DMTF: A Distributed Algorithm for Multi-team Formation

Amar Nath[a], Arun Ar and Rajdeep Niyogi[b]
IIT Roorkee, India 247667, India

Keywords: Multi-team Formation, Distributed Algorithm, Dynamic Environment, Multi-agent Coordination and Cooperation.

Abstract: Multi-team formation has received considerable attention in the last decade. Most existing approaches for team formation assume complete knowledge of the environment, and the problem is reduced to a combinatorial optimization problem. In a dynamic environmnt, the agents may be spatially distributed, have limited information, lack global knowledge, and update their knowledge by communicating with each other. In such a setting, it would be more appropriate for the agents to form teams by themselves in a distributed manner. In this paper, we suggest a distributed algorithm for multi-team formation (DMTF). The implementation of a distributed algorithm is quite challenging, and traditionally such algorithms are presented in a theoretical way. We implemented the proposed algorithm using a multi-robot simulator. The experimental results show the efficacy of the approach.

1 INTRODUCTION

Task execution in a dynamic environment may require cooperation among agents. In accomplishing missions like waste retrieval and disposal, demining, or objects manipulation in environments where direct human intervention is impossible or impractical, multi-agent systems are highly recommended. In a dynamic environment, a task may require multiple agents for its execution, the time and location of the task's arrival, the number of robots required to execute a task may not be known in advance. Moreover, a task execution may require all the members of a team to be present at the location of the task. In this paper, we consider the execution of such tasks in a dynamic environment formally defined in Section 3. Hence, the multi-team formation becomes essential in such situations to accomplish a mission. Henceforth, agents refer to physical agents, i.e., robots, and the terms agents and robots are used interchangeably.

To accomplish a mission, multiple teams are to be formed. This necessitates the design of a distributed algorithm (Lynch, 1996) that can handle multi-team formation at runtime. The robots need to communicate with each other to acquire relevant information to form a suitable team. Moreover, in such a dynamic environment, explicit coordination is preferred over implicit coordination as it provides a better and reliable way of multi-robot coordination over implicit communication (Yan et al., 2013). Implicit coordination (e.g., stigmergy behavior) is a way of coordination on a large scale of homogeneous robots, i.e., swarm robotics (Şahin, 2004).

Team formation has received a lot of attention in the MAS community. Existing approaches such as (Okimoto et al., 2016; Faliszewski et al., 2016; Lappas et al., 2009; Gutiérrez, 2016) reduce the team formation problem to combinatorial optimization problem, where the objective is to select k agents from a set of N agents, where each agent has some skills, such that a given cost criterion is optimized. These approaches (Okimoto et al., 2016; Faliszewski et al., 2016; Lappas et al., 2009) assume complete knowledge of the environment (e.g., the total number of agents in the environment, skill of an agent); current state and location of an agent are irrelevant in these approaches. Typically, in multi-agent settings, agents have limited information about an environment and they should coordinate among themselves (say, by exchanging messages) to accomplish a given task.

A novel distributed algorithm for multi-team formation (DMTF) is suggested in this paper (Section 4). The states of the robots are considered in designing the algorithm because, at a time, a robot can participate in one task (especially the multi-robot tasks, where all the robots of a team are required at task's location), and physically can be present at one loca-

[a] https://orcid.org/0000-0002-9290-4378
[b] https://orcid.org/0000/0003-1664-4882

Nath, A., Ar, A. and Niyogi, R.
DMTF: A Distributed Algorithm for Multi-team Formation.
DOI: 10.5220/0008914701520160
In *Proceedings of the 12th International Conference on Agents and Artificial Intelligence (ICAART 2020) - Volume 1*, pages 152-160
ISBN: 978-989-758-395-7; ISSN: 2184-433X

tion. A robot cannot perform multiple activities in one state at the same time. For example, a robot trying to form a team with a robot cannot express its willingness to be part of another team at the same time. Thus, states of the robots are considered, which is a distinguishing property of our algorithm. Traditionally distributed algorithms are mostly presented in a theoretical and logical way (Lynch, 1996), and the implementation of a distributed algorithm is quite challenging. We simulated the DMTF using ARGoS (*www.argos-sim.info*); a realistic multi-robot simulator (Pinciroli et al., 2012).

2 RELATED WORK

In (Okimoto et al., 2016), a centralized mission-oriented robust multi-team formation problem is suggested. A formal framework is defined, and two algorithms are provided to form robust multi-team. However, we are interested in forming multi-team in a dynamic environment in a distributed manner as agents lack global knowledge, and hence inter-robot communication is needed to form an optimal multi-teams for the tasks.

