Multi-agent Goal Delegation Using Theory of Mind

Venkatsampath Raja Gogineni[*] GOGINENI.14@WRIGHT.EDU
Sravya Kondrakunta[†] SRAVYA1@STOLAF.EDU
Michael T. Cox[*] MICHAEL.COX@WRIGHT.EDU
[*]Department of Computer Science & Engineering, Wright State University, Dayton, OH 45431 USA

[†]Department of Mathematics, Statistics and Computer Science, St. Olaf College, Northfield, MN 55057 USA

Abstract

Autonomous agents in a multi-agent system coordinate to achieve specific goals. However, in a partially observable world, current multi-agent systems are often less effective in achieving their goals. In much part, this limitation is due to an agent's lack of reasoning about other agents and their mental states. Another factor is the agent's inability to share required knowledge with others and the lack of explanations in justifying the reasons behind the goal. This paper addresses these problems by presenting a general approach for agent goal management in unexpected situations. In this approach, an agent applies two main concepts: goal reasoning - to determine what goals to pursue and share; and theory of mind - to select an agent(s) for goal delegation. Our approach presents several algorithms required for goal management in multi-agent systems. We demonstrate that these algorithms will help agents in a multi-agent context better manage their goals and improve their performance. In addition, we evaluate the performance of our multi-agent system in both a marine survey domain and in a rover domain. Finally, we compare our work to different multi-agent systems and present empirical results that support our claim.

1. Introduction

Humans work effectively in teams. They reason about people, share their knowledge and cooperate to achieve goals simultaneously. Similarly, autonomous agents in a multi-agent environment should work with each other to improve their efficiency and effectiveness. For our purposes, we define a multi-agent system as a combination of autonomous agents in an environment. Often, multi-agent systems function much better than a single agent system because of their ability to achieve assigned tasks. Besides task achievement, multi-agent systems can also share and delegate goals among themselves to overcome problems or quickly achieve existing goals. In a dynamic world, several problems can occur concurrently thus requiring autonomous agents to generate goals in response. However, given limited resources, a single agent cannot always respond to new problems and still achieve its current goals. Whereas in a multi-agent system, agents can collectively use their resources in delegating goals to others. To effectively delegate their goals, agents must reason about other agents' knowledge, select an agent to which the goal should be delegated, and share knowledge about the problem with the other agent. Acquiring such experiences improves the agents' own knowledge about other agents, which helps the agent better delegate goals in the future.

Much of the work in the multi-agent systems literature focuses on a centralized network of agents (Levesque et al., 1990; Tambe & Zhang, 2000). The focus of this paper, however, will be to work toward a decentralized multi-agent system and to develop a communication framework between the agents to better manage their goals with little human intervention. The communication framework must assist the agents in coordinating with other agents that are not necessarily designed to work together. The agents in such a system must be able to function effectively even with the introduction of new agents into the world. They must be able to learn quickly about the new agent and must be able to share and delegate goals effectively to the newer agent with little human intervention. This paper focuses on developing a solution to the problem using concepts of *Multi-agent goal reasoning* and *Theory of mind*. Multi-agent goal reasoning helps an agent to determine what goals to share and pursue among all its goals. This choice helps an agent conserve its resources and improve its performance in the real world. Theory of mind helps an agent infer other agents' beliefs, goals, and intentions. This capability gives an agent the ability to reason about sharing its goals with the right agent(s).

The hypothesis we put forth is that goal delegation using theory of mind is essential for effective goal management in multi-agent systems. This research, with the help of the concepts above, will develop a computational framework which can be used in multi-agent systems research. The work will also make use of an existing cognitive architecture called *Meta-cognitive Integrated Dual-cycle Architecture (MIDCA)*(Cox et al., 2016) for implementation purposes.

The remainder of the paper is as follows. Section 3 talks about the concept of Multi-agent goal reasoning and our approach towards goal delegation when an agent has limited resources to achieve its goals. It also talks about our approach towards accepting/rejecting delegated goals by receiving agent. Section 4 presents our Theory of Mind approach towards agent selection for goal delegation and knowledge sharing between the delegating and receiving agent. Section 5 describes the Marine Survey domain and the Rover domain used in this research. This section will describe the agent's capabilities, goals, and the unexpected events they might encounter in the respective domains. Furthermore, it also presents the experimental design and empirical results across the respective domains. It begins by introducing different multi-agent systems to compare our approach. The section then demonstrates domain descriptions, experimental designs and empirical results across the domains to test our hypothesis. Section 6 discusses related research. Finally, Section 7 presents the closing remarks and future research directions.

2. Research Problem and Example

Most of the current multi-agent systems (Dias et al., 2004; Fierro et al., 2002; Kalra et al., 2005; Stone & Veloso, 1998) take agents for granted in an environment. They often assume that the agents are either cooperative or competitive in solving problems (i.e., achieving goals). Cooperative agents (e.g., Jan't Hoen et al. (2005)) are designed to achieve their common goals. Although there is some disagreement about the definition of competitive agents, we can agree that they act selfishly to achieve their own goals. Such characteristics make coordination difficult when they are not designed to work together. Moreover, much of the multi-agent work deals with multiple agents that cooperatively search through the problem space in coming up with a plan to solve a problem (Torreño et al., 2014, 2017). This approach requires combined effort and resources from all the agents to plan for a common goal. However, this is not ideal for every task. So, our research problem is to determine how an autonomous agent in a multi-agent environment can coordinate with any agent to manage its goals while conserving resources. In this section, we demonstrate the research problem through an example in a complex naval domain.

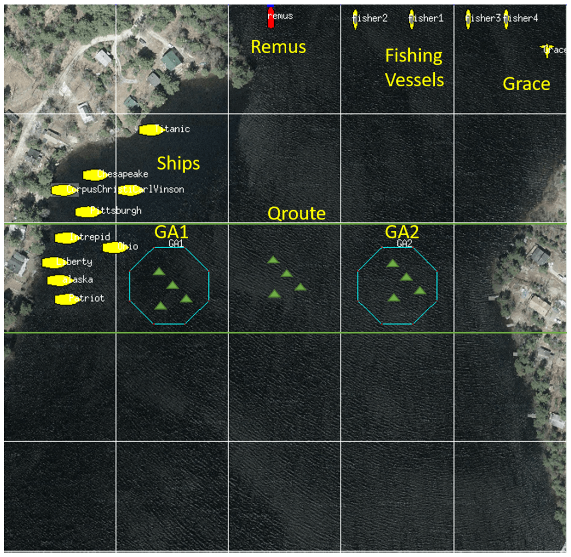

Figure 1. Simulation of the underwater mine clearance domain in Moos IvP. The Q-route extends from the left to the right side of the map and represents the path that ships (10 in yellow shown) will traverse. Grace (upper right in yellow) is a glider to survey the entire region for fish and the fishing vessels (4 in yellow shown) will catch fish in the region. The Remus AUV (red) must attempt to clear mines in green areas GA1 and GA2 to support the goals of ships reaching shore. Unexpected mines also exist within the route, and the AUV must delegate the goals.

