# Distributed Greedy Algorithm for Multi-Agent Task Assignment Problem with Submodular Utility Functions ⋆

Guannan Qu [a], Dave Brown [b], Na Li [a]

[a] *Harvard University, Cambridge, MA 02138, USA.*

[b] *Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, MA 02420, USA.*

**Abstract**

We consider a multi-agent task assignment problem where a group of agents need to select tasks from their admissible task sets. The utility of an assignment profile is measured by the sum of individual task utilities, which is a submodular function of the set of agents that are assigned to it. The objective is to find an assignment profile that maximizes the global utility. This problem is NP-hard in general. In this paper we propose an algorithm that provides an assignment profile with utility at least $1/(1+\kappa)$ of the optimal utility, where $\kappa \in [0,1]$ is a parameter for the curvature of the submodular utility functions. In the worst case, when $\kappa = 1$, our algorithm achieves utility at least $1/2$ of the optimal. Moreover, when the communication links between agents are consistent with the admissible task sets, the algorithm can be implemented distributedly and asynchronously, which means that there is no centralized coordinator and each agent selects its task using only local information and local communication based on its own time-clock.

## 1 Introduction

We consider a multi-agent task assignment problem where there are a group of agents, each of which needs to select [1] a task from its admissible task set. [2] Given an assignment profile, the utility of a task is determined by the set of agents that select it, and the global utility is the sum of all the task utilities. The objective is to find an assignment profile to maximize the global utility. The problem has many applications such as satellite-location assignment in a distributed space system [27], vehicle-target (weapon-target) assignment problem [24,1], and sensor coverage problem [18,21]. As an example, in a distributed space system, there is a group of remote sensing satellites that observe a set of locations on the Earth [30]. Each satellite can select a location on the Earth to observe, as long as the location is within its line-of-sight. For each location, the observation quality is given by a utility function on the set of satellites that are assigned to that location. This function reflects the satellites' distance to the location as well as other factors such as the atmospheric condition. The satellite location assignment problem is to assign locations to satellites such that the total utility over all the locations is maximized.

The task assignment problem is usually hard to solve as the problem is usually formulated as an integer programming problem or a set optimization problem. In [24], an algorithm has been developed based on the convex relaxation of the integer programming problem but there is no performance guarantee of the approximated solution. Alternatively, if the task utility functions are assumed to be submodular [16,25,11], (a special class of set functions, of which the functions in [24] are a special case), greedy-type algorithms can be used to find an assignment profile with an efficiency ratio being at least $1/(1+\kappa)$ [16,11,7], [3] where $\kappa \in [0,1]$ is the curvature [7] of the submodular function. For some special cases, greedy-type algorithms can even find the optimal solution [15][8].

⋆ This work is supported under NSF ECCS 1608509 and NSF CAREER 1553407. Some preliminary work appeared in NecSys 2015. Dave Brown is currently an MIT Lincoln Laboratory employee. No Laboratory funding or resources were used to produce the result/findings reported in this publication.

*Email addresses:* gqu@g.harvard.edu (Guannan Qu), david.brown@ll.mit.edu (Dave Brown), nali@seas.harvard.edu (Na Li).

[1] In this paper we will use "select" and "be assigned to" interchangeably.

[2] The admissible task set of an agent is a *given* nonempty subset of the tasks.

[3] The efficiency ratio of an assignment profile is the ratio of the global utility of the assignment profile to the optimal global utility.

A drawback of these existing algorithms is that they require a centralized coordinator with access to global information (information of all the tasks) and global communication (communication to all the agents). In many applications such as the distributed space system, it is costly to maintain such a centralized coordinator with all the information and communication needed [27,30]. A more practical scenario is that each agent acts as a decentralized decision maker that use only local information and local communication. For instance, in the distributed space system, each satellite is a decision maker and it has information and communication only with locations and satellites within its line-of-sight.

In this paper, we propose a distributed variant of the greedy algorithm [11] which we refer to as the distributed greedy algorithm. In the distributed greedy algorithm, when the communication links between agents are consistent with the admissible task sets, [4] each agent can make its own decision in an distributed and asynchronous fashion, using only local information and local communication based on its own time-clock. In addition, each iteration of the algorithm only involves a simple local maximization problem and several rounds of local communication to resolve certain 'conflicts', and is easy to implement. Moreover, we show that the distributed greedy algorithm terminates within finite iterations. Under the assumption of submodular utility functions, our algorithm produces an assignment profile with a utility that is at least $1/(1+\kappa)$ of the optimal global utility, where $\kappa \in [0,1]$ is a parameter for the curvature of the submodular utility functions. In the worst case ($\kappa = 1$), our algorithm achieves utility at least $1/2$ of the optimal. This lower bound is also proven to be tight in the sense that there exist scenarios that the efficiency ratio $1/(1+\kappa)$ is achieved. Finally, simulation results confirm the theoretical results and show that on average the distributed greedy algorithm can achieve much better efficiency ratio than $1/(1+\kappa)$.

### *1.1 Related Work*

There exists a large literature in task assignment and related problems. In multi-agent control, [23,5] studies gradient flow algorithms for sensor coverage problems where sensors adjust their locations in a geographical area to maximize coverage. Our work considers a different setting where the assignment process is of combinatorial nature. Assignment problem has also been explored in the matching literature e.g. [4,6,29], in which tasks (or objects) are assigned to agents to maximize social welfare. Auction algorithms [2,17,9,26,32] as well as other incentive mechanisms [3] are devised to solve this problem, in which tasks/objects determine their prices and agents maximize their revenues so that a good equilibrium is reached. Distributed matching and auction, especially those similar to our settings, are in general not well-studied.

The literature closest to our setting is the line of work in [31,1,22,19,28] on the multi-agent assignment problem using game design approaches. The algorithm design is decomposed to two design steps: i) local utility function for each agent is designed so that the Nash equilibrium of the resulting game achieves a good efficiency compared to the optimal solution; ii) game theory learning algorithms are designed to reach such Nash equilibrium where the convergence is usually asymptotic (convergence when the number of steps of the algorithm goes to infinity). Compared with these algorithms, we directly design the task assignment algorithm and our algorithm produces a final solution within finite number of time steps. Numerical comparison between game design methods and our approach is given in Section 4.

Some preliminary work has been published in a conference paper [27]. Compared to [27], this paper provides a refined efficiency ratio bound, allows agents to update asynchronously, and provides more simulations.

## 2 Problem Formulation and Preliminaries

### *2.1 Problem Formulation*

Consider a set of agents $\mathcal{A} := \{a_1, \ldots, a_N\}$ and a set of tasks $\mathcal{T} := \{\tau_1, \ldots, \tau_M\}$. For notational simplicity, we use $a \in \mathcal{A}$ to denote an agent and $\tau \in \mathcal{T}$ to denote a task. Each agent $a$ is associated with a *given* nonempty admissible task set $\mathcal{T}_a \subset \mathcal{T}$ which is a subset of tasks that agent $a$ can select from. An assignment profile is a subset $\Pi$ of $\mathcal{A} \times \mathcal{T}$, i.e., $\Pi \subset \mathcal{A} \times \mathcal{T}$. An element in the profile $\Pi$ is an agent-task pair, $\pi = (a, \tau) \in \Pi$, meaning that agent $a$ is assigned to task $\tau$. For an assignment profile $\Pi$, we denote $T_a(\Pi) = \{\tau \in \mathcal{T} : (a,\tau) \in \Pi\}$ to be the set of tasks that agent $a \in \mathcal{A}$ has selected and denote $A_\tau(\Pi) = \{a \in \mathcal{A} : (a,\tau) \in \Pi\}$ to be the set of agents that have selected the task $\tau \in \mathcal{T}$.

Each task $\tau \in \mathcal{T}$ is associated with a *given* utility function $U_\tau : 2^{\mathcal{A}} \to \mathbb{R}$. Given an assignment profile $\Pi$, the utility of task $\tau$ is $U_\tau(A_\tau(\Pi))$, i.e. $U_\tau(\cdot)$ evaluated at the set of agents that select task $\tau$. The global utility function $U : 2^{\mathcal{A}\times\mathcal{T}} \mapsto \mathbb{R}$ is defined by the sum of task utilities,

$$U(\Pi) = \sum_{\tau \in \mathcal{T}} U_\tau(A_\tau(\Pi)). \tag{1}$$

The objective of the agent-task assignment problem is to find an assignment profile $\Pi \subset \mathcal{A} \times \mathcal{T}$ such that the global utility function $U(\Pi)$ is maximized, subject to certain constraints, which is formally given by,

$$\max_{\Pi \subset \mathcal{A}\times\mathcal{T}} \quad U(\Pi) \tag{2a}$$
$$s.t. \quad T_a(\Pi) \subset \mathcal{T}_a, \forall a \in \mathcal{A} \tag{2b}$$
$$|T_a(\Pi)| \leq 1, \forall a \in \mathcal{A}. \tag{2c}$$

Constraint (A.1b) means that agent $a$ can only select

[4] Here we mean that two agents can communicate if their admissible task sets overlap. See the formal statement in Assumption 1.

