2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-7281-6212-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/IROS45743.2020.9341652

Distributed Near-optimal Multi-robots Coordination in Heterogeneous Task Allocation

Qinyuan Li[1] , Minyi Li [2] , Bao Quoc Vo[1], and Ryszard Kowalczyk[1,3]

Abstract— **This paper explores the heterogeneous task allocation problem in Multi-robot systems. A game-theoretic formulation of the problem is proposed to align the goal of individual robots with the system objective. The concept of Nash equilibrium is applied to define a desired solution for the task allocation problem in which each robot can allocate itself to an appropriate task group. We also introduce a market-based distributed mechanism, called `DisNE`, to allow the robots to exchange messages with tasks and move between task groups, eventually reaching an equilibrium solution. We carry out comprehensive empirical studies to demonstrate that `DisNE` achieves near-optimal system utility in significantly shorter computation times when compared with the state-of-the-art mechanisms.**

I. Introduction

Recent advances in AI have enabled autonomous robots to replace humans when conducting challenging tasks in hazardous circumstances. For some complex tasks, a single robot may be limited by its own capacity with multiple robots working together leading to better outcomes. For example, in target tracking, the more sensors working together cooperatively to collect information from different directions, the more accurately a target can be pinpointed. Situations like this, where a group of robots work cooperatively in order to perform a collective behavior are referred to as Multi-robot systems (MRS). One of the most important aspects in multi-robot systems is task allocation [1]–[3].

According to Gerkey [4], multi-robot task allocation problems can be classified into: 1) Single-task robots (ST): each robot can perform at most one task at a time, 2) multi-task robots (MT): some robots are capable of performing multiple tasks simultaneously, 3) Single-robot tasks (SR): each task is required to be completed by exactly one robot, and 4) multi-robot tasks (MR): some tasks require multiple robots to complete. In this paper, we focus on the ST-MR (single-task robots and multi-robot tasks) problems. Many real word applications can be mapped into such a category. For instance, in large-scale disaster rescue operations, each rescue robot can only assist one survivor at a time and each survivor often requires to be rescued cooperatively by multiple robots with different functions (e.g., police or firefighters free the survivor while the paramedics attend to the injuries).

For large-scale ST-MR allocation, the number of tasks and robots can be very large and it is impractical to use a centralised task allocation solution with complex communication requirements. This imposes significant challenges in terms of scalablilty and robustness for the design of task allocation mechanisms. Firstly, the computational complexity of finding the optimal allocation solution increases exponentially with the number of tasks and robots [5]. Moreover, due to the communication bottleneck, the designed mechanisms have to be robust enough to tolerate communication failures and interruptions.

Related work: Depending on the robots decision-making framework, allocation mechanisms can be either centralised or decentralised. Centralised approaches often require a robot to have a global view of the whole system to make an allocation decision. For example, in [6], the reaction function was proposed to characterise the costs to robots. Robot cost is the smallest sum of travel and wait times needed for the robot to visit all targets assigned to it. A central planner robot was employed to allocate tasks to robots based on the robots reaction functions. The design of these centralised systems are usually simple. However, they typically have high communication requirements as all robots are linked to a central planner. Furthermore, they also intrinsically suffer from a single point of failure (i.e., lack of robustness).

On the other hand, in decentralised approaches robots make their own individual decision under some designed coordination mechanism. The designed mechanisms must be scalable, robust and require low communication demands in order to work in large-scale scenarios. There is a large body of research on distributed multi-robot coordination [7]–[10]. Rajiv [11] formalised distributed robot coordination into a constraint optimisation problem (DCOP) and studied the performance of two local search mechanisms, DSA (Distributed Stochastic) and MGM (Maximum Gain Message). In particular, MGM requires full communication between robots to guarantee monotonicity. While, in DSA all robot movements are subject to some pre-defined probability, and thus, cannot avoid conflicts between robots that lead to undesired outcomes. Antidio [12] introduced the concept of service for the ST-MR problem. A distributed algorithm was proposed. The basic idea being that a robot can ask for services from other robots when it is unable to execute a task. However, this algorithm also relies on full communication between robots. The state-of-the-art mechanisms for sovling the ST-MR problem are Max-Sum and its variations. This class of mechanisms are based on factor graphs [13]–[15], where both robots and tasks are considered as nodes, and

[1] Faculty of Science, Engineering and Technology, Swinburne University of Technology, Melbourne, Australia

[2] School of Science, RMIT University, Melbourne, Australia

[3] Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland

communication between robots and tasks occur through the edges/connections between them. Robots make decisions on which task to work on based on local information sent from the tasks. Nevertheless, most existing work still leaves the robots with the problem of searching a combinatorial search space for an optimal strategy.

