IEEE Transactions on Robotics, Vol. 34, No. 6, 2018, pp. 1534-1548.
DOI: 10.1109/TRO.2018.2858292

Anonymous Hedonic Game for Task Allocation in a Large-Scale Multiple Agent System

Inmo Jang, Hyo-Sang Shin, and Antonios Tsourdos

***Abstract*—This paper proposes a novel game-theoretical autonomous decision-making framework to address a task allocation problem for a swarm of multiple agents. We consider cooperation of self-interested agents, and show that our proposed decentralized algorithm guarantees convergence of agents with *social inhibition* to a Nash stable partition (i.e., social agreement) within polynomial time. The algorithm is simple and executable based on local interactions with neighbor agents under a strongly-connected communication network and even in asynchronous environments. We analytically present a mathematical formulation for computing the lower bound of suboptimality of the outcome, and additionally show that 50% of suboptimality can be at least guaranteed if social utilities are non-decreasing functions with respect to the number of co-working agents. The results of numerical experiments confirm that the proposed framework is scalable, fast adaptable against dynamical environments, and robust even in a realistic situation.**

***Index Terms*—Distributed robot systems, Networked robots, Multi-robot systems, Task allocation, Coalition formation**

I. Introduction

Cooperation of a large number of possibly small-sized robots, called *robotic swarm*, will play a significant role in complex missions that existing operational concepts using a few large robots could not handle [1]. Even if every single robot (or called *agent*) in a swarm is incapable of accomplishing a task alone, their cooperation will lead to successful outcomes [2]–[5]. The possible applications include surveillance [6], ad-hoc network relay [7], disaster management [8], cooperative radar jamming [9], to name a few.

Due to the large cardinality of a swarm robot system, however, it is infeasible for human operators to supervise each agent directly, but needed to entrust the swarm with certain levels of decision-makings (e.g., task allocation, path planning, and individual control). Thereby, what only remains is to provide a high-level mission description, which is manageable for a few or even a single human operator. Nevertheless, there still exist various challenges in the autonomous decision-making of robotic swarms. Among them, this paper addresses a task allocation problem where the number of agents is higher than that of tasks: how to partition a set of agents into subgroups and assign the subgroups to each task. In the problem, it is assumed that each agent can be assigned to at most one task, whereas each task may require multiple agents: this case falls into ST-MR (single-task robot and multi-robot task) category [10], [11].

Inmo Jang, Hyo-Sang Shin, and Antonios Tsourdos are with Centre for Autonomous and Cyber-Physical Systems, Cranfield University, MK43 0AL, United Kingdom (e-mail: inmo.jang@cranfield.ac.uk; h.shin@cranfield.ac.uk; a.tsourdos@cranfield.ac.uk).

According to [4], [5], [12]–[14], decision-making frameworks for a robotic swarm should be *decentralized* (i.e., the desired collective behavior can be achieved by individual agents relying on local information), *scalable*, *predictable* (e.g., regarding convergence performance and outcome quality), and *adaptable* to dynamic environments (e.g., unexpected elimination or addition of agents or tasks). Moreover, the frameworks are also desirable to be *robust* against asynchronous environments because, due to the large cardinality of the system and its decentralization, it is very challenging for every agent to behave synchronously. For synchronization in practice, "artificial delays and extra communication must be built into the framework" [14], which may cause considerable inefficiency on the system. In addition, it is also preferred to be capable of accommodating different interests of agents (e.g., different swarms operated by different organizations [15]).

In this paper, we propose a novel decision-making framework based on hedonic games [16]–[18]. The task allocation problem considered is modeled as a coalition-formation game where self-interest agents are willing to form coalitions to improve their own interests. The objective of this game is to find a *Nash stable* partition, which is a social agreement where all the agents agree with the current task assignment. Despite any possible conflicts between the agents, this paper shows that if they have *social inhibition*, then a Nash stable partition can always be determined within polynomial times in the proposed framework and all the desirable characteristics mentioned above can be achieved. Furthermore, we analyze the lower bound of the outcome's suboptimality and show that 50% is at least guaranteed for a particular case. Various settings of numerical experiments validate that the proposed framework is scalable, adaptable, and robust even in asynchronous environments.

This paper is organized as follows. Section II reviews existing literature on decentralized multi-robot task allocation and introduces a recent finding in hedonic games that inspires this study. Section III proposes our decision-making framework, named *GRAPE*, and analytically proves the existence of and the polynomial-time convergence to a Nash stable partition. Section IV discusses the framework's algorithmic complexity, suboptimality, adaptability, and robustness. Section V shows that the framework can also address a task allocation problem in which each task may need a certain number of agents for completion. Numerical simulations in Section VI confirm that the proposed framework holds all the desirable characteristics. Finally, concluding remarks are followed in Section VII.

II. Related Work

A. Decentralized Coordination of Robotic Swarms

Existing approaches for task allocation problems can be categorized into two branches, depending on how agents eventually reach a converged outcome: *orchestrated* and *(fully) self-organized* approaches [19]. In the former, additional mechanism such as a negotiation or voting model is imposed so that some agents can be worse off if a specific condition is met (e.g., the global utility is better off). Alternatively, in self-organized approaches, each agent simply makes a decision without negotiating with the other agents. The latter generally induce less resource consumption in communication and computation [20], and hence they are preferable in terms of scalability. On the other hand, the former usually provide a better quality of solutions with respect to the global utility, and a certain level of suboptimality could be guaranteed [21]–[23]. A comparison result between the two approaches [20] presents that as the available information to agents becomes local, the latter becomes to outperform the former. The following review will focus on self-organized approaches because, for large-scale multiple agent systems, scalability is at least essential and each agent is likely to know information about it's immediate surrounds not the global environment.

Self-organized approaches can be categorized into *top-down approaches* and *bottom-up approaches* according to which level (i.e., an ensemble vs. individuals) is mainly focused on. Top-down approaches emphasize developing a macroscopic model for the whole system. For instance, population fractions associated with given tasks are represented as states, and the dynamics of the population fractions is modeled by Markov chains [12], [24]–[26] or differential equations [27]–[31]. Given a desired fraction distribution over the tasks, agents can converge to the desired status by following local decision policies from a macroscopic model (e.g., the associated rows or columns of the current Markov matrix). One advantage of using top-down approaches is predictability of average emergent behavior with regard to convergence speed and the quality of a stable outcome (i.e., how well the agents converge to the desired fraction distribution). However, such prediction, to the best of our knowledge, can be made mainly numerically. Besides, as top-down generated control policies regulate agents, it may be difficult to accommodate each agent's individual preference. Also, each agent may have to physically move around according to its local policy during the entire decision-making process, which may cause waste of time and energy costs in the transitioning.

Bottom-up approaches focus on designing each agent's individual rules (i.e., microscopic models) that eventually lead to a desired emergent behavior. Possible actions of a single agent can be modeled as a finite state machine [32], and a change of behavior occurs according to a probabilistic threshold model [33]. A threshold value in the model determines the decision boundary between two motions. This value is adjustable based on an agent's past experiences such as the time spent on working a task [19], [34], the success/failure rates [32], [35], and direct communication from a central unit [33]. This feature can improve system adaptability, and may have the potential to incorporate each agent's individual interest if required. However, it was shown in [35]–[41] that, to predict or evaluate an emergent performance of a swarm operated by a bottom-up approach, a macroscopic model for the system is eventually required to be developed by abstracting the microscopic model.

B. Hedonic Games

Hedonic games [16]–[18] model a conflict situation where self-interest agents are willing to form coalitions to improve their own interests. *Nash stability* [18] plays a key role since it yields a social agreement among the agents even without having any negotiation. Many researchers have investigated conditions under which a Nash stable partition is guaranteed to exist and to be determined [18], [42]–[44]. Among them, the works in [43], [44] mainly addressed *anonymous hedonic games*, in which each agent considers the size of a coalition to which it belongs instead of the identities of the members. Recently, Darmann [44] showed that selfish agents who have *social inhibition* (i.e., preference to a coalition with a fewer number of members) could converge to a Nash stable partition in an anonymous hedonic game. The author also proposed a centralized recursive algorithm that can find a Nash stable partition within $O(n_a^2 \cdot n_t)$ of iterations. Here, n_a is the number of agents and n_t is that of tasks.

C. Main Contributions

Inspired by the recent breakthrough of [44], we propose a novel decentralized framework that models the task allocation problem considered as an anonymous hedonic game. The proposed framework is a self-organized approach in which agents make decisions according to its local policies (i.e., individual preferences). Unlike top-down or bottom-up approaches reviewed in the previous section, which primarily concentrate on designing agents' decision-making policies either macroscopically or microscopically, our work instead focuses on investigating and exploiting advantages from socially-inhibitive agents, while simply letting them greedily behave according to their individual preferences. Explicitly, the main contributions of this paper are as follows:

1) This paper shows that selfish agents with social inhibition, which we refer to as *SPAO* preference (Definition 4), can reach a Nash stable partition within less algorithmic complexity compared with [44]: $O(n_a^2)$ of iterations are required[1].
2) We provide a decentralized algorithm, which is executable under a strongly-connected communication network of agents and even in asynchronous environments. Depending on the network assumed, the algorithmic complexity may be additionally increased by $O(d_G)$, where $d_G < n_a$ is the graph diameter of the network.
3) This paper analyzes the suboptimality of a Nash stable partition in term of the global utility. We firstly present a

[1]Note that the definition of *iteration* is described in Definition 5. This comparison assumes the fully-connected communication network because the algorithm in [44] is centralized.

mathematical formulation to compute the suboptimality lower bound by using the information of a Nash stable partition and agents' individual utilities. Furthermore, we additionally show that 50% of suboptimality can be at least guaranteed if the social utility for each coalition is defined as a non-decreasing function with respect to the number of members in the coalition.

4) Our framework can accommodate different agents with different interests as long as their individual preferences hold SPAO.
5) Through various numerical experiments, it is confirmed that the proposed framework is scalable, fast adaptable to environmental changes, and robust even in a realistic situation where some agents are temporarily unable to proceed a decision-making procedure and communicate with the other agents during a mission.

TABLE I
NOMENCLATURE

Symbol	Description
$\mathcal{A}$	a set of n_a agents
a_i	the i-th agent
$\mathcal{T}^*$	a set of n_t tasks
t_j	the j-th task
t_ϕ	the void task (i.e., not to work any task)
$\mathcal{T}$	a set of tasks, $\mathcal{T} = \mathcal{T}^* \cup \{t_\emptyset\}$
(t_j, p)	a task-coalition pair (i.e. to do task t_j with p participants)
$\mathcal{X}$	the set of task-coalition pairs, $\mathcal{X} = \mathcal{X}^* \cup \{t_\phi\}$, where $\mathcal{X}^* = \mathcal{T}^* \times \{1, 2, ..., n_a\}$
$\mathcal{P}_i$	agent a_i's preference relation over $\mathcal{X}$
$\succ_i$	the strong preference of agent a_i
$\sim_i$	the indifferent preference of agent a_i
$\succeq_i$	the weak preference of agent a_i
Π	a *partition*: a disjoint set that partitions the agent set $\mathcal{A}$, $\Pi = \{S_1, S_2, ..., S_{n_t}, S_\phi\}$
S_j	the (task-specific) coalition for t_j
$\Pi(i)$	the index of the task to which agent a_i is assigned given Π
d_G	the graph diameter of the agent communication network
$\mathcal{N}_i$	The neighbor agent set of agent a_i given a network

III. GROUP AGENT PARTITIONING AND PLACING EVENT

A. Problem Formulation

Let us first introduce the multi-robot task allocation problem considered in this paper and underlying assumptions.