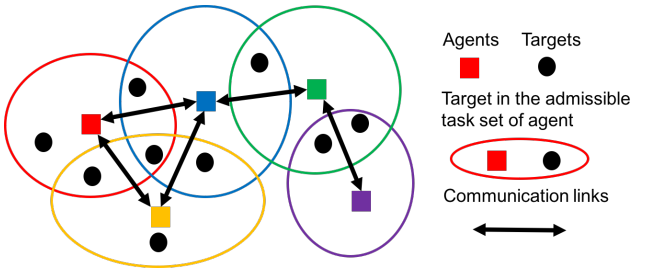


Fig. 1. Illustration of the communication graph.

tasks from its admissible task set $\mathcal{T}_a$, and constraint (A.1c) means that each agent can select at most *one* task. This can be relaxed to the case where each agent can select multiple tasks, as explained in Remark 3.

In this paper we consider the distributed solution of the optimization problem (2) where each agent selects its task based on local information and local communication. We assume each agent has access to the following information and communication links.

**Assumption 1** *Each agent has access to the following information and communication.*

- ***Local information:*** *Each agent $a$ can evaluate the utility function $U_\tau(\cdot)$ where $\tau \in \mathcal{T}_a$.*
- ***Communication graph:*** *Agent $a$ can communicate to agent $a'$ if $\mathcal{T}_a \cap \mathcal{T}_{a'} \neq \emptyset$.*

We remark here we assume the communication graph to be based on the admissible task set - two agents have a communication link if and only if their admissible task sets have a nonempty intersection (see Fig. 1 for illustration). In the applications that we consider (satellite-location, weapon-target assignment, sensor coverage), each agent and task is associated with a location in a geographical space, and the admissible task set of agent $a$ is the tasks that are geographically close to $a$, i.e. $\mathcal{T}_a = \{\tau \in \mathcal{T} : \text{dist}(\tau, a) \leq r\}$ for some $r > 0$ that is a distance threshold. In this case, when $\mathcal{T}_a \cap \mathcal{T}_{a'} \neq \emptyset$, we have $\text{dist}(a, a') \leq 2r$. This means that our assumption on communication graph is similar to that of $2r$-disk proximity graph [23, Page 78] widely adopted in multi-agent control literature. In addition, we provide another scenario in which our assumption on the communication holds. When the "task" can serve as a "communication relay", then agents do not need to directly talk to other agents but through tasks in its admissible task set instead. In this case, agents can communication if they share common admissible tasks.

### *2.2 Submodular Utility Function*

In this paper, we assume that the task utility functions satisfy the following assumption.

**Assumption 2** *For each task $\tau$, the utility function $U_\tau(\cdot)$ satisfies the following conditions.*

*(a) Nondecreasing:* $U_\tau(\tilde{A}) \leq U_\tau(A)$, $\forall \tilde{A} \subset A \subset \mathcal{A}$.

*(b) Submodular:* $U_\tau(A \cup \{a\}) - U_\tau(A) \leq U_\tau(\tilde{A} \cup \{a\}) - U_\tau(\tilde{A})$, $\forall \tilde{A} \subset A \subset \mathcal{A}$ *and* $a \in \mathcal{A}$.

*(c) Normal:* $U_\tau(\emptyset) = 0$. □

Condition (a) means that the utility of a task is nondecreasing as more agents select the task. Condition (b) is the submodular property, which says that the more agents that select a task, the less marginal utility can be brought by adding a new agent to this task. This assumption is based on the empirical belief that agents have overlap in their functionality. As a result, if $\tilde{A}$ is enlarged to $A$, there will be more functionality overlap between the additional agent $a$ and existing agents, and hence the marginal utility brought by $a$ would decrease. Condition (c) means that when a task is not selected by any agent, its utility is zero. As a result of condition (a) and (c), the utility will always be nonnegative. We also introduce the following definition of curvature to further differentiate different types of submodular functions, which has been introduced in [7].

**Definition 1** *(a) The curvature $\kappa_\tau$ of $U_\tau$ is defined as,*

$$\kappa_\tau = \max_{a \in \mathcal{A}: U_\tau(\{a\}) > 0} 1 - \frac{U_\tau(\mathcal{A}) - U_\tau(\mathcal{A}/\{a\})}{U_\tau(\{a\})}.$$

*(b) Define $\kappa = \max_{\tau \in \mathcal{T}} \kappa_\tau$.*

Per Assumption 2, it is straightforward to check that both $\kappa_\tau$ and $\kappa$ lie within $[0, 1]$. The curvature $\kappa_\tau$ (and $\kappa$) characterizes how close $U_\tau(\cdot)$ is to a "modular (additive) function", $U_\tau(A) = \sum_{a \in A} U_\tau(\{a\})$. We will show in Section 3 that $\kappa$ will directly affect the performance of our algorithm.

Submodular functions are usually referred to as the discrete counterpart to concave functions. Maximizing a submodular function has been systematically studied in the literature e.g., [10,25,11]. However, while maximizing a concave function can be solved efficiently, maximizing a submodular function is NP-hard. A widely-used class of algorithms called greedy algorithms can approximate the optimum solution with certain efficiency guarantees. For more details, we refer to [16].

### *2.3 Global Greedy Algorithm*

Problem (2) can be viewed as a variant of Problem (1.2) in [11] and it is a matroid constrained submodular maximization problem, [5] which can be solved by a greedy algorithm with bounded efficiency ratio. The greedy algorithm is provided in Algorithm 1.

This algorithm can be interpreted as follows. It starts from an empty assignment profile, and in each step, the assignment pair that brings the largest increase in the global utility is added while satisfying the constraints in (2). Because this algorithm requires a global coordinator to select the agent-task pair using all the utility

[5] Matroid constraint is a special class of set constraints and it is frequently used in combinatorial optimization [25].

**Algorithm 1** Global Greedy Algorithm

1: $k \leftarrow 0$
2: $\Pi^0 \leftarrow \emptyset$
3: $V^0 \leftarrow \{\pi \in \mathcal{A}\times\mathcal{T} : \{\pi\}$ satisfies (A.1b) and (A.1c)$\}$
4: **while** $V^k \neq \emptyset$ **do**
5: $\quad \pi_k \leftarrow \arg\max_{\pi\in V^k} U(\Pi^k \cup \{\pi\}) - U(\Pi^k)$
6: $\quad \Pi^{k+1} \leftarrow \Pi^k \cup \{\pi_k\}$
7: $\quad V^{k+1} \leftarrow \{\pi \in \mathcal{A}\times\mathcal{T} - \Pi^{k+1} :$
$\quad\quad \Pi^{k+1} \cup \{\pi\}$ satisfies (A.1b) and(A.1c)$\}$
8: $\quad k \leftarrow k+1$
9: Output assignment profile $\Pi^k$.

information, we call this algorithm as *Global Greedy Algorithm*. The algorithm has the following performance guarantee, proved in [7, Thm. 2.3]. [6]

**Theorem 1** *Assume Assumption 2 holds. Let $\Pi^*$ be an optimal assignment profile of problem (2), and $\Pi^G$ be the assignment profile produced by the global greedy algorithm. Then the efficiency ratio $r(\Pi^G) := \frac{U(\Pi^G)}{U(\Pi^*)} \geq \frac{1}{1+\kappa}$. Moreover, the bound is tight in the sense that there exist scenarios of problem (2) such that the equality holds.* □

## 3 A Distributed Greedy Algorithm

To remove the global coordinator in the global greedy algorithm, in this section we propose a distributed variant of the global greedy algorithm, which is named as the distributed greedy algorithm.

### *3.1 Distributed Greedy Algorithm*

Our algorithm does not require synchronous updating of agents, meaning that at each time step $k$, an agent can be active or inactive. Being active means that the agent will perform the decision making procedures as shown in Algorithm 2; otherwise the agent will do nothing except for passively receiving information.

In Algorithm 2, if agent $a$ is active at time step $k$, it will perform the following three main stages to decide whether to select a task or not and which task to select.

(1) **Preliminary Selection**. Agent $a$ determines $\tilde{\tau}_a^k$, which is the task that agent $a$ can bring the largest marginal utility. We refer to this process as "agent $a$ preselects $\tilde{\tau}_a^k$" because $\tilde{\tau}_a^k$ is essentially what agent $a$ would like to select, but it needs to perform the next two steps to determine whether it can actually select this task.

(2) **Conflict Resolution**. In this stage agent $a$ communicates with other agents to identify who have conflicts with it, namely those agents who are active and also preselect the same task $\tilde{\tau}_a^k$. We define this conflict set as $C_a^k$. Then, agent $a$ and agents in $C_a^k$ negotiate to let one of them actually select $\tilde{\tau}_a^k$, while others do *not* select any task in this time step. We remark here that in this algorithm we keep the negotiation rule flexible, meaning that it does not matter who will be the agent that selects $\tilde{\tau}_a^k$. Though the negotiation rule should affect the efficiency of the algorithm, as proven in our analysis it will not affect the worst-case efficiency ratio which is the focus of this paper.

