- $\Psi_i(x_i(t_{\mathsf{f},\mathcal{T}_j}); x_{\mathcal{T}_j}) := x_i - x_{\mathcal{T}_j}$, which means that the i-th agent tries to reach the position $p_{\mathcal{T}_j}$ associated with his assigned task $\mathcal{T}_j$ at time $t = t_{\mathsf{f},\mathcal{T}_j}$ and with terminal velocity $\dot{p}_{\mathcal{T}_j}$ (randomly selected).
- $\Psi_i(x_i(t_{\mathsf{f},\mathcal{T}_j}); x_{\mathcal{T}_j}) := \|p_i - p_{\mathcal{T}_j}\| - \tilde{R}_{\mathcal{T}_j}$, which means that the i-th agent tries to reach the circle (with radius $\tilde{R}_{\mathcal{T}_j}$) around his assigned task $\mathcal{T}_j$ at time $t = t_{\mathsf{f},\mathcal{T}_j} - \tau_{\mathcal{T}_j}$ and then loiters around the target until $t_{\mathsf{f},\mathcal{T}_j}$. In this work, the best entry point to enter the circle is selected by discretizing the circle in 10 points and selecting the point that minimizes the cost function [2].

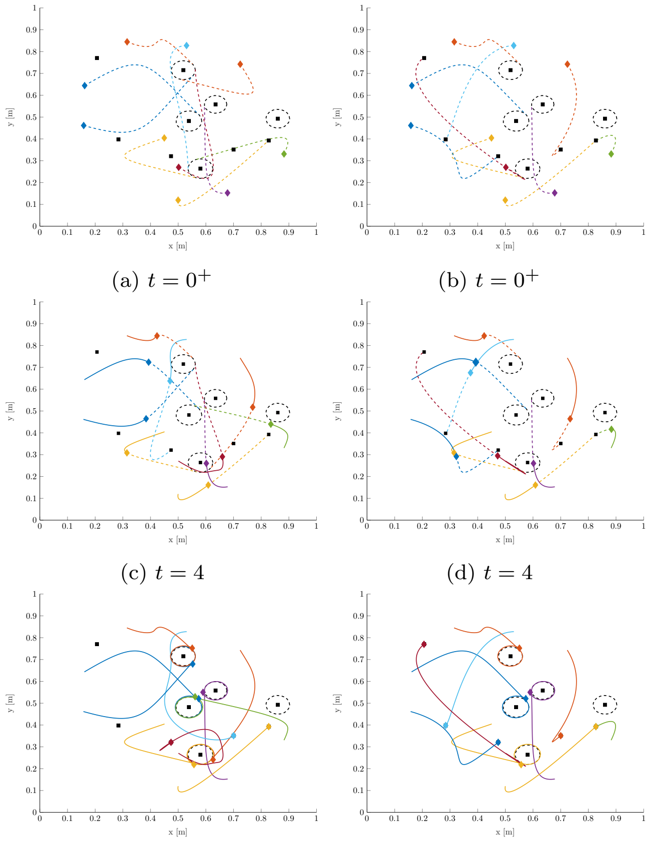
Both terminal constraints are associated with an optimal control problem with non-zero initial and terminal velocities. It turns out (see, for instance, Battin (1987)) that the optimal control input is given by

$$u_i^\star(t; t_\mathsf{f}, x_i^0, x_{\mathcal{T}_j}) = \frac{4}{t_\mathsf{f}-t}\left[\dot{p}_{\mathcal{T}_j} - \dot{p}_i(t)\right] + \frac{6}{(t_\mathsf{f}-t)^2}\left[p_{\mathcal{T}_j} - p_i(t) - \dot{p}_{\mathcal{T}_j}(t_\mathsf{f}-t)\right] \tag{7}$$

which defines a second-order differential equation with time-varying coefficients where $u_i^\star(t) = \ddot{p}_i(t)$, which is solved numerically using integration tools (ODE45) in Matlab.

While problems with zero terminal velocities have an analytical solution (Bakolas (2014)), problems with non-zero terminal velocities require more computation time due to the numerical integration.

