



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

ARTICLE INFO

Article history:

Received 10 December 2016

Received in revised form 20 October 2018

Accepted 26 February 2019

Available online 10 April 2019

Keywords:

Optimization

Submodular functions

Task assignment

Multi-agent systems

Distributed algorithms

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.

© 2019 Elsevier Ltd. All rights reserved.

1. Introduction

We consider a multi-agent task assignment problem where there are a group of agents and tasks. Each agent is given an admissible task set that is a subset of the tasks, and each agent needs to select a task from its admissible task set.¹ 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 (Qu, Brown, & Li, 2015), vehicle-target (weapon-target) assignment problem (Arslan, Marden, & Shamma, 2007; Murphey, 2000), and sensor coverage problem (Leonard & Olshevsky, 2013; Marden & Wierman, 2008). 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 (Starin & Eterno, 2011). 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 Murphey (2000), 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 (Fisher, Nemhauser, & Wolsey, 1978; Krause & Golovin, 2012; Nemhauser, Wolsey, & Fisher, 1978), (a special class of set functions, of which the functions in Murphey (2000) are a special case), greedy-type algorithms can be used to find an assignment profile whose utility is at least $1/(1+\kappa)$ of the optimal utility (Conforti & Cornuéjols, 1984; Fisher et al., 1978; Krause & Golovin, 2012), where $\kappa \in [0, 1]$ is the curvature (Conforti & Cornuéjols, 1984) of the submodular function. For some special cases, greedy-type algorithms can even find the optimal solution (Hosein, 1989; denBroeder Jr, Ellison, & Emerling, 1959).

[☆] This work has been supported under NSF CAREER, USA 1553407 and AFOSR, USA YIP. 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. The material in this paper was presented at the 5th IFAC Workshop on Distributed Estimation and Control in Networked Systems, September 10–11, 2015, Philadelphia, PA, USA. This paper was recommended for publication in revised form by Associate Editor Michael M. Zavlanos under the direction of Editor Christos G. Cassandras.

* Corresponding author.

E-mail addresses: gqu@g.harvard.edu (G. Qu), david.brown@ll.mit.edu (D. Brown), nali@seas.harvard.edu (N. Li).

¹ In this paper we will use “select” and “be assigned to” interchangeably.

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 (Qu et al., 2015; Starin & Eterno, 2011). 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 (Fisher et al., 1978) 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 (the meaning of which is defined formally in Assumption 1), each agent can make its own decision in a 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, Bullo, Cortes, and Martinez (2009) and Martinez, Cortes, and Bullo (2007) study 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. Bollobás (2012), Chakrabarti and Kale (2015) and Schrijver (2003), in which tasks (or objects) are assigned to agents to maximize social welfare. Auction algorithms (Bertsekas, 1992; Cramton, Shoham, & Steinberg, 2006; Dobzinski, Nisan, & Schapira, 2006; Lehmann, Lehmann, & Nisan, 2006; Zavlanos, Spesivtsev, & Pappas, 2008) as well as other incentive mechanisms (Blumrosen & Nisan, 2007) 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. A distributed algorithm for the matching problem has been studied in Giordani, Lujak, and Martinelli (2010). Distributed auction for task assignment has been studied in Choi, Brunet, and How (2009), Luo, Chakraborty, and Sycara (2013) and analyzed in a game theoretic framework (Choi, Kim, Johnson, & How, 2016), though these work consider a setting where a task can be selected by at most one agent, different from our setting.

The literature closest to our setting is the line of work in Arslan et al. (2007), Marden (2017), Marden and Wierman (2013), Ramaswamy, Paccagnan, and Marden (2017) and Vetta (2002) 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 (Qu et al., 2015). Compared to Qu et al. (2015), 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 $\mathcal{A} := \{a_1, \dots, a_N\}$ and a set of tasks $\mathcal{T} := \{\tau_1, \dots, \tau_M\}$. 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 Π of $\mathcal{A} \times \mathcal{T}$, i.e., $\Pi \subset \mathcal{A} \times \mathcal{T}$. An element in the profile Π is an agent–task pair, $\pi = (a, \tau) \in \Pi$, meaning that agent a is assigned to task τ . For an assignment profile Π , 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 $U_\tau : 2^{\mathcal{A}} \rightarrow \mathbb{R}$. Given an assignment profile Π , the utility of task τ is $U_\tau(A_\tau(\Pi))$, i.e. $U_\tau(\cdot)$ evaluated at the set of agents that select task τ . The global utility function $U : 2^{\mathcal{A} \times \mathcal{T}} \mapsto \mathbb{R}$ is defined by the sum of task utilities,

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

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,

$$\max_{\Pi \subset \mathcal{A} \times \mathcal{T}} U(\Pi) \quad (2a)$$

$$\text{s.t. } T_a(\Pi) \subset \mathcal{T}_a, \forall a \in \mathcal{A} \quad (2b)$$

$$|T_a(\Pi)| \leq 1, \forall a \in \mathcal{A}. \quad (2c)$$

Constraint (A.1b) means that agent a can only select 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 4 and Appendix A.