(3) **Selection Implementation**. If in the conflict resolution stage agent $a$ is elected to select $\tilde{\tau}_a^k$, it implements the selection, informs other agents of the new selection and terminates. If $a$ is *not* elected to select $\tilde{\tau}_a^k$, it will repeat this 3-stage process during the next active iteration.

[6] An earlier and weaker bound 1/2 can be found in [11, Thm. 2.1].

**Algorithm 2** Distributed Greedy Algorithm

**For each agent $a$, do the following:**
1: $k \leftarrow 0$
2: **while** true **do**
3: $\quad$ **if** agent $a$ is active at time $k$ **then**
$\quad\quad$ ▷ **Preliminary Selection:**
4: $\quad\quad$ **for** $\tau \in \mathcal{T}_a$ **do**
5: $\quad\quad\quad A_\tau^k \leftarrow$ the set of agents that have already selected $\tau$ in previous time steps.
6: $\quad\quad\quad \tilde{\rho}_{(a,\tau)}^k \leftarrow U_\tau(A_\tau^k \cup \{a\}) - U_\tau(A_\tau^k)$
7: $\quad\quad \tilde{\tau}_a^k \leftarrow \arg\max_{\tau\in\mathcal{T}_a} \tilde{\rho}_{(a,\tau)}^k$
$\quad\quad$ ▷ **Conflict Resolution:**
8: $\quad\quad$ Communicate with the agents in $\{a' \in \mathcal{A}, \tilde{\tau}_a^k \in \mathcal{T}_{a'}\}$, to identify the conflict set $C_a^k \subset \{a' \in \mathcal{A}, \tilde{\tau}_a^k \in \mathcal{T}_{a'}\}$ to be the agents $a'$ who hasn't terminated yet and is active at step $k$, and satisfy $\tilde{\tau}_{a'}^k = \tilde{\tau}_a^k$.
9: $\quad\quad$ Communicate with the agents in $C_a^k$, to arbitrarily elect one of them to select $\tilde{\tau}_a^k$.
$\quad\quad$ ▷ **Selection Implementation:**
10: $\quad\quad$ **if** $a$ is elected to select $\tilde{\tau}_a^k$ **then**
11: $\quad\quad\quad a$ implements the selection of $\tilde{\tau}_a^k$.
12: $\quad\quad\quad a$ informs agents in $\{a' \in \mathcal{A}, \tilde{\tau}_a^k \in \mathcal{T}_{a'}\}$ of the new assignment pair $(a, \tilde{\tau}_a^k)$.
13: $\quad\quad\quad$ **Terminate.**
14: $\quad k \leftarrow k+1$

We now analyze the information and communication needs in Algorithm 2 and show they are indeed *local*, satisfying Assumption 1.

**Information.** During the time step $k$ when agent $a$ is active, the information agent $a$ needs is $U_\tau(A_\tau^k \cup \{a\}) - U_\tau(A_\tau^k), \forall\tau \in \mathcal{T}_a$ (Line 5 and 6). The set function $U_\tau(\cdot)$ is local information since $\tau \in \mathcal{T}_a$. We claim that $a$ also knows $A_\tau^k$, since whenever an agent $a'$ selected $\tau$ in a previous time step, $a'$ would have already informed $a$ of its selection of $\tau$ in the previous step (Line 12).

**Communication.** During the time step $k$ when agent $a$ is active, agent $a$ needs to communicate to the agents whose admissible task set includes the preselected task

$\tilde{\tau}_a^k$, i.e., $\{a' \in \mathcal{A}, \tilde{\tau}_a^k \in \mathcal{T}_{a'}\}$ (Line 8, 9 and 12), which satisfies our definition for local communication.

**Remark 1** At each time step, no two agents select the same task due to the conflict resolution stage. As shown in Section 3.2, this is the key to ensure the efficiency guarantee of the algorithm.

We make the following assumption that each agent has to be active at least once in a while.

**Assumption 3** *Each agent $a$ is active at least once within $W_a$ consecutive time steps and $W_a$ is bounded, i.e., there exists some $W$, s.t. $W_a \leq W, \forall a$.*

The following theorem shows that the distributed greedy algorithm has the same performance guarantee as the global one.

**Theorem 2** *Under Assumption 2 and 3, every agent will terminate the algorithm within $WN$ steps. Let $\Pi^D$ be the assignment profile when the algorithm terminates, and $\Pi^*$ be an optimal assignment profile of problem (2). Then the efficiency ratio $r(\Pi^D)$ is lower bounded, $r(\Pi^D) := \frac{U(\Pi^D)}{U(\Pi^*)} \geq \frac{1}{1+\kappa}$. Moreover, the bound is tight in the sense that there exist cases of problem (2) such that the bound is approximated as close as possible.* □

**Remark 2** *Since $\kappa \in [0, 1]$, we have $\frac{1}{1+\kappa} \geq \frac{1}{2}$. Therefore, our algorithm achieves a worst-case $1/2$ efficiency ratio.*

**Remark 3** *The algorithm can be extended to the multi-selection case (each agent $a$ selects up to $n_a \geq 1$ tasks). Here we provide two ways. In the first method, we can split each agent into $n_a$ "pseudo" sub-agents, and each sub-agent has the same admissible task set as the original agent, and each sub-agent independently runs the algorithm. When the algorithm terminates, each sub-agent is assigned with a task, and the "actual" agent is assigned with $n_a$ tasks. In the second method, the agent does not terminate the algorithm after the first task is selected; instead, the agent continues running the algorithm to select the next task, while fixing the selections it has already made, until all $n_a$ tasks have been selected. For more details, see our online report [14, Appendix-A].*

### *3.2 Proof of Theorem 2*

We will first prove that the algorithm will terminate within finite time steps and $U(\Pi^*) \leq (1+\kappa)U(\Pi^D)$. Then we will show the bound is tight. In the rest of this section, we will frequently refer to the marginal utility brought by an additional agent-task pair $\pi \in \mathcal{A} \times \mathcal{T}$ to an assignment profile $\Pi \subset \mathcal{A} \times \mathcal{T}$. Hence we introduce the following notation for the marginal utility,

$$\rho_\pi(\Pi) = U(\Pi \cup \{\pi\}) - U(\Pi), \text{ for } \Pi \subset \mathcal{A} \times \mathcal{T}, \pi \in \mathcal{A} \times \mathcal{T}.$$

#### *3.2.1 Proof of the bound*

To prove the bound, we first present three lemmas. The first lemma proves an property of marginal increase of $U_\tau(\cdot)$ that involves the curvature parameter $\kappa$.

**Lemma 1** *For any $A \subset \tilde{A} \subsetneq \mathcal{A}$, and $a \in \mathcal{A}/\tilde{A}$,*

$$\begin{aligned} &U_\tau(\tilde{A} \cup \{a\}) - U_\tau(\tilde{A}) \\ &\geq (1-\kappa)(U_\tau(A \cup \{a\}) - U_\tau(A)). \end{aligned} \tag{3}$$

**Proof**

$$\begin{aligned} U_\tau(\tilde{A} \cup \{a\}) - U_\tau(\tilde{A}) &\overset{(a)}{\geq} U_\tau(\mathcal{A}) - U_\tau(\mathcal{A}/\{a\}) \\ &\overset{(b)}{\geq} (1-\kappa_\tau)(U_\tau(\{a\}) - U_\tau(\emptyset)) \\ &\overset{(c)}{\geq} (1-\kappa_\tau)(U_\tau(A \cup \{a\}) - U_\tau(A)) \\ &\overset{(d)}{\geq} (1-\kappa)(U_\tau(A \cup \{a\}) - U_\tau(A)) \end{aligned}$$

where (a) and (c) are due to submodularity, and $\tilde{A} \subset \mathcal{A}/\{a\}$ (for (a)), $\emptyset \subset A$ (for (c)). Ineq. (b) is due to the definition of $\kappa_\tau$ (see Def. 1 in Sec. 2), and (d) is due to $\kappa_\tau \leq \kappa$. □

**Lemma 2** *The global utility function $U$ defined in (1) is submodular and nondecreasing, and satisfies $U(\emptyset) = 0$.*

**Proof** For submodularity, consider $\tilde{\Pi} \subset \Pi \subset \mathcal{A} \times \mathcal{T}$, and consider $\pi = (a, \tau) \in \mathcal{A} \times \mathcal{T}$. Then,

$$\begin{aligned} U(\Pi \cup \{\pi\}) - U(\Pi) &\overset{(a)}{=} U_\tau(\{a\} \cup A_\tau(\Pi)) - U_\tau(A_\tau(\Pi)) \\ &\overset{(b)}{\leq} U_\tau(\{a\} \cup A_\tau(\tilde{\Pi})) - U_\tau(A_\tau(\tilde{\Pi})) \\ &\overset{(c)}{=} U(\tilde{\Pi} \cup \{\pi\}) - U(\tilde{\Pi}) \end{aligned} \tag{4}$$

where (a) and (c) are due to the definition of $U$ (equation (1)), and (b) is due to the submodularity of $U_\tau$. For monotinicity, for any $\Pi \subset \mathcal{A} \times \mathcal{T}$ and $\pi = (a, \tau) \in \mathcal{A} \times \mathcal{T}$, we have, $U(\Pi \cup \{\pi\}) - U(\Pi) = U_\tau(\{a\} \cup A_\tau(\Pi)) - U_\tau(A_\tau(\Pi)) \geq 0$. At last, $U(\emptyset) = 0$ follows from $U_\tau(\emptyset) = 0$ for each $\tau \in \mathcal{T}$. □

The following Lemma 3 presents an inequality that holds for general nondecreasing submodular functions.