Figure 1 shows an underwater mine clearance domain (Gogineni et al., 2018; Kondrakunta et al., 2018). The Q-route (Li, 2009) indicated by the parallel green lines in the figure, is an area through which ships transport their cargo. The blue octagons GA1 and GA2 (Green areas) are areas where mines are expected to be present and the green triangles represent these mines. The environment is also divided into a 5x5 grid consisting of 25 cells. Four different types of agents exist in this domain, and each have their own goals. Remus is an autonomous underwater vehicle whose goals are to survey certain regions (i.e., GA1 and GA2) to make the Q-route void of mines. Grace is a glider whose goals are to search certain cells in the grid to find concentrations of fish. Fishing vessels (3) have goals to catch fish; and ships (10) transport freight from the left end of the Q-route to the other end. While everyone pursues their individual goals, Remus encounters mines between GA1 and GA2 which it did not expect to be present. So it generates a new goal to search the area between GA1 and GA2 to find new mines. However, the Remus does not have enough resources (time) to achieve all these goals. If it cannot achieve all its goals within a certain time, ships traversing the Q-route might hit a mine and damage the shipments. Furthermore, underwater communication incurs a heavy cost.

In this scenario, it is an open question how the Remus can achieve all its goals. If it tries to delegate its goals to other agents, it is not clear with whom it should coordinate. Furthermore, it is unclear how to coordinate to keep the communication cost to a minimum. In critical missions such as these, addressing such issues is essential. We present an approach to address them in the next section and revisit this example throughout this paper.

3. Multi-agent Goal Reasoning

Goal reasoning is a viable computational component in managing an agent's goals (Aha, 2018; Roberts et al., 2018). In the event of unexpected events, it enables an agent to perform different goal operations. Such goal operations will help the agent make rational decisions as a response to these exogenous events. In this research, we leverage the implementation of several goal operations but focus more on goal delegation. Furthermore, in a dynamic environment where unexpected situations occur, an agent should often respond to the developing problems. An intelligent agent should generate new goals in response to the developing problems. However, with limited resources, an agent cannot always pursue its new goals and still achieve its current goals. So, to balance this conflict an agent can decide to delegate some of its goals to other agents. To perform goal delegation, an agent needs to recognize *when* to delegate its goals and determine *what* goals to delegate.

3.1 Goal Delegation

The primary step involved in a goal delegation process is to recognize the need for delegation. We use goal specific resource estimation and priority functions (Gogineni et al., 2020) to select the goals an agent can achieve, and delegate the remaining goals in the order of maximum priority.

Table 1 shows an algorithm that returns a set of delegated goals g_d when there is a need for goal delegation. *DetectDelegation* takes the following inputs: goal agenda of the current agent $(\hat{G}_c = \{g_1, g_2, ...g_c, ...g_m\})$ and current state of agent's perception (s_{cc}). While there are enough estimated resources $\hat{R}(s_{cc})$ in the world (line 2), the agent tries to compute the order of goals it can achieve $g_{achievable}$. Such a goal ordering is possible by applying goal-specific priority $\hat{P}(g, s_{cc})$ and resource estimation functions $\hat{R}(g, s_{cc})$ (Gogineni et al., 2020). The agent then selects a set of goals g_s which has maximum priority (line 3) and minimum resource consumption (line 4). Later, it computes the estimated resources to achieve selected goals g_s (line 5). Furthermore, the agent adds the selected goals to its achievable goals $g_{achievable}$ (line 6). The agent then continues executing these steps until all the goals in its goal agenda are ordered (line 7). Finally, it computes the goals it must delegate (g_d) by subtracting the goals in its agenda with the goals it can achieve (line 9). The function finally returns the set of delegated goals (line 10).

To understand the application of the above goal delegation algorithm, let us revisit the example scenario from section 2. In this example, Remus has achieved its goal to clear mines in GA1. While pursuing the goal to clear mines in GA2 by transiting from GA1 to GA2, it encounters mines between the two areas and formulates a new goal to clear mines in GA3, as shown in figure 2.

Table 1. A method for detecting goal delegation by the current agent ($agent_c$). Parameter $\hat{G}_c$ is the goal agenda of the current agent, and s_{cc} is the known observed state of $agent_c$.

```
DetectDelegation ($\hat{G}_c, s_{cc}$)
1.  $g_{achievable} \leftarrow \emptyset$
2.  $\hat{r} \leftarrow \hat{R}(\hat{G}_c, s_{cc})$                    // Estimation of agent's resources
3.  $while\ \hat{r} > 0\ do$                          // Loop until the agent has enough resources
4.      $g_s \leftarrow \underset{g \in (\hat{G}_c - g_{achievable})}{argmax} \hat{P}(g, s_{cc})$   // Select goals with maximum priority
                                                   // From the goals with the same priority,
5.      $g_s \leftarrow \underset{g \in g_s}{argmin}\ \hat{R}(g, s_{cc})$            select goals with minimum resources
6.      $\hat{r} \leftarrow \hat{r} - \hat{R}(g_s, s_{cc})$                 // Estimate the remaining resources
7.      $g_{achievable} \leftarrow g_{achievable} \cup g_s$          // Add selected goals to agent's achievable goals
8.      $if\ (\hat{G}_c - g_{achievable})\ is\ \emptyset\ then$      // Break, if agent can achieve all its goals
9.          $break$
10. $g_d \leftarrow (\hat{G}_c - g_{achievable})$               // Goals agent cannot achieve
11. $return\ g_d$                                   // Return delegated goals
```

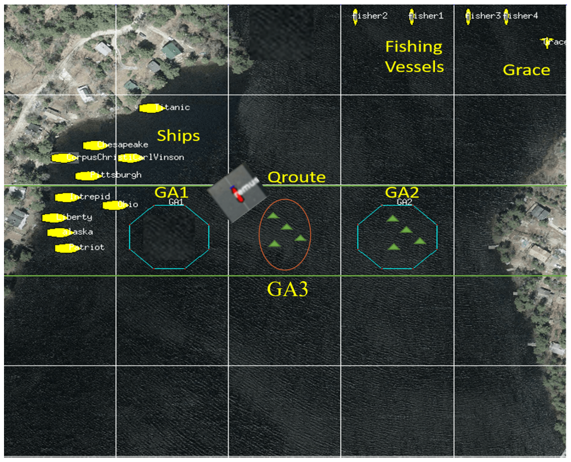

Figure 2. Simulation of the underwater mine clearance domain. Remus (red) choose to pursue clearing mines in GA3 and decides to delegate its goal to clear mines in GA3 to other agents in the environment.

Following the algorithm, the agent anticipates that it does not have enough resources (time) to achieve its goals to clear mines in GA2 and GA3. Although both goals have the same priority, clearing mines in GA3 takes less resources than in GA2. So, the agent decides to clear mines in GA3 and delegates the goal to clear mines in GA2 to other agents in the environment.

In this sub-section, we have seen how the agent uses DetectDelegation algorithm to determine when and what goals to delegate. The following section describes how a receiving agent should respond to the goal requests.

3.2 Goal Acceptance and Goal Rejection

Often an autonomous agent in a multi-agent system cannot accept all requested goals. It is limited by its resources and must reason about its current goals while accepting or delegating the remaining ones. We use DetectDelegation algorithm from section 3.1 to identify the goals the agent cannot pursue. Then, the agent rejects the requested goals that are unachievable and accepts the remaining goals.

Table 2 shows an algorithm that returns the goals the agent can achieve given the agent's goal agenda $\hat{G}_i$, current state s_{ci} and delegated goals g_d. A requested agent $agent_i$ estimates the goals it cannot achieve $g_{unachievable}$ using the *MonitorDelegation* algorithm from section 3.1 (line 1). Later $agent_i$ adds delegated goals g_d to its goal agenda $\hat{G}_i$ if they are not in unachievable goals $g_{unachievable}$ (line 2 and line 3). Finally this returns the list of goals $agent_i$ can achieve (line 4) while rejecting the remaining goals.