Contributions: To address these challenges, we propose a novel market-based coordination mechanism to tackle large-scale ST-MR task allocation problems. The main contribution of this paper is as follows:

1) inspired by game theory [16], [17], we formulate the ST-MR problem into a task allocation game. We define the strategies and payoffs to the robots such that the goal of an individual robot is coincident with system optimisation, and thus, every robot aims to make the maximum marginal contribution to system performance.
2) through the above formulation, we can apply the concept of Nash equilibrium to describe a desired solution characteristic, where no single robot can make a better marginal contribution by joining another task group, i.e., performing another task instead.
3) We further introduce a flexible and efficient market-based mechanism `DisNE` to support interactions among robots and tasks, guiding them to make movements that eventually converge in a Nash equilibrium. `DisNE` possesses anytime and monotonically This guarantees `DisNE` can return a valid and current best solution even if it is interrupted before its termination.

II. Problem description

Consider a population of m robots allocated among n tasks. $R = \{r_1, \cdots, r_m\}$ represents the set robots and $T = \{t_1, \cdots, t_n\}$ represents the set of tasks. $\mathcal{G} = \{G_1, \cdots, G_n\}$ is an allocation solution where each $G_j \in \mathcal{G}$ is the group of robots that is attached to task t_j according to $\mathcal{G}$ ($G_j \subseteq R$). Note that, the groups in a valid allocation $\mathcal{G}$ are disjoint, meaning, $\forall G_j, G_h \in \mathcal{G}$ **and** $j \neq h, G_j \cap G_h = \emptyset$. Many real word multi-robot problems have spatial and temporal constraints, and thus, some tasks might not be feasible for some robots. We model these as constraints, and for each $r_i \in R$, T_{r_i} ($T_{r_i} \subseteq T$) is the set of tasks that r_i can perform. Similarly, for each $t_j \in T$, R_{t_j} is the set of robots who can perform t_j.

For a task t_j and the associated group G_j, $\mathtt{U}_j(G_j) \in \mathbb{R}$ is the task utility of t_j achieved by the corresponding task group G_j. Task utilities are assumed to be independent, so system utility Φ is the sum of individual task utilities. In task allocation, the goal is to find an allocation solution that maximise the system utility.

$$\max_{\mathcal{G} \in \mathcal{O}} \Phi(\mathcal{G}) = \max_{\mathcal{G} \in \mathcal{O}} \sum_{G_j \in \mathcal{G}} \mathtt{U}_j(G_j) \tag{1}$$

where $\mathcal{O}$ denotes the set of all possible allocation solutions.

III. Game theoretic formulation and Equilibrium Solution

Consider a ST-MR allocation problem with n tasks and m robots, the allocation objective is to maximise system utility. We formulate the ST-MR allocation problem into a task allocation game. We can then apply the concept of Nash equilibrium in the allocation game to model the desired characteristic of a solution.

Strategy

Each robot r_i has $|T_{r_i}|+1$ strategies, strategy 0 to strategy $|T_{r_i}|$ ($| \star |$ denotes the cardinality of the set). Strategy 0 indicates the robot does not perform any task, (i.e., joins dummy group G_0) and strategy $j > 0$ means the robot performs task t_j (i.e., joins G_j).

For convenience, we also extend the representation of an allocation solution to $\mathcal{G} = \{G_0, G_1, \ldots, G_n\}$, with G_0 being the dummy task group.

Payoff

The payoff to a robot r_i playing strategy j (e.g., joining G_j), is its marginal contribution to task t_j, denoted by $\sigma_j(r_i, G_j)$, i.e., the difference in task utility when robot r_i is and is not in G_j.

$$\sigma_j(r_i, G_j) = \mathtt{U}_j(G_j) - \mathtt{U}_j(G_j \backslash \{r_i\}) \tag{2}$$

Note that, $\forall r_i \in R$, if $j = 0$ (i.e., r_i joins dummy group G_0), $\sigma_0(r_i, G_0) = 0$.

In the game formulation, each robot aims to select the task/strategy that gives it the best payoff. A robot will change strategy (i.e., move to another task group) at anytime, if doing so provides it a better payoff, until, the allocation reaches an equilibrium:

Definition 1 (Nash equilibrium): An allocation solution $\mathcal{G}$ is a Nash equilibrium, if and only if, no robot can gain a better payoff (i.e., make a higher marginal contribution) than that in $\mathcal{G}$, when other robots play their current strategy (i.e., stay in their current task groups).

When a robot changes its strategy by moving from one task group to another, its marginal contribution could be different. Prompted by this, movement value can be conceptualised to measure the gain/loss a robot achieves from its strategy change. By an abuse of notation, let $\mathcal{G}(i)$ be the index j of the task group that contains robot r_i according to the allocation solution $\mathcal{G}$. That is, if $\mathcal{G}(i) = j$ then $r_i \in G_j$.

