

Distributed Framework for Task Execution with Quantitative Skills

Amar Nath[1] and Rajdeep Niyogi[2(✉)]

[1] SLIET Deemed University, Longowal 148106, India
amarnath@sliet.ac.in
[2] Indian Institute of Technology Roorkee, Roorkee 247667, India
rajdpfec@iitr.ac.in

Abstract. Collaborative task execution is an important area of research in multi-agent systems. In some situations, the agents are spatially distributed, have limited information about the environment, and update their knowledge via exchanging messages. Distributed approaches for task execution in such situations have been suggested in the literature. In these approaches, skills of robots and skills required for task execution are represented as p-dimensional binary skill vectors. However, in real-world applications, it would be more desirable to consider real-valued skills. In this paper, we develop a distributed framework that consider quantitative representation of skills. We have performed extensive experiments on a RoboCupRescue simulation environment. The experimental results show the efficacy of the approach.

Keywords: Task execution · Distributed framework · Quantitative skills

1 Introduction

Autonomous robots can be deployed for task execution in some applications (e.g., search and rescue [1], space exploration [10], demining [12]) where direct human intervention is impossible or impractical. In some situations a coalition (or team) is formed based on a complete information of the world. In such a setting, the coalition formation problem can be viewed as a combinatorial optimization problem that selects a subset (coalition) from a given set of members. A coalition is guaranteed to be formed although the computational time is exponential. Coalition formation algorithms for this setting have been suggested in [13,14].

In this paper, task execution is done in a partially observable environment where the agents are spatially distributed and no agent has a global view of the environment. Thus complete knowledge of the world is not known in advance and it can be determined partially when the need for a coalition formation arises (i.e., at run-time). In such situations no guarantee can be given that any previously found coalition will execute the task successfully. Moreover, unlike the centralized setting, there is also no guarantee that a coalition would be formed;

© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12955, pp. 413–426, 2021.
https://doi.org/10.1007/978-3-030-87007-2_29

the coalition formation process may be unsuccessful due to, say, suitable agents are unavailable for the task and/or some potential members are not reachable. So the information about the world should be acquired autonomously at runtime by an agent, who has detected a task that is indivisible, i.e., cannot be divided into subtasks, by exchanging messages (explicit communication) with other agents. This is quite challenging due to the following dynamic aspects of the environment.

The state of an agent changes depending on whether it is navigating, trying to form a coalition, participating in a coalition formation, or executing a task. The locations of the agents change, agents may enter or exit the environment, and task arrival is not known in advance. Task execution requires that all members of a coalition should be present at the location of the task. Distributed approaches for team formation in such settings have been suggested in [2–5].

In [4,5], skills of robots and skills required for task execution are represented as p-dimensional binary skill vectors. Such a specification merely states whether or not a robot has a particular capability. Likewise for task execution, the required set of capabilities is given. If a team is selected for task execution, it means that the combined capabilities of the team members is a superset of the capabilities required for task execution. In real world situations, a task should be specified not just by what capabilities are required for its execution but also by some measure of the capabilities. Similarly, a capability vector of a robot should also specify the quantitative value of the particular capabilities. In this paper, we extend these approaches [4,5] by considering quantitative representation of capability.

The remaining part of the paper is organized as follows. Related work is given in Sect. 2. A generic formal framework is given in Sect. 3. A distributed approach for task execution is described in Sect. 4. The experimental results are given in Sect. 5. Conclusions are given in Sect. 6.

2 Related Work

The works [2–5] consider collaborative task execution in a dynamic environment where all the agents associated with the execution should be present at the location of the task. In [2–5] distributed approaches for collaborative task execution are suggested based on different assumptions. In [2,3] team size is known in advance, and each agent in a team should have a set of skills that is a superset of the set of skills of the task. This means that the team is homogeneous with respect to the skills of the task. A formal framework of a dynamic environment is given in [2] and the behavior of an agent is modeled using communicating automata. A distributed approach for task execution based on this model is suggested in [2]. A distributed approach for road clearance with multiple robots in urban search and rescue environment is considered in [3], where the behavior of a robot is modeled using communicating automata.

Unlike [2,3], a generic situation is considered in [4,5], where the combined set of skills of the agents of a team should be a superset of the skills of a task.