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 $\mathcal{T}_a \cap \mathcal{T}_{a'} \neq \emptyset$.

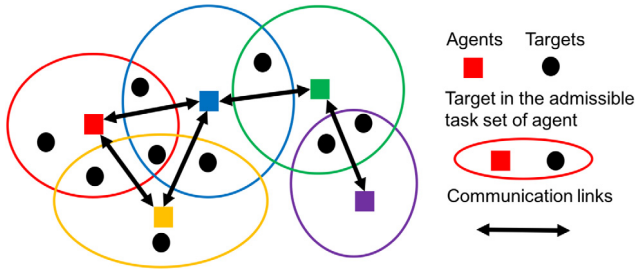


Fig. 1. Illustration of the communication graph.

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 (Martinez et al., 2007, Page 78) widely adopted in multi-agent control literature. Further, such geographical-based setting finds many real applications, including autonomous operation of underwater networks for oceanographic data collection, climate change monitoring, offshore exploration, pollution control and tracking, disaster prevention and tactical surveillance (Dunbabin & Marques, 2012; Heidemann, Ye, Wills, Syed, & Li, 2006; Hollinger, Choudhary, Qarabaqi, Murphy, Mitra, Sukhatme, Stojanovic, Singh, & Hover, 2012; Melodia, Kulhandjian, Kuo, & Demirors, 2013; Smith, Chao, Li, Caron, Jones, & Sukhatme, 2010; Villa, Gonzalez, Miljievic, Ristovski, & Morawska, 2016; Wu, Song, Varnell, & Zhang, 2014). In addition to the geographical setting, we provide another scenario in which our assumption on the communication graph holds. When a task is an entity that has communication capability with agents whose admissible sets include the task, then the task can serve as a “communication relay” and agents do not need to directly talk to other agents but through tasks in its admissible task set instead. In this case, agents can communicate 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 τ , 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: $U_\tau(\emptyset) = 0$. \square

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 Conforti and Cornuéjols (1984).

Definition 1. (a) The total curvature κ_τ of U_τ 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 κ_τ and κ lie within $[0, 1]$. The curvature κ_τ (and κ) 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 κ will directly affect the performance of our algorithm.

Remark 1. The definition of curvature used in this paper is sometimes called total curvature in literature. Recently, alternative definitions of curvatures have been proposed, e.g. the elemental curvature (Sun, Cassandra, & Meng, 2017; Wang, Moran, Wang, & Pan, 2016). How to characterize performance bounds based on these curvature definitions remains our future work.

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., Edmonds (1970), Fisher et al. (1978) and Nemhauser et al. (1978). 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 Krause and Golovin (2012).

2.3. Global greedy algorithm

Problem (2) can be viewed as a variant of Problem (1.2) in Fisher et al. (1978) and it is a matroid constrained submodular maximization problem,² which can be solved by a greedy algorithm with bounded efficiency ratio. The greedy algorithm is provided in Algorithm 1.