Problem 1. Suppose that there exist a set of n_a agents $\mathcal{A} = \{a_1, a_2, ..., a_{n_a}\}$ and a set of tasks $\mathcal{T} = \mathcal{T}^* \cup \{t_\emptyset\}$, where $\mathcal{T}^* = \{t_1, t_2, ..., t_{n_t}\}$ is a set of n_t tasks and $t_\emptyset$ is *the void task* (i.e., not to perform any task). Each agent a_i has *the individual utility* $u_i : \mathcal{T} \times |\mathcal{A}| \to \mathbb{R}$, which is a function of the task to which the agent is assigned and the number of its co-working agents (including itself) $p \in \{1, 2, ..., n_a\}$ (called *participants*). The individual utility for t_ϕ is zero regardless of the participants. Since every agent is considered to have limited capabilities to finish a task alone, the agent can be assigned to at most one task. The objective of this task allocation problem is to find an assignment that maximizes *the global utility*, which is the sum of individual utilities of the entire agents. The problem described above is defined as follows:

$$\max_{\{x_{ij}\}} \sum_{\forall a_i \in \mathcal{A}} \sum_{\forall t_j \in \mathcal{T}} u_i(t_j, p) x_{ij}, \tag{1}$$

subject to

$$\sum_{\forall t_j \in \mathcal{T}} x_{ij} \leq 1, \quad \forall a_i \in \mathcal{A}, \tag{2}$$

$$x_{ij} \in \{0, 1\}, \quad \forall a_i \in \mathcal{A}, \forall t_j \in \mathcal{T}, \tag{3}$$

where x_{ij} is a binary decision variable that indicates whether or not task t_j is assigned to agent a_i.

The term *social utility* refers to the sum of individual utilities within any agent group.

Assumption 1 (*Homogeneous agents with limited capabilities*)**.** This paper considers a large-scale multi-robot system of homogeneous agents since the realisation of a swarm can be in general achieved through mass production [4]. Therefore, each individual utility u_i is concerned with the cardinality of co-working agents. Note that agents in this paper may have different preferences with respect to the given tasks, e.g., for an agent, a spatially closer task is more preferred, whereas this may not be the case for another agent. Besides, noting that "mass production favors robots with fewer and cheaper components, resulting in lower cost but also reduced capabilities [45]", it is also assumed that each agent can be only assigned to perform at most a single task. According to [10], such a robot is called a *single-task* (ST) robot.

Assumption 2 (*Agents' communication*)**.** The communication network of the entire agents is at least *strongly-connected*, i.e., there exists a directed communication path between any two arbitrary agents. Given a network, $\mathcal{N}_i \subseteq \mathcal{A}$ denotes a set of neighbor agents for agent a_i.

Assumption 3 (*Multi-robot-required tasks*)**.** Every task is a *multi-robot* (MR) task, meaning that the task may require multiple robots [10]. For now, we assume that each task can be performed even by a single agent although it may take a long time. However, in Section V, we will also address a particular case in which some tasks need at least a certain number of agents for completion.

Assumption 4 (*Agents' pre-known information*)**.** Every agent a_i only knows its own individual utility function u_i with regard to every task, while not being aware of those of the other agents. Through communication, however, they can notice which agent currently chooses which task, i.e., *partition* (Definition 2). Note that the agents do not necessarily have to know the true partition information at all the time. Each agent may own its locally-known partition information.

B. Proposed Game-theoretical Approach: GRAPE

Let us transform Problem 1 into an anonymous hedonic game event where every agent selfishly tends to join a coalition according to its preference.

Definition 1 (*GRAPE*)**.** An instance of *GRoup Agent Partitioning and placing Event* (GRAPE) is a tuple $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ that consists of (1) $\mathcal{A} = \{a_1, a_2, ..., a_{n_a}\}$, a set of n_a agents; (2) $\mathcal{T} = \mathcal{T}^* \cup \{t_\phi\}$, a set of tasks; and (3) $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_{n_a})$, an n_a-tuple of preference relations of the agents. For each agent a_i, $\mathcal{P}_i$ describes its *preference relation* over the set of task-coalition pairs $\mathcal{X} = \mathcal{X}^* \cup \{t_\phi\}$, where $\mathcal{X}^* = \mathcal{T}^* \times \{1, 2, ..., n_a\}$; a task-coalition pair (t_j, p) is interpreted as "to do task t_j with p participants". For any task-coalition pairs $x_1, x_2 \in \mathcal{X}$, $x_1 \succ_i x_2$ implies that agent a_i strongly prefers x_1 to x_2, and $x_1 \sim_i x_2$ means that the preference regarding x_1 and x_2 is indifferent. Likewise, $\succeq_i$ indicates the weak preference of agent a_i.

Note that agent a_i's preference relation can be derived from its individual utility $u_i(t_j, p)$ in Problem 1. For instance, given that $u_i(t_1, p_1) > u_i(t_2, p_2)$, it can be said that $(t_1, p_1) \succ_i (t_2, p_2)$.

Definition 2 (*Partition*)**.** Given an instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ of GRAPE, a *partition* is defined as a set $\Pi = \{S_1, S_2, ..., S_{n_t}, S_\phi\}$ that disjointly partitions the agent set $\mathcal{A}$. Here, $S_j \subseteq \mathcal{A}$ is the *(task-specific) coalition* for executing task t_j such that $\cup_{j=0}^{n_t} S_j = \mathcal{A}$ and $S_j \cap S_k = \emptyset$ for $j \neq k$. S_ϕ is the set of agents who choose the void task t_ϕ. Note that this paper interchangeably uses S_0 to indicate S_ϕ. Given a partition Π, $\Pi(i)$ indicates the index of the task to which agent a_i is assigned. For example, $S_{\Pi(i)}$ is the coalition that the agent belongs to, i.e., $S_{\Pi(i)} = \{S_j \in \Pi \mid a_i \in S_j\}$.

The objective of GRAPE is to determine a stable partition that all the agents agree with. In this paper, we seek for a *Nash stable* partition, which is defined as follows:

Definition 3 (*Nash stable*)**.** A partition Π is said to be *Nash stable* if, for every agent $a_i \in \mathcal{A}$, it holds that $(t_{\Pi(i)}, |S_{\Pi(i)}|) \succeq_i (t_j, |S_j \cup \{a_i\}|)$, $\forall S_j \in \Pi$.

In other words, in a Nash stable partition, every agent prefers its current coalition to joining any of the other coalitions. Thus, every agent does not have any conflict within this partition, and no agent will not unilaterally deviate from its current decision.

Remark 1 (*An advantage of Nash stability: low communication burden on agents*)**.** The rationale behind the use of Nash stability among various stable solution concepts in hedonic games [16], [46]–[48] is that it can reduce communication burden between agents required to reach a social agreement. In the process of converging to a Nash stable partition, an agent does not need to get any permission from the other agents when it is willing to deviate. This property may not be the case for the other solution concepts. Therefore, each agent is only required to notify its altered decision without any negotiation. This fact can reduce inter-agent communication in the proposed approach.

C. SPAO Preference: Social Inhibition

This section introduces the key condition, called *SPAO*, that enables our proposed approach to provide all the desirable properties described in Section I, and then explains its implications.

Definition 4 (*SPAO*)**.** Given an instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ of GRAPE, it is said that the preference relation of agent a_i with respect to task t_j is *SPAO (Single-Peaked-At-One)* if it holds that, for every $(t_j, p) \in \mathcal{X}^*$, $(t_j, p_1) \succeq_i (t_j, p_2)$ for any $p_1, p_2 \in \{1, ..., n_a\}$ such that $p_1 < p_2$. Besides, we say that an instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ of GRAPE is SPAO if the preference relation of every agent in $\mathcal{A}$ with respect to every task in $\mathcal{T}^*$ is SPAO.

For an example, suppose that $\mathcal{P}_i$ is such that

$$(t_1, 1) \succ_i (t_1, 2) \succeq_i (t_1, 3) \succ_i (t_2, 1) \sim_i (t_1, 4) \succ_i (t_2, 2).$$

This preference relation indicates that agent a_i has $(t_1, 1) \succ_i (t_1, 2) \succeq_i (t_1, 3) \succ_i (t_1, 4)$ for task t_1, and $(t_2, 1) \succ_i (t_2, 2)$ for task t_2. According to Definition 4, the preference relation for each of the tasks holds SPAO. For another example, given that

$$(t_1, 1) \succ_i (t_1, 2) \succeq_i (t_1, 3) \succ_i (t_2, 2) \sim_i (t_1, 4) \succ_i (t_2, 1),$$

the preference relation regarding task t_1 holds SPAO, whereas this is not the case for task t_2 because of $(t_2, 2) \succ_i (t_2, 1)$.

This paper only considers the case in which every agent has SPAO preference relations regarding all the given tasks. Such agents prefer to execute a task with smaller number of collaborators, namely, they have *social inhibition*.

Remark 2 (*Implications of SPAO*)**.** SPAO implies that an agent's individual utility should be a monotonically decreasing function with respect to the size of a coalition. In practice, SPAO can often emerge. For instance, experimental and simulation results in [49, Figures 3 and 4] show that the total work capacity resulted from cooperation of multiple robots does not proportionally increase due to interferences of the robots. In such a *non-superadditive* environment [50], assuming that an agent's individual work efficiency is considered as its individual utility, the individual utility monotonically drops as the number of collaborators enlarges even though the social utility is increased. For another example, SPAO also arises when individual utilities are related with shared-resources. As more agents use the same resource simultaneously, their individual productivities become diminished (e.g., traffic affects travel times [51] [52, Example 3]). As the authors in [50] pointed out, a non-superadditive case is more realistic than a superadditive case: agents in a superadditive environment always attempt to form the grand coalition whereas those in a non-superadditive case are willing to reduce unnecessary costs. Note that social utility functions are not restricted so that they can be either monotonic or non-monotonic.

Remark 3 (*Cooperation of selfish agents with different interests*)**.** The proposed framework can accommodate selfish agents who greedily follow their individual preferences as long as the preferences hold SPAO. This implies that the framework may be utilized for a combination of swarm systems from different organisations under the condition that the multiple systems satisfy SPAO.

D. Existence of and Convergence to a Nash Stable Partition

Let us prove that if an instance of GRAPE holds SPAO, there always exists a Nash stable partition and it can be found within polynomial time.

Definition 5 (*Iteration*)**.** This paper uses the term *iteration* to represent an iterative stage in which a single arbitrary agent compares the set of selectable task-coalition pairs given an existing partition and then exclusively updates the partition according to its individual preference. This paper refers to this agent as the *deciding agent* at the iteration. Based on the resultant partition, another deciding agent performs the same process at the next iteration, and this continues until every agent does not deviate from a specific partition, which is, in fact, a Nash stable partition.

Assumption 5 (*Mutual exclusion algorithm*)**.** To choose the deciding agent at each iteration, a *mutual exclusion* (or called *mutex*) algorithm is required in practice. In this section, for simplicity of description, we assume that all the agents are fully-connected, by which they somehow select and know the deciding agent. However, in Section III-E, we will present a distributed mutex algorithm that enables the proposed approach to be executed under a strongly-connected communication network even in an asynchronous manner.