Definition 2 (Movement value): The movement value, written as $mv^i_{j\to h}$, for a robot r_i moving from its current task t_j to another task t_h (i.e., leaving G_j to join G_h), forming $G'_h = G_h \cup \{r_i\}$, is the difference between its marginal contribution to task t_h (in G'_h) and to t_j (in G_j):

$$mv^i_{j\to h} = \sigma_h(r_i, G'_h) - \sigma_j(r_i, G_j), \quad G'_h = G_h \cup \{r_i\} \tag{3}$$

Obviously, when $j = h$, $mv^i_{j\to h} = 0$.

Remark 1: Due to the movement of r_i, the structure of the task groups for t_j and t_h (i.e., G_j and G_h) is changed. Therefore, changes in the marginal contributions of the other member robots in G_j and G_h are also triggered. Although, the marginal contribution of robot r_i to the dummy task is always zero, regardless of its group structure.

Remark 2: The movement value $mv^i_{j\to h}$, for robot r_i moving from task group G_j to G_h, is the gain in payoff r_i can obtain.

Proposition 1: Given an allocation solution $\mathcal{G}$, the movement value $mv^i_{j\to h}$ for robot r_i leaving its current task group G_j $(j = \mathcal{G}(i))$ to join another task group G_h, is the change in overall system utility, caused by the change of utilities in tasks t_j and t_h.

Proof: Due to such a movement, allocation solution $\mathcal{G}'$ with $G'_j = G_j \backslash \{r_i\}$ and $G'_h = G_h \cup \{r_i\}$ is formed.

$$\begin{aligned}\Phi(\mathcal{G}') - \Phi(\mathcal{G}) &= \big(\mathtt{U}_h(G'_h) + \mathtt{U}_j(G'_j)\big) - \big(\mathtt{U}_h(G_h) + \mathtt{U}_j(G_j)\big)\\ &= \big(\mathtt{U}_h(G'_h) - \mathtt{U}_h(G_h)\big) - \big(\mathtt{U}_j(G_j) - \mathtt{U}_j(G'_j)\big)\\ &= \sigma_h(r_i, G'_h) - \sigma_j(r_i, G_j) = mv^i_{j\to h}\end{aligned}$$

■

Corollary 1: An allocation solution $\mathcal{G}$ is a Nash equilibrium, if and only if, $\forall r_i \in R$, and $\forall t_h \in T_{r_i}$, $mv^i_{j\to h} \leq 0$, where $j = \mathcal{G}(i)$.

Remark 3: Given a set of tasks and robots, there might exists more than one Nash equilibrium allocation solutions. A Nash equilibrium is not necessarily the optimal solution. Nonetheless, as demonstrated through a comprehensive set of experiments, the quality of the a Nash equilibrium solution achieved through a greedy strategy is as good as the state-of-the-art algorithms.

IV. The proposed DisNE mechanism

In this section, the DisNE mechanism is introduced. DisNE is a market-based multiple-round mechanism. Each task is governed by a manager who is in charge of communicating with the robots. Note that, for simplicity, robots do not communicate with the dummy task. In each round, each robot can propose to leave its current task group and join another based on the potential improvement to its current payoff (i.e., the movement value). Such that robots allocate/reallocate themselves to tasks and system utility monotonically increases towards a Nash equilibrium. The flow diagram of each round is shown in Fig 1.

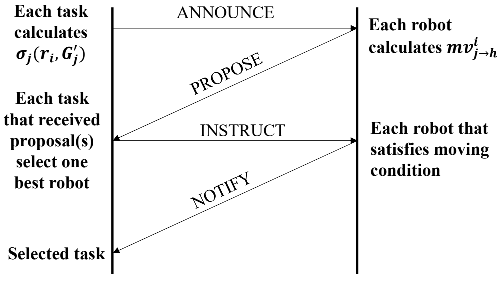

Each task calculates $\sigma_j(r_i, G'_j)$
ANNOUNCE
Each robot calculates $mv^i_{j\to h}$
PROPOSE
Each task that received proposal(s) select one best robot
INSTRUCT
Each robot that satisfies moving condition
NOTIFY
Selected task

Fig. 1: The flow diagram of each round

A. The DisNE mechanism

In each round of the DisNE mechanism, there are four steps involved: ANNOUNCE, PROPOSE, INSTRUCT, and NOTIFY.