Thus the team size is not known in advance; rather determined at run-time in [4,5]. This means that the team is heterogeneous with respect to the skills of the task. The primary difference between the types of tasks considered in [2,3] and [4,5] is as follows. In order to execute a task in [2,3], all the agents of the team should perform a same action (e.g., when a team is pushing a heavy box, all the agents of the team need to perform the action 'push'). In order to execute a task in [4,5], different agents of the team need to perform a different action (e.g., when a team is transporting an object, one agent may be involved in pushing, another involved in grabbing, and some other involved in obstacle avoidance). In this paper the type of tasks considered is the same as that in [4,5].

Whereas, [4,5] consider p-dimensional binary skill vectors, this paper considers a more general p-dimensional skill vectors where each element can be a non-negative real number. Such a representation of skills of robots and tasks has been considered in [6]. In [6] a coalition problem is considered such that there is no coalition imbalance, which informally refers to the degree of dependence on some members of a coalition. The notion of imbalance is beyond the scope of this paper.

Team formation in a networked multi-agent system is considered in [11], where each agent has a single fixed skill. Each task is associated a size and a corresponding sized skill vector is formed. In [11] a team formation algorithm is suggested where an agent joins a team using any one of the two strategies based on preferential attachment and performance and referrals. However, quantitative representation of skills is not considered in [11].

Distributed approaches for team formation have been suggested in [15–20]. However, representation of skills of agents and tasks are not quite relevant in these works. Unlike some works [15,18–20] where the roles of the robots are fixed in advance, the role of a robot is determined at run-time in this paper as well as in [2–5].

The centralized approaches for team formation [13,14] represent skills of agents and tasks by sets. The objective is to select some agents such that the combined skills of the agents cover all the skills of the task.

3 A Formal Framework for Task Execution

The intention behind the following definition is to capture (i) dynamic nature of robots: number of robots present in the environment and the locations of the robots change with time and (ii) dynamic nature of tasks: number of tasks in the environment and the locations of the tasks change with time. Moreover, this dynamic nature of the robots and tasks is not known in advance.

Definition 1 *(*Dynamic Environment*) [2]. A global view (snapshot) of an environment E, with a nonempty finite set of locations L, taken at time t, is given by a 4-tuple $E^t = \langle \mathcal{R}, \mathcal{T}, loc, loc' \rangle$ where $\mathcal{R}$ is a nonempty finite set of robots present in the environment at time t, and $\mathcal{T}$ is a nonempty finite set of tasks in the environment at time t, $loc : \mathcal{R} \times \mathbb{N} \rightarrow L$, is a function that gives the location*

of a robot at a discrete instant of time represented by the set of timestamps $\mathbb{N}$*,* $loc' : \mathcal{T} \times \mathbb{N} \rightarrow L$*, is a function that gives the location of a task at a discrete instant of time.*

Definition 2. *(*Robot*) A robot is defined as a 10-tuple* $r = \langle id, s, \psi, \mu, m_{snd}, m_{rcv}, bool, c, \theta, \theta' \rangle$ *where id is a unique identifier,* $s \in S$ *is a state from among a possible nonempty finite set of states* S*,* ψ *is a capability vector representing the capabilities,* $\mu : L \mapsto L$ *is a mapping from one location to another to represent mobility,* m_{snd}*,* $m_{rcv} \in M$ *is the message sent (received) from among a possible nonempty finite set of messages* M*, bool is a Boolean variable that is true/false depending on whether or not the robot is inside/outside an environment (if bool is true, location of robot is* $l_r \in L$*), c is a minimum cost that the robot charges for a task,* $\theta, \theta' \in (0, 1]$ *denote remaining battery backup and battery consumption rate respectively of the robot.*

A robot with higher θ ensures that it will not fail due to more remaining battery backup. A robot with lower θ' ensures that it will last for a longer period of time.

Definition 3 *(*Capability vector*). A p-dimensional capability vector v is denoted as* $v = [\alpha_1, \ldots, \alpha_p]$*, where* α_i *is a non-negative real number to give a quantitative representation of a particular capability.*

In an environment, we can assume that for the execution of a task, one or more capabilities may be required from a set of capabilities, e.g., holding, pushing, obstacle detection, and vision.

Example: Let the robots r_1, r_2, r_3, and r_4 have the following capabilities; r_1: navigation, r_2: holding, r_3: holding and pushing, r_4: pushing. The capability vectors are represented as $\psi_{r_1} = [30.5, 0, 0]$, $\psi_{r_2} = [0, 20, 0]$, $\psi_{r_3} = [0, 10.6, 10]$, and $\psi_{r_4} = [0, 0, 20.9]$ where the first component of the vector denotes navigation, the second holding, and the third pushing. This is shown in Table 1.