A distributed approach for coalition or team formation is given in (Shehory and Kraus, 1998; Abdallah and Lesser, 2004; Coviello and Franceschetti, 2012; Tosic and Agha, 2005). The work (Shehory and Kraus, 1998) suggest greedy distributed set partitioning and set covering algorithms for coalition formation that does not require considering the state and location of an agent. In our work the team formation algorithm is designed based on state of an agent. Type of messages are not given in (Shehory and Kraus, 1998), whereas different types of messages with specific meaning is considered in our work.

The work (Abdallah and Lesser, 2004) proposed an algorithm for coalition formation using an underlying hierarchical organizational structure. A manager tries to form a coalition for a given task by looking at the capabilities of its child nodes. If it is not successful, it calls a manager above her in the hierarchy, who then creates sub-tasks and tries to solve the sub-tasks. This process continues until no more managers can be given a sub-task.

In (Tosic and Agha, 2005), an algorithm suggested for group formation is based on the distributed computation of maximal cliques in the underlying network, where two agents are connected by an edge if they can communicate among themselves. A team formation strategy suggested in (Coviello and Franceschetti, 2012) is based on two kinds of agents: leader and follower. Agents only have local knowledge of the underlying network. A leader can communicate with only the neighbors in its visibility range by sending a request message after looking at some stability constraints. A follower upon receipt of a request message replies either accept or reject depending on whether it wants to be part of the team or not.

Our strategy of team/coalition formation differs from the above approaches (Abdallah and Lesser, 2004; Coviello and Franceschetti, 2012; Tosic and Agha, 2005) in the following manner. Our approach does not use any hierarchical structure among agents as in (Abdallah and Lesser, 2004). In (Abdallah and Lesser, 2004), the coalition formation process is delegated to other managers, whereas in our approach either the initiator succeeds or fails in the team formation process; delegation is not allowed. In our approach, an initiator can communicate with anyone of agents and is not as restrictive as in (Coviello and Franceschetti, 2012). In our approach, there are different types of messages whereas in (Coviello and Franceschetti, 2012) there are only request and accept/reject messages.

Multi-robot coalition formation has been studied by (Vig and Adams, 2006), where each robot has some capabilities and a coalition has to perform some task for which some capabilities are required. The authors consider the problem of coalition formation such that there is no coalition imbalance, which happens when a large part of resources is ascribed to any robot in a coalition. A balance coefficient is thus defined and it is used to form a suitable coalition for a given task. In our algorithm, all the agents in a coalition have required capabilities (skills) for task execution. Thus the notion of imbalance does not arise in our case. Moreover, our algorithm is fully distributed which is not the case in (Vig and Adams, 2006).

Auction-based approaches for team formation (task allocation) are suggested in (Gerkey and Mataric, 2002; Kong, 2015; Xie et al., 2018). A bidder agent has some resources (e.g., data center, CPU) (Kong, 2015), who may bid for multiple auctioneers concurrently. This does not have any influence on software agents. However, when we move to physical agents (eg, robots), a robot cannot be a member of multiple teams at any point of time simply because the tasks may be at different locations, and a robot cannot be at two different locations at the same time, even though a robot may have the capability to perform multiple tasks at a time.

In our work a non-initiator robot (the bidder) will not express its willingness to multiple initiators (auctioneers) concurrently; when more than one request message arrives, the robot stores the requests in its local queue. Having one or more resources specified in the auction is a sufficient condition for an agent to make a bid (Kong, 2015). Having the required skills for a task is a necessary but not a sufficient condition for a robot to express its willingness to be part of a team, in our work. A robot's behavior, in our work, is determined by its current state, whereas in (Gerkey and Mataric, 2002; Kong, 2015) states need not be taken into consideration.

A work (Dukeman and Adams, 2017) also considers a similar kind of problem, i.e., multi-robots tasks problem. However, the authors have used a centralized approach to solve the problem. In (Xie et al., 2018), a decentralized approach for object transport via implicit communication is presented. Our approach uses a distributed approach to form and assign a team to a task at run-time via explicit communication which is different from the approaches given in (Dukeman and Adams, 2017) and (Xie et al., 2018). In (Hayano et al., 2016), a task consists of a finite number of subtasks, and each subtask is assigned to a suitable agent who has the required capability for the subtask. A team for the task can be formed if each subtask can be assigned to some suitable agent. In (Hayano et al., 2016) a task is divisible which is not the case in our work. Moreover, in our work two or more agents execute a task.

In (Gunn and Anderson, 2015), the authors describe a framework for dynamic heterogeneous team formation for robotic urban search and rescue. The task discovery is made by a member of a team, and it is sent to the team coordinator for assignment. The team coordinator performs the task assignment, ensuring the task will be carried out by a robot with the necessary capabilities. However, in a distributed system, no robot knows the states, locations, and skills of other robots (i.e., the absence of global knowledge). Thus, the robots should communicate among themselves in order to acquire relevant information for task execution without the intervention of any central authority. This necessitates the design of a distributed algorithm for task execution in such a dynamic environment. In our approach unlike (Gunn and Anderson, 2015), every robot has a similar level of priority, and each of them can perform the task management activities, i.e., searching, team/coalition formation by acquiring the information from the robots available in the environment at that moment in time.