- ANNOUNCE. In this step, each task t_j calculates the marginal contribution that every robot $r_i \in R_{t_j}$ provides to its group based on Equation 2. Then, t_j announces to each of the robots in R_{t_j} the corresponding marginal contribution. In the first round, every task completes this process, from that point on, only the tasks that have changed group structures complete this process. (lines 1–4, Mechanism 1: DisNE).
- PROPOSE. Upon receipt of announcement from the tasks, each robot $r_i \in R$ re-calculates its movement values for tasks in T_{r_i} according to Equation 3 (lines 5–7, Mechanism 1: DisNE). Then, r_i explores possible improvement to its payoff. If r_i has a positive maximum movement value (i.e., improvement can be made to its payoff), it will make *proposal(s)*. Let pv_i be the proposal value of robot r_i, which is the maximum movement value of r_i, and let target-tasks be the tasks to which r_i has the maximum movement value. r_i then proposes pv_i to the target-tasks, as well as its current task t_j $(j = \mathcal{G}(i))$ if $j \neq 0$ (lines 8–12, Mechanism 1: DisNE).
- INSTRUCT. Each task t_j that received *proposal(s)* selects the robot who proposed the maximum proposal value $\max pv_i$[1]. Then, t_j sends the *accept* instruction to the selected robot and the *reject* instruction to the others (lines 13, 14, Mechanism 1: DisNE).
- NOTIFY. Robots decide whether or not to move based on the following conditions:

Mechanism 1: DisNE.

```
Step: ANNOUNCE Initialised by each task $t_j \in T$ when
      round=1; or each task $t_j$ that has changed group structure
      when round $\geq 2$.
 1: for $r_i \in R_{t_j}$ do
 2:     Computes $\sigma_j(r_i, G'_j)$, $G'_j = G_j \cup \{r_i\}$
 3:     Announces $\sigma_j(r_i, G'_j)$ to $r_i$
 4: end for
Step: PROPOSE Initialised by each robot $r_i \in R$.
 5: for $t_h \in T_{r_i}$ do
 6:     Calculates $mv^i_{j\to h}$, $j = \mathcal{G}(i)$
 7: end for
 8: if $\max_{t_h \in T_{r_i}} mv^i_{j\to h} > 0$ then
 9:     $pv_i \leftarrow \max_{t_h \in T_{r_i}} mv^i_{j\to h}$
10:     target-tasks $\leftarrow \arg\max_{t_h \in T_{r_i}} mv^i_{j\to h}$
11:     Proposes to target-tasks, and $t_j$ if $j \neq 0$, with $pv_i$.
12: end if
Step: INSTRUCT Initialised by each task $t_j$ that has re-
      ceived proposal(s).
13: Selects the robot with the $\max pv_i$.
14: Sends accept to the selected robot and reject(s) to the
    others.
Step: NOTIFY Initialised by each robot $r_i$ that satisfies
      Definition 3.
15: Selects one accepted-target-task.
16: Sends confirm to the selected task, and $t_j$ if $j \neq 0$.
17: Moves to its selected target-task.
```

[1] If the task receives the same maximum proposal value from more than one robots, it randomly selects one.

Definition 3 (Moving conditions): i) robot r_i has not been allocated, $\mathcal{G}(i) = 0$ and it receives *accept(s)* from its target-task(s). ii) robot r_i has a current task t_j, $j = \mathcal{G}(i)$ and $j \neq 0$ and it receives *accepts* from both its target-task(s) and its current task t_j.

Each robot r_i that meets condition i): sends *confirm* to an accepted-target-task. Each robot r_i that meets condition ii): sends *confirm* to both an accepted-target-task and its current task t_j. r_i then moves to its selected target-task[2] (lines 15-17, Mechanism 1: `DisNE`).

After Step NOTIFY, `DisNE` then returns to the ANNOUNCE step. The process continues, until no robots make *proposal(s)*. This means that no robot has incentive to deviate from its current task group, meaning that a Nash Equilibrium has been reached.

Remark 4: `DisNE` can work with any initial group configuration (i.e., as a mechanism to improve an existing allocation). If the initial allocation is an equilibrium, after the step ANNOUNCE, no robot will propose and the mechanism stops at the same equilibrium.