Table 1. Capability vector of the robots

Capability vector	Capability		
	navigation	holding	pushing
ψ_{r_1}	30.5	0	0
ψ_{r_2}	0	20	0
ψ_{r_3}	0	10.6	10
ψ_{r_4}	0	0	20.9

Definition 4 *(*Capability vector operators*). Let* v_1, v_2 *be two p-dimensional capability vectors,* $v_1 = [\alpha_1, \ldots, \alpha_p]$*,* $v_2 = [\beta_1, \ldots, \beta_p]$*, where* α_i, β_i *are capabilities represented by non-negative real numbers. Let* $v_j[i]$ *denote the* i^{th} *component of* v_j*, i.e.,* α_i *or* β_i*. We define the following operators on the capability vectors as:*

- $v_1 = v_2$ *equals true iff* $\forall i \in \{1, \ldots, p\},\ v_1[i] = v_2[i]$.
- $v_1 \oplus v_2$ *equals true iff* $\exists i \in \{1, \ldots, p\},\ (v_2[i] > 0) \wedge (v_1[i] > 0)$.
- $v_1 \odot v_2$ *equals true iff* $\forall i \in \{1, \ldots, p\},\ v_2[i] > 0 \rightarrow (v_1[i] \geqslant v_2[i])$.

Two vectors are equal if all the corresponding components of the vectors are equal. $v_1 \oplus v_2$ equals true denotes that there is at least one component for which the values of the capabilities in both the vectors are greater than zero. $v_1 \odot v_2$ equals true denotes that for all components where the value of capability of v_2 is greater than zero, the value of capability of v_1 should be at least the value of v_2.

Let $v_1 = [20, 0, 12]$, $v_2 = [5, 2, 8]$. It is easy to see that v_1 and v_2 are not equal. $v_1 \oplus v_2$ is true since for the first component the values are 20 and 5 respectively which exceed 0. $v_1 \odot v_2$ is not true. The first component of v_2 is 5 which is greater than 0, and the first component of v_1 is 20 which is greater than 5. The same is true for the third component. However, for the second component, the value of v_2 is 2 which is greater than 0 but the value of v_1 is less than 2.

Definition 5 *(*Necessary and sufficient conditions for task execution*). Let v_1, v_2 be the p-dimensional capability vectors of a robot (a coalition) and a task τ respectively. If $v_1 \oplus v_2$ is true, it means that the robot (the coalition) can participate in the task execution. $v_1 \oplus v_2$ is true is a necessary but not a sufficient condition for task execution. $v_1 \odot v_2$ is true is a sufficient condition for the robot (the coalition) to execute the task.*

Definition 6 *(*Task*) [4]. 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., move (carry) box* `B` *to location l', lift desk* `D`*), $l = loc'(\tau, t)$ is the location where the task arrived, t is the time at which the task arrived, and Ψ is a p-dimensional capability vector required to execute the task. The task is indivisible i.e., it cannot be divided into subtasks.*

When a robot detects a task τ, it acquires the information about τ, i.e., the p-dimensional capability vector $\Psi = [\beta_1, \ldots, \beta_p]$ required to execute the task. For instance, $\Psi = [\beta_1, \beta_2, \beta_3, \beta_4] = [0, 20, 0, 30]$ means that 20 units of capability β_2 and 30 units of capability β_4 are required.

Definition 7 (Capability vector of a coalition). *Let $\Gamma = \{r_1, \ldots, r_k\}$ be a coalition of k robots and the p-dimensional capability vector of the robots are $\psi_{r_1}, \ldots, \psi_{r_k}$. The ith component of the p-dimensional capability vector of the coalition Γ is defined as, $\Upsilon_\Gamma[i] = \psi_{r_1}[i] + \ldots + \psi_{r_k}[i]$, $i = 1, \ldots, p$.*

Example: Consider three robots r_1, r_2, and r_3 with 4-dimensional capability vectors $\psi_{r_1} = [10, 20, 0, 0]$, $\psi_{r_2} = [0, 0, 0, 20]$, and $\psi_{r_3} = [5, 15, 0, 5]$. The capability vector of a coalition $\Gamma = \{r_1, r_2, r_3\}$ is obtained by the point-wise addition of the corresponding capabilities of the members of the coalition. This is shown in Table 2.