**Lemma 3** $U(\Pi \cup \tilde{\Pi}) \leq U(\tilde{\Pi}) + \sum_{\pi \in \Pi - \tilde{\Pi}} \rho_\pi(\tilde{\Pi}), \forall \Pi, \tilde{\Pi} \subset \mathcal{A} \times \mathcal{T}$.

**Proof** Let $\Pi - \tilde{\Pi} = \{\pi_1, \pi_2, \ldots, \pi_m\}$. Let $\tilde{\Pi}_k = \tilde{\Pi} \cup \{\pi_1, \pi_2, \ldots, \pi_k\}$ for $k = 1, 2, \ldots, m$. Let $\tilde{\Pi}_0 = \tilde{\Pi}$. Note that $\tilde{\Pi}_m = \tilde{\Pi} \cup (\Pi - \tilde{\Pi}) = \Pi \cup \tilde{\Pi}$. Then,

$$\begin{aligned} U(\Pi \cup \tilde{\Pi}) - U(\tilde{\Pi}) &= \sum_{k=0}^{m-1} (U(\tilde{\Pi}_{k+1}) - U(\tilde{\Pi}_k)) \\ &\overset{(a)}{=} \sum_{k=0}^{m-1} \rho_{\pi_{k+1}}(\tilde{\Pi}_k) \\ &\overset{(b)}{\leq} \sum_{k=0}^{m-1} \rho_{\pi_{k+1}}(\tilde{\Pi}) = \sum_{\pi \in \Pi - \tilde{\Pi}} \rho_\pi(\tilde{\Pi}) \end{aligned}$$

where (a) is due to $\tilde{\Pi}_{k+1} = \tilde{\Pi}_k \cup \{\pi_{k+1}\}$ and the definition of $\rho_{\pi_{k+1}}(\tilde{\Pi}_k)$; (b) follows from $\tilde{\Pi} \subset \tilde{\Pi}_k$ and $U$ is submodular. □

We are now ready to prove the bound.

**Proof** We first show that the algorithm will terminate within at most $WN$ time steps. Let $\Pi^k$ be the assignment profile at the beginning of time step $k$. It is easy to see that, if there is at least one agent active at time step $k$, then $|\Pi^{k+1}| - |\Pi^k| \geq 1$. Combining this with the fact that at least one agent will be active at least once within any time window of length $W$ and the fact that $|\Pi^k|$ cannot exceed $N$, we have that all the agents must terminate within $WN$ time steps.

Let $K$ be the time step in which the last assignment is made. Then the final assignment profile generated by the distributed greedy algorithm is $\Pi^D = \Pi^{K+1}$.

Let $\rho_k = U(\Pi^{k+1}) - U(\Pi^k)$, for $k = 0, 1, \ldots, K$. Let $\Delta^k = \{a : \exists \tau \in \mathcal{T}_a \text{ s.t. } (a, \tau) \in \Pi^{k+1} - \Pi^k\}$, i.e. the set of agents that make their selections in time step $k$. We also let $\tau_a^D$ be the selection of $a$ in the greedy assignment profile $\Pi^D$, i.e. $\Pi^D = \cup_{a \in \mathcal{A}} \{(a, \tau_a^D)\}$. We have the following facts about $\Delta^k$.

(i) The sets $\{\Delta^k : k = 0, 1, \ldots, K\}$ are disjoint. This is because by definition $\Delta^k$ is the set of agents that make their selections at time step $k$, and each agent can make its selection only once.

(ii) $\cup_{k=0}^{K} \Delta^k = \mathcal{A}$. This is because all the agents must have made their selections when the algorithm terminates.

(iii) For $a \in \Delta^k$, $\tau_a^D = \arg\max_{\tau \in \mathcal{T}_a} \rho_{(a,\tau)}(\Pi^k)$. To see this, since $a \in \Delta^k$, $a$ finalizes its selection in iteration $k$ and its selection is $\tau_a^D = \tilde{\tau}_a^k$ (Line 11 of the algorithm). Next, for any $\tau \in \mathcal{T}_a$, $A_\tau^k = A_\tau(\Pi^k)$ (Line 5). Hence, by Line 7, $\tau_a^D = \tilde{\tau}_a^k = \arg\max_{\tau \in \mathcal{T}_a} U_\tau(A_\tau^k \cup \{a\}) - U_\tau(A_\tau^k) = \arg\max_{\tau \in \mathcal{T}_a} U_\tau(A_\tau(\Pi^k) \cup \{a\}) - U_\tau(A_\tau(\Pi^k)) = \arg\max_{\tau \in \mathcal{T}_a} \rho_{(a,\tau)}(\Pi^k)$.

(iv) Because of the conflict-resolution stage (Line 8 and 9), for two different agents $a, a' \in \Delta^k$, their selections must be different, i.e. $\tau_a^D \neq \tau_{a'}^D$. Hence, $\rho_k = U(\Pi^{k+1}) - U(\Pi^k) = \sum_{a \in \Delta^k} U_{\tau_a^D}(A_{\tau_a^D}(\Pi^k) \cup \{a\}) - U_{\tau_a^D}(A_{\tau_a^D}(\Pi^k)) = \sum_{a \in \Delta^k} \rho_{(a,\tau_a^D)}(\Pi^k)$.

Without loss of generality, we assume in the optimal assignment profile $\Pi^*$ each agent $a$ has to select a task (if not, we can assign a task to the agents that haven't selected any task while we don't decrease the global utility). Denote the task that $a$ selects in $\Pi^*$ be $\tau_a^*$. By Lemma 3, we have

$$U(\Pi^* \cup \Pi^D) \leq U(\Pi^D) + \sum_{\pi \in \Pi^* - \Pi^D} \rho_\pi(\Pi^D) \tag{5}$$

$$= U(\Pi^D) + \sum_{\substack{a \in \mathcal{A} \\ (a,\tau_a^*) \notin \Pi^D}} \rho_{(a,\tau_a^*)}(\Pi^D) \tag{6}$$

$$= U(\Pi^D) + \sum_{k=0}^{K} \sum_{\substack{a \in \Delta^k \\ \tau_a^* \neq \tau_a^D}} \rho_{(a,\tau_a^*)}(\Pi^D) \tag{7}$$

$$\leq U(\Pi^D) + \sum_{k=0}^{K} \sum_{\substack{a \in \Delta^k \\ \tau_a^* \neq \tau_a^D}} \rho_{(a,\tau_a^*)}(\Pi^k) \tag{8}$$

$$\leq U(\Pi^D) + \sum_{k=0}^{K} \sum_{\substack{a \in \Delta^k \\ \tau_a^* \neq \tau_a^D}} \rho_{(a,\tau_a^D)}(\Pi^k) \tag{9}$$

Equality (6) is a restatement of (5) using the fact that $\Pi^* = \cup_{a \in \mathcal{A}} \{(a, \tau_a^*)\}$. (7) is due to Facts (i-ii) ($\mathcal{A}$ is a disjoint union of $\Delta^k$ where $k$ ranges from 0 to $K$), as well as the fact that $(a, \tau_a^*) \notin \Pi^D \iff \tau_a^* \neq \tau_a^D$. (8) follows from the submodularity of $U$ and $\Pi^k \subset \Pi^D$. (9) follows from $\tau_a^* \in \mathcal{T}_a$ and fact (iii), the maximality of $\tau_a^D$.