Lemma 1. *Given an instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ of GRAPE that is SPAO, suppose that a new agent $a_r \notin \mathcal{A}$ holding a SPAO preference relation with regard to every task in $\mathcal{T}$ joins $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ in which a Nash stable partition is already established. Then, the new instance $(\tilde{\mathcal{A}}, \mathcal{T}, \mathcal{P})$, where $\tilde{\mathcal{A}} = \mathcal{A} \cup \{a_r\}$, also (1) satisfies SPAO; (2) contains a Nash stable partition; and (3) the maximum number of iterations required to re-converge to a Nash stable partition is $|\tilde{\mathcal{A}}|$.*

Proof. Given a partition Π, for agent a_i, *the number of additional co-workers tolerable in its coalition* is defined as:

$$\Delta_{\Pi(i)} \coloneqq \min_{S_j \in \Pi \setminus \{S_{\Pi(i)}\}} \max_{\Delta \in \mathbb{Z}} \{\Delta \mid (t_{\Pi(i)}, |S_{\Pi(i)}| + \Delta) \succeq_i (t_j, |S_j \cup \{a_i\}|)\}. \tag{4}$$

Due to the SPAO preference relation, this value satisfies the following characteristics: (a) if Π is Nash stable, for every agent a_i, it holds that $\Delta_{\Pi(i)} \geq 0$; (b) $\Delta_{\Pi(i)} < 0$ implies that agent a_i is willing to deviate to another coalition at a next iteration; and (c) for the agent a_i who deviated at the last iteration and updated the partition as Π', it holds that $\Delta_{\Pi'(i)} \geq 0$.

From Definition 4, it is clear that the new instance $(\tilde{\mathcal{A}}, \mathcal{T}, \mathcal{P})$ still holds SPAO. Let Π_0 denote a Nash stable partition in the original instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$. When a new agent $a_r \notin \mathcal{A}$ decides to execute one of the tasks in $\mathcal{T}$ and creates a new partition Π_1, it holds that $\Delta_{\Pi_1(r)} \geq 0$, as shown in (c). If there is no existing agent $a_q \in \mathcal{A}$ whose $\Delta_{\Pi_1(q)} < 0$, then the new partition Π_1 is Nash stable.

Suppose that there exists at least an agent a_q whose $\Delta_{\Pi_1(q)} < 0$. Then, the agent must be one of the existing members in the coalition that agent a_r selected in the last iteration. As agent a_q moves to another coalition and creates a new partition Π_2, the previously-deviated agent a_r holds $\Delta_{\Pi_2(r)} \geq 1$. In other words, an agent who deviates to a coalition and expels one of the existing agents in that coalition will not deviate again even if another agent joins the coalition in a next iteration. This implies that at most $|\tilde{\mathcal{A}}|$ of iterations are required to hold $\Delta_{\tilde{\Pi}(i)} \geq 0$ for every agent $a_i \in \tilde{\mathcal{A}}$, where the partition $\tilde{\Pi}$ is Nash stable. □

Lemma 1 is essential not only for the existence of and convergence to a Nash stable partition but also for fast adaptability to dynamic environments.

Theorem 1 (*Existence*)**.** *If $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ is an instance of GRAPE holding SPAO, then a Nash stable partition always exists.*

Proof. This theorem will be proved by induction. Let $M(n)$ be the following mathematical statement: for $|\mathcal{A}| = n$, if an instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ of GRAPE is SPAO, then there exists a Nash stable partition.

Base case: When $n = 1$, there is only one agent in an instance. This agent is allowed to participate in its most preferred coalition, and the resultant partition is Nash stable. Therefore, $M(1)$ is true.

Induction hypothesis: Assume that $M(k)$ is true for a positive integer k such that $|\mathcal{A}| = k$.

Induction step: Suppose that a new agent $a_i \notin \mathcal{A}$ whose preference relation regarding every task in $\mathcal{T}$ is SPAO joins the instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$. This induces a new instance $(\tilde{\mathcal{A}}, \mathcal{T}, \mathcal{P})$ where $\tilde{\mathcal{A}} = \mathcal{A} \cup \{a_i\}$ and $|\tilde{\mathcal{A}}| = k + 1$. From Lemma 1, it is clear that the new instance also satisfies SPAO and has a Nash stable partition $\tilde{\Pi}$. Consequently, $M(k+1)$ is true. By mathematical induction, $M(n)$ is true for all positive integers $n \geq 1$. □

Theorem 2 (*Convergence*)**.** *If $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ is an instance of GRAPE holding SPAO, then the number of iterations required to determine a Nash stable partition is at most $|\mathcal{A}| \cdot (|\mathcal{A}| + 1)/2$.*

Proof. Suppose that, given a Nash stable partition in an instance where there exists only one agent, we add another arbitrary agent and find a Nash stable partition for this new instance, and repeat the procedure until all the agents in $\mathcal{A}$ are included. From Lemma 1, if a new agent joins an instance in which the current partition is Nash stable, then the maximum number of iterations required to find a new Nash stable partition is the number of the existing agents plus one. Therefore, it is trivial that the maximum number of iterations to find a Nash stable partition of an instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ is given as

$$\sum_{k=1}^{|\mathcal{A}|} k = |\mathcal{A}| \cdot (|\mathcal{A}| + 1)/2. \tag{5}$$

Note that this polynomial-time convergence still holds even if the agents are initialized to a random partition. Suppose that we have the following setting: the entire agents $\mathcal{A}$ are firstly not movable from the existing partition, except a set of free agents $\mathcal{A}' \subseteq \mathcal{A}$; whenever the agents $\mathcal{A}'$ find a Nash stable partition Π', one arbitrary agent in $a_r \in \mathcal{A} \setminus \mathcal{A}'$ additionally becomes liberated and deviates from the current coalition $S_{\Pi'(r)}$ to another coalition in Π'. In this setting, from the viewpoint of the agents in $\mathcal{A}' \setminus S_{\Pi'(r)}$, the newly liberated agent

is considered as the new agent in Lemma 1. Accordingly, we can still utilize the lemma for the agents in $\mathcal{A}' \setminus S_{\Pi'(r)} \cup \{a_r\}$. The agents also can find a Nash stable partition if one of them moves to $S_{\Pi'(r)}$ during the process, because, due to a_r, it became $\Delta_{\Pi'(i)} \geq 1$ for every agent $a_i \in S_{\Pi'(r)} \setminus \{a_r\}$. In a nutshell, the agents $\mathcal{A}' \cup \{a_r\}$ can converge to a Nash stable partition within $|\mathcal{A}' \cup \{a_r\}|$, which is equivalent to Lemma 1. Hence, Theorem 1 and this theorem are also valid for the case when the initial partition of the agents are randomly given. □

E. Decentralized Algorithm

In the previous section, it was assumed that only one agent is somehow chosen as the deciding agent at each iteration under the fully-connected network. On the contrary, in this section, we propose a decentralized algorithm, as shown in Algorithm 1, in which every agent does decision making based its local information and affects its neighbors simultaneously under a strongly-connected network. Despite that, we show that Theorems 1 and 2 still hold thanks to our proposed distributed mutex subroutine shown in Algorithm 2. The details of the decentralized main algorithm are as follows.

Algorithm 1 Decision-making algorithm for each agent a_i

// Initialisation
1: satisfied $\leftarrow false$; $r^i \leftarrow 0$; $s^i \leftarrow 0$
2: $\Pi^i \leftarrow \{S_\phi = \mathcal{A}, S_j = \phi \ \forall t_j \in \mathcal{T}\}$
// Decision-making process begins
3: **while** $true$ **do**
// Make a new decision if necessary
4: **if** satisfied $= false$ **then**
5: $(t_{j*}, |S_{j*}|) \leftarrow \arg\max_{\forall S_j \in \Pi^i}(t_j, |S_j \cup \{a_i\}|)$
6: **if** $(t_{j*}, |S_{j*}|) \succ_i (t_{\Pi^i(i)}, |S_{\Pi^i(i)}|)$ **then**
7: Join S_{j*} and update Π^i
8: $r^i \leftarrow r^i + 1$
9: $s^i \in \text{unif}[0, 1]$
10: **end if**
11: satisfied $= true$
12: **end if**
// Broadcast the local information to neighbor agents
13: Broadcast $M^i = \{r^i, s^i, \Pi^i\}$ and receive M^k from its neighbors $\forall a_k \in \mathcal{N}_i$
// Select the valid partition from all the received messages
14: Construct $\mathcal{M}^i_{rcv} = \{M^i, \forall M^k\}$
15: $\{r^i, s^i, \Pi^i\}$, satisfied $\leftarrow$ D-MUTEX($\mathcal{M}^i_{rcv}$)
16: **end while**

Each agent a_i has local variables such as Π^i, satisfied, r^i, and s^i (Line 1–2). Here, Π^i is the agent's locally-known partition; satisfied is a binary variable that indicates whether or not the agent is satisfied with Π^i such that it does not want to deviate from its current coalition; $r^i \in \mathbb{Z}^+$ is an integer variable to represent how many times Π^i has evolved (i.e., the number of iterations happened for updating Π^i until that moment); and $s^i \in [0, 1]$ is a uniform-random variable that is generated whenever Π^i is newly updated (i.e., a random time stamp). Given Π^i, agent a_i examines which coalition is the most preferred among others, assuming that the other agents remain at the existing coalitions (Line 5). Then, the agent joins the newly found coalition if it is strongly preferred than its current coalition. In this case, the agent updates Π^i to reflect its new decision, increases r^i, and generates a new random time stamp s^i (Line 6–10). In any case, the agent ascertained that the currently-selected coalition is the most preferred, so the agent becomes satisfied with Π^i (Line 11). Then, agent a_i generates and sends a message $M^i := \{r^i, s^i, \Pi^i\}$ to its neighbor agents, and vice versa (Line 13).

Since every agent locally updates its locally-known partition simultaneously, one of the partitions should be regarded as if it were the partition updated by a deciding agent at the previous iteration. We refer to this partition as *the valid partition* at the iteration. The distributed mutex subroutine in Algorithm 2 enables the agents to recognize the valid partition among all the locally-known current partitions even under a strongly-connected network and in asynchronous environments. Before executing this subroutine, each agent a_i collects all the messages received from its neighbor agents $\forall a_k \in \mathcal{N}_i$ (including M^i) as $\mathcal{M}^i_{rcv} = \{M^i, \forall M^k\}$. Using this message set, the agent examines whether or not its own partition Π^i is valid. If there exists any other partition Π^k such that $r^k > r^i$, then the agent considers Π^k more valid than Π^i. This also happens if $r^k = r^i$ and $s^k > s^i$, which indicates the case where Π^k and Π^i have evolved over the same amount of times, but the former has a higher time stamp. Since Π^k is considered as more valid, agent a_i will need to re-examine if there is a more preferred coalition given Π^k in the next iteration. Thus, the agent sets satisfied as $false$ (Line 3–10 in Algorithm 2). After completing this subroutine, depending on satisfied, each agent proceeds the decision-making process again (i.e., Line 4–12 in Algorithm 1) and/or just broadcasts the existing locally-known partition to its neighbor agents (Line 13 in Algorithm 1).

Algorithm 2 Distributed Mutex Subroutine

1: **function** D-MUTEX($\mathcal{M}^i_{rcv}$)
2: satisfied $\leftarrow true$
3: **for** each message $M_k \in \mathcal{M}^i_{rcv}$ **do**
4: **if** $(r^k > r^i)$ or $(r^k = r^i$ & $s^k > s^i)$ **then**
5: $r^i \leftarrow r^k$
6: $s^i \leftarrow s^k$
7: $\Pi^i \leftarrow \Pi^k$
8: satisfied $\leftarrow false$
9: **end if**
10: **end for**
11: **return** $\{r^i, s^i, \Pi^i\}$, satisfied
12: **end function**

In a nutshell, the distributed mutex algorithm makes sure that there is only one valid partition that dominates (or will finally dominate depending on the communication network) any other partitions. In other words, multiple partitions locally evolve, but one of them eventually survives as long as a strongly-connected network is given. From each partition's viewpoint, it can be regarded as being evolved by a random sequence of the agents under the fully-connected network. Thus, the partition becomes Nash stable within the polynomial time as shown in Theorem 2. In an extreme case, we may

encounter multiple Nash stable partitions at the very last. Nevertheless, thanks to the mutex algorithm, one of them can be distributedly selected by the agents. All the features imply that agents using Algorithm 1 can find a Nash stable partition in a decentralized manner and Theorems 1 and 2 still hold.