Distributed coalition value calculation (DCVC) is given in (Rahwan, 2007). Cost calculation of a team is done by an initiator in our work. The work (Rahwan, 2007), however, does not consider the problem of distributed multi team formation, which is attempted in our work. In this paper, the task arrival time (at what time it is detected) and location are not known a priori; hence, task searching and coalition/team formation activities are performed by a robot at runtime. The (Skobelev et al., 2018) covers the problem of aggregate and final assembly of complex technical objects at the ramp-up stage. Coalition formation techniques have been implemented on some multi-agent programming contest domains in (Rodrigues et al., 2018). The work (Abbasi et al., 2017) suggest a feedback control system for motion control of the robots.

3 FORMAL FRAMEWORK

Definition 3.1. (**Dynamic Environment**). A global view (snapshot) of an environment $\mathcal{E}$, with a set of locations L, taken at time t, is given by a 3-tuple $\mathcal{E}^t = \langle \mathcal{R}^t, \mathcal{T}^t, f \rangle$, where $\mathcal{R}^t = \{r_1, \ldots, r_n\}$ is the set of robots present in the environment at time t, and $\mathcal{T}^t = \{\tau_1, \ldots, \tau_m\}$ is the set of tasks (e.g., multi-robot tasks) in the environment at time t, $f : \mathcal{R}^t \times \mathbb{N} \mapsto L$, is a function that gives the location of a robot at a discrete instant of time represented by the set of natural numbers $\mathbb{N}$.

In this work agents referred to physical agents, i.e., robots. Each robot $i \in \mathcal{R}$, where $1 \leqslant i \leqslant n$, has a p-dimensional skill vector, $\chi_i = [\alpha_1, \ldots, \alpha_p]$ and at any instant of time it may be in any state from the set of states $\mathcal{S}$= {IDLE, READY, PROMISE, BUSY}. In a dynamic environment, the state of a physical agent (robot) is crucial as a robot can participate in one task, and can be present at one location at a time. The significance of the states are: IDLE state means that a robot is not executing any task and it is free to take any responsibility; BUSY state means that a robot is engaged in executing a task (individually or jointly); READY state indicate that a robot has detected a task and it has started the team formation process for it; and PROMISE state means that an robot has expressed its willingness to be part of a team as robots are assumed to be independent, autonomous and strictly collaborative, not selfish and whenever possible they express their willingness to be part of a team.

Definition 3.2. (**Agent**). An agent is defined as a 4-tuple $\langle \mathcal{S}, \chi, id, b \rangle$, where $\mathcal{S}$ is a set of states, χ is a non-null $p-$dimensional skill vector, id is the identifier, and b is Boolean variable to denote whether or not the agent is within the environment.

The value of a particular skill α_i (where $i = 1 \ldots p$) $\in \chi_i$, may be 1 or 0 depending on whether a robot i possess a particular skill or not. A robot can enter the environment $\mathcal{E}$ at any point in time, but can leave only if it is in IDLE state. In an environment, we can assume that for the execution of any task, one or more skill/s may be required. One way of expressing a set of skills is by using a p-dimensional vector, where p is the number of skills, a robot may possess. The i^{th} component of the vector represents a particular skill, say griping.

Definition 3.3. (**Task**). A task τ is specified by a 4-tuple $\tau = \langle \nu, l, t, \Psi \rangle$ where, ν is the name of a task (e.g., transferring or transporting a box B from location l to location l', lift desk D) etc., where $l \in L$ is the location where the task is arrived/detected, t is the time at which the task arrived, and Ψ is a non-null p-dimensional skill vector required to execute the task.

When a robot detects a task τ, it acquires the information about τ, i.e., p-dimensional skill vector $\Psi_\tau = [\beta_1, \ldots, \beta_p]$, needed to execute the task. We use two binary operators $\odot$ and $\oplus$ that evaluate to true/false. Let v_1 and v_2 be the skill vectors of a robot and task respectively. The intuitive meaning of the operators are: $v_1 \oplus v_2$ holds, when a robot possess at least one skill for a given task; which signifies that a robot can be a member of a team to execute the task; $v_1 \odot v_2$ holds, when a robot possesses all the skills and possibly more for a given task. (Definition 3.4).