A dynamic task allocation is then performed and presented through the dynamic map of the allocation. Fig. 2 (b,d,f) illustrates trajectories of the agents in the absence of communication constraints (or limitations) while the agents in Fig. 2 (a,c,e) can only communicate [3] with the other agents within range $\varrho = 0.3$. Due to the fact that the computation time required for the numerical integration is substantial, we only consider scenarios with $n = 10$ agents and $p = 10$ tasks. 5 tasks are fixed targets with non-zero terminal velocities $\dot{p}_{\mathcal{T}_j} \in [-0.1, 0.1]$ (black squares). The other 5 tasks are dynamic, the agents need to loiter at a radius $\tilde{R}_{\mathcal{T}_j} \in [0.032, 0.048]$ and complete one loop at velocity $\dot{p}_{\mathcal{T}_j} \in [-0.1, 0.1]$ for a time $\tau_{\mathcal{T}_j}$ such that $\tau_{\mathcal{T}_j}/t_\mathsf{f} \in [0.15, 0.25]$ (black dashed circle around a dashed square).



(a) $t = 0^+$ (b) $t = 0^+$

(c) $t = 4$ (d) $t = 4$

(e) $t = 10$ (f) $t = 10$

Fig. 2. Dynamic task allocation for the range constrained (left, $\varrho = 0.3$, $\mathcal{U} = 1.804$) and range unconstrained (right, $\varrho \to \infty$, $\mathcal{U} = 1.515$) cases ($n = p = 10$, $t_\mathsf{f} = 10$).

In Fig. 2 where the range is limited to 0.3, several agents are allocated to the same task because they are not in communication with all the other agents. They thus estimate their utility solely based on the reward of the task while their marginal utility is actually lower. When the agents come closer and enter the communication range, the agents start assessing their marginal utility correctly and thus consider other tasks that might increase their own utility. Toward the end of the simulation (Fig. 2(e)), the agents' trajectory is subject to sharper changes of direction (e.g. the red and light blue curves) compared to the range unconstrained case (Fig. 2(f)).

Fig. 3 and 4 quantitatively show the allocation presented above, for which the data from the range unconstrained case (blue lines) are constant over time. When the communication is limited, the total utility increases step-by-step as the agents start communicating with their neighbors (red line in Fig. 4) but still remains lower than the utility obtained when the communication is not limited. It is worth noting that even though the

[1] Source code available at `https://github.com/MartinBraquet/task-allocation-auctions`.

[2] The best solution can also be found by optimal control methods in a systematic / rigorous way and will be considered in further work

[3] The communication range is not shown in the figure for clarity.

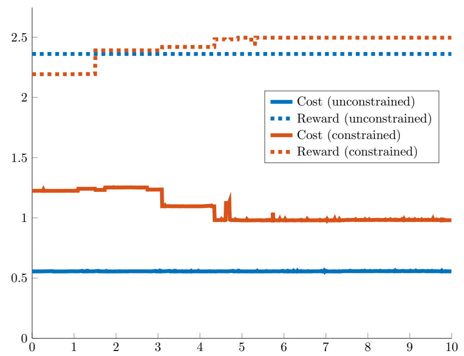


Fig. 3. Total cost and reward.

final reward is higher for the constrained case (dashed red line in Fig. 3), the higher cost produced by abrupt trajectory changes makes its final utility lower than the utility generated without communication limitation. Finally, the noisy curves are due to approximation errors in the numerical integration. In Fig. 5, the global utility increases progressively with the communication range.

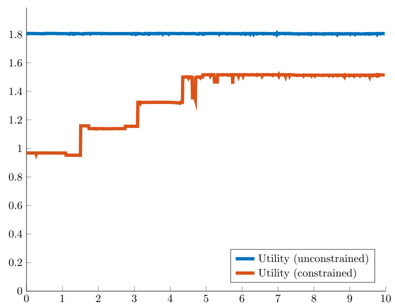


Fig. 4. Total utility.

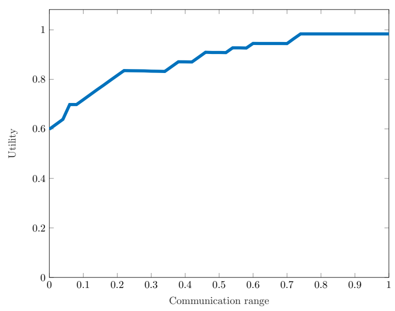


Fig. 5. Impact analysis of the communication range on the utility.