Definition 8 (Task execution by a coalition). *A coalition Γ, of size at least 2, can execute a task $\tau = \langle \nu, l, t, \Psi \rangle$, if $\Upsilon_\Gamma \odot \Psi$ is true.*

Table 2. Capability vector of a coalition Γ

Capability vector	Capability			
	α_1	α_2	α_3	α_4
ψ_{r_1}	10	20	0	0
ψ_{r_2}	0	0	0	20
ψ_{r_3}	5	15	0	5
$\Upsilon_\Gamma[i] = \psi_{r_1}[i] + \psi_{r_1}[i] + \psi_{r_3}[i]$	15	35	0	25

Definition 9 *(*Cost of a task execution*). Let $\Gamma = \{r_1, \ldots, r_k\}$, $k \geqslant 2$, be a coalition that can execute a task $\tau = \langle \nu, l, t, \Psi \rangle$ where each member of the coalition was located at $l_{r_i} = loc(r_i, t')$, for some $t' > t$. The cost of coalition Γ for executing τ is $\mathcal{C}_{\langle \Gamma, \tau \rangle} = \sum_{r_i \in \Gamma} (c_{\langle r_i, \tau \rangle} \times \frac{1}{\theta_{r_i}} + d(l_{r_i}, l) \times \theta'_{r_i})$, where d denotes Euclidean distance.*

Definition 10 *(*Dominant Coalition*). Let $\Gamma_1, \ldots, \Gamma_n$ be the possible coalitions for executing a task τ. We call Γ_i a dominant coalition if $\mathcal{C}_{\langle \Gamma_i, \tau \rangle} \leqslant \mathcal{C}_{\langle \Gamma_j, \tau \rangle}$ $\forall\ j \in \{1, \ldots, n\} \setminus \{i\}$.*

4 Distributed Approach for Task Execution

We present a distributed approach for collaborative task execution by extending [4,5] by considering quantitative representation of capabilities. 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 assumption is made on the network topology. A robot can enter the environment at any point in time, but can exit only if it is in IDLE state. No message is delivered to a robot who is outside the environment.

Let a robot i, in IDLE state, detect a task $\tau = \langle \nu, l, t, \Psi \rangle$ such that $\psi_i \oplus \Psi$ is true whereas $\psi_i \odot \Psi$ is not true. So the robot starts the coalition formation process and changes it's state from IDLE to ANALYZE. In order to form a coalition, i communicates with other robots. We refer to i as an initiator, and the other robots as non-initiators.

Team Formation: Initiator: To form a coalition, i broadcasts a Request message within its range (a local broadcast), starts its timer and waits for some time, say Δ. A Request message contains a field Ψ' that indicates the remaining capability required for executing the task; $\Psi' = \Psi - \psi_i$. For instance, if $\Psi = [10, 5, 4, 0]$ and $\psi_i = [6, 3, 0, 5]$, then $\Psi' = [4, 2, 4, 0]$ which is obtained by subtracting the corresponding components. Note that the last component of Ψ is 0, which means that this component is not required. So in Ψ' this component is kept as it is, i.e., 0. Now consider another situation where $\Psi = [10, 5, 4, 2]$ and ψ_i is same as above. Consider the fourth component. The capability value of i

for this component exceeds that of the task. No other robot is thus required to fulfill this component. So for this situation, $\Psi' = [4, 2, 4, 0]$ which is same as in the previous situation.

During the period when the timer is ON, if a Willing message is received, the counter c is incremented by one and the capabilities are recorded in a matrix. If a Willing message is received when the timer is OFF, a Not-required message is sent. If a coalition cannot be formed for the task, Not-required message is sent and the state of the initiator becomes IDLE. This situation manifests failure to form a team. If a coalition can be formed, a dominant coalition is found. Then Confirm and Not-required messages are sent to the respective non-initiators, and the state of the initiator becomes WAITING.

In the WAITING state the initiator waits for the non-initiators to arrive at the location of the task. When all the members arrive, the state becomes BUSY. Now the task is executed. Upon completion of the task, state becomes IDLE.

Team Formation: Non-initiator: A non-initiator j, who is in IDLE state and has some capability to execute the task responds to the Request message by sending a Willing message to the initiator. In other cases, the non-initiator just ignores the message. The following cases may arise. A non-initiator receives a Request message from another initiator in PROMISE state. Since a non-initiator engages itself with one initiator at a time, so this message is ignored. The non-initiator does not have any one of the capability of the task. So it ignores the message.