IV. Analysis

A. Algorithmic Complexity (Scalability)

Firstly, let us discuss the running time for the proposed framework to find a Nash stable partition. This paper refers to a unit time required for each agent to proceed the main loop of Algorithm 1 (Line 4–15) as a *time step*. Depending on the communication network considered, especially if it is not fully-connected, it may be possible that some of the given agents have to execute this loop to just propagate their locally-known partition information without affecting r_i as Line 8. Because this process also spends a unit time step, we call it as *dummy iteration* to distinguish from a *(normal) iteration*, which increases r_i.

Notice that such dummy iterations happen sequentially at most d_G times before a normal iteration occurs, where d_G is the graph diameter of the communication network. Hence, thanks to Theorem 2, the total required time steps until finding a Nash stable partition is $O(d_G n_a^2)$. For the fully-connected network case, it becomes $O(n_a^2)$ because of $d_G = 1$. Note that this algorithmic complexity is less than that of the centralized algorithm, i.e., $O(n_a^2 \cdot n_t)$, in [44].

Every agent at each iteration investigates $n_t + 1$ of selectable task-coalition pairs including t_ϕ given a locally-known valid partition (as shown in Line 5 in Algorithm 1). Therefore, the computational overhead for an agent is $O(n_t)$ per any iteration. With consideration of the total required time steps, the running time of the proposed approach for an agent can be bounded by $O(d_G n_t n_a^2)$. Note that the running time in practice can be much less than the bound since Theorem 2 was conservatively analyzed, as described in the following remark.

Remark 4 (*The number of required iterations in practice*)**.** Algorithm 1 allows the entire agents in $\mathcal{A}$ to be involved in the decision-making process, whereas, in the proof for Theorem 2, a new agent can be involved after a Nash stable partition of existing agents is found. Since agents using Algorithm 1 do not need to find every Nash stable partition for each subset of the agents, unnecessary iterations can be reduced. Hence, the number of required iterations in practice may become less than that shown in Theorem 2, which is also supported by the experimental results in Section VI-B.

Let us now discuss the communication overhead for each agent per iteration. Given a network, agent a_i should communicate with $|\mathcal{N}_i|$ of its neighbors, and the size of each message grows with regard to n_a. Hence, the communication overhead of the agent is $O(|\mathcal{N}_i| \cdot n_a)$. It could be quadratic if $|\mathcal{N}_i|$ increases in proportion to n_a. However, this would rarely happen in practice due to spatial distribution of agents and physical limits on communication such as range limitation. Instead, $|\mathcal{N}_i|$ would be most likely saturated.

Remark 5 (*Communication overhead vs. Running time*)**.** To reduce the communication overhead, we may impose *the maximum number of transactions per iteration*, denoted by n_c, on each agent. Even so, Theorems 1 and 2 are still valid as long as the union of underlying graphs of the communication networks over time intervals becomes connected. However, in return, the number of dummy iterations may increase, so does the framework's running time. In an extreme case where $n_c = 1$ (i.e., unicast mode), dummy iterations may happen in a row at most n_a times. Thus, the total required time steps until finding a Nash stable partition could be $O(n_a^3)$, whereas the communication overhead is $O(n_a)$. In short, the running time of the framework can be traded off against the communication overhead for each agent per iteration.

B. Suboptimality

This section investigates the *suboptimality lower bound* (or can be called *approximation ratio*) of the proposed framework in terms of the global utility, i.e., the objective function in Equation (1). Given a partition Π, the global utility value can be equivalently rewritten as

$$J = \sum_{\forall a_i \in \mathcal{A}} u_i(t_{\Pi(i)}, |S_{\Pi(i)}|). \tag{6}$$

Note that we can simply derive $\{x_{ij}\}$ for Equation (1) from Π for Equation (6), and vice versa. Let J_{GRAPE} and J_{OPT} represent the global utility of a Nash stable partition obtained by the proposed framework and the optimal value, respectively. This paper refers to the fraction of J_{GRAPE} with respect to J_{OPT} as the *suboptimality* of GRAPE, denoted by α, i.e.,

$$\alpha := J_{GRAPE}/J_{OPT}. \tag{7}$$

The lower bound of the suboptimality can be determined by the following theorem.

Theorem 3 (*Suboptimality lower bound: general case*)**.** *Given a Nash stable partition Π obtained by GRAPE, its suboptimality in terms of the global utility is lower bounded as follows:*

$$\alpha \geq J_{GRAPE}/(J_{GRAPE} + \lambda), \tag{8}$$

where

$$\lambda \equiv \sum_{\forall S_j \in \Pi} \max_{a_i \in \mathcal{A}, p \leq |\mathcal{A}|} \left\{ p \cdot \left[u_i(t_j, p) - u_i(t_j, |S_j \cup \{a_i\}|) \right] \right\} \tag{9}$$

Proof. Let Π^* denote the optimal partition for the objective function in Equation (6). Given a Nash stable partition Π, from Definition 3, it holds that, $\forall a_i \in \mathcal{A}$,

$$u_i(t_{\Pi(i)}, |S_{\Pi(i)}|) \geq u_i(t^*_{j \leftarrow i}, |S_j \cup \{a_i\}|), \tag{10}$$

where $t^*_{j \leftarrow i}$ indicates task $t_j \in \mathcal{T}$ to which agent a_i should have joined according to the optimal partition Π^*; and $S_j \in \Pi$ is the coalition for task t_j whose participants follow the Nash stable partition Π.

The right-hand side of the inequality in Equation (10) can be rewritten as

$$\begin{aligned} u_i(t^*_{j \leftarrow i}, |S_j \cup \{a_i\}|) = u_i(t^*_{j \leftarrow i}, |S^*_j|) - \\ \left\{ u_i(t^*_{j \leftarrow i}, |S^*_j|) - u_i(t^*_{j \leftarrow i}, |S_j \cup \{a_i\}|) \right\}, \end{aligned} \tag{11}$$

where $S_j^* \in \Pi^*$ is the ideal coalition of task $t_{j\leftarrow i}^*$ that maximizes the objective function.

By summing over all the agents, the inequality in Equation (10) can be said that

$$\begin{aligned}\sum_{\forall a_i \in \mathcal{A}} & u_i(t_{\Pi(i)}, |S_{\Pi(i)}|) \\ & \geq \sum_{\forall a_i \in \mathcal{A}} u_i(t_{j\leftarrow i}^*, |S_j^*|) \\ & \quad - \sum_{\forall a_i \in \mathcal{A}} \{u_i(t_{j\leftarrow i}^*, |S_j^*|) - u_i(t_{j\leftarrow i}^*, |S_j \cup \{a_i\}|)\}.\end{aligned} \tag{12}$$

The left-hand side of the inequality in Equation (12) represents the objective function value of the Nash stable partition Π, i.e., J_{GRAPE}, and the first term of the right-hand side is the optimal value, i.e., J_{OPT}. The second term in the right-hand side can be interpreted as the summation of the utility lost of each agent caused by the belated decision to its optimal task, provided that the other agents still follow the Nash stable partition.

The upper bound of the second term is given by

$$\sum_{j=1}^{n_t} |S_j^*| \cdot \max_{a_i \in S_j^*} \{u_i(t_{j\leftarrow i}^*, |S_j^*|) - u_i(t_{j\leftarrow i}^*, |S_j \cup \{a_i\}|)\}. \tag{13}$$

This is at most

$$\sum_{\forall S_j \in \Pi} \max_{a_i \in \mathcal{A}, p \leq |\mathcal{A}|} L_{ij}[p] \equiv \lambda, \tag{14}$$

where $L_{ij}[p] = p \cdot (u_i(t_j, p) - u_i(t_j, |S_j \cup \{a_i\}|))$.

Hence, the inequality in Equation (12) can be rewritten as

$$J_{GRAPE} \geq J_{OPT} - \lambda.$$

Dividing both sides by J_{GRAPE} and rearranging them yield the suboptimality lower bound of the Nash stable partition, as given by Equation (8). □

Although Theorem 3 does not provide a fixed-value lower bound, it can be determined as long as a Nash stable partition and agents' individual utility functions are given. Nevertheless, as a special case, if the social utility for any coalition is non-decreasing (or monotonically increasing) in terms of the number of co-working agents, then we can obtain a fixed-value lower bound for the suboptimality of a Nash stable partition.

Theorem 4 (*Suboptimality lower bound: a special case*)**.** *Given an instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ of GRAPE, if (i) the social utility for any coalition is non-decreasing with regard to the number of participants, i.e., for any $S_j \subseteq \mathcal{A}$ and $a_l \in \mathcal{A} \setminus S_j$, it holds that*

$$\sum_{\forall a_i \in S_j} u_i(t_j, |S_j|) \leq \sum_{\forall a_i \in S_j \cup \{a_l\}} u_i(t_j, |S_j \cup \{a_l\}|),$$

and (ii) all the individual utilities can derive SPAO preference relations, then a Nash stable partition Π obtained by GRAPE provides at least 50% of suboptimality in terms of the global utility.

Proof. Firstly, we introduce some definitions and notations that facilitate to describe this proof. Given a partition Π of an instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$, the global utility is denoted by

$$V(\Pi) := \sum_{\forall a_i \in \mathcal{A}} u_i(t_{\Pi(i)}, |S_{\Pi(i)}|). \tag{15}$$

We use operator $\oplus$ as follows. Given any two partitions $\Pi^A = \{S_0^A, ..., S_{n_t}^A\}$ and $\Pi^B = \{S_0^B, ..., S_{n_t}^B\}$,

$$\Pi^A \oplus \Pi^B := \{S_0^A \cup S_0^B,\ S_1^A \cup S_1^B, ...,\ S_{n_t}^A \cup S_{n_t}^B\}.$$

Since $\cup_{j=0}^{n_t} S_j^A = \cup_{j=0}^{n_t} S_j^B = \mathcal{A}$, there may exist the same agent a_i even in two different coalitions in $\Pi^A \oplus \Pi^B$. For instance, suppose that $\Pi^A = \{\{a_1\}, \{a_2\}, \{a_3\}\}$ and $\Pi^B = \{\emptyset, \{a_1, a_3\}, \{a_2\}\}$. Then, $\Pi^A \oplus \Pi^B = \{\{a_1\}, \{a_1, a_2, a_3\}, \{a_2, a_3\}\}$. We regard such an agent as two different agents in $\Pi^A \oplus \Pi^B$. Accordingly, the operation may increase the number of total agents in the resultant partition.