Table 2. A method to accept/reject a delegated goal by the selected agent $agent_i$. Parameter $\hat{G}_i$ is the goal agenda of the selected agent, s_{ci} is the known observed state of $agent_i$, and g_d is the set of goals delegated by the current agent $(agent_c)$ to the selected agent $(agent_i)$.

```
GoalAcceptReject ($\hat{G}_i, s_{ci}, g_d$)        // $agent_i'$s goal agenda, its current state and delegated goals
1.  $g_{unachievable} \leftarrow DetectDelegation(\hat{G}_i \cup g_d, s_{ci})$   // Obtain unachievable goals
2.  $if\ g_d \notin g_{unachievable}$                         // If delegated goals are achievable
3.      $\hat{G}_i \leftarrow \hat{G}_i \cup g_d$                          // Add delegated goals to goal agenda
4.  $return\ \hat{G}_i$
```

From the example in the previous section, let us assume that Remus delegates its goals to clear mines in GA2 to Grace. Since Grace is an underwater glider and cannot clear mines, following the GoalAcceptReject algorithm, the requested goal remains in the list of unachievable goals. Therefore, Grace rejects the requested goal. In the next section, we will see how goal sharing can help Remus and Grace achieve all the goals.

In this section, we have seen using DetectDelegation algorithm, when an agent should delegate its goals and what it should delegate. Furthermore, using GoalAcceptReject, we have also seen how a receiving agent should respond to the requested goals. In the following section, we will look at the theory of mind approach to whom an agent should delegate its goals.

4. Theory of Mind

In a distributed multi-agent environment, an individual agent often acts on its own. It is not feasible for the agent to know all the information about other agents' goals and current states. So to delegate goals, the agent is limited by its knowledge about other agents. Theory of Mind refers to reasoning about other agents' mental states using the agent's knowledge about other agents.

In our multi-agent system, we developed a theory of mind approach to select an agent among all the other agents in the environment for goal delegation. This approach is represented in section 4.1. In addition to the agent selection approach and agent should also share required knowledge with the selected agent. Such knowledge can help the selected agent to successfully achieve the delegated goals. This knowledge sharing algorithm is represented in the section 4.2.

4.1 Agent Selection

A delegating agent must select another agent capable of achieving its delegated goals among all the other agents in the environment. Selecting an agent must use its knowledge about other agents' states, goals, and domain knowledge to simulate their ability to achieve the delegated goal. To perform such an agent selection, we use landmarks (Hoffmann et al., 2004) for the delegated goals and select the agent that takes minimum cost to achieve the first landmark. Landmarks are the states that exist in all possible plans to achieve the goal.

Table 3 represents the algorithm that the current agent $agent_c$ applies to select an $agent_i$ to delegate its goals g_d. The *AgentSelection* algorithm takes the following inputs: all the candidate agents in the environment *CA* = $\{agent_1, agent_2, ..., agent_k\}$, their domain knowledge $(\Sigma_1, \Sigma_2, ..., \Sigma_k)$ where Σ is a state transition system represented as (S, A, γ), $agent_c$'s own domain knowledge Σ_c and set of goals to delegate g_d. $Agent_c$ then computes the landmarks $(L_1, ..., L_m)$ to achieve the delegated goals g_d (line 1). Later, it selects $agent_i$ that has the minimum estimated cost to reach the first landmark (line 2 and line 3). Finally, $agent_c$ makes a delegation request to $agent_i$.

Table 3. A method for selecting an agent ($agent_i$) by the current agent ($agent_c$) to delegate its goals g_d. Parameter *CA* is the set of all candidate agents in the environment, $(\Sigma_1, ..., \Sigma_k)$ are corresponding candidate agents' domain knowledge known to the current agent ($agent_c$), Σ_c is the current agent's domain knowledge, s_{cc} is the known observed state of $agent_c$, and g_d is the set of goals to delegate by $agent_c$.

```
AgentSelection($(CA, (\Sigma_1, ..., \Sigma_k), \Sigma_c, s_{cc}, g_d)$)
1.  $L[1, 2, ..m] \leftarrow Landmarks(\Sigma_c, s_{cc}, g_d)$      // Obtain landmarks to achieve delegated goals
                                                         // Select agent with minimum cost
2.  $agent\_index \leftarrow \underset{j \in CA}{argmin}\ \text{cost}(\hat{\Pi}(\Sigma_j, s_{cj}, L[1])$   // to achieve the first landmark
3.  $agent_i \leftarrow$ CA $[agent\_index]$
4.  $return\ agent_i$
```

To better understand the algorithm, let us revisit the example from the underwater mine clearance domain as shown in the figure 3. Remus decides to delegate the goal to clear mines in GA2. For simplicity, let us make an assumption that Grace is now equipped with mine clearing equipment. To delegate its goals, Remus follows the AgentSelection algorithm to obtain the landmarks for the goal. These landmarks include the state of an agent being at the location and identifying mines at GA2. Remus then estimates the planning cost of every agent based on its knowledge about others and selects the agent Grace to delegate its goals. Grace not only has the capability to identify mines but also takes less time to reach GA2 compared to all the other agents in the environment.

Now that the agent has decided on whom to delegate its goals. In the next section, we will see our theory of mind approach on what knowledge it should choose to share with the receiving agent.

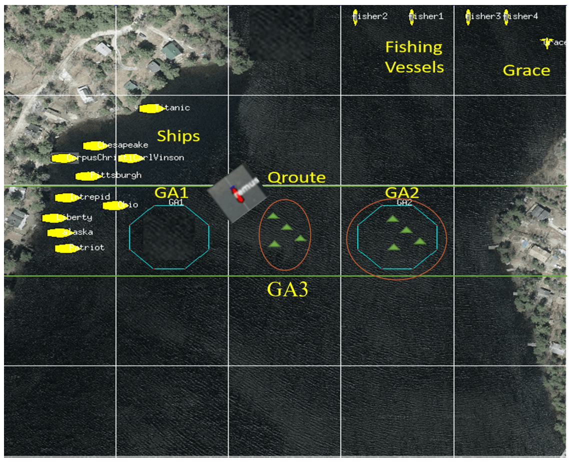

Figure 3. Simulation of the underwater mine clearance domain. Remus (red) choose to pursue clearing mines in GA3 and decides to delegate its goal to clear mines in GA2 to other agents in the environment.

4.2 Knowledge Sharing

Given the selected agent $agent_i$ from section 4.1 and the goals to delegate g_d from 3.1, the delegating agent $agent_c$ should share required knowledge to achieve the goal. However, due to partial observability and the distributed nature of the multi-agent environment, $agent_c$ cannot know what knowledge the other agents necessarily require. However, $agent_c$ should reason using its knowledge to determine some information required by $agent_i$ to achieve the delegated goals. To obtain such information, the agent computes expectations by planning to achieve the first landmark using its own knowledge. These expectations are the future states of the requesting agent when it tries to achieve the goal. Furthermore, the agent then chooses to share all the knowledge related to the attained expectations that it believes $agent_i$ does not know. Such knowledge will help the receiving agent understand the problems it should avoid and the benefits from available opportunities.