B. Illustration

Capability Task	f_1	f_2	f_3	f_4	f_5
t_1	✓	✓	✓	✗	✗
t_2	✓	✗	✗	✓	✓

(a) Capabilities required by tasks

Capability robot	f_1	f_2	f_3	f_4	f_5
r_1	0	0	0	7	6
r_2	4	3	8	8	4
r_3	0	9	8	0	0
r_4	4	9	0	7	0

(b) Competency values of robots

TABLE I: An example ST-MR problem

Consider a simply example for demonstration with task requirements and robot capabilities shown in Tab Ia and Tab Ib. Each f_l represents a capability to be requested or provided. "✓" in Tab Ia indicates the task requests the capability, "✗" means it does not. Each number in Tab Ib can be viewed as the competency value e_i^l of the provided capability f_l of robot r_i. Denote F_{t_j} as the set of capabilities t_j requests. A simple task utility function[3] is given by the sum of the maximum competency values of the group members:

$$\mathtt{U}_j(G_j) = \sum_{f_l \in F_{t_j}} \max_{r_i \in G_j} e_i^l \tag{4}$$

The entire process of the proposed `DisNE` mechanism is illustrated in Fig 2. In the beginning all robots are unallocated. Each table demonstrates the received marginal contributions of robots ("MC.") and the movement values of robots ("MV."). Under the table, the arrow between a robot and a task represents the messages exchanged. The number next to an arrow indicates the proposal value of a robot and *accept* or *reject* is the instruction the task gives to the robot. The red arrow indicates the robot notifies its movement to the task.

- In the first round: all the robots are in G_0, so the maximum movement value of all robots is positive. r_1 has its maximum movement value $mv^1_{0\to 2} = 13$, so it proposes to t_2 with proposal value $pv_1 = mv^1_{0\to 2} = 13$. Similarly, r_2 proposes to task t_2 with $pv_2 = 16$, r_3 proposes to t_1 with $pv_3 = 17$, and r_4 proposes to task t_1 with $pv_4 = 13$. For tasks, t_1 receives two proposal values $pv_3 = 17$ and $pv_4 = 13$, and 17 is the maximum proposed by r_3. t_1 then sends *accept* to r_3 and sends *reject* to r_4. t_2 receives proposal values $pv_1 = 13$ and $pv_2 = 16$, it then sends *reject* to r_1 and *accept* to r_2. Therefore, robot r_2 (respectively, r_3) receive *accept* from t_2 (respectively, t_1). Then, r_2 notifies its movement to t_2 and moves to G_2, r_3 sends *confirm* to t_1 then moves to G_1.
- In the second round: Both tasks t_1 and t_2 have their group structures changed, so t_1 and t_2 re-calculate and announce the marginal contributions of the corresponding robots. In this round, only r_1 and r_4 have a positive maximum movement value. r_1 proposes to t_2 with proposal value $pv_1 = 2$ and r_4 proposes to t_1 with $pv_4 = 4$. Both t_1 and t_2 receive one proposal, so they all send *accept*. Then, r_1 notifies t_2 and moves to group G_2. r_4 notifies t_1 and moves to group G_1.
- In the third round: t_1 and t_2 again re-compute and announce the marginal contributions. After each robot recalculates its movement values, no robots make further proposal(s) as none of them have a positive movement value. This indicates a Nash equilibrium solution has been reached with $G_0 = \emptyset, G_1 = \{r_3, r_4\}, G_2 = \{r_1, r_2\}$.

C. Property

Theorem 1: Based on `DisNE`, the formed allocation solution $\mathcal{G}$ is a Nash equilibrium.

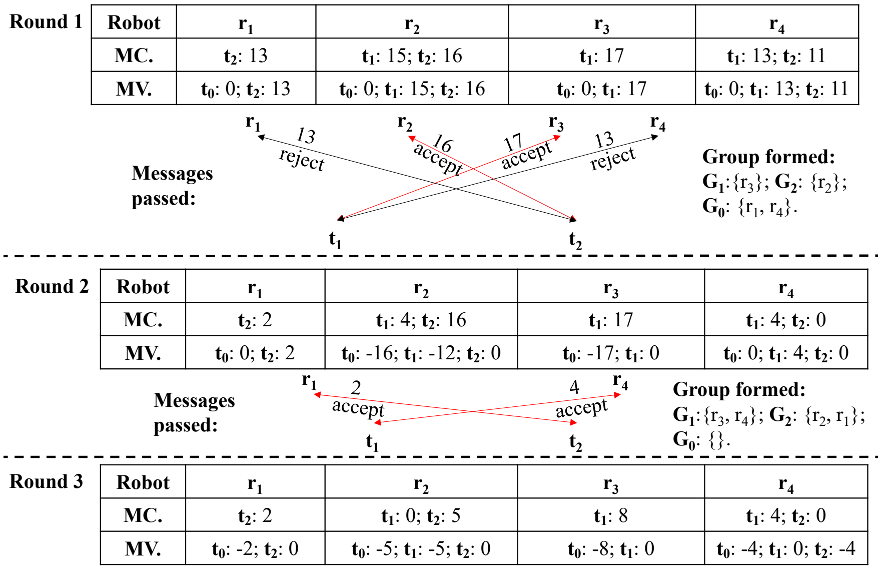
Proof: `DisNE` naturally terminates when no robots propose. In such a case, $\forall r_i \in R$ and $\forall t_h \in T_{r_i}$, $mv^i_{\mathcal{G}(i)\to h} \leq 0$. Following Corollary 1, the formed solution $\mathcal{G}$ is a Nash equilibrium. ■

Theorem 2: The proposed `DisNE` mechanism guarantees anytime monotonicity.