Definition 3.4. (**Skill Vector**). Let v_1, v_2 be two p-dimensional skill vectors, $v_1 = [\alpha_1, \ldots, \alpha_p]$, $v_2 = [\beta_1, \ldots, \beta_p]$, where α_i, β_i is either 0 or 1. Let $v_j[i] = \beta_i$ denote the i th component of v_j. We define the following operations on the skill vectors as: (i) ($v_1 = v_2$) holds iff ($\forall i,\ i \in \{1, \ldots, p\},\ v_1[i] = v_2[i]$) holds; (ii) ($v_1 \oplus v_2$) holds iff ($\exists i,\ i \in \{1, \ldots, p\}$, s.t. $v_2[i] \wedge v_1[i]$ holds; (iii) ($v_1 \odot v_2$) iff ($\forall i,\ i \in \{1, \ldots, p\}$, s.t. ($v_2[i] \rightarrow v_1[i]$)) holds; (iv) $v_1\ op\ v_2 = [v_1[1]\ op\ v_2[1], \ldots, v_1[i]\ op\ v_2[i], \ldots, v_1[p]\ op\ v_2[p]]$; where $op \in \{\vee, \wedge\}$ denotes Boolean operations on the vectors; and (v) $op\ (v_1, \ldots, v_n) = (...((v_1\ op\ v_2)\ op\ v_3) \ldots\ op\ v_n)$.

Definition 3.5. (Skill Vector of a Team). Let $\Gamma = \{x_1, \ldots, x_k\}$ be a team of k robots, where $x_1, \ldots, x_k$ are the variables and they are place-holders for $r_1, \ldots, r_n$. We define a p-dimensional skill vector of the team Γ, Υ_Γ as, $\Upsilon_\Gamma = \vee(\chi_{x_1}, \ldots, \chi_{x_k})$.

Definition 3.6. (**Cost of a Task Execution**). Let $\Gamma = \{x_1, \ldots, x_n\}$ be a team that can execute a task $\tau = \langle \nu, l, t, \Psi \rangle$ where each member of the team was located at $loc(x_i, t')$, $t' > t$. The cost of a team Γ for executing τ is $\mathcal{C}_{\langle \Gamma, \tau \rangle} = \sum_{x_i \in \Gamma} \mu_{\langle x_i, \tau \rangle}$ where $\mu_{\langle x_i, \tau \rangle} = \mathbf{p}(x_i, \tau) \times \frac{1}{\theta_{x_i}} + \mathbf{d}(loc(x_i, t'), l) \times \theta'_{x_i}$ where, $\theta_{x_i}, \theta'_{x_i} \in (0, 1]$ denote remaining battery backup and battery consumption rate respectively of x_i, $\mathbf{p}(x_i, \tau)$ is the basic price of x_i for τ, $\mathbf{d}(l_1, l_2)$ is the Euclidean distance between l_1 to l_2. A robot with higher θ ensures that it will not fail due to its more remaining battery backup. A robot with lower θ' ensures that it will last for a longer period of time.

Definition 3.7. (**Dominant Team**). Let $\Gamma_1, \ldots, \Gamma_{n'}$ be the possible teams for executing a task τ. We call Γ_i a dominant team if $\mathcal{C}_{\langle \Gamma_i, \tau \rangle} \leqslant \mathcal{C}_{\langle \Gamma_j, \tau \rangle}\ \forall\ j \in \{1, \ldots, n'\} \backslash \{i\}$.

Definition 3.8. (**Conditions for Multi-team Formation**). Tasks $\tau_1 = \langle \nu_1, l_1, t, \Psi_1 \rangle$ and $\tau_2 = \langle \nu_2, l_2, t, \Psi_2 \rangle$ can be executed simultaneously if the following conditions hold:

- There exists a set $\mathcal{R}_1$ of available robots at some time $t'_1 > t$, such that $\Upsilon_{\mathcal{R}_1} \odot \Psi_1$ holds, and at some time $t''_1 > t'_1$, $loc_r = l_1$ for all $r \in \mathcal{R}_1$.
- There exists a set $\mathcal{R}_2$ of available robots at some time $t'_2 > t$, such that $\Upsilon_{\mathcal{R}_2} \odot \Psi_2$ holds, and at some time $t''_2 > t'_2$, $loc_r = l_2$ for all $r \in \mathcal{R}_2$.
- $\mathcal{R}_1 \cap \mathcal{R}_2 = \emptyset$.

Problem Statement: We assume that there is a finite number of agents, situated at different locations of an environment E. No agent knows the total number of agents in the environment. No agent knows the skills, current state and current location of another agent. The only way for an agent to update its knowledge is via sending messages to another agent(s). Design a distributed algorithm for dominant multi-team formation simultaneously.

4 DISTRIBUTED ALGORITHM FOR MULTIPLE-TEAM FORMATION (DMTF)

We consider a wireless network that is lossless, message delay is finite, data is not corrupted during transmission. Messages are delivered in a FIFO (first-in-first-out) manner. No message is delivered to an agent who is outside the environment. No assumption is made on the network topology. Agents are cooperative and whenever possible they expresses their willingness to be part of a team.