Table 4 formally represents the knowledge sharing algorithm which outputs the states (s_{share}) that $agent_c$ should share with $agent_i$. KnowledgeSharing algorithm takes the following input: $agent_i$'s domain knowledge $\Sigma_i = (S_i, A_i, \gamma_i)$, $agent_c$'s known state of $agent_i$ (s_{ci}), $agent_c$'s current state of the world (s_{cc}) and the first Landmark L_1. $agent_c$ plans $\langle a_1, a_2, ..., a_n \rangle$ to achieve the first landmark (line 1). $agent_c$ then computes expectations for every action of the plan (line 5). These expectations comes from the preconditions and the effects of the actions. Later $agent_c$ then tries to retrieve all the states (s_r) that are abstractly related to the computed expectations (s_{ei}) and its current state (s_{cc}) (line 6). Abstractly related states are those states that have similar predicates or contain arguments that exists in both the given state representations. Later, $agent_c$ adds it to the states s_{share} to share with $agent_i$ while removing the states it has already shared s_{ci} (line 7). Finally $agent_c$ updates its knowledge about $agent_i$ (line 8) and returns the states s_{share} to share (line 9).

To understand the KnowledgeSharing algorithm, let us revisit the example from the previous section. Remus chooses Grace to delegate its goal (i.e., to clear mines in GA2). Now it uses the KnowledgeSharing algorithm to share the required knowledge. First, Remus using its own

Table 4. A method for computing the knowledge the current agent ($agent_c$) must share with the delegated agent ($agent_i$). Parameter Σ_i is the known domain knowledge of $agent_i$, s_{ci} is the currently observed state of $agent_i$, s_{cc} is the known observed state of $agent_c$, and $L[1]$) is the first landmark.

```
KnowledgeSharing ($\Sigma_i, s_{ci}, s_{cc}, L[1]$)
1.  $\langle a_1, a_2, ...a_n \rangle \leftarrow \hat{\Pi}(\Sigma_i, s_{ci}, L[1])$        // Estimated actions to achieve first Landmark
2.  $s_{share} \leftarrow \emptyset$                                  // Knowledge states to share with $agent_i$
3.  $s_{ei} \leftarrow \emptyset$                                     // The expected states of $agent_i$
4.  $for\ a\ in\ \langle a_1, a_2, ...a_n \rangle$
5.      $s_{ei} \leftarrow s_{ei} \cup pre(a) \cup a^+ - a^-$     // Expectations of $agent_i$ stemming from the results of action $a$
6.      $s_r \leftarrow AbstractRelatedStates(s_{ei}, s_{cc})$   // Related states of $agent_c$ to expectations
7.      $s_{share} \leftarrow s_{share} \cup (s_r - s_{ci})$          // Add related states to share with $agent_i$
8.  $s_{ci} \leftarrow s_{ci} \cup s_{share}$                            // Update knowledge of $agent_i$
9.  $return\ s_{share}$
```

knowledge about Grace, obtains expectations from its plan to achieve the first landmark (i.e., being at GA2 to identify mines). These expectations include the future state of Grace being at the Q-route (area between the two parallel lines in 3. Since Remus already knows the location of some mines in the Q-route (GA3), it shares those locations with Grace. This information will help Grace traverse those regions carefully or avoid them altogether to stay safe and continue pursuing the delegated goal.

After the requesting agent decides on *when*, *what*, to *whom*, and *how* to delegate its goals, the next step is to coordinate with the receiving agent for successful delegation. To perform this operation, the agent generates the following goal $requested(agent_c, agent_i, g_d, s_{share})$ where $agent_c$ is the requesting agent, $agent_i$ is the receiving agent, g_d is the delegated goal, s_{share} is the information to share with $agent_i$.

The requesting agent $agent_c$ will plan to achieve the goal. It then executes its plan to successfully make the request. Furthermore, the agent waits for an acknowledgement form $agent_i$ before continuing to pursue its own goals.

5. Experimental Design and Evaluation

We have implemented our work in the marine life survey domain and the rover domain to test our hypothesis. The marine life survey domain is a simulated version of surveying underwater marine life in the real world. The Rover domain is one of the classic planning benchmark domains introduced as a simple representation of the NASA Mars Exploration missions. The agents in this domain are equipped with different but possibly overlapping sensors to perform experiments on a planet's surface. While both environments are partially observable and distributed in nature, they are different in many ways. For example, they differ in the goals they pursue, the observations made, the problems that occur, and the challenges they encounter.

This section describes each of these domains in detail and provides experimental evidence to our claims. It also introduces different multi-agent systems in comparison to our work. Furthermore, we have performed experiments with different design parameters across both domains. In the following subsections we will look at these experimental designs and the empirical results in detail.

5.1 Comparison with Different Multi-agent Systems

To evaluate the performances of each of our algorithms introduced in previous sections and to compare them with external multi-agent systems we introduced seven variations. They are as follows,

1. Ideal: Unexpected events (i.e., anomalies) do not occur in the ideal multi-agent system's environment. Everything goes according to the plan, and the agents can achieve their own goals without goal delegation. Having this type of multi-agent system implemented in a domain will help us understand the highest performance the system can perform at any given time. As such, it provides and upper bound on performance.

2. Traditional goal reasoning: In the traditional multi-agent goal reasoning condition, agents encounter anomalies and, in response, they perform relevant goal operations (except goal delegation). These goal operations include goal selection, goal formulation, goal change, and goal achievement. Comparing this multi-agent system with others will enable us to understand the importance of delegating goals to other agents using our theory of mind approach.

3. Delegation (*DetectDelegation*, *AgentSelection*): In the multi-agent delegation condition, agents perform different goal operations in response to the encountered anomalies. Furthermore, they decide when and what goals to delegate using the *DetectDelegation* algorithm (see 1) and finally coordinate them to the agent(s) determined by the *AgentSelection* algorithm (see 3 . Here that the receiving agent will always accept the goals delegated to it. However, it prioritizes achieving its own goals first and only later achieves the delegated goals. This multi-agent condition will show us the impact of the algorithms mentioned above on a traditional multi-agent goal reasoning condition.

4. Accept_reject (*DetectDelegation*, *AgentSelection*, *GoalAcceptReject*): The accept_reject multi-agent condition enhances goal delegation with an additional decision-making ability for the receiving agent. It can either accept or reject the delegated goals as determined by the *GoalAcceptReject* algorithm (see 2. Note that if the receiving agent(s) denies the given goal(s), the requesting agent will then delegate its goals to the next best agent provided by the *AgentSelection* algorithm.

5. Knowledge sharing (*DetectDelegation*, *AgentSelection*, *GoalAcceptReject*, *KnowledgeSharing*): Agents in the knowledge sharing delegation condition do everything similar to the agents in the Accept_reject multi-agent condition. In addition, they share the required knowledge with the receiving agent determined by the *KnowledgeSharing* algorithm (see 4). This multi-agent condition will enable us to evaluate the impact of sharing such knowledge with the receiving agent. Note that the receiving agent(s) will prioritize achieving their own goals first and later pursue the delegated goals.

6. Random (*DetectDelegation*, *RandomAgentSelection*): The multi-agent aandom condition shares the same capabilities as the traditional goal reasoning multi-agent system. However, agents in this multi-agent system will decide when and what goals to delegate using the *DetectDelegation* algorithm and randomly choose an agent to pursue its goals. Comparing this multi-agent system with the goal delegation multi-agent system will help us understand the impact of the *AgentSelection* algorithm on the agents' performance.

7. Auction mechanism (external multi-agent system): The auction mechanism is the most standard agent-coordination approach in the multi-agent systems community. In this multi-agent condition, requesting agents will make their goals available for bidding. Other agents will take the goals and provide their cost to achieve them. The requesting agent will then delegate the goals to the receiving agent with the lowest bid. Note that the agents still use the *DetectDelegation* algorithm to determine when and what to delegate. Furthermore, the receiving agent will still prioritize achieving its own goals over the given goals.