Algorithm 1 Global Greedy Algorithm

```

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

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 information, we call this algorithm as *Global Greedy Algorithm*.

² Matroid constraint is a special class of set constraints and it is frequently used in combinatorial optimization (Nemhauser et al., 1978).

The algorithm has the following performance guarantee, proved in Conforti and Cornuéjols (1984, Thm. 2.3).³

Theorem 1. Assume Assumption 2 holds. Let Π^* be an optimal assignment profile of problem (2), and Π^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. \square

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.

Algorithm 2 Distributed Greedy Algorithm

For each agent a , do the following:

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

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.

- (i) **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.

- (ii) **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.
- (iii) **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.

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 $U_\tau(A_\tau^k \cup \{a\}) - U_\tau(A_\tau^k)$, $\forall \tau \in \mathcal{T}_a$ (Lines 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_τ^k , since whenever an agent a' selected τ in a previous time step, a' would have already informed a of its selection of τ 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., $\{a' \in \mathcal{A}, \tilde{\tau}_a^k \in \mathcal{T}_{a'}\}$ (Lines 8, 9 and 12), which satisfies our definition for local communication. In terms of communication complexity, if d_a is the number of neighbors that a has in the communication graph, then Line 8 can be implemented by agent a sending a 's preselection $\tilde{\tau}_a^k$ to d_a other agents; Line 9 can be done without communication, using any predefined lexicographical order; Line 12 involves sending $\tilde{\tau}_a^k$ to d_a other agents. These show that for each iteration, each agent needs to communicate at most $O(d_a B)$ bits to its neighbors where B is the number of bits required to represent a task.

Remark 2. 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 Assumptions 2 and 3, every agent will terminate the algorithm within WN steps. Let Π^D be the assignment profile when the algorithm terminates, and Π^* 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. \square

³ An earlier and weaker bound $1/2$ can be found in Fisher et al. (1978, Thm. 2.1).

Remark 3. 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 4. 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 [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 a property of marginal increase of $U_\tau(\cdot)$ that involves the curvature parameter κ .

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

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

Proof.

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

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 κ_τ (see [Definition 1](#) in Section 2), and (d) is due to $\kappa_\tau \leq \kappa$. \square

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,

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

where (a) and (c) are due to the definition of U (equation (1)), and (b) is due to the submodularity of U_τ . For monotonicity, 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 $\tau \in \mathcal{T}$. \square

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

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

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. \square

We are now ready to prove the bound.

Proof. We first show that the algorithm will terminate within at most WN time steps. Let Π^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 $k = 0, 1, \dots, K$. 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 τ_a^D be the selection of a in the greedy assignment profile Π^D , i.e. $\Pi^D = \cup_{a \in \mathcal{A}} \{(a, \tau_a^D)\}$. We have the following facts about Δ^k .

- (i) The sets $\{\Delta^k : k = 0, 1, \dots, K\}$ are disjoint. This is because by definition Δ^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 Π^* each agent a has to select a task (if not, we can assign a task to the agents that have not selected any task while we do not decrease the global utility). Denote the task that a selects in Π^* be τ_a^* . By [Lemma 3](#), we have

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

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

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

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

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

Equality (6) is a restatement of (5) using the fact that $\Pi^* = \bigcup_{a \in \mathcal{A}} \{(a, \tau_a^*)\}$. (7) is due to Facts (i–ii) (\mathcal{A} is a disjoint union of Δ^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 τ_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,