Next, define $\hat{\Pi}^k = \Pi^* \cup \Pi^k$, and then $\hat{\Pi}^0 = \Pi^*$, $\hat{\Pi}^{K+1} = \Pi^* \cup \Pi^D$. Hence,

$$\begin{aligned}
&U(\Pi^* \cup \Pi^D) \\
&= U(\Pi^*) + \sum_{k=0}^{K} (U(\hat{\Pi}^{k+1}) - U(\hat{\Pi}^k)) \\
&= U(\Pi^*) \\
&+ \sum_{k=0}^{K} \sum_{\substack{a \in \Delta^k \\ \tau_a^* \neq \tau_a^D}} \left[ U_{\tau_a^D}(A_{\tau_a^D}(\hat{\Pi}^k) \cup \{a\}) - U_{\tau_a^D}(A_{\tau_a^D}(\hat{\Pi}^k)) \right]
\end{aligned} \tag{10}$$

$$\begin{aligned}
&\geq U(\Pi^*) + (1 - \kappa) \\
&\times \sum_{k=0}^{K} \sum_{\substack{a \in \Delta^k \\ \tau_a^* \neq \tau_a^D}} \left[ U_{\tau_a^D}(A_{\tau_a^D}(\Pi^k) \cup \{a\}) - U_{\tau_a^D}(A_{\tau_a^D}(\Pi^k)) \right]
\end{aligned} \tag{11}$$

$$= U(\Pi^*) + (1 - \kappa) \sum_{k=0}^{K} \sum_{\substack{a \in \Delta^k \\ \tau_a^* \neq \tau_a^D}} \rho_{(a,\tau_a^D)}(\Pi^k) \tag{12}$$

where (10) is because $\hat{\Pi}^{k+1} - \hat{\Pi}^k = (\Pi^{k+1} - \Pi^k) - \Pi^* = \{(a, \tau_a^D) : a \in \Delta^k, \tau_a^D \neq \tau_a^*\}$ and the fact that for any two distinct $a, a' \in \Delta^k$, $\tau_a^D \neq \tau_{a'}^D$ (fact (iv)). Inequality (11) is due to the property associated with the curvature (Lemma 1), and $A_{\tau_a^D}(\hat{\Pi}^k) \supset A_{\tau_a^D}(\Pi^k)$. Equality (12) is due to the conflict resolution step, for any two distinct $a, a' \in \Delta^k$, $\tau_a^D \neq \tau_{a'}^D$ (fact (iv)).

Combining (12) and (9), we have

$$U(\Pi^*) \leq U(\Pi^D) + \kappa \sum_{k=0}^{K} \sum_{\substack{a \in \Delta^k \\ \tau_a^* \neq \tau_a^D}} \rho_{(a,\tau_a^D)}(\Pi^k)$$

$$
\begin{aligned}
&\le U(\Pi^D) + \kappa \sum_{k=0}^{K} \sum_{a \in \Delta^k} \rho_{(a,\tau_a^D)}(\Pi^k) \\
&= U(\Pi^D) + \kappa \sum_{k=0}^{K} \rho_k \qquad (13) \\
&= (1+\kappa) U(\Pi^D)
\end{aligned}
$$

where in (13) we have used fact (iv). So we are done.

### *3.2.2 Proof of the tightness of the bound*

For $\kappa = 0$, the bound is always tight, and for $\kappa = 1$ the tightness is discussed in our conference paper [27]. So we assume $\kappa \in (0,1)$. Consider two tasks $\mathcal{T} = \{\tau, \tilde{\tau}\}$, and $2N$ agents $\mathcal{A} = \{a_1, a_2, \ldots, a_N, \tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_N\}$. The admissible task set of all agents are set to be $\mathcal{T}$, i.e. all the tasks. Define $\gamma_i = \frac{(1-\kappa)^{i-1}\kappa}{1+\kappa}$ $(i = 1, 2, \ldots)$. Now we define the task utility functions $U_\tau$ and $U_{\tilde{\tau}}$. For $A \subset \mathcal{A}$, define

$$
\begin{aligned}
U_\tau(A) &= \sum_{i:a_i \in A} \gamma_i + \sum_{\substack{i:\tilde{a}_i \in A \text{ s.t.} \\ i=1 \text{ or } a_{i-1} \in A}} \gamma_i + \sum_{\substack{i:\tilde{a}_i \in A \text{ s.t.} \\ i>1, a_{i-1} \notin A}} \frac{1}{1-\kappa}\gamma_i \\
U_{\tilde{\tau}}(A) &= \sum_{i:\tilde{a}_i \in A} \gamma_i + \sum_{\substack{i:a_i \in A \text{ s.t.} \\ i=1 \text{ or } \tilde{a}_{i-1} \in A}} \gamma_i + \sum_{\substack{i:a_i \in A \text{ s.t.} \\ i>1, \tilde{a}_{i-1} \notin A}} \frac{1}{1-\kappa}\gamma_i.
\end{aligned}
$$

The functions are the same as the one used in [7, Pf. of Thm. 2.12], and are nondecreasing, submodular and normal (satisfying Assumption 2) with curvature $\kappa$. For completeness we include a proof of these properties in our online report [14, Appendix-B].

We claim that when we run the distributed greedy algorithm, after the $(k-1)$'th iteration, agent $a_1, \ldots, a_k$ selects $\tau$, agent $\tilde{a}_1, \ldots, \tilde{a}_k$ selects $\tilde{\tau}$, and other agents remain unassigned. We now show the claim for $k = 1$. In the 0'th iteration, $U_\tau(\{a_1\}) - U_\tau(\emptyset) = \gamma_1$, $U_{\tilde{\tau}}(\{a_1\}) - U_{\tilde{\tau}}(\emptyset) = \gamma_1$, so we can make $a_1$ preselect $\tau$ (since ties can be settled arbitrarily in Line 7 of the algorithm); for the same reason, we can make $\tilde{a}_1$ preselect $\tilde{\tau}$. Regardless of what other agents preselect, we can make $a_1$ and $\tilde{a}_1$ implement their selections (since conflict can be resolved arbitrarily in Line 9 of the algorithm) and let other agents remain unassigned. So we have proven the claim for $k = 1$.

Let the claim hold for $k$. Then in the $k$'th iteration, $a_{k+1}$ calculates the marginal increase it can bring to the two tasks:

$$
\begin{aligned}
U_\tau(\{a_1, \ldots, a_{k+1}\}) - U_\tau(\{a_1, \ldots, a_k\}) &= \gamma_{k+1} \\
U_{\tilde{\tau}}(\{\tilde{a}_1, \ldots, \tilde{a}_k, a_{k+1}\}) - U_{\tilde{\tau}}(\{\tilde{a}_1, \ldots, \tilde{a}_k\}) &= \gamma_{k+1}.
\end{aligned}
$$

Hence, $a_{k+1}$ preselects $\tau$. Similarly, $\tilde{a}_{k+1}$ preselects $\tilde{\tau}$. Again, regardless of what others preselect, we can make $a_{k+1}$ and $\tilde{a}_{k+1}$ implement their selection. Hence the claim is true for $k+1$.

Due to the above reasoning, the final assignment profile of our algorithm $\Pi^D$ is such that $a_k$ selects $\tau$ and $\tilde{a}_k$ selects $\tilde{\tau}$. The resulting global utility is

$$
\begin{aligned}
U(\Pi^D) &= U_\tau(\{a_1, \ldots, a_N\}) + U_{\tilde{\tau}}(\{\tilde{a}_1, \ldots, \tilde{a}_N\}) \\
&= 2\sum_{i=1}^{N} \gamma_i = 2\frac{1-(1-\kappa)^N}{1+\kappa}.
\end{aligned}
$$

The optimal assignment profile $\Pi^*$ is such that $a_k$ selects $\tilde{\tau}$ and $\tilde{a}_k$ selects $\tau$. The resulting global utility is,

$$
\begin{aligned}
U(\Pi^*) &= U_\tau(\{\tilde{a}_1, \ldots, \tilde{a}_N\}) + U_{\tilde{\tau}}(\{a_1, \ldots, a_N\}) \\
&= 2(\gamma_1 + \frac{1}{1-\kappa}\sum_{i=2}^{N} \gamma_i) = 2(1 - \frac{(1-\kappa)^{N-1}}{1+\kappa}).
\end{aligned}
$$

So the efficiency ratio is $\frac{1}{1+\kappa}\frac{1-(1-\kappa)^N}{1-\frac{(1-\kappa)^{N-1}}{1+\kappa}}$, which converges to $\frac{1}{1+\kappa}$ when $N \to \infty$.