The DMTF algorithm consists of send and receive functions given in Algorithm 1 and 2 respectively. In order to form a team for the execution of task τ, i communicates with other robots. We refer to i as an initiator, and the other robots as non-initiators. The overall team formation process is given in Algorithm 1 and Algorithm 2. To form a team, an initiator i broadcasts a Request message $\langle id_i, \nu_\tau, l_\tau, (\Psi_\tau - \chi_i) \rangle$ within communication range and waits for some time, say Δ; id_i is the identifier of initiator i, l_τ is name of the task τ, Ψ_τ is skill of task τ, and χ_i is skill of initiator i. Here,

$(\Psi_\tau - \chi_i)$ means the remaining skills required for executing the task.

A non-initiator j, who has at least one skill required to execute the task (i.e., $\chi_{r_1} \oplus \Psi_\tau$), will send either a Willing (whose format is $\langle id_j, \chi_j, \mathbf{p}_j, \theta'_j, \theta_j, loc_j \rangle$), if it is in IDLE state or an Engaged message if its state is other than IDLE. The initiator increases its counter c when it receives a Willing message from non-initiator. The initiator constructs a matrix A_S$[a_{ij}]$ of dimension $(c+1) \times p$, to keep the record of skills received via Willing message. The matrix is initialized with all zero, i.e., $a_{ij} = 0$; $\forall i \in n, \forall j \in p$). If a particular skill is available in non-initiator's skill vector χ_j, the value on that cell in the matrix A_S[] is updated with value 1. The column-wise union of each column of the matrix A_S[] is done after computing all Willing messages (after Δ time has elapsed). Eventually, initiator checks whether the task τ can be executed by this available skill vector. If yes, a dominant team is selected as per Definition 3.6, which is a combinatorial optimization problem where k elements are chosen from a set of size c; this problem is known to be NP-hard (Okimoto et al., 2015). In the computation of cost (Definition 3.6), the distance between two locations is assumed to be the Euclidean distance. Otherwise, the task is postponed temporarily. Thus team formation for a task is not guaranteed any time.

The initiator sends a Confirm message to those robots $(k-1)$ who are finally selected for the team and it sends Not-required message to those robots $(c-(k-1))$ who had sent Willing message but finally are not selected for team. Also, i changes its state from READY to IDLE or PROMISE depending on its queue status. The matrix construction and updation are illustrated by an example later in this section. The proposed distributed algorithm is non-blocking since a timer is used. If there was no timer, an initiator would have waited indefinitely and thereby forcing some non-initiators to wait indefinitely as well; thus the system would be blocked.

The *receive* function of a robot is given in Algorithm 2. The computations are done based on the current state that may be IDLE (line no. 19-26), PROMISE (line no. 27-42), BUSY (line no. 14-17 and 44-46), and READY (line no. 2-13). Within a state, the type of message is checked and appropriate actions are taken. For example, when state is IDLE, if a Request message is received, it becomes PROMISE, the identifier of the sender is en-queued, and flag is set to true; all these actions are done atomically (denoted by $\langle \ldots \rangle$). Now the robot sends a Willing message to the sender (initiator) and flag is set to false (line no. 24-25 Algorithm 1).

A robot j maintains a local queue Q which keeps the identifiers of the senders, based on the incoming Request messages. The Q is used to avoid starvation since more than one initiator may send Request messages at the same instant in time. When a robot goes to BUSY state, it means, it is in task execution state, which takes considerable time. So, maintaining the Q for this period is of no use, and thus Q is made empty. The Boolean variables $flag$ and $flag'$ are used to control the sending of Willing and Engaged messages respectively.

Algorithm 1: DMTF-Send function.

```
   Send function of initiator $i$ :
1  $state$ := READY; broadcast Request$_i$ ;
2  create a matrix A_S$[a_{ij}]_{n \times p}$;        $\diamond$ to keep the record of skills
3  start timer and wait for $\Delta$ unit of time;
4  if (team formed for task $\tau$) then
5      for each possible team, calculate cost as per Definition 3.6
         and select the team that has minimum cost, say $\Gamma$ of size $k$;
6      send Confirm$_i$ to $(k-1)$ members of $\Gamma$;
7      send Not-Required$_i$ to $(c-(k-1))$ non-initiators;
8      $\langle state :=$ BUSY; make $Q_i$ empty$\rangle$;
9      initiate execution of $\tau$ when $(k-1)$ members of $\Gamma$ arrive at
         $l$;
10 else
11     send Not-Required$_i$ to $c$ non-initiators;
12     if $Q_i = \varnothing$ then
13         $state :=$ IDLE;
14     else
15         $state :=$ PROMISE;
16         send Willing$_i$ to the front element of $Q_i$;
17     end
18 end
19 if Willing message is received in a state $\neq$ READY from $j$ then
20     send Not-Required$_i$ to $j$;
21 end
   Send function of Non-initiator $j$ for a task $\tau = \langle \nu, l, t, \Psi \rangle$:
22 case $Q_j \neq \varnothing$ and flag = true do
23     send Willing$_j$ to the front element of $Q_j$; $flag := false$;
24 end
25 case $Q_j \neq \varnothing$ and flag$'$ = true do
26     send Engaged$_j$ to the rear element of $Q_j$; $flag' := false$;
27 end
```