$$U(\Pi^* \cup \Pi^D) = U(\Pi^*) + \sum_{k=0}^K (U(\hat{\Pi}^{k+1}) - U(\hat{\Pi}^k)) \quad (10)$$

$$= U(\Pi^*) + \sum_{k=0}^K \sum_{\substack{a \in \Delta^k \\ \tau_a^* \neq \tau_a^D}} [U_{\tau_a^D}(A_{\tau_a^D}(\hat{\Pi}^k) \cup \{a\}) - U_{\tau_a^D}(A_{\tau_a^D}(\hat{\Pi}^k))] \quad (10)$$

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

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

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 (9) and (12), we have

$$U(\Pi^*) \leq U(\Pi^D) + \kappa \sum_{k=0}^K \sum_{\substack{a \in \Delta^k \\ \tau_a^* \neq \tau_a^D}} \rho_{(a, \tau_a^D)}(\Pi^k) \quad (13)$$

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

$$= U(\Pi^D) + \kappa \sum_{k=0}^K \rho_k$$

$$= (1 + \kappa)U(\Pi^D)$$

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 (Qu et al., 2015). So we assume $\kappa \in (0, 1)$. Consider two tasks $\mathcal{T} = \{\tau, \tilde{\tau}\}$, and $2N$ agents $\mathcal{A} = \{a_1, a_2, \dots, a_N, \tilde{a}_1, \tilde{a}_2, \dots, \tilde{a}_N\}$. The admissible task set of all agents are set to be \mathcal{T} , i.e. all the tasks. Define $\gamma_i = \frac{(1-\kappa)^{i-1}\kappa}{1+\kappa}$ ($i = 1, 2, \dots$). Now we define the task utility functions U_τ and $U_{\tilde{\tau}}$. For $A \subset \mathcal{A}$, define

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

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

The functions are similar to the one used in Conforti and Cornuéjols (1984, Pf. of Thm. 2.12), and are nondecreasing, submodular and normal (satisfying Assumption 2) with curvature κ . For completeness we include a proof of these properties in Appendix B.

We claim that when we run the distributed greedy algorithm, after the $(k-1)$ 'th iteration, agent a_1, \dots, a_k selects τ , agent $\tilde{a}_1, \dots, \tilde{a}_k$ selects $\tilde{\tau}$, and other agents remain unassigned. We now show the claim for $k=1$. In the 0'th iteration, $U_\tau(\{a_1\}) - U_\tau(\emptyset) = \gamma_1$, $U_{\tilde{\tau}}(\{\tilde{a}_1\}) - U_{\tilde{\tau}}(\emptyset) = \gamma_1$, so we can make a_1 preselect τ (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:

$$U_\tau(\{a_1, \dots, a_{k+1}\}) - U_\tau(\{a_1, \dots, a_k\}) = \gamma_{k+1}$$

$$U_{\tilde{\tau}}(\{\tilde{a}_1, \dots, \tilde{a}_{k+1}\}) - U_{\tilde{\tau}}(\{\tilde{a}_1, \dots, \tilde{a}_k\}) = \gamma_{k+1}.$$

Hence, a_{k+1} preselects τ . 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 Π^D is such that a_k selects τ and \tilde{a}_k selects $\tilde{\tau}$. The resulting global utility is

$$U(\Pi^D) = U_\tau(\{a_1, \dots, a_N\}) + U_{\tilde{\tau}}(\{\tilde{a}_1, \dots, \tilde{a}_N\})$$

$$= 2 \sum_{i=1}^N \gamma_i = 2 \frac{1 - (1-\kappa)^N}{1+\kappa}.$$

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

$$U(\Pi^*) = U_\tau(\{\tilde{a}_1, \dots, \tilde{a}_N\}) + U_{\tilde{\tau}}(\{a_1, \dots, a_N\})$$

$$= 2(\gamma_1 + \frac{1}{1-\kappa} \sum_{i=2}^N \gamma_i) = 2(1 - \frac{(1-\kappa)^{N-1}}{1+\kappa}).$$

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 $N \rightarrow \infty$.

4. Case study

In this section, we numerically test the performance of the proposed algorithm. All tests are run on MATLAB 2016a using HP Z640 Workstation with Intel Xeon CPU E5-2620 v4. We use the weapon-target model (Murphey, 2000) to numerically test

Table 1
Performance and running time results for different N, M .

N	M	r_{eff}	r_{appr}	r_{global}	t_{dist}	t_{global}	t_{cvx}	t_{opt}
10	10	0.9956	0.9944	0.9968	0.0050s	0.0033s	1.5893s	0.0037s
20	20	0.9964	0.9949	0.9984	0.0022s	0.0023s	2.3308s	14.5061s
30	30	–	0.9949	1.0037	0.0020s	0.0058s	3.1574s	–
40	40	–	0.9867	0.9979	0.0034s	0.0142s	4.9813s	–
50	50	–	0.9859	0.9981	0.0036s	0.0269s	6.1459s	–
60	60	–	0.9834	0.9955	0.0059s	0.0426s	6.6049s	–
70	70	–	0.9806	0.9941	0.0077s	0.0715s	7.9140s	–
80	80	–	0.9799	0.9948	0.0097s	0.1084s	9.6759s	–
90	90	–	0.9798	0.9946	0.0130s	0.1479s	11.5562s	–
100	100	–	0.9804	0.9954	0.0118s	0.1994s	14.9043s	–

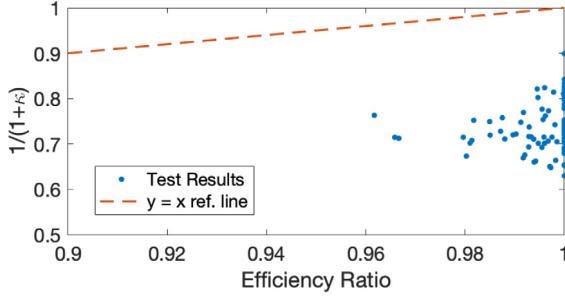


Fig. 2. Case I. Each dot represents one run of the algorithm with one realization of the problem parameters. The x-axis represents the efficiency ratio; while the y-axis is the theoretical efficiency ratio $\frac{1}{1+\kappa}$ calculated using (14).

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 $\mathcal{T}_a = \{\tau \in \mathcal{T} : \text{dist}(a, \tau) < r_a\}$, where $\text{dist}(\cdot, \cdot)$ 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 τ as $p_{a,\tau} = \frac{1}{1+\exp(-\alpha_a \text{dist}(a,\tau) + \beta_a)}$ which is an increasing function of the distance between a and τ . The parameters α_a, β_a characterizes how the failure probability increases w.r.t. the distance between a and target τ . The utility of a target τ is the probability that it gets destroyed times the value of τ , V_τ . Formally, the utility is $U_\tau(\mathcal{A}) = V_\tau(1 - \prod_{a \in \mathcal{A}: \tau \in \mathcal{T}_a} p_{a,\tau})$. We can verify that the utility function satisfies [Assumption 2](#). Moreover, for this particular problem we can calculate the curvature κ ([Definition 1](#)),