Evaluating the performance of each process in our goal delegation approach (2-5) while comparing them to some external multi-agent systems (6-7) is the primary intent behind the introduction of these eight multi-agent conditions. In the following sections, we will see the implementation of these multi-agent systems in different experimental scenarios of the marine life survey domain and the rover domain.

5.2 The Marine Life Survey Domain

Surveying marine environments using *autonomous underwater vehicles (AUV*s) is often time-limited and challenging. Figure 4 shows the region of Gray's Reef National Marine Sanctuary located on the inner shelf of the South Atlantic Bight off the coast of Savannah, GA. The red area represents the research area and is restricted for public access. In this research-only region, our team regularly deploys AUVs such as custom robotic fish and Slocum gliders to evaluate and test new platforms to perform missions of detecting areas of fish hot spots. During these missions, AUVs typically surface to communicate regularly or respond to forced interruptions. In this area, marine scientists have previously tagged the fish with acoustic transmitters that constantly ping at a pre-determined frequency. These acoustic signals help the AUVs detect fish using their acoustic detection sensors. These AUVs can also classify unique pings, ignoring multiple pings from the same fish.

Before actual deployment, we simulated the marine life survey domain (Kondrakunta et al., 2021) using the Moos-IvP (Benjamin et al., 2010) framework. This simulator replicates the underwater environment close to the real world. For simulation, we have divided a portion of the research area (see lower right of Figure 4) into twenty-five cells. The red dots represent the thousand fish that pings at every seventeen time steps. The highlighted region around the agent is the acoustic fish tag detection sensor. At Gray's Reef, the detection radius varies with environmental conditions, but currently, the simulator assumes it to be ten meters. An agent can identify hot spots based on the number of pings.

5.3 Experimental Design

We have introduced two naturally occurring discrepancies in the marine life survey domain: Remora attacks and flow events. Remora attacks are attachments to the AUV made by sea creatures that hinder movement. They act randomly with a specific rate of occurrence. These Remora attacks negatively affect an agent's speed, and enough attacks could disable an individual platform from moving. An agent can successfully respond to a remora attack by formulating a goal to be free from the organism by gliding backward. Flow events occur at a specific location in the marine life survey domain. These events disable the agent and push them out of the region. In such a case, an agent

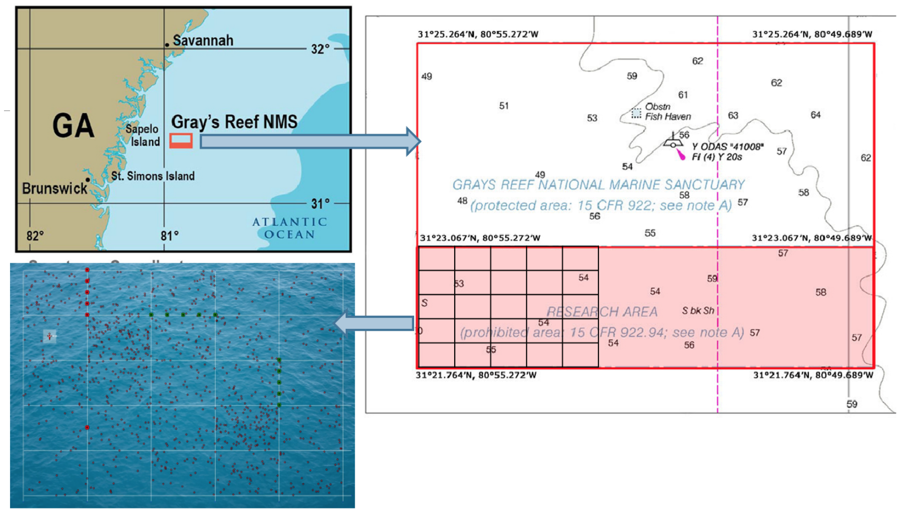

Figure 4. Gray's Reef National Marine Sanctuary is located off the coast of Georgia and contains a research area shown in the insert shaded in pink. Within this, we represent a 5x5 subsection in simulation. This grid contains fish hot-spots that are of interest to marine scientists. The highlighted square around the agent indicates the sensor range for detecting acoustic fish tags (the small red dots).

can only delegate its goals and ask the operator for help to initiate a rescue operation. Note that the agents can only communicate when they surface, and they surface at regular intervals of 10 units in simulation time.

Figure 5 shows the 5x5 region of the marine life survey domain. Three agents exist in this multi-agent system, namely Grace, Franklin, and Remus. All agents are provided with initial goals to survey a specific part of the 5x5 region. For example, Grace is responsible for achieving nine survey goals represented as the blue region in the figure. Similarly, Franklin and Remus have eight goals to survey the green and red areas. Furthermore, as shown in the figure, two specific flow-affected cells are at (2,0) and (0,2). When agents survey these flow-affected cells, they are pushed far from the area and are disabled. However, they can delegate goals and call for help. Similarly, there are randomly occurring Remora attacks, as mentioned above. In addition, nine fish hot spots exist in this scenario, represented as the cells with yellow borders in the figure.

Each trial in the above-mentioned experimental setting has different initial starting positions for the agents. Therefore, we have performed two experiments with varying randomness for each multi-agent condition, and an experiment contains a hundred separate trials. This randomness affects the occurrence of Remora attacks, thereby changing the performance of the multi-agent system as a whole. This experiment aims to measure the performance of a multi-agent system in terms of the percentage of goals achieved and the identification of fish hot spots.

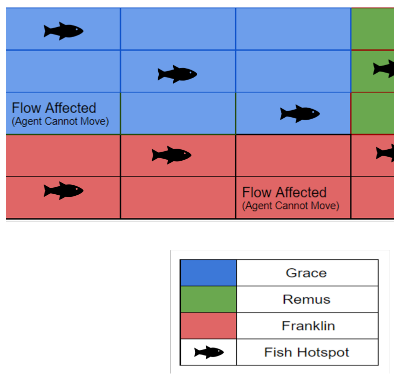

Figure 5. Experimental design to evaluate eight different multi-agent conditions in the marine life survey domain (Gogineni et al., 2021). There are three underwater gliders (Franklin, Grace, and Remus) and nine hot spots in the scenario. The hot spot location are evenly distributed between the 3 agents to survey.

5.4 Empirical Results

Figure 6 depicts the results of the eight different multi-agent conditions. The X-axis represents the time taken by the agents in these multi-agent conditions to achieve their goals, while the Y-axis represents the percentage of goals achieved. As mentioned in the previous section, an experiment is run twice with different seed values for every condition and there are hundred initial positions for each experiment. Therefore, every point in the graph averages two hundred trails. Furthermore, every trial has a simulation time limit of six hundred seconds. Note that one unit of simulation time is forty times the actual time in real world. If an agent following our approach (goal delegation using theory of mind) is affected by the flow, it decides to delegates its goals to an agent with the minimum planning cost and shares the affected region's knowledge.

Overall, the results show that the agents in the multi-agent knowledge sharing condition performs better and achieves goals quicker than all the others in the environment except for the agents in an ideal multi-agent condition, which operates in perfect conditions. Nevertheless, the significant performance of the agents in the knowledge sharing multi-agent condition compared to other multi-agent conditions shows that our approach towards goal delegation using theory of mind has proven effective. Note that our experimental setup does not allow agents in any practical multi-agent condition to achieve one hundred percent of the goals. Because trials exist where all agents start at the flow-affected regions, and no agent can achieve any goals.