Proof: In each round, consider a group-pair $(G_j - G_h)_i$ consists of task group G_j that has robot r_i leaving, and task group G_h accepts r_i joining. Since each task group allows either the joining or leaving movement of one robot, a group-pair is linked with a single robot that has a positive movement value. For the same reason, group-pairs are disjoint or intersect on dummy task group G_0. For each group-pair that is disjoint with any other group-pairs, ($j \neq h \neq 0$), system utility changed from the movement of robot r_i, is not related to any other task utility change apart from t_j and t_h. For

[2] If the robot receives more than one *accepts* from its target-tasks, it randomly selects one.

[3] Note that `DisNE` does not limited to the use of this task utility function, it can handle various types of task utility functions. For example, in rescue scenario task utility function can be defined according to time budget.



Round 1

Robot	$\mathbf{r_1}$	$\mathbf{r_2}$	$\mathbf{r_3}$	$\mathbf{r_4}$
MC.	$\mathbf{t_2}$: 13	$\mathbf{t_1}$: 15; $\mathbf{t_2}$: 16	$\mathbf{t_1}$: 17	$\mathbf{t_1}$: 13; $\mathbf{t_2}$: 11
MV.	$\mathbf{t_0}$: 0; $\mathbf{t_2}$: 13	$\mathbf{t_0}$: 0; $\mathbf{t_1}$: 15; $\mathbf{t_2}$: 16	$\mathbf{t_0}$: 0; $\mathbf{t_1}$: 17	$\mathbf{t_0}$: 0; $\mathbf{t_1}$: 13; $\mathbf{t_2}$: 11

Messages passed:

Group formed: $\mathbf{G_1}$:{r_3}; $\mathbf{G_2}$: {r_2}; $\mathbf{G_0}$: {r_1, r_4}.

Round 2

Robot	$\mathbf{r_1}$	$\mathbf{r_2}$	$\mathbf{r_3}$	$\mathbf{r_4}$
MC.	$\mathbf{t_2}$: 2	$\mathbf{t_1}$: 4; $\mathbf{t_2}$: 16	$\mathbf{t_1}$: 17	$\mathbf{t_1}$: 4; $\mathbf{t_2}$: 0
MV.	$\mathbf{t_0}$: 0; $\mathbf{t_2}$: 2	$\mathbf{t_0}$: -16; $\mathbf{t_1}$: -12; $\mathbf{t_2}$: 0	$\mathbf{t_0}$: -17; $\mathbf{t_1}$: 0	$\mathbf{t_0}$: 0; $\mathbf{t_1}$: 4; $\mathbf{t_2}$: 0

Messages passed:

Group formed: $\mathbf{G_1}$:{r_3, r_4}; $\mathbf{G_2}$: {r_2, r_1}; $\mathbf{G_0}$: {}.

Round 3

Robot	$\mathbf{r_1}$	$\mathbf{r_2}$	$\mathbf{r_3}$	$\mathbf{r_4}$
MC.	$\mathbf{t_2}$: 2	$\mathbf{t_1}$: 0; $\mathbf{t_2}$: 5	$\mathbf{t_1}$: 8	$\mathbf{t_1}$: 4; $\mathbf{t_2}$: 0
MV.	$\mathbf{t_0}$: -2; $\mathbf{t_2}$: 0	$\mathbf{t_0}$: -5; $\mathbf{t_1}$: -5; $\mathbf{t_2}$: 0	$\mathbf{t_0}$: -8; $\mathbf{t_1}$: 0	$\mathbf{t_0}$: -4; $\mathbf{t_1}$: 0; $\mathbf{t_2}$: -4

No robots make proposal(s), an equilibrium solution has been reached.

Fig. 2: Illustration of the allocation process

each group-pair that intersects with other group-pairs on G_0, ($j = 0$ or $h = 0$), changed system utility from the movement of robot r_i is only related to the non-zero task. In both cases, according to Proposition 1, the movement of each of the robots independently increases system utility by $mv^i_{j\to h} > 0$. Such that DisNE is able to guarantee monotonicity. ■

Theorem 3: The number of messages passing in DisNE is polynomial.