6. CONCLUSION

In this paper, we have presented an auction-based framework to address dynamic task allocation problems for multi-agent systems with state-dependent utilities and various task characteristics (such as terminal constraints, completion time, etc.). Our greedy approach offers a practical, yet efficient, solution to a class of realistic and challenging dynamic task allocation problems for autonomous mobile agents. In the case of large fleets of autonomous systems, scalability issues may arise due to the high computation time. In our future work, we will explore ways to integrate tools from machine learning and multi-agent reinforcement learning in our auction-based decentralized task allocation framework in order to reduce the computational cost and be able to address more challenging problems.

REFERENCES

Arslan, G., Marden, J.R., and Shamma, J.S. (2007). Autonomous Vehicle-Target Assignment: A Game-Theoretical Formulation. *J. Dyn. Syst. Meas. Control*, 129(5), 584–596.

Bakolas, E. (2014). A decentralized spatial partitioning algorithm based on the minimum control effort metric. *2014 American Control Conference*, 5264–5269.

Bakolas, E. and Lee, Y. (2021). Decentralized game-theoretic control for dynamic task allocation problems for multi-agent systems. In *2021 American Control Conference (ACC)*, 3219–3224.

Battin, R. (1987). An introduction to the mathematics and methods of astrodynamics. 559–561.

Capitan, J., Spaan, M.T., Merino, L., and Ollero, A. (2013). Decentralized multi-robot cooperation with auctioned POMDPs. *The International Journal of Robotics Research*, 32(6), 650–671.

Choi, H., Brunet, L., and How, J.P. (2009). Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, 25(4), 912–926.

Gautier, P., Johann, L.D., and Diguet, J.P. (2020). Comparison of Market-based and DQN methods for Multi-Robot processing Task Allocation (MRpTA). *IEEE International Conference on Robotic Computing (IRC)*.

Gerkey, B.P. and Mataric, M.J. (2002). Sold!: Auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18(5), 758–768.

Gerkey, B.P. and Matarić, M.J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9), 939–954.

Hayat, S., Yanmaz, E., Bettstetter, C., and Brown, T.X. (2020). Multi-objective drone path planning for search and rescue with quality-of-service requirements. *Autonomous Robots*, 44(7), 1183–1198.

Kim, K.S., Kim, H.Y., and Choi, H.L. (2019). Minimizing communications in decentralized greedy task allocation. *Journal of Aerospace Information Systems*, 16, 1–6.

Luo, L., Chakraborty, N., and Sycara, K. (2012). Competitive analysis of repeated greedy auction algorithm for online multi-robot task assignment. In *2012 IEEE International Conference on Robotics and Automation*, 4792–4799.

Nanjanath, M. and Gini, M. (2010). Repeated auctions for robust task execution by a robot team. *Robotics and Autonomous Systems*, 58(7), 900 – 909.

Otte, M., Kuhlman, M.J., and Sofge, D. (2020). Auctions for multi-robot task allocation in communication limited environments. *Autonomous Robots*, 44(3), 547–584.

Phillips, S. and Parra, F. (2021). A case study on auction-based task allocation algorithms in multi-satellite systems. *AIAA Scitech 2021 Forum*.

Qu, G., Brown, D., and Li, N. (2019). Distributed greedy algorithm for multi-agent task assignment problem with submodular utility functions. *Automatica*, 105, 206 – 215.

Rahili, S., Riviere, B., and Chung, S.J. (2020). Distributed adaptive reinforcement learning: A method for optimal routing. *arXiv preprint arXiv:2005.01976*.

Shin, H.S., Li, T., and Segui-Gasco, P. (2019). Sample greedy based task allocation for multiple robot systems. *arXiv preprint arXiv:1901.03258*.

Whitbrook, A., Meng, Q., and Chung, P.W.H. (2019). Addressing robustness in time-critical, distributed, task allocation algorithms. *Applied intelligence*, 49(1), 1–15.