In PROMISE state, non-initiator robot receives either Confirm or Not-required messages. If non-initiator receive a Confirm message, its approach towards initiator/task's location (Algorithm 2, line no. 31-33). If non-initiator receives Not-required message, then it change its state from PROMISE to IDLE (Algorithm 2, line no. 34-41). The non-initiator who receives the CONFIRM message and moves towards initiator location, sends a beacon signal to acknowledge that it has reached. When all selected robots reach task τ location, they synchronize and align themselves to be ready for task execution. Eventually, initiator commands for task execution and they

Algorithm 2: DMTF-Receive function.

```
Receive function of Initiator j :
1  $c := 0$;  // to count the number of Willing messages
2  case state = READY do
3      case msg = Request do
4          ⟨enqueue($Q_j, i$); $flag' := true$⟩
5      end
6      case msg = Willing do
7          $c := c + 1$;
8          update available matrix A_S$[a_{ij}]_{n \times p}$;
9      end
10     case msg = Engaged do
11         skip;
12     end
13 end
14 case state = BUSY do
15     case msg = Request do
16         skip;
17     end
18 end
Receive function of Non-initiator j :
19 case state = IDLE do
20     case msg = Request and $\chi_j \oplus \Psi_\tau$ do
21         ⟨state := PROMISE; enqueue($Q_j, i$) if i is not
             present in $Q_j$; flag := true⟩
22     end
23     case msg = Request and $\neg(\chi_j \oplus \Psi_\tau)$ do
24         skip;
25     end
26 end
27 case state = PROMISE and $\chi_j \oplus \Psi_\tau$ do
28     case msg = Request do
29         ⟨enqueue($Q_j, i$) if i is not present in
             $Q_j$; flag$' := true$⟩
30     end
31     case msg = Confirm do
32         ⟨state := BUSY; make $Q_j$ empty; move to $loc_i$⟩
33     end
34     case msg = Not-Required do
35         dequeue($Q_j$);
36         if $Q_j = \varnothing$ then
37             state := IDLE;
38         else
39             flag := true;
40         end
41     end
42 end
43 case state = BUSY do
44     case msg = Request do
45         skip;
46     end
47 end
```

jointly execute the task.

All the robots of the team including initiator, start the task execution (transportation) and their state becomes BUSY now. After finishing the task team is dissolved and robots (initiator and non-initiator) changes their state to IDLE. In this way, all the robots of the environment behave according to Algorithms 1 and 2 and accomplish the mission. Wherever no task is left of a mission, the algorithm terminates.

5 EXPERIMENTAL RESULTS

In the environment, prototypes in AGRoS, 5 robots and 4 different tasks (transportation of object/obstacle) are present. The 6-dimensional skill vector required for executing the tasks is as follows: task $\Psi_{\tau_1} = [1,0,1,1,0,1]$, task $\Psi_{\tau_2} = [0,0,0,0,1,1]$, task $\Psi_{\tau_3} = [1,0,0,0,0,1]$, and task $\Psi_{\tau_4} = [1,0,0,0,0,0]$. The task τ_1, τ_2, τ_3 and τ_4 are represented with large disc, large box, small disk and small box respectively. The battery back required for executing each task is considered to be $\geqslant 50\%$. Initially, battery backup θ of each robot is assumed to be 100%. The details of a robot is given in Table 1.

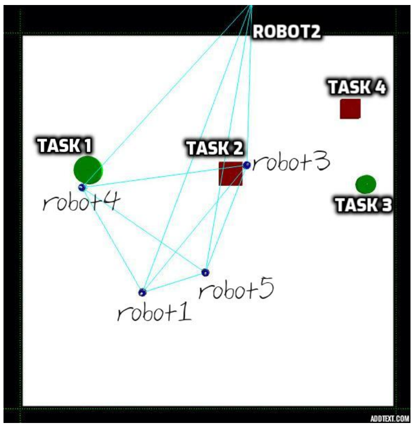
Table 1: The value of skill vector χ, basic price **p**, and battery consumption coefficient θ' of each robot

Robot	Skills_set (χ_i)	(**p**)	θ'
r_1	$[1,0,0,0,0,0]$	40	.5
r_2	$[0,0,0,0,0,1]$	50	.6
r_3	$[1,1,0,0,0,0]$	70	.7
r_4	$[0,0,0,0,1,1]$	70	.8
r_5	$[0,0,1,1,0,0]$	60	.6