Proof: For m robots and n tasks, assuming DisNE runs for K rounds, in the worst case, there are $m \cdot n$ marginal contributions announced by tasks in each round. Each robot proposes to n tasks, so the number of proposals is $m \cdot n$. Thus, the instructions sent by tasks are $m \cdot n$. Each robot notifies to at most two tasks and the confirm messages are $2 \cdot m$. Therefore, the worst case message passing of DisNE is $O(m^4 \cdot n^3 \cdot K)$. ■

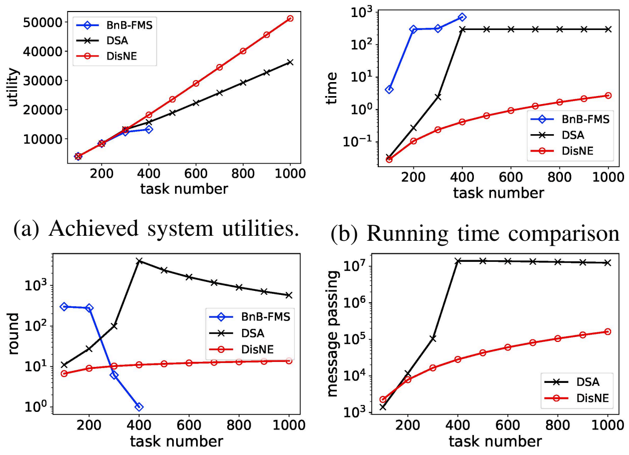
Indeed, the actual message passing in DisNE is far less than the worst case. Only the tasks that have changed group structures need to re-calculate and announce new marginal contribution values. Moreover, only the robots with positive maximum movement values will participate in the Steps after ANNOUNCE.

V. Experiments and Evaluation

This section presents the experiment results on the performance of the proposed DisNE mechanism, in comparison with the Distributed Stochastic Algorithm (DSA), as well as the Branch-and-Bound Fast-Max-Sum (BnB-FMS).

A. Parameter setting

In the experiment, both task requirements and robots' enclosed skills are randomly generated from 10 possible capabilities. The competency values e^l_i are random numbers in the range $[0, 10]$. The function defined in Equation 4 is used to calculate task utilities. For BnB-FMS, as in [15], we set the number of rounds at $|A| + |T|$. Regarding DSA, the predefined control parameter[4] p is set at 0.7. The experiment investigates the performance of different mechanisms on a different number of tasks with $|T| \in \{100, 200, \cdots, 1000\}$. The number of robots is double the number of tasks $|R| = 2 \cdot |T|$. Maximum density[5] d is set at 4% of $|T|$, e.g., with 100 tasks, the maximum number of edges of a task or a robot is 4. For each number of tasks, we randomly generate 1000 experiment instances. We set a time bound for each experiment instance to 300 seconds, and all the mechanisms will terminate and return the current constructed solution upon reaching this time bound.



(a) Achieved system utilities.

(b) Running time comparison

(c) Execution round comparison

(d) Number of messages passed

Fig. 3: Results bounded by the time constraint.

[4] p indicates the degree of parallel execution. Readers are referred to [18] for detailed analysis on the performance of DSA with different p.

[5] d indicates the maximum number of tasks that a robot can choose and the maximum number of robots that can join a task group.

B. Experiment Results

With a small number of tasks ($|T| \leq 300$), the achieved system utilities show little difference among the different mechanisms (see Fig. 3a). However, when the task number is larger than 300, both the performance of DSA and BnB-FMS decrease. Moreover, since the search space of BnB-FMS increase exponentially with the density parameter d, it can only run up to 400 tasks and 800 robots ($d = 400 \times 4\% = 16$). When $|T| > 400$ (resulting in $d \geq 17$), it was unable to complete even one round of message passing within the time constraint, and thus, the results are not shown in Fig. 3.

Fig. 3b compares the running time of the mechanisms. We can observe that DisNE requires significantly shorter running time than that required by either BnB-FMS or DSA. As mentioned before, BnB-FMS is not able to complete even one round of computation and message passing within the time constraint when there are more than 400 tasks (see the average number of rounds for BnB-FMS in Fig. 3c) [6]. The running time of DSA quickly reaches the time bound (when $|T| = 400$) as the number of tasks increases. This is because when the density d increases, there is a higher probability that more robots will change their strategies together in a conflicting way, causing "thrashing". As a results, it typically requires a lot more rounds before an equilibrium is reached. This observation is consistent with the results shown in Fig. 3c. In contrast, DisNE experiences minimum growth in the number of rounds required. With 1000 tasks and 2000 robots, on average, DisNE obtains an equilibrium allocation solution within 3 seconds and 14 rounds.