Next comes the performance of agents in the multi-agent accept_reject condition, which gives the receiving agent the decision-making ability to accept or reject delegated goals. In this experiment, if the requested agent is also at a flow-effected region, it refuses the goal, thereby helping the requesting agent give its goals to the subsequent agents. However, not sharing the requesting agent's knowledge of flow-affected areas keeps it significantly lower than Knowledge sharing.

Agents in the multi-agent auction mechanism performs close to the agents in the accept_reject condition given enough time. Minimum cost bids help the requesting agent(s) determine the most capable agents and avoid agents affected by the flow. However, it takes at least twenty units of time

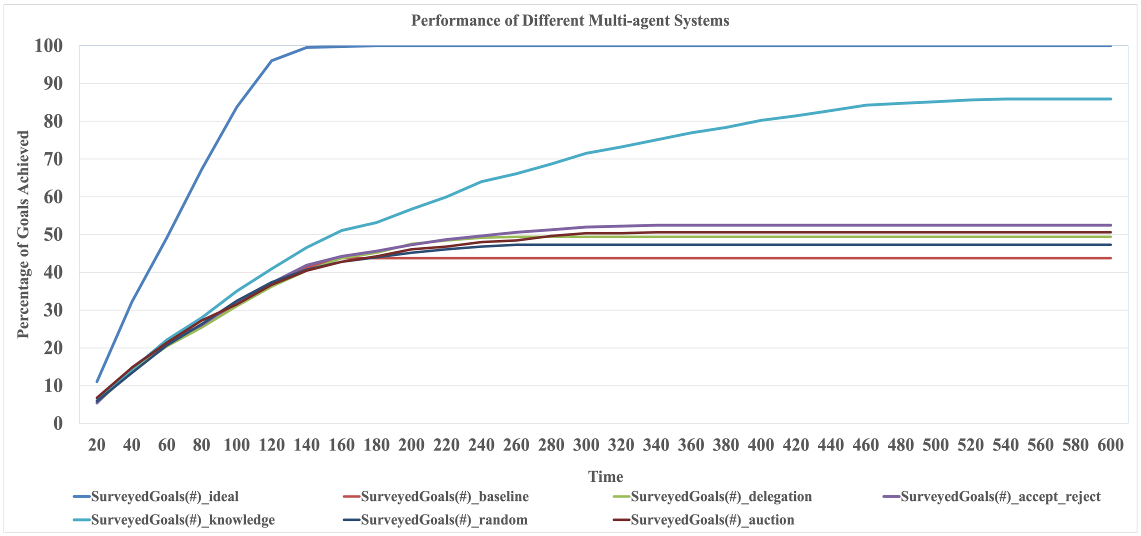

Figure 6. Percentage of goals achieved by agents in different multi-agent conditions in the marine life survey domain. On the X-axis is the time to achieve the goals, and on the Y-axis is the percentage of goals achieved.

for successful bidding. The communication overhead thus keeps its performance significantly lower in the interval 160 to 260 time units.

Finally, the observation that the agents in the delegation multi-agent system (with agent selection capability) performing better than the agents in the random and traditional goal reasoning conditions is self-explanatory.

5.5 The Rovers Domain

The rovers domain is a simplified problem representation of the NASA Mars Exploration Rover missions launched in 2003. As the name suggests, agents in this mission are a collection of rovers equipped with different sensors. They must travel between various waypoints collecting data and transmitting it back to a lander. Typical goals of the agent involve traversing the planet's surface to perform science gathering experiments such as soil sample analysis, rock sample analysis, capturing photographs of different objectives.

These rovers have different capabilities to perform the mission, sometimes overlapping with each other. Several challenges make their goals difficult to achieve. For example, some rovers cannot traverse specific waypoints. Similarly, not all rovers possess the tools to achieve their mission, thus making it a good problem domain for multi-agent coordination. In addition, exogenous events exist that damage the rovers' sensors. Furthermore, data transmission is also constrained by the visibility of the lander from the waypoints. Because of the complexity in the domain, it is one of the benchmark problem domains in the planning competitions held at International Conference on Automated Planning and Scheduling (ICAPS)[1] every two years (Coles et al., 2012).

1. https://www.icaps-conference.org/competitions/

5.6 Experimental Design

For the Rover domain, we have obtained hundred problem sets from the Competition of Distributed and Multi-agent Planners (CoDMAP)[2] track held at the international planning competition. From the hundred, forty problem sets have been actually in the competition to evaluate different planners, while the remaining sixty are from the problem generator used by the competition. Each problem contains a different number of goals and agents. At the start of each trial, we have uniformly and randomly allocated these goals to the agents. However, since every agent does not possess the required tools to achieve their assigned goals, they can delegate them to other agents. In addition, we have introduced events that randomly damage the equipment of these agents. Equipment in this domain includes high and low-resolution cameras and tools to perform soil/rock sample analysis.

Figure 7 shows the distribution of agents from the problem set. On the X-axis are the number of agents and on the Y-axis are the number of problems/trials containing these agents. Our trials include a maximum of ten distributed agents and a minimum of one agent operating in the domain.

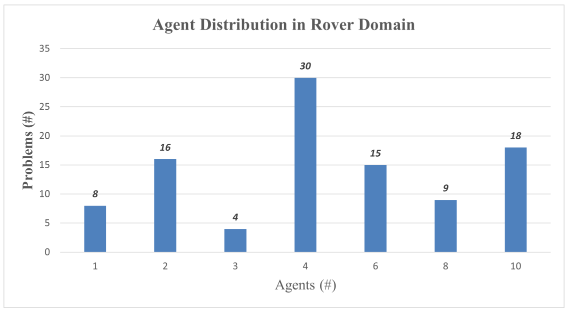

Figure 7. Distribution of agents across the 100 benchmark problems sets in the rovers domain.

For every multi-agent condition described in section 5.1, we have performed three experiments with the same hundred trials mentioned above. However, the experiments vary in the randomness of the initial goals allocated and the occurrence of exogenous events that damage tools. In the next section, we will see how the results obtained are similar to the results from the marine survey domain.

5.7 Empirical Results

Figure 8 depicts the results of the agents in the seven different multi-agent conditions. The X-axis represents the time taken by agents in the the multi-agent conditions to achieve their goals, while the Y-axis represents the percentage of goals achieved. As mentioned in the previous section, a trail is run three times with different seed values for agents in every condition. There are a hundred trials for each experiment, and each trial contains n different number of goals and agents. Therefore, every point in the graph averages three hundred trials. Note that the units for time here are MIDCA cycles. As mentioned previously, MIDCA is a cognitive architecture providing the goal reasoning framework for implementing goal delegation. If an agent following our approach (goal delegation

2. http://agents.fel.cvut.cz/codmap/

using theory of mind) cannot pursue its assigned goals due to the reasons mentioned above, it decides to delegates its goals to an agent with the minimum planning cost. It shares the knowledge that it lost its sensor traversing a waypoint, and explains the motivations behind the goals.