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

For each agent a , we draw W_a uniformly from $\{1, \dots, 10\}$, 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 [Fig. 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. From [Fig. 2](#), we can see

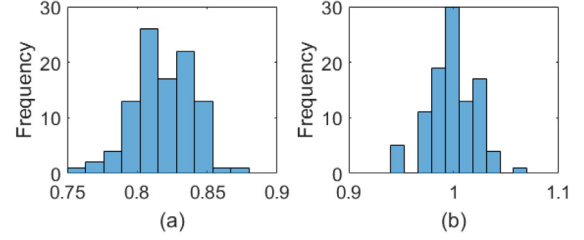


Fig. 3. Case II. Part (a) (left) is histogram for the approximated efficiency ratio (ratio of the distributed greedy solution to the convex relaxation solution). Part (b) (right) is histogram for the ratio of distributed greedy solution to global greedy solution.

that the efficiency ratio is indeed better than the theoretic value $\frac{1}{1+\kappa}$, confirming the statement of [Theorem 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 [Murphey \(2000\)](#)), where the convex relaxation is solved using CVX ([Grant & Boyd, 2008; Grant, Boyd, & Ye, 2008](#)). 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 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 [Fig. 3\(b\)](#) the distributed greedy algorithm has a similar performance as the global greedy algorithm.

To provide a complete picture on how the performance and the running time scales with N and M , we run our algorithm, the brutal force method (intractable for $N \geq 30$), the convex relaxation and the global greedy algorithm for different N 's and M 's. For each N and M , we run the algorithms on 10 randomly generated instances using the parameters in Case I, and calculate the average running time of the algorithms (t_{dist} for distributed greedy algorithm, t_{global} for global greedy algorithm, t_{cvx} for the convex relaxation method, t_{opt} for the brutal force search method), and the efficiency ratio r_{eff} , the approximated efficiency ratio r_{appr} , and the ratio of distributed greedy solution to global greedy solution r_{global} . The results are summarized in [Table 1](#).

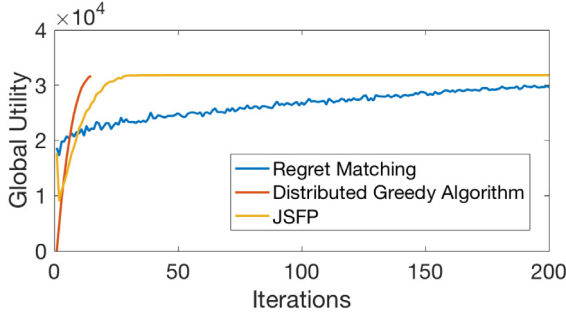


Fig. 4. Comparison of Distributed Greedy Algorithm with Regret Matching and Joint Strategy Fictitious Play. The x-axis is the number of iterations while the y-axis is the global utility.

At last, we compare our algorithm with the game design approach in Arslan et al. (2007). In particular, we select the agent utility to be the marginal contribution (Wonder Life Utility (Arslan et al., 2007, Sec. 3.4)), and we test two learning algorithms, the regret matching (Arslan et al., 2007, Sec. 4.2) and joint strategy fictitious play (JSFP) (Marden, Arslan, & Shamma, 2009). For both algorithms we use the version with fading memory (setting $\rho = 0.05$) and inertial (setting $\alpha = 0.8$).⁴ 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. 4.⁵ Further, we do 10 runs of the algorithms on randomly generated instances, and find that the three algorithms reach assignments with similar utility values, with the value of the distributed greedy algorithm being on average 0.9956 of that of regret matching, and 0.9980 of that of JSFP. The average number of iterations it takes to reach the final value is 15.2 for distributed greedy algorithm, 543.0 for regret matching, and 170.8 for JSFP. From these results, it can be seen that all algorithms reach similar level of utility value. The regret matching converges relatively slowly, while JSFP and our algorithm converge faster. However, our algorithm has a clear termination rule and terminates early, while JSFP has no clear termination rules, and usually stays at a value slightly below the final value for a long time before jumping to the final value.

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.

⁴ For the meaning of ρ and α , please see Arslan et al. (2007, Sec. 4.2) and Marden et al. (2009, Sec 2.4, Sec. 3).

⁵ The per-iteration computation and communication cost of Regret Matching and JSFP is similar to that of the distributed greedy algorithm.

Appendix 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

$$\max_{\Pi \in \mathcal{A} \times \mathcal{T}} U(\Pi) \quad (\text{A.1a})$$

$$\text{s.t. } T_a(\Pi) \subset \mathcal{T}_a, \forall a \in \mathcal{A} \quad (\text{A.1b})$$

$$|T_a(\Pi)| \leq n_a, \forall a \in \mathcal{A}. \quad (\text{A.1c})$$

The multi-selection version for our distributed greedy algorithm is provided in Algorithm 3. For notational simplicity, we present the synchronous form. Algorithm 3 is different from the single selection version Algorithm 2 in Line 2, 12, 14 and 15. Specifically, in Algorithm 3, each agent a keeps track of variable 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_a tasks. We comment that selections already made by an agent a will affect later iterations of the algorithm. In detail, in Line 6 of Algorithm 3, A_τ^k is the set of agents that have already selected τ . If a itself has selected τ 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 τ again.