Now if a Confirm message is received when its state is PROMISE, j changes its state to APPROACH and starts moving towards the location l of the task. On the other hand, if a Not-required message is received when its state is PROMISE, j changes its state to IDLE and disengages itself from the coalition formation process. When j has come very close to l, it sends an Arrived message to the initiator i, and changes its state to BUSY. Thereafter the task would be executed and after its completion the state changes to IDLE. After completion of the task τ, the coalition is dissolved and the robots (initiator and non-initiators) change their states to IDLE.

5 Experiments and Results

5.1 Experimental Setup

To evaluate the distributed approach given in the previous Section, we used ARGoS [7], a realistic multi-robot simulator. The experiments were carried out by running the 3.0.0-beta50 version of ARGoS on Intel® Core™ i5-2600 CPU@3.40 GHz×8, 8 GB RAM and 64-bit Ubuntu operating system. Each experiment has been run multiple times.

The algorithm can also be simulated using other multi-robot simulators, for example, V-REP [8], ROS-Gazebo [9]. V-REP has a very feature-rich simulation environment that includes a scene and a model editor, large library of models,

real-time mesh manipulation. ROS-Gazebo can utilize complex 3D meshes and physics engines. ARGoS, on the other hand, is a lightweight alternative that is open-source, and suitable for multi-robot applications. ARGoS is a multi-physics robot simulator. The code run in ARGoS can be directly deployed on a real robot system. We can customize ARGoS easily by adding new plug-ins. Based on these features we chose the ARGoS multi-robot simulator for the implementation.

Implementation of the distributed approach requires addressing some challenging issues, that are: (i) to control the movement of a robot to avoid obstacle or another robot based on proximity sensor data, where the sensor detects an obstacle or another robot, (ii) control speed and velocity of a robot, (iii) synchronizing the robots for task execution, (iv) to control the movement of a robot when boundaries are detected using motor-ground sensors, and (v) communication among robots using range_and_bearing sensor. We were able to suitably customize the functionalities of ARGoS to address these issues. We have implemented the distributed framework on a real-world environment.

5.2 Road Clearance Environment

A RoboCupRescue simulation is taken as a case study that mimic a disaster scenario of a city [1]. In the RoboCupRescue simulation environment, heterogeneous field agents (ambulance agent, police force agent, and fire-brigade agent) work collaboratively for rescue operations. Here, the role of police force agents is crucial, as they clear blocked roads to allow the other agents to access disaster sites.

A road clearance scenario is constructed (see Fig. 1) where the arena size is $(10\,\mathrm{m} \times 10\,\mathrm{m} \times 2\,\mathrm{m})$ and the borders are denoted by green light. The obstacles are represented by circular disk (colored green) and rectangular box (colored red) while the robots are shown in blue. The goal is to shift all the obstacles from road (gray) to the sides of the road (white).

A *LED* light is kept on top of each object for detection. When a foot-bot robot comes within a range of an object, the colored_blob_omnidirectional camera detects the *LED* light. Based on the color of the *LED* light, the robot can determine the type of the object. We took 5 robots and 4 different types of obstacles in the environment that were placed randomly for each experiment. The 6-dimensional capability vectors required for executing the tasks are given in Table 3.

Table 3. Capability vectors of the tasks

Tasks	Capability vector of the task
#1 (large disk)	$[20, 10, 0, 30, 0, 0]$
#2 (large box)	$[10, 0, 0, 0, 0, 10]$
#3 (small disk)	$[10, 20, 0, 0, 0, 0]$
#4 (small box)	$[0, 0, 0, 0, 20, 10]$