Using the definitions described above, condition (i) implies that

$$V(\Pi^A) \leq V(\Pi^A \oplus \Pi^B). \tag{16}$$

From now on, we will show that $\frac{1}{2} V(\Pi^*) \leq V(\hat{\Pi})$, where $\Pi^* = \{S_0^*, S_1^*, ..., S_{n_t}^*\}$ is an optimal partition and $\hat{\Pi} = \{\hat{S}_0, \hat{S}_1, ..., \hat{S}_{n_t}\}$ is a Nash stable partition. By doing so, this theorem can be proved. From the definition in Equation (15), it can be said that

$$\begin{aligned}V(\hat{\Pi} \oplus \Pi^*) = & \sum_{\forall a_i \in \mathcal{A}} u_i(t_{\hat{\Pi}(i)}, |\hat{S}_{\hat{\Pi}(i)} \cup S_{\hat{\Pi}(i)}^*|) \\ & + \sum_{\forall a_i \in \mathcal{A}^-} u_i(t_{\Pi^*(i)}, |\hat{S}_{\Pi^*(i)} \cup S_{\Pi^*(i)}^*|),\end{aligned} \tag{17}$$

where $\mathcal{A}^-$ is the set of agents whose decisions follow not the Nash stable partition $\hat{\Pi}$ but only the optimal partition Π^*. Due to condition (ii), the first term of the right-hand side in Equation (17) is no more than

$$\sum_{\forall a_i \in \mathcal{A}} u_i(t_{\hat{\Pi}(i)}, |\hat{S}_{\hat{\Pi}(i)}|) \equiv V(\hat{\Pi}). \tag{18}$$

Likewise, the second term is also at most

$$\sum_{\forall a_i \in \mathcal{A}^-} u_i(t_{\Pi^*(i)}, |\hat{S}_{\Pi^*(i)} \cup \{a_i\}|). \tag{19}$$

By the definition of Nash stability (i.e., for every agent $a_i \in \mathcal{A}$, $u_i(t_{\hat{\Pi}(i)}, |\hat{S}_{\hat{\Pi}(i)}|) \geq u_i(t_j, |\hat{S}_j \cup \{a_i\}|)$, $\forall \hat{S}_j \in \hat{\Pi}$), the above equation is at most

$$\sum_{\forall a_i \in \mathcal{A}^-} u_i(t_{\hat{\Pi}(i)}, |\hat{S}_{\hat{\Pi}(i)}|), \tag{20}$$

which is also no more than, because of $\mathcal{A}^- \subseteq \mathcal{A}$,

$$\sum_{\forall a_i \in \mathcal{A}} u_i(t_{\hat{\Pi}(i)}, |\hat{S}_{\hat{\Pi}(i)}|) \equiv V(\hat{\Pi}). \tag{21}$$

Accordingly, the left-hand side of Equation (17) holds the following inequality:

$$V(\hat{\Pi} \oplus \Pi^*) \leq 2V(\hat{\Pi}). \tag{22}$$

Thanks to Equation (16), it follows that

$$V(\Pi^*) \leq V(\hat{\Pi} \oplus \Pi^*).$$

Therefore, $V(\Pi^*) \leq 2V(\hat{\Pi})$, which completes this proof. □

C. Adaptability

Our proposed framework is also adaptable to dynamic environments such as unexpected addition or loss of agents or tasks, owing to its fast convergence to a Nash stable partition. Thanks to Lemma 1, if a new agent additionally joins an ongoing mission in which an assignment was already determined, the number of iterations required for converging to a new Nash stable partition is at most the number of the total agents. Responding to any environmental change, the framework is able to establish a new agreed task assignment within polynomial time.

D. Robustness in Asynchronous Environments

In the proposed framework, for every iteration, each agent does not need to wait until nor ensure that its locally-known information has been propagated to a certain neighbor group. Instead, as described in Remark 5, it is enough for the agent to receive the local information from one of its neighbors, to make a decision, and to send the updated partition back to some of its neighbors. Temporary disconnection or non-operation of some agents may cause dummy iterations additionally. However, it does not affect the existence of, the convergence to, and the suboptimality of a Nash stable partition under the proposed framework, which is also supported by Section VI-E.

V. GRAPE with Minimum Requirements

This section addresses another task allocation problem where each task may require at least a certain number of agents for its completion. This problem can be defined as follows.

Problem 2. Given a set of agents $\mathcal{A}$ and a set of tasks $\mathcal{T}$, the objective is to find an assignment such that

$$\max_{\{x_{ij}\}} \sum_{\forall a_i \in \mathcal{A}} \sum_{\forall t_j \in \mathcal{T}} u_i(t_j, p) x_{ij}, \tag{23}$$

subject to

$$\sum_{\forall a_i \in \mathcal{A}} x_{ij} \geq R_j, \quad \forall t_j \in \mathcal{T}, \tag{24}$$

$$\sum_{\forall t_j \in \mathcal{T}} x_{ij} \leq 1, \quad \forall a_i \in \mathcal{A}, \tag{25}$$

$$x_{ij} \in \{0, 1\}, \quad \forall a_i \in \mathcal{A}, \forall t_j \in \mathcal{T}, \tag{26}$$

where $R_j \in \mathbb{N} \cup \{0\}$ is the number of minimum required agents for task t_j, and all the other variables are identically defined as those in Problem 1. Here, it is considered that, for $\forall a_i \in \mathcal{A}$ and $\forall t_j \in \mathcal{T}$,

$$u_i(t_j, p) = 0 \quad \text{if } p < R_j \tag{27}$$

because task t_j cannot be completed in this case. Note that any task t_j without such a requirement is regarded to have $R_j = 0$.

For each task t_j having $R_j > 0$, even if $u_i(t_j, p)$ is monotonically decreasing when $p \geq R_j$, the individual utility can not be simply transformed to a preference relation holding SPAO because of Equation (27). Thus, we need to modify the utility function to yield alternative values for the case when $p < R_j$. We refer to the modified utility as *auxiliary individual utility* $\tilde{u}_i$, which is defined as

$$\tilde{u}_i(t_j, p) = \begin{cases} u_i^0(t_j, p) & \text{if } p \leq R_j \\ u_i(t_j, p) & \text{otherwise,} \end{cases} \tag{28}$$

where $u_i^0(t_j, p)$ is the *dummy utility* of agent a_i with regard to task t_j when $p \leq R_j$.

The dummy utility is intentionally used also for the case when $p = R_j$ in order to find an assignment that holds Equation (24). For this, the auxiliary individual utility should satisfy the following condition.

Condition 1. For every agent $a_i \in \mathcal{A}$, its preference relation $\mathcal{P}_i$ holds that, for any two tasks $t_j, t_k \in \mathcal{T}$,

$$(t_j, R_j) \succ_i (t_k, R_k + 1).$$

This condition enables every agent to prefer a task for which the number of co-working agents is less than its minimum requirement, over any other tasks whose requirements are already fulfilled. Under this condition, as long as the agent set $\mathcal{A}$ is such that $|\mathcal{A}| \geq \sum_{\forall t_j \in \mathcal{T}} R_j$ and a Nash stable partition is found, the resultant assignment satisfies Equation (24).

Proposition 1. *Given an instance of Problem 2 where $u_i(t_j, p)$ $\forall i$ $\forall j$ is a monotonically decreasing function with regard to $\forall p \geq R_j$, if the dummy utilities $u_i^0(t_j, p)$ $\forall i$ $\forall j$ in (28) are set to satisfy Condition 1 and SPAO for $\forall p \leq R_j$, then all the resultant auxiliary individual utilities $\tilde{u}_i(t_j, p)$ $\forall i$ $\forall j$ $\forall p$ can be transformed to a n_a-tuple of preference relations $\mathcal{P}$ that hold Condition 1 as well as SPAO for $\forall p \in \{1, ..., n_a\}$. In the corresponding instance of GRAPE $(\mathcal{A}, \mathcal{T}, \mathcal{P})$, a Nash stable partition can be determined within polynomial times as shown in Theorems 1 and 2 because of SPAO, and the resultant partition can satisfy Equation (24) due to Condition 1.*

Let us give an example. Suppose that there exist 100 agents $\mathcal{A}$, and 3 tasks $\mathcal{T} = \{t_1, t_2, t_3\}$ where only t_3 has its minimum requirement $R_3 = 5$; for every agent $a_i \in \mathcal{A}$, individual utilities for t_1 and t_2, i.e., $u_i(t_1, p)$ and $u_i(t_2, p)$, are much higher than that for t_3 in $\forall p \in \{1, ..., 100\}$. We can find a Nash stable partition for this example, as described in Proposition 1, by setting $u_i^0(t_j, p) = \max_{\forall t_j}\{u_i(t_j, R_j + 1)\} + \beta$ for $\forall p \leq R_j$, $\forall a_i \in \mathcal{A}$, where $\beta > 0$ is an arbitrary positive constant.

After a Nash stable partition is found, in order to compute the objective function value in (23), the original individual utility function u_i should be used instead of the auxiliary one $\tilde{u}_i$.

Proposition 2. *Given a Nash stable partition Π obtained by implementing Proposition 1, its suboptimality bound α is such that*

$$\alpha \geq \frac{J_{GRAPE}}{J_{GRAPE} + \tilde{\lambda}} \cdot \frac{J_{GRAPE}}{J_{GRAPE} + \delta}. \tag{29}$$

Here, $\delta \equiv \tilde{J}_{GRAPE} - J_{GRAPE}$, where $\tilde{J}_{GRAPE}$ (or J_{GRAPE}) is the objective function value in (23) using $\tilde{u}_i$ (or using u_i) given the Nash stable partition. Likewise, $\tilde{\lambda}$ is the value in (9)

using $\tilde{u}_i$. In addition to this, if every $\tilde{u}_i$ satisfies the conditions for Theorem 4, then

$$\alpha \geq \frac{1}{2} \cdot \frac{J_{GRAPE}}{J_{GRAPE} + \delta}. \tag{30}$$

Proof. Since the Nash stable partition Π is obtained by using $\tilde{u}_i$, it can be said from Equations (7) and (8) that

$$\frac{\tilde{J}_{GRAPE}}{\tilde{J}_{OPT}} \geq \frac{\tilde{J}_{GRAPE}}{\tilde{J}_{GRAPE} + \tilde{\lambda}}. \tag{31}$$

Due to the fact that $\tilde{u}_i(t_j, p) \geq u_i(t_j, p)$ for $\forall i, j, p$, it is clear that $\tilde{J}_{GRAPE} \geq J_{GRAPE}$ and $\tilde{J}_{OPT} \geq J_{OPT}$. By letting that $\delta := \tilde{J}_{GRAPE} - J_{GRAPE}$, the left term in (31) is at most $(J_{GRAPE} + \delta)/J_{OPT}$. Besides, the right term in (31) is a monotonically-increasing function with regard to $\tilde{J}_{GRAPE}$, and thus, it is lower bounded by $J_{GRAPE}/(J_{GRAPE} + \tilde{\lambda})$. From this, Equation (31) can be rewritten as Equation (29) by multiplying $J_{GRAPE}/(J_{GRAPE} + \delta)$.

Likewise, for the case when every $\tilde{u}_i$ satisfies the conditions for Theorem 4, it can be said that $\tilde{J}_{GRAPE} \geq 1/2 \cdot \tilde{J}_{OPT}$, which can be transformed into Equation (30) as shown above. □

Notice that if $\delta = 0$ for the Nash stable partition in Proposition 2, then the suboptimality bounds become equivalent to those in Theorems 3 and 4.

VI. Simulation and Results

This section validates the performances of the proposed framework with respect to its scalability, suboptimality, adaptability against dynamic environments, and robustness in asynchronous environments.

A. Mission Scenario and Settings

1) Utility functions: Firstly, we introduce the social and individual utilities used in this numerical experiment. We consider that if multiple robots execute a task together as a coalition, then they are given a certain level of reward for the task. The amount of the reward varies depending on the number of the co-working agents. The reward is shared with the agents, and each agent's individual utility is considered as the shared reward minus the cost required to personally spend on the task (e.g., fuel consumption for movement). In this experiment, *the equal fair allocation rule* [53], [54] is adopted. Under the rule, a task's reward is equally shared among the members. Therefore, the individual utility of agent a_i executing task t_j with coalition S_j is defined as

$$u_i(t_j, |S_j|) = r(t_j, |S_j|)/|S_j| - c_i(t_j), \tag{32}$$

where $r(t_j, |S_j|)$ is the reward from task t_j when it is executed by S_j together, and $c_i(t_j)$ is the cost that agent a_i needs to pay for the task. Here, we simply set the cost as a function of the distance from agent a_i to task t_j. We set that if $u_i(t_j, |S_j|)$ is not positive, agent a_i prefers to join S_ϕ over S_j.