Algorithm 3 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 τ 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 has not terminated and satisfies $\tilde{\tau}_{a'}^k = \tilde{\tau}_a^k$.
 - 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** $|S_a^{k+1}| = n_a$ **then**
 - 15: **Terminate.**
 - 16: $k \leftarrow k + 1$

In the following theorem, we prove that Algorithm 3 will achieve a $1/2$ -efficiency guarantee compared with the optimal solution of (A.1).

Theorem 3. Under Assumption 2, all the agents will terminate Algorithm 3 within NQ steps where $Q = \max_a n_a$ and N is the number of agents. Let Π^D be the assignment profile after the algorithm terminates, and let Π^* 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 NQ time steps. Let Π^k be the assignment profile at the beginning of time step k . It is easy to see that, at each iteration at least one assignment is made, i.e. $|\Pi^{k+1}| - |\Pi^k| \geq 1$. Combining this with the fact that $|\Pi^k|$ cannot exceed NQ , we have that all the agents must terminate within NQ 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 $k = 0, 1, \dots, K$. 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 $i_a^k = |T_a(\Pi^k)|$, i.e. the number of tasks that a has selected at the beginning of iteration k . It is clear that $0 \leq i_a^k \leq n_a$, $i_a^{K+1} = n_a$. We have the following facts about Δ^k .

- (i) Let $X = \cup_{a \in \mathcal{A}} \{(a, j) : j = 1, \dots, n_a\}$ and $Y = \cup_{k=0}^K \{(a, i_a^{k+1}) : a \in \Delta^k\}$. We claim $X = Y$. To see this, let $(a, j) \in X$, then $1 \leq j \leq n_a$. Note i_a^k , the number of tasks a has selected at the beginning of iteration k , is a nondecreasing integer sequence w.r.t. k that starts from $i_a^0 = 0$ and reaches $i_a^{K+1} = n_a$. Further, $i_a^{k+1} = i_a^k + 1$ whenever $a \in \Delta^k$; $i_a^{k+1} = i_a^k$ when $a \notin \Delta^k$. This means there must be a k s.t. $a \in \Delta^k$ and $i_a^{k+1} = j$. Hence $(a, j) \in Y \Rightarrow X \subset Y$. For the other direction, let $(a, i_a^{k+1}) \in Y$, then since i_a^{k+1} is an integer between 1 and n_a , we must have $(a, i_a^{k+1}) \in X$. Hence $X = Y$.
- (ii) For $a \in \Delta^k$, and any $\tau \in \mathcal{T}_a$, $A_\tau^k = A_\tau(\Pi^k)$ (Line 5 in Algorithm 3). Hence, the preselected 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 Algorithm 3), 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 Π^* each agent a has to select n_a tasks (if not, we can assign more tasks to the agents that have not selected enough tasks while we do not decrease the global utility). Denote the set of tasks that a selects in Π^* be $\tau_a^*(1), \tau_a^*(2), \dots, \tau_a^*(n_a)$. By Lemma 2, we have

$$\begin{aligned} U(\Pi^*) &\leq U(\Pi^D) + \sum_{\pi \in \Pi^* - \Pi^D} \rho_\pi(\Pi^D) \\ &\leq U(\Pi^D) + \sum_{\pi \in \Pi^*} \rho_\pi(\Pi^D) \end{aligned} \quad (\text{A.2a})$$

$$= U(\Pi^D) + \sum_{a \in \mathcal{A}} \sum_{i=1}^{n_a} \rho_{(a, \tau_a^*(i))}(\Pi^D) \quad (\text{A.2b})$$

$$= U(\Pi^D) + \sum_{k=0}^K \sum_{a \in \Delta^k} \rho_{(a, \tau_a^*(i_a^{k+1}))}(\Pi^D) \quad (\text{A.2c})$$

$$\leq U(\Pi^D) + \sum_{k=0}^K \sum_{a \in \Delta^k} \rho_{(a, \tau_a^*(i_a^{k+1}))}(\Pi^k) \quad (\text{A.2d})$$

$$\leq U(\Pi^D) + \sum_{k=0}^K \sum_{a \in \Delta^k} \rho_{(a, \tilde{\tau}_a^k)}(\Pi^k) \quad (\text{A.2e})$$

$$\begin{aligned} &= U(\Pi^D) + \sum_{k=0}^K \rho_k \\ &= 2U(\Pi^D) \end{aligned} \quad (\text{A.2f})$$

where (A.2a) follows from the nondecreasing property of U , i.e. $\rho_\pi(\Pi^D) \geq 0, \forall \pi \in \mathcal{A} \times \mathcal{T}$. Equality (A.2b) is a restatement of (A.2a) using the fact that $\Pi^* = \cup_{a \in \mathcal{A}} \{(a, \tau_a^*(j)) : j = 1, \dots, n_a\}$. Equality (A.2c) is a reordering of the summation indexes using Fact (i). Inequality (A.2d) follows from the submodularity of U and $\Pi^k \subset \Pi^D$. Inequality (A.2e) follows from Fact (ii), the maximality of $\tilde{\tau}_a^k$. Equality (A.2f) follows from Fact (iii). \square

Appendix B. Derivation for Section 3.2.2

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

Recall $U(\cdot) : 2^{\mathcal{A}} \rightarrow \mathbb{R}$ is a set function defined on $\mathcal{A} = \{a_1, \dots, a_N, \tilde{a}_1, \dots, \tilde{a}_N\}$, 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 $\gamma_i \geq 0$. Then, 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 γ_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 γ_i if A contains a 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

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

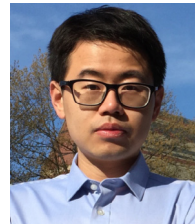