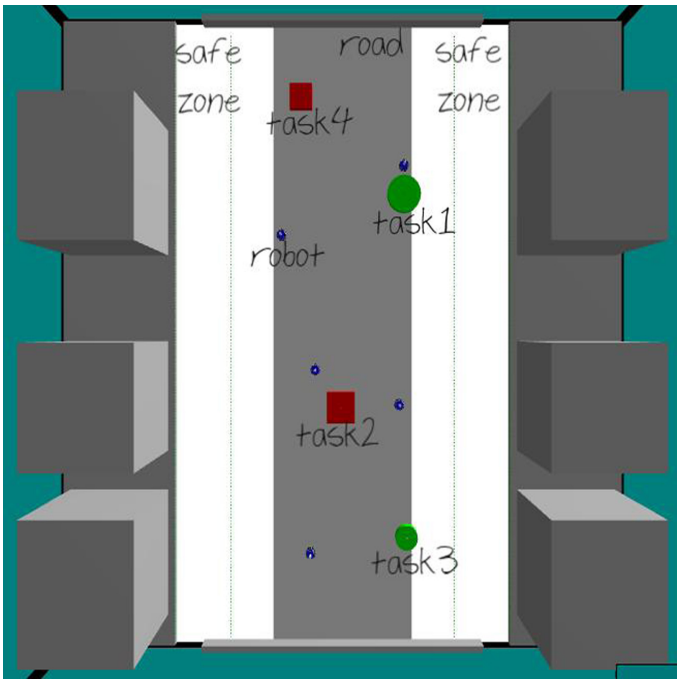
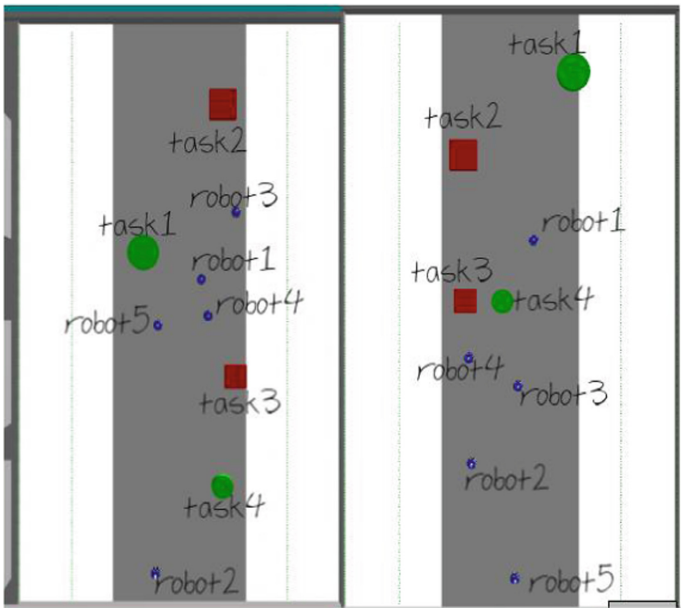


Fig. 1. A snapshot of the environmental setup of dimension (10 m × 10 m × 2 m) (Color figure online)

Table 4. Capability vector and other detail of the robots

Robot	Capability vector of the robot	c	θ'
r_1	$[20, 0, 0, 0, 10, 0]$	60	0.6
r_2	$[0, 20, 0, 0, 0, 10]$	40	0.4
r_3	$[10, 20, 0, 10, 0, 0]$	80	0.5
r_4	$[10, 0, 0, 0, 10, 10]$	60	0.8
r_5	$[0, 0, 10, 20, 0, 0]$	80	0.7

The tasks $task1$, $task2$, $task3$, and $task4$ are represented with large disk, large box, small disk, and small box respectively. The capability vector and other details of each robot are given in Table 4; θ is taken as 1.0 for each robot.



(a) (b)

Fig. 2. Initial setting for experiments 1 and 2 of road clearance environment

In the environment, the robots start exploring the road to search for obstacles. The initial positions of the robots and tasks are shown in Figs. 2-a. and 2-b for two different experiments. If a robot finds an obstacle (see Fig. 3) and it cannot execute it alone, it starts the coalition formation process. The robots who are selected in the coalition reach the location of the obstacle (see Fig. 3). Then they grab the obstacle and make the final orientation for moving. They synchronize themselves and finally move with obstacle toward the white space and place the obstacle there (see Fig. 4). Then the coalition is dissolved. This process continues until all the obstacles are removed from the road.