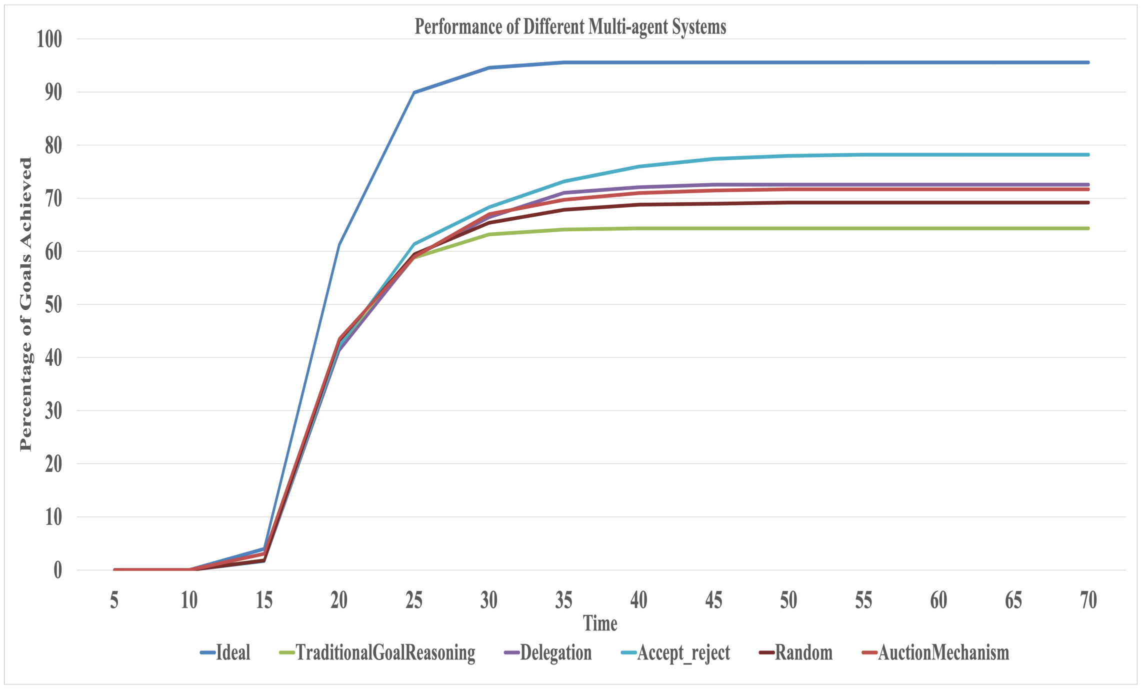

Figure 8. Percentage of goals achieved by agents in different multi-agent conditions in the rovers domain. On the X-axis is the time to achieve the goals, and on the Y-axis is the percentage of goals achieved.

All agents are provided with initial goals at ten simulation time units (See figure 8) and will remain in operation until the deadline of hundred time units. Although the results show a similar pattern compared to the results from the marine life survey domain, a few key observations are to be made.

Agents in the multi-agent ideal condition do not achieve all their goals. Since these benchmark problems evaluate planners, some goals are too hard to solve. In our work, we use the fast-downward planner (Helmert, 2006) to plan for the goals. Note that some problems are too easy to solve for the agents, which is why we see agents in the traditional goal reasoning condition achieving almost 63% of goals in the given time. Furthermore, agents in the delegation condition perform slightly higher than the agents in the auction mechanism condition because of the communication overhead during agent coordination.

Irrespective of the disparities between complex and straightforward problems, agents in the multi-agent knowledge sharing condition achieve close to 90% of the goals proving again the effectiveness of goal delegation using theory of mind. At the same time, agents in the accept_reject condition reaches 79% of goals followed by the performance of agents in the delegation at around 73% of goals proving that goal delegation alone is not sufficient. Finally, agents in the multi-agent auction mechanism achieve around 72% of the goals, and at the lower tier, we have the performances of agents in the random conditions at 67% goal achievement and traditional goal reasoning condition at 63%.

In summary, the results here and those from 5.2 support our main hypothesis that goal delegation using theory of mind is essential for effective goal management in multi-agent systems.

6. Related research

There are three different classifications of systems in which multiple autonomous agents work together (Szymak, 2011). They are: centralized, decentralized and hybrid. Centralized multi-agent systems (Levesque et al., 1990) represent a central architecture between agents, which implies that the agents are assumed to be cooperative and kind during the problem solving process. In contrast, decentralized multi-agent systems (Wooldridge & Jennings, 1999) represent autonomous control of agents, which are assumed to be independent, cooperative or competitive, depending on the situation they experience. Hybrid multi-agent systems represent agents following a combination of both decentralized and multi-agent systems. We are more interested in decentralized multi-agent systems, as they allow other agents to be added to the system easily. For the purpose of this discussion, when we refer to multi-agent systems we are exclusively referencing decentralized systems. Wooldridge and Jennings (Wooldridge & Jennings, 1999) presents a formalized approach for multi-agent systems to pursue goals in a Cooperative Problem-Solving (CPS) Process. However, multi-agent coordination remains a central problem in steps following Recognition.

Goal-driven autonomy and goal reasoning more generally are relatively new areas of research compared to many technical areas of AI, but they are active in the cognitive systems community. Cox (2007) was motivated to find the reason behind the origin of goals. This paper was implemented in the Wumpus world domain. The goal of the agent is to reach a destination while avoiding the pits in the world. Later on, the concept of GDA was incorporated into a cognitive architecture called MIDCA (Paisner et al., 2013). Goal management has been a key focus of goal reasoning research, hence GDA allows the agent to dynamically perform certain goal operations: selection, monitoring, transformation, formulation and several others. Kondrakunta (2017); Kondrakunta & Cox (In Press) presents a goal selection strategy to look at the cost-benefit ratio during goal selection. This work closely aligns with one of our goal selection strategies applied in our work. Although we leverage the implementation of goal operations from the above mentioned works, our prime focus in this paper is to address goal delegation.

Theory of mind is an ability of an agent to infer other agents' goals, beliefs, emotions and intentions. There are mainly two broad theories which are widely accepted from a psychological stand point. Theory Theory (Gopnik & Wellman, 1994), states that children hold a naïve psychological theory to infer goals, beliefs and emotions of others. This knowledge is used to predict behaviors of others. Moreover, as children grow up they develop their psychological awareness to better infer mental states of others. Simulation Theory of Empathy Goldman (1992), states that children simulate the actions of others to predict the behavior of others. Furthermore, there are several hybrid theories that include a part of Theory Theory and Simulation Theory of Empathy, some of which are Intentional stance and Structure-Mapping theory of analogy. Intentional stance (Burke et al., 2001;

Dennett, 1983) is a concept of theory of mind which states that an agent can predict other agent's beliefs and desires given the other agent's purpose and place in the world. Structure-Mapping theory (Gentner, 1983) (SMT) of analogy is a theory of analogy and similarity. SMT focuses on humans' ability to see structural similarities across dissimilar cases. From a computational standpoint, Rabkina et al. (2020) proposed that an agent can better recognize goals of other agents by externally observing their actions using an implementation of Analogical Theory of Mind.

7. Conclusions and Future Research

In this paper, we presented an approach for a distributed multi-agent system. The agents in the system work together when unexpected events happen. They follow a theory of mind approach to delegate goals and share the required knowledge. This paper explicitly develops algorithms to approach goal delegation, agent selection, knowledge sharing, and goal acceptance rejection. These algorithms improve the performance of a multi-agent system when uncertain events are bound to occur, which is often the case in the real world. Furthermore, to support our claims of robustness and generality, we introduced our approach in two different research domains. The data supports our claims that a multi-agent system following our approach outperforms all the real-world multi-agent systems introduced in sub-section 5.