This shows that the marginal increase brought by a_N is always γ_N . For the marginal increase brought by a_i where $i < N$, it is γ_i , or a smaller value $(1 - \kappa) \gamma_i$ if A contains a specific element (\tilde{a}_{i+1}). This shows that, enlarging A will only leave $U(A \cup \{a_i\}) - U(A)$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

The above reasoning shows that U is submodular. For the curvature of U , we do the following calculations. For $i = 1$, $\frac{U(A) - U(A - \{\tilde{a}_1\})}{U(\{\tilde{a}_1\})} = 1$; For $i > 1$, $\frac{U(A) - U(A - \{\tilde{a}_i\})}{U(\{\tilde{a}_i\})} = \frac{\gamma_i}{\frac{1}{1-\kappa} \gamma_i} = 1 - \kappa$. For $i = N$, $\frac{U(A) - U(A - \{a_N\})}{U(\{a_N\})} = 1$; For $i < N$, $\frac{U(A) - U(A - \{a_i\})}{U(\{a_i\})} = \frac{(1-\kappa) \gamma_i}{\gamma_i} = (1 - \kappa)$. This shows that the curvature of U is κ .

References

- Arslan, Gürdal, Marden, Jason R., & Shamma, Jeff S. (2007). Autonomous vehicle-target assignment: A game-theoretical formulation. *Journal of Dynamic Systems, Measurement, and Control*, 129(5), 584–596.
- Bertsekas, Dimitri P. (1992). Auction algorithms for network flow problems: A tutorial introduction. *Computational Optimization and Applications*, 1(1), 7–66.
- Blumrosen, Liad, & Nisan, Noam (2007). Combinatorial auctions. *Algorithmic Game Theory*, 267, 300.

- Bollobás, Béla (2012). *Graph theory: an introductory course*, Vol. 63. Springer Science & Business Media.
- Bullo, Francesco, Cortes, Jorge, & Martinez, Sonia (2009). *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton University Press.
- Chakrabarti, Amit, & Kale, Sagar (2015). Submodular maximization meets streaming: Matchings, matroids, and more. *Mathematical Programming*, 154(1–2), 225–247.
- Choi, Han-Lim, Brunet, Luc, & How, Jonathan P. (2009). Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, 25(4), 912–926.
- Choi, Han-Lim, Kim, Keum-Seong, Johnson, Luke B., & How, Jonathan P. (2016). Potential game-theoretic analysis of a market-based decentralized task allocation algorithm. In *Distributed autonomous robotic systems* (pp. 207–220). Springer.
- Conforti, Michele, & Cornuéjols, Gérard (1984). 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.
- Cramton, Peter, Shoham, Yoav, & Steinberg, Richard (2006). *Combinatorial Auctions*. MIT Press.
- Dobzinski, Shahar, Nisan, Noam, & Schapira, Michael (2006). Truthful randomized mechanisms for combinatorial auctions. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing* (pp. 644–652). ACM.
- Dunbabin, Matthew, & Marques, Lino (2012). Robots for environmental monitoring: Significant advancements and applications. *IEEE Robotics & Automation Magazine*, 19(1), 24–39.
- Edmonds, Jack (1970). In G. Goos, J. Hartmanis, & J. van Leeuwen (Eds.), *Submodular functions, matroids, and certain polyhedra* (p. 11). Springer.
- Fisher, Marshall L., Nemhauser, George L., & Wolsey, Laurence A. (1978). *An analysis of approximations for maximizing submodular set functions—II*. Springer.
- Giordani, Stefano, Lujak, Marin, & Martinelli, Francesco (2010). A distributed algorithm for the multi-robot task allocation problem. In *International conference on industrial, engineering and other applications of applied intelligent systems* (pp. 721–730). Springer.
- Grant, Michael, Boyd, Stephen (2008). Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, & H. Kimura (Eds.), *Lecture notes in control and information sciences, Recent advances in learning and control* (pp. 95–110). Springer-Verlag Limited.
- Grant, Michael, Boyd, Stephen, & Ye, Yinyu (2008). CVX: Matlab Software for disciplined convex programming.
- Heidemann, John, Ye, Wei, Wills, Jack, Syed, Affan, & Li, Yuan (2006). Research challenges and applications for underwater sensor networking. In *Wireless communications and networking conference, 2006. WCNC 2006. IEEE, Vol. 1* (pp. 228–235). IEEE.
- Hollinger, Geoffrey A., Choudhary, Sunav, Qarabagi, Parastoo, Murphy, Christopher, Mitra, Urbashi, Sukhatme, Gaurav S., Stojanovic, Milica, Singh, Hanu-mant, & Hover, Franz (2012). *Underwater data collection using robotic sensor networks*. Institute of Electrical and Electronics Engineers (IEEE).
- Hosein, Patrick A. (1989). *A class of dynamic nonlinear resource allocation problems: Technical report*, DTIC Document.
- denBroeder Jr, G. G., Ellison, R. E., & Emerling, L. (1959). On optimum target assignments. *Operations Research*, 7(3), 322–326.
- Krause, Andreas, & Golovin, Daniel (2012). Submodular function maximization. *Tractability: Practical Approaches to Hard Problems*, 3, 19.
- Lehmann, Benny, Lehmann, Daniel, & Nisan, Noam (2006). Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 55(2), 270–296.
- Leonard, Naomi Ehrlich, & Olshevsky, Alex (2013). Nonuniform coverage control on the line. *IEEE Transactions on Automatic Control*, 58(11), 2743–2755.
- Luo, Lingzhi, Chakraborty, Nilanjan, & Sycara, Katia (2013). Distributed algorithm design for multi-robot task assignment with deadlines for tasks. In *Robotics and automation (ICRA), 2013 IEEE international conference on* (pp. 3007–3013). IEEE.