This experiment considers two types of tasks. For the first type, a task's reward becomes higher as the number of participants gets close to a specific desired number. We refer to such a task as a *peaked-reward* task, and its reward can be defined as

$$r(t_j, |S_j|) = \frac{r_j^{\max} \cdot |S_j|}{n_j^d} \cdot e^{-|S_j|/n_j^d + 1}, \tag{33}$$

where n_j^d represents the desired number, and $r_j^{\max}$ is the peaked reward in case that n_j^d of agents are involved in. Consequently, the individual utility of agent a_i with regard to task t_j becomes the following equation:

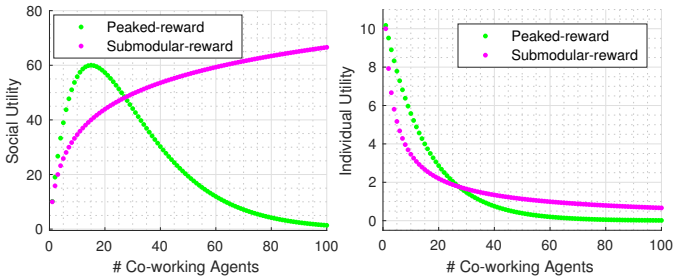
$$u_i(t_j, |S_j|) = \frac{r_j^{\max}}{n_j^d} \cdot e^{-|S_j|/n_j^d + 1} - c_i(t_j). \tag{34}$$

For the second type, a task's reward becomes higher as more agents are involved, but the corresponding marginal gain decreases. This type of tasks is said to be *submodular-reward*, and the reward can be defined as

$$r(t_j, |S_j|) = r_j^{\min} \cdot \log_{\epsilon_j}(|S_j| + \epsilon_j - 1), \tag{35}$$

where $r_j^{\min}$ indicates the reward obtained if there is only one agent involved, and $\epsilon_j > 1$ is the design parameter regarding the diminishing marginal gain. The resultant individual utility becomes as follows:

$$u_i(t_j, |S_j|) = r_j^{\min} \cdot \log_{\epsilon_j}(|S_j| + \epsilon_j - 1)/|S_j| - c_i(t_j). \tag{36}$$

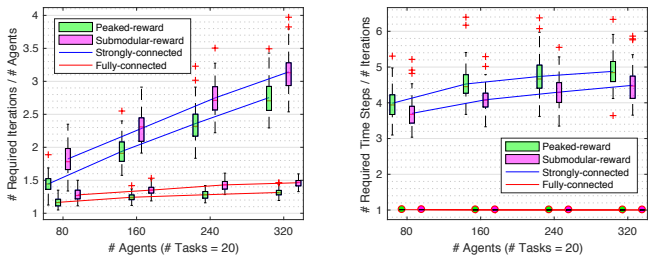


(a) Social utility (b) Individual utility

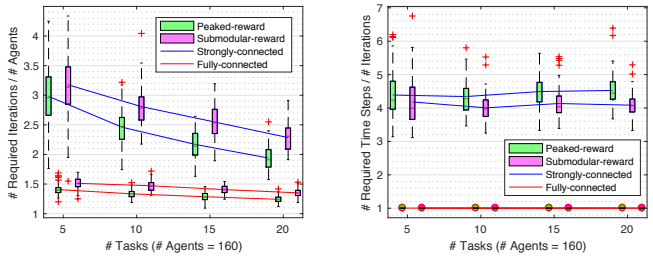
Fig. 1. Examples of utility functions used in the numerical experiment are shown, depending on the two different task types (i.e., peaked-reward and submodular-reward): **(a)** the social utility of a coalition; **(b)** an agent's individual utility.

Figure 1 illustrates examples of the social utilities and individual utilities for the task types introduced above. For simplification, agents' costs are ignored in the figure. We set $r_j^{\max}$, n_j^d, $r_j^{\min}$ and ϵ_j to be 60, 15, 10, and 2, respectively. Notice that the individual utilities are monotonically decreasing in both cases, as depicted in Figure 1(b). Therefore, given a mission that entails these task types, we can generate an instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ of GRAPE that holds SPAO.

2) Parameters generation: In the following sections, we will mainly utilize Monte Carlo simulations. At each run, n_t tasks and n_a agents are uniform-randomly located in a 1000 $m \times 1000$ m arena and a 250 $m \times 250$ m arena within there, respectively. For a scenario including peaked-reward tasks, $r_j^{\max}$ is randomly generated from a uniform distribution over $[1000, 2000] \times n_a/n_t$, and n_j^d is set to be the rounded value of $(r_j^{\max}/\sum_{\forall t_k \in \mathcal{T}^*} r_k^{\max}) \times n_a$. For a scenario including submodular-reward tasks, ϵ_j is set as 2, and $r_j^{\min}$ is uniform-randomly generated over $[1000, 2000] \times 1/\log_{\epsilon_j}(n_a/n_t + 1)$.



(a) $n_a \in \{80, 160, 240, 320\}$ with fixed $n_t = 20$



(b) $n_t \in \{5, 10, 15, 20\}$ with fixed $n_a = 160$

Fig. 2. Convergence performance of the proposed framework is shown, depending on communication networks (i.e., Strongly-connected vs. Fully-connected) and utility function types (i.e., Peaked-reward vs. Submodular-reward) with different number of agents and tasks: **(Left)** the number of (normal) iterations happened relative to that of agents; **(Right)** the number of time steps happened (i.e., normal and dummy iterations) relative to that of iterations.

3) Communication network: Given a set of agents, their communication network is strongly-connected in a way that only contains a bidirectional minimum spanning tree with consideration of the agents' positions. Furthermore, we also consider the fully-connected network in some experiments in order to examine the influence of the network. The communication network is randomly generated at each instance, and is assumed to be sustained during a mission except the robustness test simulations in Section VI-E.

B. Scalability

To investigate the effectiveness of n_t and n_a upon the scalability of the proposed approach, we conduct Monte Carlo simulations with 100 runs for the scenarios introduced in Section VI-A with a fixed $n_t = 20$ and various $n_a \in \{80, 160, 240, 320\}$ and for those with $n_a = 160$ and $n_t \in \{5, 10, 15, 20\}$. Figure 2 shows the statistical results using box-and-whisker plots, where the green boxes indicate the results from the scenarios with the peaked-reward tasks and the magenta boxes are those with the submodular-reward tasks. The blue and red lines connecting the boxes represent the average value for each test case (n_a, n_t) under a strongly-connected network and the fully-connected network, respectively.

The left subfigure in Figure 2(a) shows that the ratio of the number of required (normal) iterations to that of agents linearly increases as more agents are involved. This implies that the proposed framework has quadratic complexity with regard to the number of agents (i.e., $C_1 n_a^2$), as stated in Theorem 2, but with C_1 being much less than $\frac{1}{2}$, which is the value from the theorem. C_1 can become even lower (e.g., $C_1 = 5 \times 10^{-4}$ in the experiments) under the fully-connected network. Such C_1 being smaller than $\frac{1}{2}$ may be explained by Remark 4: the algorithmic efficiency of Algorithm 1 can reduce unnecessary iterations that may be induced in the procedure of the proof for Theorem 2.

On the other hand, the left subfigure in Figure 2(b) shows that the number of required iterations decreases with regard to the number of tasks. This trend may be caused by the fact that more selectable options provided to the fixed number of agents can reduce possible conflicts between the agents.

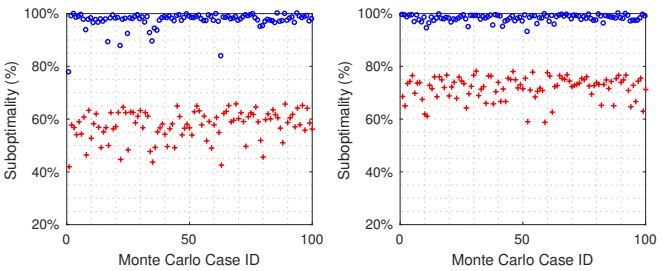
Furthermore, in the two results, the trends regarding either n_a or n_t have higher slopes under a strongly-connected network than those under the fully-connected network. This is because the former condition is more sensitive to conflicts between agents, and thus causes additional iterations. For example, agents at the middle nodes of the network may change their decisions (and thus increase the number of iterations) while the local partition information of the agent at one end node is being propagated to the other end nodes. Such unnecessary iterations in the middle might not have occurred if the agents at all the end nodes were directly connected to each other.

The right subfigures in Figure 2(a) and (b) indicate that approximately 3–4 times of dummy iterations, compared with the required number of normal iterations, are additionally needed under a strongly-connected network. Noting that the mean values of the graph diameter d_G for the instances with $n_a \in \{80, 160, 240, 320\}$ are $36, 58, 75$ and 92, respectively, the results show that the amount of dummy iterations happened is much less than the bound value, which is d_G as pointed out in Section IV-A. On the contrary, under the fully-connected network, there is no need of such a dummy iteration, and thus the required number of iterations and that of time steps are the same.

C. Suboptimality

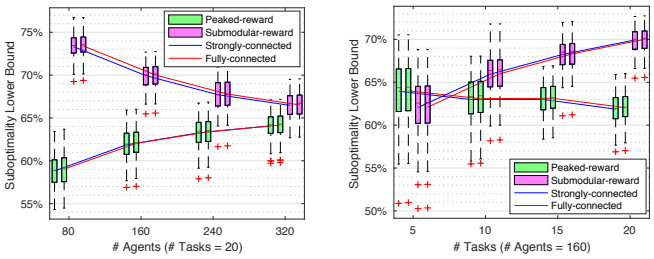
This section examines the suboptimality of the proposed framework by using Monte Carlo simulations with 100 instances. In each instance, there are $n_t = 3$ of tasks and $n_a = 12$ of agents who are strongly-connected. Figure 3 presents the true suboptimality of each instance, which is the ratio of the global utility obtained by the proposed framework to that by a brute-force search, i.e., J_{GRAPE}/J_{OPT}, and the lower bound given by Theorem 3. A blue circle and a red cross in the figure indicate the true suboptimality and the lower bound, respectively. The results show that the framework provides near-optimal solutions in almost all cases and the suboptimality of each Nash stable partition is enclosed by the corresponding lower bound.

The suboptimality may be improved if the agents are allowed to investigate a larger search space, for example, possible coalitions caused by co-deviation of multiple agents. However, this strategy in return may increase communication transactions between the agents because they have to notice each other's willingness unless their individual utility functions are known to each other, which is in contradiction to Assumption 4. Besides, the computational overhead for each



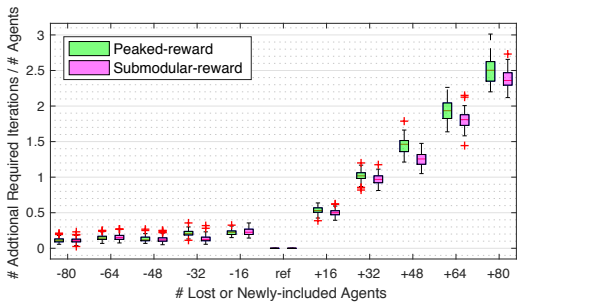
(a) Peaked-reward task (b) Submodular-reward task

Fig. 3. True suboptimality of a Nash stable partition obtained by GRAPE for each run of the Monte Carlo simulation (denoted by a blue circle) and its lower bound provided by Theorem 3 (denoted by a red cross) under a strongly-connected communication network: **(a)** the scenarios with peaked-reward tasks; **(b)** the scenarios with submodular-reward tasks

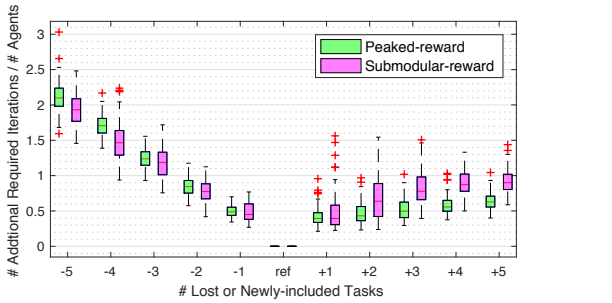


(a) (b)

Fig. 4. The suboptimality lower bound, given by Theorem 3, of a Nash stable partition obtained by GRAPE, depending on communication networks (i.e., Strongly-connected vs. Fully-connected) and utility function types (i.e., Peaked-reward vs. Submodular-reward): **(a)** fixed $n_t = 20$ with varying $n_a \in \{80, 160, 240, 320\}$; **(b)** fixed $n_a = 160$ with varying $n_t \in \{5, 10, 15, 20\}$



(a) Dynamic agents



(b) Dynamic tasks

Fig. 5. The plot shows the number of additional iterations required for re-converging to a Nash stable partition relative to the number of agents in the case when some agents or tasks are partially lost or newly involved (Baseline: $n_t = 10$, $n_a = 160$, and a Nash stable partition was already found). Negative values in the x-axis indicate that the corresponding number of existing agents or tasks are lost. Positive values indicate that the corresponding number of new agents or tasks are included in an ongoing mission. A strongly-connected communication network is used.

agent per iteration also becomes more expensive than $O(n_t)$, which is the complexity for unilateral searching as shown in Section IV-A. Hence, the resultant algorithm's complexity may hinder its practical applicability to a large-scale multiple agent system.