## 4 Case Study

We use the weapon-target model [24] to numerically test our distributed greedy algorithm. Here agents are weapons and tasks are targets. We consider $N$ agents and $M$ tasks, both of which are embedded as points in a three dimensional box $[0,1]^3$, and we generate them uniformly random. We define $\mathcal{T}_a = \{\tau \in \mathcal{T} : dist(a,\tau) < r_a\}$, where $dist(\cdot,\cdot)$ means Euclidean distance between two points, and $r_a$ is a parameter. This definition means that $\mathcal{T}_a$ includes all tasks whose distances to $a$ are less than $r_a$. For each $a$ and $\tau \in \mathcal{T}_a$, define the probability of $a$ failing to destroy $\tau$ as $p_{a,\tau} = \frac{1}{1+\exp(-\alpha_a dist(a,\tau)+\beta_a)}$ which is an increasing function of the distance between $a$ and $\tau$. The parameters $\alpha_a, \beta_a$ characterizes how the failure probability increases wr.t. the distance between $a$ and target $\tau$. The utility of a target $\tau$ is the probability that it gets destroyed times the value of $\tau$, $V_\tau$. Formally, the utility is $U_\tau(A) = V_\tau(1 - \prod_{a \in A: \mathcal{T}_a \ni \tau} p_{a,\tau})$. We can verify that the utility function satisfies Assumption 2. Moreover, for this particular problem we can calculate the curvature $\kappa$ (Definition 1),

$$
\begin{aligned}
\kappa &= \max_{\tau \in \mathcal{T}} \max_{a \in A: \mathcal{T}_a \ni \tau} 1 - \frac{U_\tau(\mathcal{A}) - U_\tau(\mathcal{A}/\{a\})}{U_\tau(\{a\})} \\
&= \max_{\tau \in \mathcal{T}} \max_{a \in A: \mathcal{T}_a \ni \tau} 1 - \prod_{\substack{a' \in \mathcal{A}: \\ \text{s.t. } \mathcal{T}_{a'} \ni \tau, a' \neq a}} p_{a',\tau}. \qquad (14)
\end{aligned}
$$

For each agent $a$, we draw $W_a$ uniformly from $\{1, \ldots, 10\}$, and after each time step $a$ is active, the next time step that $a$ is active is determined by a random draw from the next $W_a$ time steps. As for conflict resolution, we let the agents resolve conflicts uniformly random.

We test two cases. For case I, we choose $N = 20$, $M = 20$, $r_a \sim [0.2, 0.4]$, $\alpha_a \sim [0.2, 0.5]$, $\beta_a \sim [-2, -1]$, $V_\tau \sim$

$[10, 100]$, where "$\sim$" means "uniformly sampled from". Case I is a small scale system, for which we can compute the optimal solution through brutal force search. We do 100 runs with random parameters and plot the efficiency ratio of the resulting solution vs. theoretical efficiency ratio bound $\frac{1}{1+\kappa}$ (calculated using (14)) in Figure 2. The average running time of the distributed greedy algorithm is 0.0025s, and the average time to compute the exact solution through brutal force is 30.33s. [7] From Fig. 2, we can see that the efficiency ratio is indeed better than the theoretic value $\frac{1}{1+\kappa}$, confirming the statement of Thm. 2.

For case II, we choose $N = 100$, $M = 200$, $r_a \sim [0.1, 0.2]$, $\alpha_a \sim [1, 2]$, $\beta_a \sim [2, 3]$, $V_\tau \sim [10, 100]$. Case II is a large scale system, whose optimal solution is intractable to compute. Hence we compute an *approximated efficiency ratio* instead, which is the ratio of the distributed greedy solution to a convex relaxation of the original problem (see RSWTA-URR in [24]). The optimal value of the convex relaxation is an upper bound of the optimum value of the original problem, therefore the approximated efficiency ratio is a lower bound for the true efficiency ratio. Apart from the convex relaxation, we also compare our algorithm with the global greedy algorithm and compute the ratio of the distributed greedy solution to the global greedy solution. We do 100 runs for case II, and plot the approximated efficiency ratio in Fig. 3 (a), and the distributed greedy to global greedy ratio in Fig. 3 (b). The average running time of the convex relaxation method [8] is 23.7s, of distributed greedy algorithm is 0.014s, of global greedy algorithm is 0.095s. For the parameter ranges of this case, the bound $\frac{1}{1+\kappa}$ becomes very close to $1/2$ (the largest among the 100 runs is 0.5004). From Fig. 3 (a) we can see that the efficiency ratios of the proposed algorithm are all larger than 0.7, confirming the statement of Theorem 2. Moreover, from Figure 3 (b) the distributed greedy algorithm has a similar performance as the global greedy algorithm.

At last, we compare our algorithm with the game design approach in [1]. In particular, we select the agent utility to be the marginal contribution (Wonder Life Utility [1, Sec. 3.4]), and we test two learning algorithms, the regret matching [1, Sec. 4.2] and joint strategy fictitious play (JSFP) [20]. For both algorithms we use the version with fading memory (setting $\rho = 0.05$) and inertial (setting $\alpha = 0.8$). [9] We set $N = 500, M = 1000$, and we draw parameters from $r_a \sim [0.2, 0.4]$, $\alpha_a \sim [1, 2]$,$\beta_a \sim [1, 3]$,$V_r \sim [10, 100]$. For our algorithm we set $W_a = 1$ (all agents conduct synchronous updates). We conduct one random run of the algorithms, and we plot the global utility of the algorithms versus iteration in Fig. 4. [10]

[7] All simulations are run on MATLAB 2016a using HP Z640 Workstation with Intel Xeon CPU E5-2620 v4.

[8] We solve the convex problem using CVX[13,12].

[9] For the meaning of $\rho$ and $\alpha$, please see [1, Sec. 4.2] and [20, Sec 2.4, Sec. 3].

[10] The per-iteration computation and communication cost of Regret Matching and JSFP is similar to that of the distributed greedy algorithm.

It can be seen that the regret matching converges relatively slow, while JSFP and our algorithm converge very fast and achieve similar global utility. However, our algorithm has a clear termination, while JSFP has no clear termination rules.

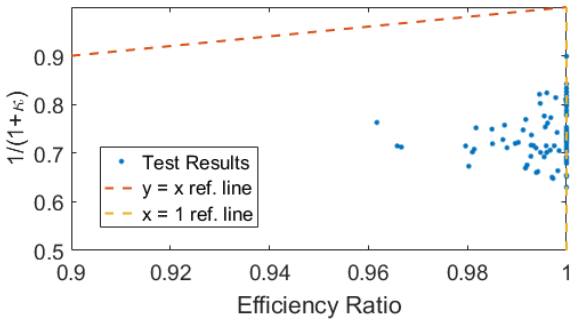


Fig. 2. Case I. Each dot represents one run of the algorithm with one realization of the problem parameters. The $x$-axis represents the efficiency ratio; while the $y$-axis is the theoretical efficiency ratio $\frac{1}{1+\kappa}$ calculated using (14).

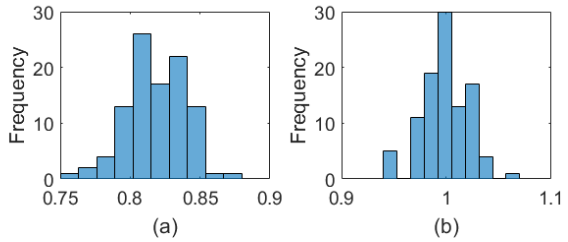


Fig. 3. Case II. Part (a) (left) is histogram for the approximated efficiency ratio (ratio of the distributed greedy solution to the convex relaxation solution). Part (b) (right) is histogram for the ratio of distributed greedy solution to global greedy solution.

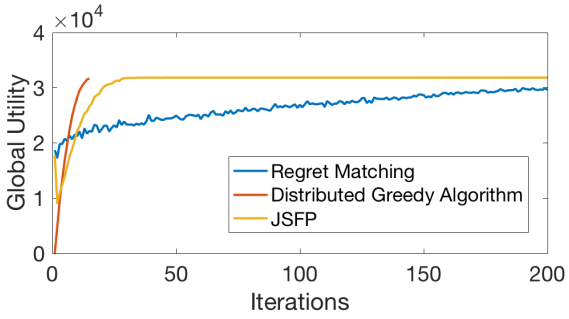


Fig. 4. Comparison of Distributed Greedy Algorithm with Regret Matching and Joint Strategy Fictitious Play. The $x$-axis is the number of iterations while the $y$-axis is the global utility.

## 5 Conclusion

In this paper, we study the problem where a set of agents select tasks to maximize a global utility function. We develop a distributed greedy algorithm whose efficiency ratio is lower bounded by $1/(1+\kappa)$, where $\kappa \in [0,1]$ is a problem dependent curvature parameter. In the future, we will explore the design of distributed algorithms that use more flexible information and communication structure, e.g., the communication is not necessarily based on the admissible task set as in this paper. We will also consider the assignment problem in a dynamic environment, in which the admissible task sets and task utilities are evolving over time.

## References


[1] Gürdal Arslan, Jason R Marden, and Jeff S Shamma. Autonomous vehicle-target assignment: A game-theoretical formulation. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):584–596, 2007.

[2] Dimitri P Bertsekas. Auction algorithms for network flow problems: A tutorial introduction. *Computational optimization and applications*, 1(1):7–66, 1992.

[3] Liad Blumrosen and Noam Nisan. Combinatorial auctions. *Algorithmic game theory*, 267:300, 2007.

[4] Béla Bollobás. *Graph theory: an introductory course*, volume 63. Springer Science & Business Media, 2012.

[5] Francesco Bullo, Jorge Cortes, and Sonia Martinez. *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton University Press, 2009.

[6] Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: Matchings, matroids, and more. *Mathematical Programming*, 154(1-2):225–247, 2015.