- Marden, Jason R. (2017). The role of information in distributed resource allocation. *IEEE Transactions on Control of Network Systems*, 4(3), 654–664.
- Marden, Jason R., Arslan, Gürdal, & Shamma, Jeff S. (2009). Joint strategy fictitious play with inertia for potential games. *IEEE Transactions on Automatic Control*, 54(2), 208–220.
- Marden, Jason R., & Wierman, Adam (2008). Distributed welfare games with applications to sensor coverage. In *Decision and control, 2008. CDC 2008. 47th IEEE conference on* (pp. 1708–1713). IEEE.
- Marden, Jason R., & Wierman, Adam (2013). Distributed welfare games. *Operations Research*, 61(1), 155–168.
- Martinez, Sonia, Cortes, Jorge, & Bullo, Francesco (2007). Motion coordination with distributed information. *IEEE Control Systems*, 27(4), 75–88.
- Melodia, Tommaso, Kulhandjian, Hovannes, Kuo, Li-Chung, & Demircan, Emre-can (2013). Advances in underwater acoustic networking. In *Mobile ad hoc networking: Cutting edge directions* (pp. 804–852). Wiley Online Library.
- Murphey, Robert A. (2000). Target-based weapon target assignment problems. In *Nonlinear assignment problems* (pp. 39–53). Springer.
- Nemhauser, George L., Wolsey, Laurence A., & Fisher, Marshall L. (1978). An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1), 265–294.
- Qu, Guannan, Brown, Dave, & Li, Na (2015). Distributed greedy algorithm for satellite assignment problem with submodular utility function. *IFAC-PapersOnLine*, 48(22), 258–263.
- Ramaswamy, Vinod, Paccagnan, Dario, & Marden, Jason R. (2017). The impact of local information on the performance of multiagent systems. arXiv preprint arXiv:1710.01409.
- Schrijver, Alexander (2003). *Combinatorial optimization: polyhedra and efficiency*, Vol. 24. Springer Science & Business Media.
- Smith, Ryan N., Chao, Yi, Li, Peggy P., Caron, David A., Jones, Burton H., & Sukhatme, Gaurav S. (2010). Planning and implementing trajectories for autonomous underwater vehicles to track evolving ocean processes based on predictions from a regional ocean model. *International Journal of Robotics Research*, 29(12), 1475–1497.
- Starin, Scott R., & Eterno, John (2011). In James Richard Wertz, David F. Everett, & Jeffery John Puschell (Eds.), *Space mission engineering: the new SMAD*. Microcosm Press.
- Sun, Xinmiao, Cassandras, Christos G., & Meng, Xiangyu (2017). A submodularity-based approach for multi-agent optimal coverage problems. In *Decision and control (CDC), 2017 IEEE 56th annual conference on* (pp. 4082–4087). IEEE.
- Vetta, Adrian (2002). 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* (pp. 416–425). IEEE.
- Villa, Tommaso Francesco, Gonzalez, Felipe, Miljievic, Branka, Ristovski, Zoran D., & Morawska, Lidia (2016). An overview of small unmanned aerial vehicles for air quality measurements: Present applications and future perspectives. *Sensors*, 16(7), 1072.
- Wang, Zengfu, Moran, Bill, Wang, Xuezhi, & Pan, Quan (2016). Approximation for maximizing monotone non-decreasing set functions with a greedy method. *Journal of Combinatorial Optimization*, 31(1), 29–43.
- Wu, Wencen, Song, Aijun, Varnell, Paul, & Zhang, Fumin (2014). Cooperatively mapping of the underwater acoustic channel by robot swarms. In *Proceedings of the international conference on underwater networks & systems* (p. 20). ACM.
- Zavlanos, Michael M., Spesivtsev, Leonid, & Pappas, George J. (2008). A distributed auction algorithm for the assignment problem. In *Decision and control, 2008. CDC 2008. 47th IEEE conference on* (pp. 1212–1217). IEEE.



Guannan Qu received his B.S. degree in Electrical Engineering from Tsinghua University in Beijing, China in 2014. Since 2014 he has been a graduate student in the School of Engineering and Applied Sciences at Harvard University. His research interest lies in network control and optimization.



David Brown received the Masters degree in computational science and engineering in 2015 at Harvard University. Since 2015 he has been research staff at MIT's Lincoln Laboratory. His current research interests include networked systems, space systems, and electromagnetic sensing.



Na Li received her B.S. degree in mathematics and applied mathematics from Zhejiang University in China and her PhD degree in Control and Dynamical systems from the California Institute of Technology in 2013. She is an Associate Professor in the School of Engineering and Applied Sciences in Harvard University. She was a postdoctoral associate of the Laboratory for Information and Decision Systems at Massachusetts Institute of Technology. Her research lies in the design, analysis, optimization, and control of distributed network systems, with particular applications to cyber-physical network systems. She received NSF CAREER Award in 2016, AFOSR Young Investigator Award in 2017, ONR Young Investigator Award in 2019.