We intend to extend this research to incorporate explanations and goal sharing. Explanations can help the delegating agent to justify the reasons behind the delegated goals to the selected agent but can also play a major role in the agent coordination process. For example, in the case of a selected agent rejecting the delegated goals, explaining the reasons behind the rejection will help improve the delegating agent's agent selection process.

Moreover, in this paper, we assumed that an individual agent is capable enough to achieve its own goals. However, in the real-world an individual agent often requires the help of several other agents to perform a given goal. Such a process is called goal sharing. We want to incorporate goal sharing within our approach by leveraging the concept of *Hierarchical goal networks (HGN)*. HGNs can split a high-level goal into several sub-goals to share with capable agents. Such goal sharing could further improve the performance of a multi-agent system.

In summary, results obtained from both the domains suggest that agents operating in a distributed multi-agent environment can effectively manage their individual and shared goals by following our approach. They decide when and what goals to delegate using our *DetectDelegation* algorithm, whom to delegate using *AgentSelection* algorithm, and *KnowledgeSharing* algorithm.

Acknowledgements

This research was supported by the National Science Foundation under grant 1849131 and by the Office of Naval Research under grant N00014-18-1-2009.

References

Aha, D. W. (2018). Goal reasoning: Foundations, emerging applications, and prospects. *AI Magazine*, *39*, 3–24.

Benjamin, M. R., Schmidt, H., Newman, P. M., & Leonard, J. J. (2010). Nested autonomy for unmanned marine vehicles with moos-ivp. *Journal of Field Robotics*, *27*, 834–875.

Burke, R., Isla, D., Downie, M., Ivanov, Y., & Blumberg, B. (2001). Creature smarts: The art and architecture of a virtual brain. *Proceedings of the Computer Game Developers Conference* (pp. 147–166). Citeseer.

Coles, A., Coles, A., Olaya, A. G., Jiménez, S., López, C. L., Sanner, S., & Yoon, S. (2012). A survey of the seventh international planning competition. *AI Magazine*, *33*, 83–88.

Cox, M. T. (2007). Perpetual self-aware cognitive agents. *AI magazine*, *28(1)*, 32–45.

Cox, M. T., Alavi, Z., Dannenhauer, D., Eyorokon, V., Munoz-Avila, H., & Perlis, D. (2016). MIDCA: A metacognitive, integrated dual-cycle architecture for self-regulated autonomy. *Thirtieth AAAI Conference on Artificial Intelligence* (pp. 3712–3718). AAAI Press.

Dennett, D. C. (1983). Taking the intentional stance seriously. *Behavioral and Brain Sciences*, *6*, 379–390.

Dias, B. M., Zlot, R., Zinck, M., Gonzalez, J. P., & Stentz, A. (2004). A versatile implementation of the traderbots approach for multirobot coordination. *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*.

Fierro, R., Song, P., Das, A., & Kumar, V. (2002). Cooperative control of robot formations. In *Cooperative control and optimization*, 73–93. Springer.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive science*, *7*, 155–170.

Gogineni, V. R., Kondrakunta, S., & Cox, M. T. (2021). Multi-agent goal delegation. *Proceedings of the 9th Goal Reasoning Workshop*.

Gogineni, V. R., Kondrakunta, S., Molineaux, M., & Cox, M. T. (2018). Application of case-based explanations to formulate goals in an unpredictable mine clearance domain. *Proceedings of the ICCBR Workshop on Case-Based Reasoning for the Explanation of Intelligent Systems* (pp. 42–51).

Gogineni, V. R., Kondrakunta, S., Molineaux, M., & Cox, M. T. (2020). Case-based explanations and goal specific resource estimations. *Proceedings of Thirty-Third International Flairs Conference* (pp. 407–412). AAAI.

Goldman, A. I. (1992). In defense of the simulation theory. *Mind & Language*, *7 (1-2)*, 104–119.

Gopnik, A., & Wellman, H. M. (1994). The theory theory. *Mapping the mind: Domain specificity in cognition and culture*, (p. 257–293).

Helmert, M. (2006). The fast downward planning system. *Journal of Artificial Intelligence Research*, *26*, 191–246.

Hoffmann, J., Porteous, J., & Sebastia, L. (2004). Ordered landmarks in planning. *Journal of Artificial Intelligence Research*, *22*, 215–278.

Jan't Hoen, P., Tuyls, K., Panait, L., Luke, S., & La Poutre, J. A. (2005). An overview of cooperative and competitive multiagent learning. *Proceedings of the First International Conference on*

Learning and Adaption in Multi-Agent Systems (pp. 1–46). Springer.

Kalra, N., Ferguson, D., & Stentz, A. (2005). Hoplites: A market-based framework for planned tight coordination in multirobot teams. *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 1170–1177). IEEE.

Kondrakunta, S. (2017). *Implementation and evaluation of goal selection in a cognitive architecture*. Master's thesis, Wright State University.

Kondrakunta, S., & Cox, M. T. (In Press). Autonomous goal selection operations for agent-based architectures. *Proceedings from Advances in Artificial Intelligence and Applied Cognitive Computing*. Las Vegas, NV: Springer.

Kondrakunta, S., Gogineni, V. R., Molineaux, M., Munoz-Avila, H., Oxenham, M., & Cox, M. T. (2018). Toward problem recognition, explanation and goal formulation. *Sixth Goal Reasoning Workshop at IJCAI/FAIM-2018*.

Kondrakunta, S., et al. (2021). The rational selection of goal operations and the integration of search strategies with goal-driven autonomy. *Proceedings of the Ninth Annual Conference on Advances in Cognitive Systems*. Cognitive Systems Foundation.

Levesque, H. J., Cohen, P. R., & Nunes, J. H. T. (1990). On acting together. *AAAI*. Menlo Park, CA.

Li, P.-C. (2009). *Planning the optimal transit for a ship through a mapped minefield*. Master's thesis, Naval Postgraduate School Monterey.

Paisner, M., Maynord, M., Cox, M. T., & Perlis, D. (2013). Goal-driven autonomy in dynamic environments. *In Goal Reasoning: Papers from the ACS Workshop* (pp. 79–94).

Rabkina, I., Kathnaraju, P., Roberts, M., Wilson, J., Forbus, K., & Hiatt, L. (2020). Recognizing the goals of uninspectable agents. *Proceedings of the Eighth Annual Conference on Advances in Cognitive Systems*. Cognitive Systems Foundation.

Roberts, M., Borrajo, D., Cox, M., & Yorke-Smith, N. (2018). Special issue on goal reasoning. *AI Communications*, *31*, 115–116.

Stone, P., & Veloso, M. (1998). Towards collaborative and adversarial learning: A case study in robotic soccer. *International Journal of Human-Computer Studies*, *48*, 83–104.

Szymak, P. (2011). Comparison of centralized, dispersed and hybrid multiagent control systems of underwater vehicles team. *Solid State Phenomena* (pp. 114–121). Trans Tech Publications.

Tambe, M., & Zhang, W. (2000). Towards flexible teamwork in persistent teams. *Autonomous Agents and Multi-Agent Systems*, *3*, 159–183.

Torreño, A., Onaindia, E., Komenda, A., & Štolba, M. (2017). Cooperative multi-agent planning: A survey. *ACM Computing Surveys (CSUR)*, *50*, 1–32.

Torreño, A., Onaindia, E., & Sapena, O. (2014). Fmap: Distributed cooperative multi-agent planning. *Applied Intelligence*, *41*, 606–626.

Wooldridge, M. J., & Jennings, N. R. (1999). The cooperative problem-solving process. *Journal of Logic and Computation*, *9(4)*, 563–592.