[7] Michele Conforti and Gérard Cornuéjols. Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the rado-edmonds theorem. *Discrete applied mathematics*, 7(3):251–274, 1984.

[8] GG denBroeder Jr, RE Ellison, and L Emerling. On optimum target assignments. *Operations Research*, 7(3):322–326, 1959.

[9] Shahar Dobzinski, Noam Nisan, and Michael Schapira. Truthful randomized mechanisms for combinatorial auctions. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 644–652. ACM, 2006.

[10] Jack Edmonds. Submodular functions, matroids, and certain polyhedra. *Edited by G. Goos, J. Hartmanis, and J. van Leeuwen*, page 11, 1970.

[11] Marshall L Fisher, George L Nemhauser, and Laurence A Wolsey. *An analysis of approximations for maximizing submodular set functionsII*. Springer, 1978.

[12] Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008.

[13] Michael Grant, Stephen Boyd, and Yinyu Ye. Cvx: Matlab software for disciplined convex programming, 2008.

[14] Na Li Guannan Qu, Dave Brown. Distributed greedy algorithm for multi-agent task assignment problem with submodular utility functionsn. https://scholar.harvard.edu/files/gqu/files/qu2018submodular.pdf. Accessed: 2018-02-19.

[15] Patrick A Hosein. A class of dynamic nonlinear resource allocation problems. Technical report, DTIC Document, 1989.

[16] Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability: Practical Approaches to Hard Problems*, 3:19, 2012.

[17] Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 55(2):270–296, 2006.

[18] Naomi Ehrich Leonard and Alex Olshevsky. Nonuniform coverage control on the line. *IEEE Transactions on Automatic Control*, 58(11):2743–2755, 2013.

[19] Jason R Marden. The role of information in distributed resource allocation. *IEEE Transactions on Control of Network Systems*, 4(3):654–664, 2017.

[20] Jason R Marden, Gürdal Arslan, and Jeff S Shamma. Joint strategy fictitious play with inertia for potential games. *IEEE Transactions on Automatic Control*, 54(2):208–220, 2009.

[21] Jason R Marden and Adam Wierman. Distributed welfare games with applications to sensor coverage. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 1708–1713. IEEE, 2008.

[22] Jason R Marden and Adam Wierman. Distributed welfare games. *Operations Research*, 61(1):155–168, 2013.

[23] Sonia Martinez, Jorge Cortes, and Francesco Bullo. Motion coordination with distributed information. *IEEE Control Systems*, 27(4):75–88, 2007.

[24] Robert A Murphey. Target-based weapon target assignment problems. In *Nonlinear Assignment Problems*, pages 39–53. Springer, 2000.

[25] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978.

[26] Yoav Shoham Peter Cramton and Richard Steinberg. *Combinatorial Auctions*. MIT Press, 2006.

[27] Guannan Qu, Dave Brown, and Na Li. Distributed greedy algorithm for satellite assignment problem with submodular utility function. *IFAC-PapersOnLine*, 48(22):258–263, 2015.

[28] Vinod Ramaswamy, Dario Paccagnan, and Jason R Marden. The impact of local information on the performance of multiagent systems. *arXiv preprint arXiv:1710.01409*, 2017.

[29] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

[30] Scott R. Starin and John Eterno. *Space mission engineering: the new SMAD*, chapter Spacecraft Attitude Determination and Control Systems. Microcosm Press, 2011.

[31] Adrian Vetta. Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 416–425. IEEE, 2002.

[32] Michael M Zavlanos, Leonid Spesivtsev, and George J Pappas. A distributed auction algorithm for the assignment problem. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 1212–1217. IEEE, 2008.


## A Extension to Multi-Selection Case

In this section, we consider the case that each agent $a$ can select up to $n_a$ tasks. The corresponding global

optimization problem is

$$\max_{\Pi \subset \mathcal{A} \times \mathcal{T}} \quad U(\Pi) \tag{A.1a}$$
$$s.t. \quad T_a(\Pi) \subset \mathcal{T}_a, \forall a \in \mathcal{A} \tag{A.1b}$$
$$|T_a(\Pi)| \leq n_a, \forall a \in \mathcal{A}. \tag{A.1c}$$

The multi-selection algorithm DistGreedyMulti is given in Algorithm 3. It is a modification of the single selection version (Algorithm 2, referred to as DistGreedy hereafter), and we highlight in red the part that is different from DistGreedy. From Algorithm 3, we can see that the difference of DistGreedyMulti compared with DistGreedy is that in DistGreedyMulti, each agent keeps track of varaible $S_a^k$ ("$S$" stands for "selection" here), which is the set of tasks that $a$ has already selected up to iteration $k$, and that $a$ keeps the algorithm running until it has selected $n_s$ tasks. We comment that selections already made by an agent

**Algorithm 3** DistGreedyMulti: Distributed Greedy Algorithm with Multiple Selections

**For each agent $a$, do the following:**

1: $k \leftarrow 0$
2: $S_a^k \leftarrow \emptyset$
3: **while** true **do**
▷ **Preliminary Selection:**
4: **for** $\tau \in \mathcal{T}_a$ **do**
5: $A_\tau^k \leftarrow$ the set of agents that have already selected $\tau$ in previous time steps.
6: $\tilde{\rho}_{(a,\tau)}^k \leftarrow U_\tau(A_\tau^k \cup \{a\}) - U_\tau(A_\tau^k)$
7: $\tilde{\tau}_a^k \leftarrow \arg\max_{\tau \in \mathcal{T}_a} \tilde{\rho}_{(a,\tau)}^k$
▷ **Conflict Resolution:**
8: Communicate with the agents in $\{a' \in \mathcal{A}, \tilde{\tau}_a^k \in \mathcal{T}_{a'}\}$, to identify the conflict set $C_a^k \subset \{a' \in \mathcal{A}, \tilde{\tau}_a^k \in \mathcal{T}_{a'}\}$ to be the agents $a'$ who hasn't terminated yet and is active at step $k$, and satisfy $\tilde{\tau}_{a'}^k = \tilde{\tau}_a^k$.
9: Communicate with the agents in $C_a^k$, to arbitrarily elect one of them to select $\tilde{\tau}_a^k$.
▷ **Selection Implementation:**
10: **if** $a$ is elected to select $\tilde{\tau}_a^k$ **then**
11: $a$ implements the selection of $\tilde{\tau}_a^k$.
12: $S_a^{k+1} \leftarrow S_a^k \cup \{\tilde{\tau}_a^k\}$
13: $a$ informs agents in $\{a' \in \mathcal{A}, \tilde{\tau}_a^k \in \mathcal{T}_{a'}\}$ of the new assignment pair $(a, \tilde{\tau}_a^k)$.
14: **if** $S_a^{k+1} = |n_s|$ **then**
15: **Terminate.**
16: $k \leftarrow k + 1$

$a$ will affect later iterations of the algorithm. In details, in Line 6 of DistGreedyMulti, $A_\tau^k$ is the set of agents that have already selected $\tau$. If $a$ itself has selected $\tau$ in previous iterations, then $a \in A_\tau^k$, which leads to $\tilde{\rho}_{a,\tau}^k = U_\tau(A_\tau^k \cup \{a\}) - U_\tau(A_\tau^k) = 0$, effectively preventing $a$ from selecting $\tau$ again.

In the following Theorem, we prove that DistGreedyMulti will achieve a 1/2-efficiency guarantee compared with the optimal solution of (A.1).

**Theorem 3** *Under Assumption 2 and 3, all the agents will terminate* DistGreedyMulti *within $DNQ$ steps where $Q = \max_a n_a$. Let $\Pi^D$ be the assignment profile after the algorithm terminates, and let $\Pi^*$ be the optimal assignment profile in problem (A.1). Then the efficiency ratio $r(\Pi^D)$ is lower bounded by 1/2, i.e.,*

$$r(\Pi^D) := \frac{U(\Pi^D)}{U(\Pi^*)} \geq \frac{1}{2}.$$

**Proof** We first show that the algorithm will terminate within maximum $DNQ$ time steps. Let $\Pi^k$ be the assignment profile at the beginning of time step $k$. It is easy to see that, if there is at least one agent active at time step $k$, then $|\Pi^{k+1}| - |\Pi^k| \geq 1$. Combining this with the fact that at least one agent will be active at least once within any time window of length $D$ and the fact that $|\Pi^k|$ cannot exceed $NQ$, we have that all the agents must terminate within $DNQ$ time steps.

Let $K$ be the time step in which the final assignment is made. Then the final assignment profile generated by the distributed greedy algorithm is $\Pi^D = \Pi^{K+1}$.