Figure 4 depicts the suboptimality lower bounds for the large-size problems that were previously addressed in Section VI-B. It is clearly shown that the agent communication network does not make any effect on the suboptimality lower bound of a Nash stable partition. Although there is no universal trend of the suboptimality with regard to n_a and n_t in both utility types, it is suggested that the features of the lower bound given by Theorem 3 can be influenced by the utility functions considered. In the experiments, the suboptimality bound averagely remain above than 60–70 %.

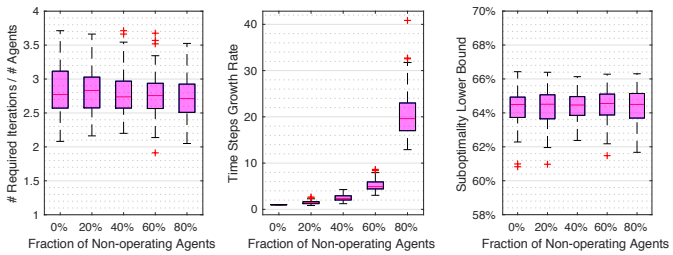
D. Adaptability

This section discusses the adaptability of our proposed framework in response to dynamic environments such as unexpected inclusion or loss of agents or tasks. Suppose that there are 10 tasks and 160 agents in a mission, and a Nash stable partition was already found as a baseline. During the mission, the number of agents (or tasks) changes; the range of the change is from losing 50% of the existing agents (or tasks) to additionally including new ones as much as 50% of them. For each dynamical environment, we perform a Monte Carlo simulation with 100 instances, randomly including or excluding a subset of the corresponding number of agents or tasks. Here, we consider a strongly-connected communication network.

Figure 5(a) illustrates that the more agents are involved additionally, the more iterations are required for re-converging to a new Nash stable partition. This is because the inclusion of a new agent may lead to additional iterations at most as much as the number of the total agents including the new agent (as shown in Lemma 1). On the contrary, the loss of existing agents does not seem to have any apparent relation with the number of iterations. A possible explanation is that the exclusion of an existing agent is favorable to the other agents due to SPAO preferences. This stimulates only a limited number of agents who are preferred to move to the coalition where the excluded agent was. This feature induces fewer additional iterations to reach a new Nash stable partition, compared with the case of adding a new agent.

Figure 5(b) shows that eliminating existing tasks causes more iterations than including new tasks. This can be explained by the fact that removing any task releases the agents performing the task free and it results in extra iterations at least as much as the number of the freed agents. On the other hand, adding new tasks induces relatively fewer additional iterations because only some of the existing agents are attracted to these tasks.

In summary, as the ratio of the number of agents to that of tasks increases, the number of additional iterations for



(a) (b) (c)

Fig. 6. Robustness test in asynchronous environments at scenarios with $n_t = 10, n_a = 160$, and the submodular-reward tasks: the plot shows the effectiveness of the fraction of non-operating agents with regard to: **(a)** the number of iterations happened until convergence relative to that of agents; **(b)** the ratio of the time steps happened to those for the normal case; **(c)** the suboptimality lower bound by Theorem 3.

convergence to a new Nash stable partition also increases. This result corresponds to the trend described in Section VI-B, i.e., the left subfigures in Figure 2(a) and (b). In all the cases of this experiment, the number of additionally induced iterations still remains at the same order of the number of the given agents, which implies that the proposed framework provides excellent adaptability.

E. Robustness in Asynchronous Environments

This section investigates the robustness of the proposed framework in asynchronous environments. This scenario assumes that a certain fraction of the given agents, which are randomly chosen at each time step, somehow can not execute Algorithm 1 and even can not communicate with other normally-working neighbor agents. We refer to such agents as *non-operating* agents. Given that $n_t = 5$ and $n_a = 40$, the fractions of the non-operating agents are set as $\{0, 0.2, 0.4, 0.6, 0.8\}$. In each case, we conduct 100 instances of Monte Carlo experiments for which the submodular-reward tasks are used.

Figure 6(a) presents that the number of (normal) iterations required for converging to a Nash stable partition remains at the same level regardless of the fraction of the non-operating agents. Despite that, the required time steps increase as more agents become non-operating, as shown in Figure 6(b). Note that *time steps growth rate* means the ratio of the total required time steps to those for the case when all the agents operate normally. These findings indicate that, due to communicational discontinuity caused by the non-operating agents, the framework may take more time to wait for these agents to operate again and then to disseminate locally-known partition information over the entire agents. Accordingly, dummy iterations may increase in asynchronous environments, but the proposed framework is still able to find a Nash stable partition. Furthermore, the resultant Nash stable partition's suboptimality lower bound obtained by Theorem 3 is not affected, as presented in Figure 6(c).

F. Visualization

We have $n_a = 320$ agents and $n_t = 5$ tasks. The initial locations of the given agents are randomly generated, and the overall formation shape is different in each test scenario such as being circle, skewed circle, and square (denoted by Scenario #1, #2, and #3, respectively). The tasks are also randomly located away from the agents. In this simulation, each agent is able to communicate with its nearby agents within a radius of 50 m. Here, the submodular-reward tasks are used.

Figure 7 shows the visualized task allocation results, where the circles and the squares indicate the positions of the agents and the tasks, respectively. The lines between the circles represent the communication networks of the agents. The colored agents are assigned to the same colored task, for example, yellow agents belong to the team for executing the yellow task. The size of a square indicates the reward of the corresponding task. The cost for an agent with regard to a task is considered as a function of the distance from the agent to the task. The allocation results seem to be reasonable with consideration of the task rewards and the costs.

The number of iterations required to find a Nash stable partition is 1355, 1380, and 1295 for Scenario #1, #2, and #3, respectively. The number of dummy iterations happened is just 20–30% of that of the iterations. This value is much fewer than the results in Figure 2 because the networks considered here are more connected than those in Section VI-B.

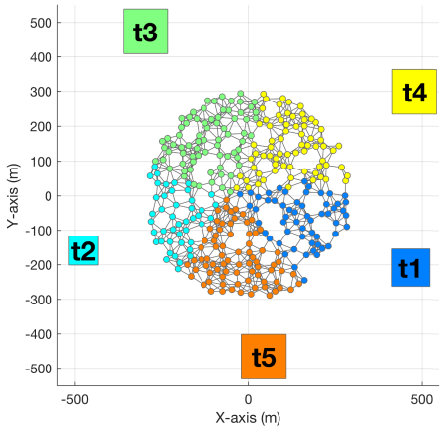
VII. Conclusion

This paper proposed a novel game-theoretical framework that addresses a task allocation problem for a robotic swarm consisting of self-interested agents. We showed that selfish agents whose individual interests are transformable to SPAO preferences can converge to a Nash stable partition by using the proposed decentralized algorithm, which is straightforward and executable even in asynchronous environments and under a strongly-connected communication network. We analytically and experimentally presented that the proposed framework provides scalability, a certain level of guaranteed suboptimality, adaptability, robustness, and the potential to accommodate different interests of agents.

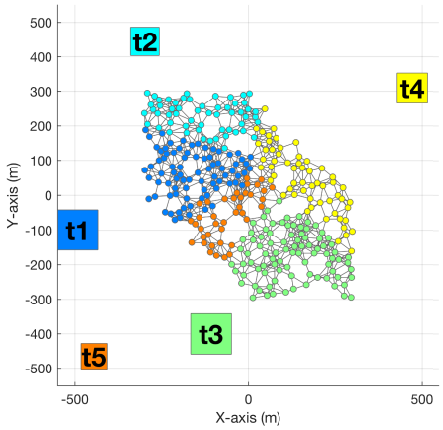
As this framework can be considered as a new sub-branch of self-organized approaches, one of our ongoing works is to compare it with one of the existing methods. Defining a fair scenario for both methods is non-trivial and requires careful consideration; otherwise, a resultant unsuitable scenario may provide biased results. Secondly, another natural progression of this study is to relax anonymity of agents and thus to consider a combination of the agents' identities. Experimentally, we have often observed that heterogeneous agents with social inhibition also can converge to a Nash stable partition. More research would be needed to analyze the quality of a Nash stable partition obtained by the proposed framework in terms of min-max optimisation because our various experiments showed that the outcome provides individual utilities to agents in a balanced manner.

Acknowledgment

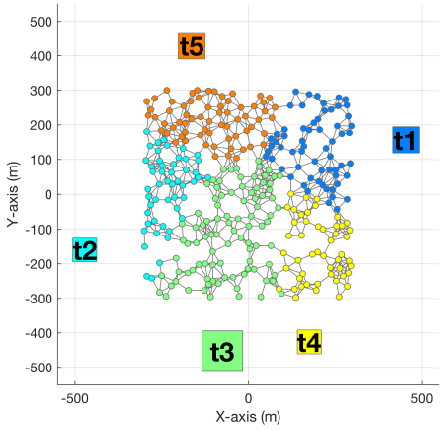
The authors gratefully acknowledge that this research was supported by International Joint Research Programme with Chungnam National University (No. EFA3004Z)



(a) Scenario #1



(b) Scenario #2



(c) Scenario #3

Fig. 7. Visualized task allocation results are shown with different geographic scenarios ($n_t = 5$, $n_a = 320$). Each square and its size represent each task's position and its reward (or demand), respectively. The circles and the lines between them indicate the positions of agents and their communication network, respectively. The color of each circle implies that the corresponding agent is assigned to the same colored task.

REFERENCES

[1] H.-S. Shin and P. Segui-Gasco, "UAV Swarms: Decision-Making Paradigms," in *Encyclopedia of Aerospace Engineering*. John Wiley & Sons, 2014, pp. 1–13.

[2] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot Task Allocation: A Review of the State-of-the-Art," in *Cooperative Robots and Sensor Networks 2015*. Cham: Springer International Publishing, 2015, vol. 604, pp. 31–51.

[3] A. Jevtić, Á. Gutierrez, D. Andina, and M. Jamshidi, "Distributed Bees Algorithm for Task Allocation in Swarm of Robots," *IEEE Systems Journal*, vol. 6, no. 2, pp. 296–304, 2012.

[4] E. Sahin, "Swarm Robotics: From Sources of Inspiration to Domains of Application," in *Swarm Robotics*. Berlin: Springer, 2005, pp. 10–20.