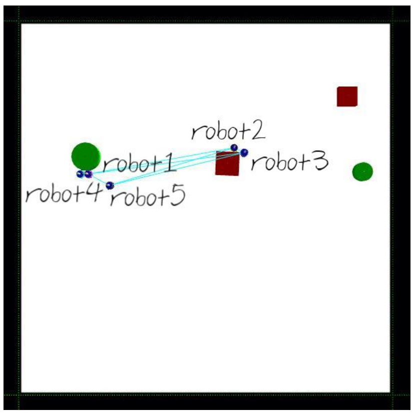
5.1 Multi-team Formation Scenario

In the dynamic environment, the arrival time and location of the task are not known a priori. In such a situation, multiple tasks of a mission may be detected at the same time by different robots (initiators). To execute the tasks, multiple teams are required to be formed simultaneously. The algorithm should be capable enough to handle this situation. The proposed algorithm handles this situation easily as shown in Figure 1. If a sufficient number of robots are available in the environment (conditions for multi-team formation satisfies as given in Definition 3.8) at the time of the team formation process, then multiple teams can be formed successfully. The multi-team can execute multi-tasks in parallel. The illustration of multi-team formation process is given in Figure 1 and corresponding formed team are presented in Tables 2 and 3.

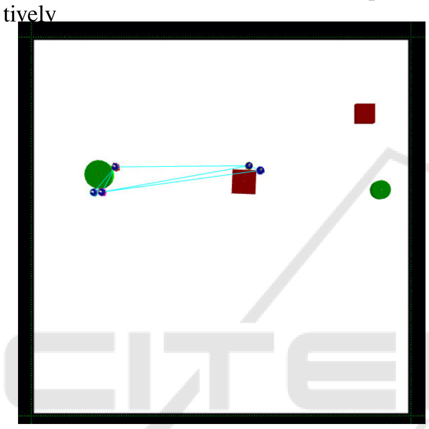
At some moment in time, two robots r_4 and r_3 detect two tasks τ_1 and τ_2 respectively. The task τ_1 is represented by large disc green in color, while the τ_2 is shown with a large box and red color. Both robots r_4 and r_3 start the team formation process simultaneously. Two teams are formed successfully. The members of team 1 and 2 are given in Table 2. The results



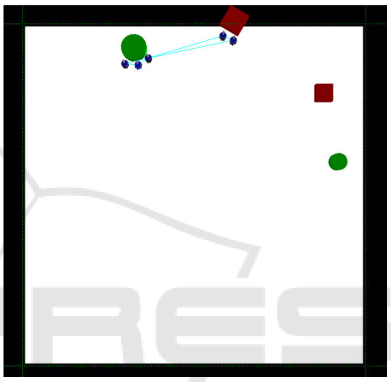
(a) r_4 and r_3 detect the τ_1 and τ_2 respectively



(b) r_4 and r_3 have formed their corresponding teams and selected robots coming them



(c) Non-initiators have reached their corresponding task's location



(d) τ_1 and τ_2 are being executed

Figure 1: Multi-team formation and dual task execution in parallel

for another run of the algorithm are given in Table 3. In this way, we can conclude that the proposed distributed algorithm can handle the multi-team formation.

Table 2: Team formation for the execution of tasks (τ_1, τ_2), simultaneously

Tasks	Task's initiator	team	Cost
τ_1	r_4	$\{r_4, r_1, r_5\}$	1353.45
τ_2	r_3	$\{r_3, r_2\}$	785.78
Total_Cost			**1149.23**
Total_Time			**123.27**

Table 3: Team formation for the execution of tasks (τ_3, τ_4), simultaneously

Tasks	Task's initiator	team	Cost
τ_3	r_2	$\{r_2, r_4\}$	1145.85
τ_4	r_1	$\{r_1\}$	646.97
Total_Cost			**1792.82**
Total_Time			**102.42**

5.2 Discussion

We present some challenges encountered during the implementation. The implementation of the proposed distributed algorithm (given in Section 4) consists of challenges like how to explore, navigate, avoid collision, control the speed of a robot when it comes near other objects (robot or obstacle), rotation and alignment to reach the exact location, grab the object, synchronization while execution. We managed these issues during the implementation of DMTF in ARGoS. We have also addressed another subtle situation, where two robots are present at the location of a task and both are eligible for executing the task. Now, they may start the team formation process simultaneously for the same task, which should not happen. This situation is handled as follows. A robot can sense its neighbor by an omni-directional camera sensor. The robots will exchange their identities and skill vectors by Beacon signals. The robot who has more skills will become the initiator. If their skill vectors are same, the robot with lower identifier will become the initiator.