Let $\rho_k = U(\Pi^{k+1}) - U(\Pi^k)$, for $k = 0, 1, \ldots, K$. Let $\Delta^k = \{a : \exists \tau \in \mathcal{T}_a \text{ s.t. } (a, \tau) \in \Pi^{k+1} - \Pi^k\}$, i.e. the set of agents that make a selection in time step $k$. For each $a$ and $k$, define $i_a^k = |T_a(\Pi^{k+1})|$, i.e. the number of tasks that $a$ has selected at the end of iteration $k$. It is clear that $0 \leq i_a^k \leq n$, $i_a^K = n_a$. We have the following facts about $\Delta^k$.

(i) If $a \in \Delta^k$, $i_a^k = i_a^{k-1} + 1$ (with the convention $i_a^{-1} = 0$) per the definition of $\Delta^k$. Otherwise, $i_a^k = i_a^{k-1}$. This implies that there are exactly $n_a$ indices $k$ such that $a \in \Delta^k$.

(ii) For $a \in \Delta^k$, and any $\tau \in \mathcal{T}_a$, $A_\tau^k = A_\tau(\Pi^k)$ (Line 5 in DistGreedyMulti). Hence, the pre-selected task (Line 7) $\tilde{\tau}_a^k = \arg\max_{\tau \in \mathcal{T}_a} U_\tau(A_\tau^k \cup \{a\}) - U_\tau(A_\tau^k) = \arg\max_{\tau \in \mathcal{T}_a} U_\tau(A_\tau(\Pi^k) \cup \{a\}) - U_\tau(A_\tau(\Pi^k)) = \arg\max_{\tau \in \mathcal{T}_a} \rho_{(a,\tau)}(\Pi^k)$.

(iii) Because of the conflict-resolution stage (Line 8 and 9 in DistGreedyMulti), for two different agents $a, a' \in \Delta^k$, their selections must be different, i.e. $\tilde{\tau}_a^k \neq \tilde{\tau}_{a'}^k$. Hence, $\rho_k = U(\Pi^{k+1}) - U(\Pi^k) = \sum_{a \in \Delta^k} U_{\tilde{\tau}_a^k}(A_{\tilde{\tau}_a^k}(\Pi^k) \cup \{a\}) - U_{\tilde{\tau}_a^k}(A_{\tilde{\tau}_a^k}(\Pi^k)) = \sum_{a \in \Delta^k} \rho_{(a,\tilde{\tau}_a^k)}(\Pi^k)$.

Without loss of generality, we assume in the optimal assignment profile $\Pi^*$ each agent $a$ has to select $n_s$ tasks (if not, we can assign more tasks to the agents that haven't selected enough tasks while we don't decrease the global utility). Denote the set of tasks that $a$ selects in $\Pi^*$ be

$\tau_a^*(1), \tau_a^*(2), \ldots, \tau_a^*(n_s)$. By Lemma 2, we have

$$\begin{aligned}
U(\Pi^*) &\le U(\Pi^D) + \sum_{\pi \in \Pi^* - \Pi^D} \rho_\pi(\Pi^D) \\
&\le U(\Pi^D) + \sum_{\pi \in \Pi^*} \rho_\pi(\Pi^D) && \text{(A.2a)} \\
&= U(\Pi^D) + \sum_{a \in \mathcal{A}} \sum_{i=1}^{n_a} \rho_{(a, \tau_a^*(i))}(\Pi^D) && \text{(A.2b)} \\
&= U(\Pi^D) + \sum_{k=0}^{K} \sum_{a \in \Delta^k} \rho_{(a, \tau_a^*(i_a^k))}(\Pi^D) && \text{(A.2c)} \\
&\le U(\Pi^D) + \sum_{k=0}^{K} \sum_{a \in \Delta^k} \rho_{(a, \tau_a^*(i_a^k))}(\Pi^k) && \text{(A.2d)} \\
&\le U(\Pi^D) + \sum_{k=0}^{K} \sum_{a \in \Delta^k} \rho_{(a, \tilde{\tau}_a^k)}(\Pi^k) && \text{(A.2e)} \\
&= U(\Pi^D) + \sum_{k=0}^{K} \rho_k && \text{(A.2f)} \\
&= 2U(\Pi^D)
\end{aligned}$$

where (A.2a) follows from the nondecreasing property of $U$, i.e. $\rho_\pi(\Pi^D) \ge 0, \forall \pi \in \mathcal{A} \times \mathcal{T}$. Inequality (A.2b) is a restatement of (A.2a) using the fact that $\Pi^* = \cup_{a \in \mathcal{A}} \cup_{i=1}^{n_a} \{(a, \tau_a^*(i))\}$. Equality (A.2c) is due to Fact (i). (8) follows from the submodularity of $U$ and $\Pi^k \subset \Pi^D$. (9) follows from $\tau_a^* \in \mathcal{T}_a$ and fact (iii), the maximality of $\tilde{\tau}_a^k$. (A.2f) follows from fact (iv). □

## B Derivation for Section 3.2.2

Recall that in Section 3.2.2, we claim function $U_\tau$ and $U_{\tilde{\tau}}$ are nondecreasing, submodular and normal with curvature $\kappa$. We now prove this claim. Without loss of generality, we only prove the claim for $U_\tau$, and the proof for $U_{\tilde{\tau}}$ is the same. For notational simplicity, we will drop the subscript and simply refer to $U_\tau$ as $U$.

Recall $U(\cdot) : 2^{\mathcal{A}} \to \mathbb{R}$ is a set function defined on $\mathcal{A} = \{a_1, \ldots, a_N, \tilde{a}_1, \ldots, \tilde{a}_N\}$, given by,

$$U(A) = \sum_{i: a_i \in A} \gamma_i + \sum_{\substack{i: \tilde{a}_i \in A \text{ s.t.} \\ i=1 \text{ or } a_{i-1} \in A}} \gamma_i + \sum_{\substack{i: \tilde{a}_i \in A \text{ s.t.} \\ i>1, a_{i-1} \notin A}} \frac{1}{1-\kappa} \gamma_i$$

where $\gamma_i = \frac{(1-\kappa)^{i-1}\kappa}{1+\kappa}$. It is clear that $U(\emptyset) = 0$, and $U(\cdot)$ is nondecreasing since $\gamma_i \ge 0$. Then, we have the following, for $\tilde{a}_i \notin A$, we have

$$U(A \cup \{\tilde{a}_i\}) - U(A) = \begin{cases} \gamma_i & \text{if } i = 1 \text{ or } a_{i-1} \in A \\ \frac{1}{1-\kappa}\gamma_i & \text{if } i > 1 \text{ and } a_{i-1} \notin A \end{cases}$$

This shows that the marginal increase brought by $\tilde{a}_1$ is always $\gamma_1$. For the marginal increase brought by $\tilde{a}_i$ where $i > 1$, it is $\frac{1}{1-\kappa}\gamma_i$, or a smaller value $\gamma_i$ if $A$ contains an specific element $(a_{i-1})$. This shows that, enlarging $A$ will only make $U(A \cup \{\tilde{a}_i\}) - U(A)$ unchanged or make it smaller.

Similarly, for $a_i \notin A$, we have

$$\begin{aligned}
&U(A \cup \{a_i\}) - U(A) \\
&= \begin{cases} \gamma_i & \text{if } i = N \text{ or } \tilde{a}_{i+1} \notin A \\ \gamma_i + (1 - \frac{1}{1-\kappa})\gamma_{i+1} & \text{if } i < N \text{ and } \tilde{a}_{i+1} \in A \end{cases} \\
&= \begin{cases} \gamma_i & \text{if } i = N \text{ or } \tilde{a}_{i+1} \notin A \\ (1-\kappa)\gamma_i & \text{if } i < N \text{ and } \tilde{a}_{i+1} \in A \end{cases}
\end{aligned}$$

This shows that the marginal increase brought by $a_N$ is always $\gamma_N$. For the marginal increase brought by $a_i$ where $i < N$, it is $\gamma_i$, or a smaller value $(1-\kappa)\gamma_i$ if $A$ contains an specific element $(\tilde{a}_{i+1})$. This shows that, enlarging $A$ will only leave $U(A \cup \{a_i\}) - U(A)$ unchanged or make it smaller.

The above reasoning shows that $U$ is submodular. For the curvature of $U$, we do the following calculations. For $i = 1$,

$$\frac{U(\mathcal{A}) - U(\mathcal{A} - \{\tilde{a}_1\})}{U(\{\tilde{a}_1\})} = 1$$

For $i > 1$

$$\frac{U(\mathcal{A}) - U(\mathcal{A} - \{\tilde{a}_i\})}{U(\{\tilde{a}_i\})} = \frac{\gamma_i}{\frac{1}{1-\kappa}\gamma_i} = 1 - \kappa$$

For $i = N$,

$$\frac{U(\mathcal{A}) - U(\mathcal{A} - \{a_N\})}{U(\{a_N\})} = 1$$

For $i < N$,

$$\frac{U(\mathcal{A}) - U(\mathcal{A} - \{a_i\})}{U(\{a_i\})} = \frac{(1-\kappa)\gamma_i}{\gamma_i} = (1 - \kappa)$$

This shows that the curvature of $U$ is $\kappa$.