[5] M. Dorigo, M. Birattari, and M. Brambilla, "Swarm robotics," *Scholarpedia*, vol. 9, no. 1, p. 1463, 2014.

[6] M. Saska, V. Vonásek, J. Chudoba, J. Thomas, G. Loianno, and V. Kumar, "Swarm Distribution and Deployment for Cooperative Surveillance by Micro-Aerial Vehicles," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 84, no. 1-4, pp. 469–492, 2016.

[7] I. Bekmezci, O. K. Sahingoz, and S. Temel, "Flying Ad-Hoc Networks (FANETs): A Survey," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013.

[8] M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz, "Help from the Sky: Leveraging UAVs for Disaster Management," *IEEE Pervasive Computing*, vol. 16, no. 1, pp. 24–32, 2017.

[9] I. Jang, J. Jeong, H.-S. Shin, S. Kim, A. Tsourdos, and J. Suk, "Cooperative Control for a Flight Array of UAVs and an Application in Radar Jamming," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 8011–8018, 2017.

[10] B. P. Gerkey and M. J. Matarić, "A Formal Analysis and Taxonomy of Task Allocation in Multi-robot Systems," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[11] G. A. Korsah, A. Stentz, and M. B. Dias, "A Comprehensive Taxonomy for Multi-robot Task Allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[12] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic and Distributed Control of a Large-Scale Swarm of Autonomous Agents," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1103–1123, 2017.

[13] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm Robotics: a Review from the Swarm Engineering Perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[14] L. Johnson, S. Ponda, H.-L. Choi, and J. How, "Asynchronous Decentralized Task Allocation for Dynamic Environments," in *Infotech@Aerospace*, St.Louis, MO, USA, 2011.

[15] C. M. Clark, R. Morton, and G. A. Bekey, "Altruistic relationships for optimizing task fulfillment in robot communities," *Distributed Autonomous Robotic Systems 8*, pp. 261–270, 2009.

[16] J. H. Drèze and J. Greenberg, "Hedonic Coalitions: Optimality and Stability," *Econometrica*, vol. 48, no. 4, pp. 987–1003, 1980.

[17] S. Banerjee, H. Konishi, and T. Sönmez, "Core in a Simple Coalition Formation Game," *Social Choice and Welfare*, vol. 18, no. 1, pp. 135–153, 2001.

[18] A. Bogomolnaia and M. O. Jackson, "The Stability of Hedonic Coalition Structures," *Games and Economic Behavior*, vol. 38, no. 2, pp. 201–230, 2002.

[19] A. Brutschy, G. Pini, C. Pinciroli, M. Birattari, and M. Dorigo, "Self-organized Task Allocation to Sequentially Interdependent Tasks in Swarm Robotics," *Autonomous Agents and Multi-Agent Systems*, vol. 28, no. 1, pp. 101–125, 2014.

[20] N. Kalra and A. Martinoli, "A Comparative Study of Market-Based and Threshold-Based Task Allocation," in *Distributed Autonomous Robotic Systems 7*, Tokyo, Ed. Springer Japan, 2006, pp. 91–101.

[21] Y. Zhang and L. E. Parker, "Considering Inter-task Resource Constraints in Task Allocation," *Autonomous Agents and Multi-Agent Systems*, vol. 26, no. 3, pp. 389–419, 2013.

[22] H. L. Choi, L. Brunet, and J. P. How, "Consensus-based Decentralized Auctions for Robust Task Allocation," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 912–926, 2009.

[23] P. Segui-Gasco, H.-S. Shin, A. Tsourdos, and V. J. Segui, "Decentralised Submodular Multi-Robot Task Allocation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hamburg, Germany, 2015, pp. 2829–2834.

[24] B. Acikmese and D. S. Bayard, "Markov Chain Approach to Probabilistic Guidance for Swarms of Autonomous Agents," *Asian Journal of Control*, vol. 17, no. 4, pp. 1105–1124, 2015.

[25] I. Chattopadhyay and A. Ray, "Supervised Self-Organization of Homogeneous Swarms Using Ergodic Projections of Markov Chains," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1505–1515, 2009.

[26] N. Demir and B. Acikmese, "Probabilistic Density Control for Swarm of Decentralized ON-OFF Agents with Safety Constraints," in *American Control Conference*, Chicago, IL, USA, 2015, pp. 5238–5244.

[27] S. Berman, A. Halasz, M. A. Hsieh, and V. Kumar, "Optimized Stochastic Policies for Task Allocation in Swarms of Robots," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 927–937, 2009.

[28] A. Halasz, M. A. Hsieh, S. Berman, and V. Kumar, "Dynamic Redistribution of a Swarm of Robots among Multiple Sites," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, USA, 2007, pp. 2320–2325.

[29] M. A. Hsieh, A. Halasz, S. Berman, and V. Kumar, "Biologically Inspired Redistribution of a Swarm of Robots among Multiple Sites," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 121–141, 2008.

[30] T. W. Mather and M. A. Hsieh, "Macroscopic Modeling of Stochastic Deployment Policies with Time Delays for Robot Ensembles," *The International Journal of Robotics Research*, vol. 30, no. 5, pp. 590–600, 2011.

[31] A. Prorok, M. A. Hsieh, and V. Kumar, "The Impact of Diversity on Optimal Control Policies for Heterogeneous Robot Swarms," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 346–358, 2017.

[32] T. H. Labella, M. Dorigo, and J.-L. Deneubourg, "Division of Labor in a Group of Robots Inspired by Ants' Foraging Behavior," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 1, no. 1, pp. 4–25, 2006.

[33] E. Castello, T. Yamamoto, Y. Nakamura, and H. Ishiguro, "Foraging Optimization in Swarm Robotic Systems Based on an Adaptive Response Threshold Model," *Advanced Robotics*, vol. 28, no. 20, pp. 1343–1356, 2014.

[34] H. Kurdi, J. How, and G. Bautista, "Bio-Inspired Algorithm for Task Allocation in Multi-UAV Search and Rescue Missions," in *AIAA Guidance, Navigation, and Control Conference*, San Diego, CA, USA, 2016.

[35] W. Liu, A. F. T. Winfield, J. Sa, J. Chen, and L. Dou, "Towards Energy Optimization: Emergent Task Allocation in a Swarm of Foraging Robots," *Adaptive Behavior*, vol. 15, no. 3, pp. 289–305, 2007.

[36] W. Liu and A. F. T. Winfield, "Modeling and Optimization of Adaptive Foraging in Swarm Robotic Systems," *The International Journal of Robotics Research*, vol. 29, no. 14, pp. 1743–1760, 2010.

[37] A. Martinoli, K. Easton, and W. Agassounon, "Modeling Swarm Robotic Systems: a Case Study in Collaborative Distributed Manipulation," *The International Journal of Robotics Research*, vol. 23, no. 4, pp. 415–436, 2004.

[38] K. Lerman, A. Martinoli, and A. Galstyan, "A Review of Probabilistic Macroscopic Models for Swarm Robotic Systems," in *Swarm Robotics*. Berlin: Springer, 2005, pp. 143–152.

[39] N. Correll and A. Martinoli, "System Identification of Self-Organizing Robotic Swarms," in *Distributed Autonomous Robotic Systems 7*. Tokyo: Springer Japan, 2006, pp. 31–40.

[40] A. Prorok, N. Correll, and A. Martinoli, "Multi-level Spatial Modeling for Stochastic Distributed Robotic Systems," *The International Journal of Robotics Research*, vol. 30, no. 5, pp. 574–589, 2011.

[41] A. Kanakia, J. Klingner, and N. Correll, "A Response Threshold Sigmoid Function Model for Swarm Robot Collaboration," *Distributed Autonomous Robotic Systems*, pp. 193–206, 2016.

[42] D. Dimitrov and S. C. Sung, "Top responstiveness and Nash stability in coalition formation games," *Kybernetika*, vol. 42, no. 4, pp. 453–460, 2006.

[43] A. Darmann, E. Elkind, S. Kurz, J. Lang, J. Schauer, and G. Woeginger, "Group Activity Selection Problem," in *Proceedings of the 8th International Conference on Internet and Network Economics*, Liverpool, UK, 2012, pp. 156–169.

[44] A. Darmann, "Group Activity Selection from Ordinal Preferences," in *Algorithmic Decision Theory. ADT 2015. Lecture Notes in Computer Science*, ser. Lecture Notes in Computer Science, T. Walsh, Ed. Berlin, Heidelberg: Springer Cham, 2015, vol. 9346, pp. 35–51.

[45] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable Self-assembly in a Thousand-robot Swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[46] S. C. Sung and D. Dimitrov, "On Myopic Stability Concepts for Hedonic Games," *Theory and Decision*, vol. 62, no. 1, pp. 31–45, 2007.

[47] M. Karakaya, "Hedonic Coalition Formation Games: A New Stability Notion," *Mathematical Social Sciences*, vol. 61, no. 3, pp. 157–165, 2011.

[48] H. Aziz and F. Brandl, "Existence of Stability in Hedonic Coalition Formation Games," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, Valencia, Spain, 2012, pp. 763–770.

[49] J. Guerrero and G. Oliver, "Multi-robot Coalition Formation in Real-time Scenarios," *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1295–1307, 2012.

[50] O. Shehory and S. Kraus, "Feasible Formation of Coalitions Among Autonomous Agents in Nonsuperadditive Environments," *Computational Intelligence*, vol. 15, no. 3, pp. 218–251, 1999.

[51] C. Nam and D. A. Shell, "Assignment Algorithms for Modeling Resource Contention in Multirobot Task Allocation," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 889–900, 2015.

[52] L. B. Johnson, H.-L. Choi, and J. P. How, "The Role of Information Assumptions in Decentralized Task Allocation: A Tutorial," *IEEE Control Systems*, vol. 36, no. 4, pp. 45–58, 2016.

[53] W. Saad, Z. Han, M. Debbah, and A. Hjørungnes, "A Distributed Coalition Formation Framework for Fair User Cooperation in Wireless Networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 9, pp. 4580–4593, 2009.

[54] W. Saad, Z. Han, T. Basar, M. Debbah, and A. Hjørungnes, "Hedonic Coalition Formation for Distributed Task Allocation among Wireless Agents," *IEEE Transactions on Mobile Computing*, vol. 10, no. 9, pp. 1327–1344, 2011.

Inmo Jang received his BSc and MSc on mechanical and aerospace engineering from Seoul National University, South Korea, in 2008 and 2010, respectively. He worked in Korea Aerospace Industries Ltd., from 2010 to 2014 and in Korea Institute of Aviation Safety Technology from 2014 to 2015, respectively. Since 2015, he has been working towards a PhD on multi-robot system coordination in Cranfield University, UK.

Hyo-Sang Shin received his BSc from Pusan National University in 2004 and gained an MSc on flight dynamics, guidance and control in Aerospace Engineering from KAIST and a PhD on cooperative missile guidance from Cranfield University in 2006 and 2010, respectively. He is currently Reader on Guidance, Control and Navigation Systems in Autonomous and Intelligent Systems Group at Cranfield University. His current research interests include cooperative control of multiple agent systems, multiple target tracking and information-driven sensing.

Antonios Tsourdos obtained a MEng in electronic, control and systems engineering from the University of Sheffield (1995), an MSc in systems engineering from Cardiff University (1996), and a PhD in nonlinear robust missile autopilot design and analysis from Cranfield University (1999). He is a Professor of Control Engineering with Cranfield University, and was appointed Head of the Centre for Cyber-Physical Systems in 2013. He was a member of the Team Stellar, the winning team for the UK MoD Grand Challenge (2008) and the IET Innovation Award (Category Team, 2009).