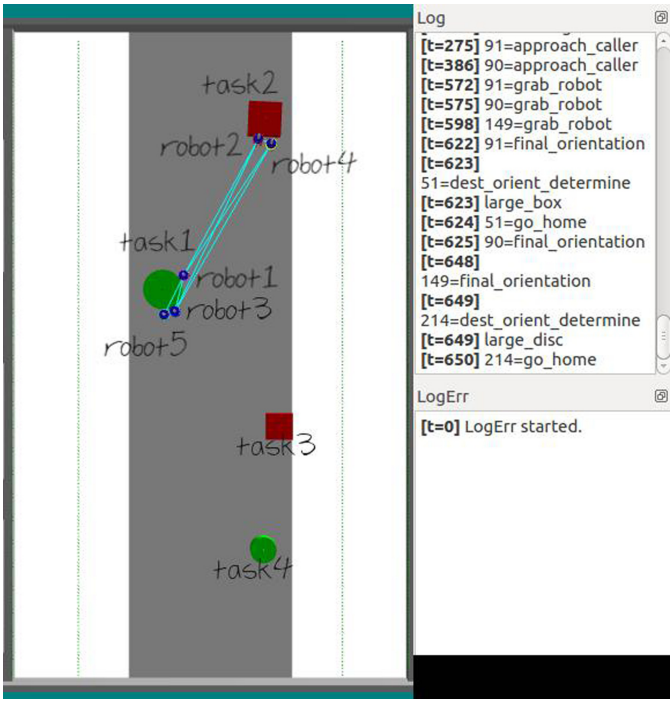


Fig. 3. r_3 and r_2 detect $task1$ and $task2$ respectively and form their coalitions

The experimental results are given in Tables 5, 6. The results indicate that an optimal coalition size cannot be decided in advance in a dynamic environment. For example, when the $task1$ is initialized with different robots, different coalitions are formed. The time to execute a task is the sum of the time taken for task detection, coalition formation, and placing an obstacle to a side of the road. The total time taken to execute all the tasks in experiments 1 and 2 are 1078.2 s and 1120.4 s respectively. The difference in time is due to the initial settings of the experiments—Fig. 2.

The $task1$ is initialized by robot r_3 in experiment 1; by r_1 in experiment 2 (see Tables 5 and 6). It can be seen that the same task, say ($task1$), is executed by different coalitions $[r_1, r_3, r_5]$ and $[r_1, r_2, r_3, r_5]$, in experiments 1 and 2 respectively. It is clear from these results that the initiator as well as the members of a coalition are decided at run-time.

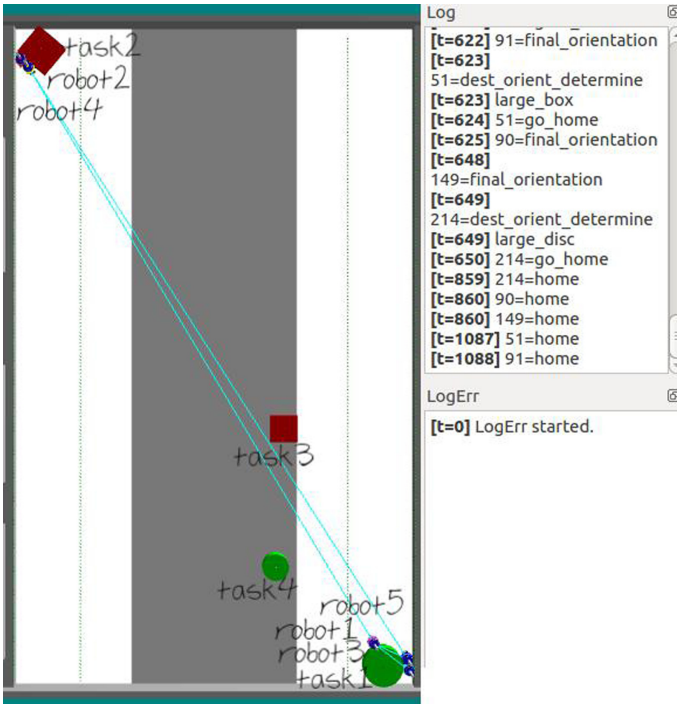


Fig. 4. Robots drop the obstacles on a side of the road

Table 5. Results of experiment 1 (road clearance environment)

Tasks	Task's initiator	Coalition	Cost
#1	r_3	$\{r_1, r_3, r_5\}$	230.00
#2	r_4	$\{r_2, r_4\}$	140.00
#3	r_2	$\{r_2, r_3\}$	125.00
#4	r_1	$\{r_1, r_2, r_4\}$	171.60

Table 6. Results of experiment 2 (road clearance environment)

Tasks	Task's initiator	Coalition	Cost
#1	r_1	$\{r_1, r_2, r_3, r_5\}$	275.00
#2	r_4	$\{r_1, r_4\}$	130.00
#3	r_2	$\{r_2, r_3\}$	121.00
#4	r_1	$\{r_1, r_4\}$	145.60

6 Conclusion

In this paper, we considered collaborative task execution in a dynamic environment, where the location and time of task arrival, the number of robots and location of the robots are not known in advance. A formal framework for cooperative task execution is provided. The distributed approach suggested in this paper considered quantitative representation of capability, and so it generalizes previous works that considered binary representation of capabilities.