Fig 3d further examines into the number of messages passing between agents and tasks in DSA and DisNE[7]. DSA is a general multi-robot cooperation mechanism in which message passing happens only at the beginning when tasks announce marginal contributions. Then, robots either stay unchanged or move to their preferred tasks. Even though in DisNE the amount of message-passing in a single round may be greater than that in DSA, in DisNE there are considerably less execution rounds than in DSA (see Fig. 3c). Therefore, the total amount of message-passing in DisNE is significantly less than that in DSA.

VI. Conclusions and Future Works

This paper focused on ST-MR in large-scale heterogeneous task allocation problems. A game-theoretic based formulation was proposed in which each robot is aiming to achieve the best strategy so that it makes the maximum marginal contribution to system utility. The Nash equilibrium concept was then applied to model a near-optimal solution in which no single robot can improve system performance by moving to another task group, given that all the other robots remain unchanged. An efficient distributed allocation mechanism, called DisNE was proposed, to achieve a Nash equilibrium solution. We ran comprehensive experiments and the results demonstrate the proposed mechanism significantly outperforms the state-of-art mechanisms. Future work aims to explore allocation problems where tasks are interdependent.

References

[1] 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.

[2] F. Faruq, D. Parker, B. Laccrda, and N. Hawes, "Simultaneous task allocation and planning under uncertainty," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3559–3564.

[3] V. Tereshchuk, J. Stewart, N. Bykov, S. Pedigo, S. Devasia, and A. G. Banerjee, "An efficient scheduling algorithm for multi-robot task allocation in assembling aircraft structures," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3844–3851, 2019.

[4] B. P. Gerkey and M. J. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *I. J. Robotics Res.*, vol. 23, pp. 939–954, 2004.

[5] J. A. Adams *et al.*, "Coalition formation for task allocation: Theory and algorithms," *Autonomous Agents and Multi-Agent Systems*, vol. 22, no. 2, pp. 225–248, 2011.

[6] X. Zheng and S. Koenig, "Reaction functions for task allocation to cooperative agents," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 559–566.

[7] M. A. Batalin and G. S. Sukhatme, "Sensor network-based multi-robot task allocation," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 2. IEEE, 2003, pp. 1939–1944.

[8] M. G. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak, and A. J. Kleywegt, "Simple auctions with performance guarantees for multi-robot task allocation," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 1. IEEE, 2004, pp. 698–705.

[9] L. Jin, S. Li, H. M. La, X. Zhang, and B. Hu, "Dynamic task allocation in multi-robot coordination for moving target tracking: A distributed approach," *Automatica*, vol. 100, pp. 75–81, 2019.

[10] D. K. Jha, "Algorithms for task allocation in homogeneous swarm of robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3771–3776.

[11] R. T. Maheswaran, J. P. Pearce, and M. Tambe, "A family of graphical-game-based algorithms for distributed constraint optimization problems," in *Coordination of large-scale multiagent systems*. Springer, 2006, pp. 127–146.

[12] A. Viguria, I. Maza, and A. Ollero, "S+ t: An algorithm for distributed multirobot task allocation based on services for improving robot cooperation," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 3163–3168.

[13] F. Fioretto, E. Pontelli, and W. Yeoh, "Distributed constraint optimization problems and applications: A survey," *Journal of Artificial Intelligence Research*, vol. 61, pp. 623–698, 2018.

[14] S. D. Ramchurn, A. Farinelli, K. S. Macarthur, and N. R. Jennings, "Decentralized coordination in robocup rescue," *The Computer Journal*, vol. 53, no. 9, pp. 1447–1461, 2010.

[15] K. S. Macarthur, R. Stranders, S. Ramchurn, and N. Jennings, "A distributed anytime algorithm for dynamic task allocation in multi-agent systems," in *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[16] A. Kanakia, B. Touri, and N. Correll, "Modeling multi-robot task allocation with limited information as global game," *Swarm Intelligence*, vol. 10, no. 2, pp. 147–160, 2016.

[17] K. Garapati, J. J. Roldán, M. Garzón, J. del Cerro, and A. Barrientos, "A game of drones: Game theoretic approaches for multi-robot task allocation in security missions," in *Iberian Robotics conference*. Springer, 2017, pp. 855–866.

[18] W. Zhang and Z. Xing, "Distributed breakout vs. distributed stochastic: A comparative evaluation on scan scheduling," in *AAMAS-02 Workshop on Distributed Constraint Reasoning*, 2002.

[6] We tested over 50 randomly generated instances of 500 tasks, BnB-FMS consistenly requires more than 700 seconds to complete a single round of computation and message passing in all instances.

[7] We only compare the amount of message passing in both DSA and DisNE. BnB-FMS follows a different communication mechanism and message structure, and thus, not comparable with the other two. Also, most of the computation in BnB-FMS lies on local optimisation computation rather than message passing.