6 CONCLUSION

We considered a type of multiple team formation problem in a dynamic multi-agent setting, where agents have limited information about an environment (e.g., total number of agents present in the environment, skill, state and location of other robots) and they should coordinate among themselves by exchanging messages to accomplish a given mission composed of multi-robot tasks. We presented a distributed algorithm DMTF that can form multiple teams simultaneously, which to the best of our knowledge, is the first attempt.

Implementing a distributed algorithm is nontrivial. A prototype model of the proposed algorithm is developed in ARGoS; a multi-robot simulation environment, and it was tested through intensive simulations. The simulation results are quite encouraging, exhibit the expected behavior of the algorithm, and illustrate how multiple teams are formed. Our algorithm could be useful in real-world applications where control is distributed, and no central entity is responsible for controlling the activities of other robots, and the roles of robots are not decided in advance. As part of our future work, we wish to implement the prototype model on real robots and test the efficiency of the system.

ACKNOWLEDGMENT

The authors thank the anonymous reviewers for their valuable comments that were helpful for improving the paper. The third author was in part supported by a research grant from Google.

REFERENCES

Abbasi, E., Ghayour, M., and Danesh, M. (2017). Virtual leader-follower formation control of multi quadrotors by using feedback linearization controller. In *ICRoM*, pages 614–619. IEEE.

Abdallah, S. and Lesser, V. (2004). Organization-based cooperative coalition formation. In *Int. Conf. on IAT*, pages 162–168.

Coviello, L. and Franceschetti, M. (2012). Distributed team formation in multi-agent systems: stability and approximation. In *51st Annual Conf. on Decision and Control*, pages 2755–2760.

Dukeman, A. and Adams, J. A. (2017). Hybrid mission planning with coalition formation. *AAMAS*, 31(6):1424–1466.

Faliszewski, P., Slinko, A., and Talmon, N. (2016). Voting-based group formation. In *IJCAI*.

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

Gunn, T. and Anderson, J. (2015). Dynamic heterogeneous team formation for robotic urban search and rescue. *Journal of Computer and System Sciences*, 81(3):553–567.

Gutiérrez, e. a. (2016). The multiple team formation problem using sociometry. *Computers & Operations Research*, 75:150–162.

Hayano, M., Miyashita, Y., and Sugawara, T. (2016). Switching behavioral strategies for effective team formation by autonomous agent organization. In *ICAART*, pages 56–65.

Kong, e. a. (2015). An auction-based approach for group task allocation in an open network environment. *The Computer Journal*, 59(3):403–422.

Lappas, T., Liu, K., and Terzi, E. (2009). Finding a team of experts in social networks. In *ACM SIGKDD*, pages 467–476. ACM.

Lynch, N. A. (1996). *Distributed algorithms*. Morgan Kaufmann.

Okimoto, T., Ribeiro, T., Bouchabou, D., and Inoue, K. (2016). Mission oriented robust multi-team formation and its application to robot rescue simulation. In *IJCAI*, pages 454–460.

Okimoto, T., Schwind, N., Clement, M., Ribeiro, T., Inoue, K., and Marquis, P. (2015). How to form a task-oriented robust team. In *AAMAS*, pages 395–403.

Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G., Ducatelle, F., et al. (2012). Argos: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm intelligence*, 6(4):271–295.

Rahwan, T. (2007). *Algorithms for coalition formation in multi-agent systems*. PhD thesis, University of Southampton.

Rodrigues, T. K. et al. (2018). Constrained coalition formation among heterogeneous agents for the multi-agent programming contest.

Şahin, E. (2004). Swarm robotics: From sources of inspiration to domains of application. In *International workshop on swarm robotics*, pages 10–20. Springer.

Shehory, O. and Kraus, S. (1998). Methods for task allocation via agent coalition formation. *Artificial intelligence*, 101(1-2):165–200.

Skobelev, P., Eliseev, V., Mayorov, I., Travin, V., Zhilyaev, A., and Simonova, E. (2018). Designing distributed multi-agent system for aggregate and final assembly of complex technical objects on ramp-up stage. In *ICAART (1)*, pages 250–257.

Tosic, P. and Agha, G. (2005). Maximal clique based distributed coalition formation for task allocation in large-scale multi-agent systems. In *Int. Workshop on Massively Multiagent Systems*, pages 104–120.

Vig, L. and Adams, J. A. (2006). Multi-robot coalition formation. *IEEE Trans. on Robotics*, 22(4):637–649.

Xie, B., Chen, S., Chen, J., and Shen, L. (2018). A mutual-selecting market-based mechanism for dynamic coalition formation. *International Journal of Advanced Robotic Systems*, 15(1):1729881418755840.

Yan, Z., Jouandeau, N., and Cherif, A. A. (2013). A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, 10(12):399.