A RoboCupRescue simulation environment is taken for implementing the distributed approach using ARGoS–a multi-robot simulation environment. Several experiments were carried out. The results are quite encouraging and it demonstrates how robots are executing the tasks and that different coalitions are formed in different simulation runs for the same task. Implementation of the distributed framework is quite challenging, and that is part of our future work.

Acknowledgments. The authors thank the anonymous reviewers of ICCSA 2021 for their valuable suggestions. The second author was in part supported by a research grant from Google.

References

1. Tadokoro, S., et al.: The robocup-rescue project: a robotic approach to the disaster mitigation problem. In: ICRA 2000, pp. 4089–4094 (2000)
2. Nath, A., Arun, A.R., Niyogi, R.: An approach for task execution in dynamic multirobot environment. In: Mitrovic, T., Xue, B., Li, X. (eds.) AI 2018. LNCS (LNAI), vol. 11320, pp. 71–76. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03991-2_7
3. Nath, A., Arun, A.R., Niyogi, R.: A distributed approach for road clearance with multi-robot in urban search and rescue environment. Int. J. Intell. Robot. Appl. **3**(4), 392–406 (2019). https://doi.org/10.1007/s41315-019-00111-5
4. Nath, A., Arun, A.R., Niyogi, R.: DMTF: a distributed algorithm for multi-team formation. In: ICAART 2020, vol. 1, pp. 152–160 (2020)
5. Nath, A., Arun, A.R., Niyogi, R.: A distributed approach for autonomous cooperative transportation in a dynamic multi-robot environment. In: SAC 2020, pp. 792–799 (2020)
6. Vig, L., Adams, J.A.: Multi-robot coalition formation. IEEE Trans. Rob. **22**(4), 637–649 (2006)
7. Pinciroli, C., et al.: ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. Swarm Intell. **6**(4), 271–295 (2012)
8. Rohmer, E., Singh, S.P., Freese, M.: V-REP: a versatile and scalable robot simulation framework. In: IROS 2013, pp. 1321–1326 (2013)
9. Sokolov, M., Lavrenov, R., Gabdullin, A., Afanasyev, I., Magid, E.: 3D modelling and simulation of a crawler robot in ROS/Gazebo. In: Proceedings of the 4th International Conference on Control, Mechatronics and Automation, pp. 61–65 (2016)
10. Agmon, N., Stone, P.: Leading ad hoc agents in joint action settings with multiple teammates. In: AAMAS 2012, pp. 341–348 (2012)

11. Gaston, M.E., DesJardins, M.: Agent-organized networks for dynamic team formation. In: AAMAS 2005, pp. 230–237 (2005)
12. Hemapala, M., Belotti, V., Michelini, R., Razzoli, R.: Humanitarian demining: path planning and remote robotic sweeping. Ind. Robot Int. J. **36**(2), 146–156 (2009)
13. Okimoto, T., Ribeiro, T., Bouchabou, D., Inoue, K.: Mission oriented robust multi-team formation and its application to robot rescue simulation. In: IJCAI 2016, pp. 454–460 (2016)
14. Lappas, T., Liu, K., Terzi, E.: Finding a team of experts in social networks. In: SIGKDD 2009, pp. 467–476 (2009)
15. Abdallah, S., Lesser, V.: Organization-based cooperative coalition formation. In: International Conference on Intelligent Agent Technology, pp. 162–168 (2004)
16. Coviello, L., Franceschetti, M.: Distributed team formation in multi-agent systems: stability and approximation. In: CDC 2012, pp. 2755–2760 (2012)
17. Tošić, P.T., Agha, G.A.: Maximal clique based distributed coalition formation for task allocation in large-scale multi-agent systems. In: International Workshop on Massively Multiagent Systems, pp. 104–120 (2004)
18. Gunn, T., Anderson, J.: Dynamic heterogeneous team formation for robotic urban search and rescue. J. Comput. Syst. Sci. **81**(3), 553–567 (2015)
19. Gerkey, B.P., Mataric, M.J.: Sold!: auction methods for multirobot coordination. IEEE Trans. Robot. Autom. **18**(5), 758–768 (2002)
20. Kong, Y., Zhang, M., Ye, D.: An auction-based approach for group task allocation in an open network environment. Comput. J. **59**(3), 403–422 (2016)