Proceedings of the Ninth Goal Reasoning Workshop Submitted 10/2021; published 11/2021

Multi-agent Goal Delegation

Venkatsampath Raja Gogineni GOGINENI.4@WRIGHT.EDU
Sravya Kondrakunta KONDRAKUNTA.2@WRIGHT.EDU
Michael T. Cox MICHAEL.COX@WRIGHT.EDU
Department of Computer Science & Engineering, Wright State University, Dayton, OH 45431 USA

Abstract

Autonomous agents in a multi-agent system work with each other to achieve their goals. However, In a partially observable world, current multi-agent systems are often less effective in achieving their goals. This limitation is due to the agents' lack of reasoning about other agents and their mental states. Another factor is the agents' inability to share required knowledge with other agents. This paper addresses the limitations by presenting a general approach for autonomous agents to work together in a multi-agent system. In this approach, an agent applies two main concepts: goal reasoning- to determine what goals to pursue and share; Theory of mind-to select an agent(s) for sharing goals and knowledge. We evaluate the performance of our multi-agent system in a Marine Life Survey Domain and compare it to another multi-agent system that randomly selects agent(s) to delegates its goals.

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 to improve their efficiency and effectiveness. We define a multi-agent system as an interacting combination of autonomous agents in an environment for our purposes. Often, multi-agent systems function much better than single-agent systems because of their ability to achieve assigned tasks. Besides task achievement, multi-agent systems can also share and delegate goals to overcome problems or quickly achieve existing goals. Several problems can occur concurrently in a dynamic world, 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 delegate goals to others. To effectively delegate their goals, agents must reason about other agents' knowledge, select an agent to delegate goals, and share knowledge about the problem with the other agent. Acquiring such experiences improves the agents' knowledge about other agents, which helps the agent better delegate goals in the future.

The remainder of the paper is as follows. Section 2 describes the concept of goal-driven marine autonomy, a framework we used to integrate high-level and low-level autonomy mechanisms. 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. Section 4 presents our Theory of Mind approach towards agent selection for goal delegation and knowledge sharing between the del-

egating and receiving agent. It also talks about our approach towards accepting/rejecting delegated goals by receiving agent. Section 5 describes the Marine Life Survey domain Kondrakunta et al. (2021) followed by experimental design in section 6. The next section 7 presents the evaluation of our multi-agent system with another multi-agent system which delegates goals to a randomly selected agent(s). Section 8 discusses related research. Finally, Section 9 presents the closing remarks and future research directions.

2. Goal-Driven Marine Autonomy (GDMA)

The objective of *goal-driven marine autonomy (GDMA)* is to integrate the high-level abstract reasoning of cognitive systems with the preciseness of control theory for complex continuous domains. Such an integration enables the building of robust IPS for applications in dynamic and unpredictable marine environments. We exploit results from a research area called *goal-driven autonomy (GDA)* that focuses on managing the goals of cognitive systems and merge them with specifics of robotic control theory in what we term *control-driven autonomy (CDA)* (see Figure 1).

With respect to actions, GDA involves goal management and strategies using goal predicate structures and hierarchical task network plans. CDA involves task net optimization and path planning using waypoints and motion commands. With respect to perception, GDA involves problem recognition and state interpretation using problem schemas and explanation patterns. CDA involves sensor fusion and communication generation using vector field models and grid maps. Currently, we have developed the top and bottom layers shown in this figure and await future research to provide the mechanisms of managing uncertainty with belief spaces (but see Lin et al. (2020) for preliminary work toward this goal). Furthermore, we have yet to deploy platforms with GDA in field trials. Instead, our AUVs employ CDA. Thus, when we use the term "agent," we are referring to GDA and CDA in simulation. When eventually validated and fielded, the resulting GDMA agent aboard an AUV will instantiate our initial IPS.

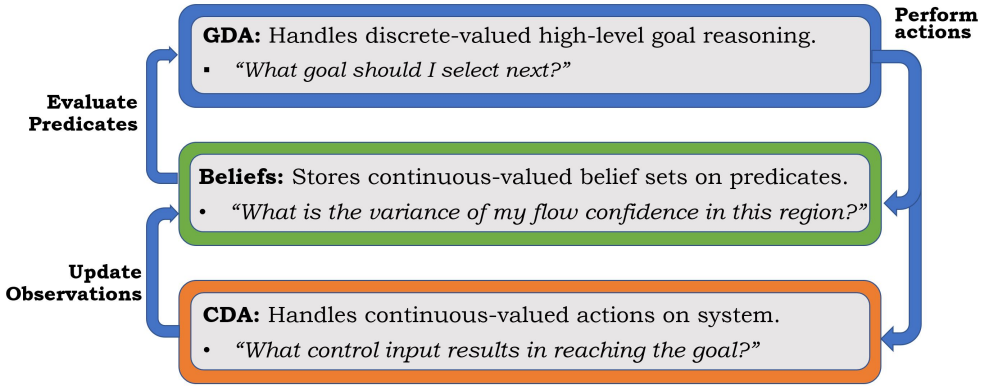

Figure 1. A multilayer approach to goal-driven marine autonomy. At the highest level, GDA performs goal operations given beliefs about the environment. At the lowest level, control manages the actuators and updates continuous sensor values. Mediating between the two, sets of possible states are mapped to discrete predicates and actions are translated to control parameters.

3. Multi-agent Goal Reasoning

Autonomous agents must be able to function effectively even with the introduction of new agents into the world. The agents 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. 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. The following section 3.1 introduces goal delegation and its need in multi-agent systems.

3.1 Goal Delegation

In a dynamic environment where unexpected situations occur, an agent should often respond to the developing problems. Therefore, an intelligent agent should generate goals in response to these developing problems. However, with limited resources, an agent cannot pursue its new goals and still achieve its current goals. So, to achieve all its goals, an agent should delegate some of its goals to other agents.

The primary step involved in a goal delegation process is to recognize the need for delegation. 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).

4. Theory of Mind

In a distributed multi-agent environment, often an individual agent 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 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. Furthermore, following this approach also aids the delegating agent in sharing 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

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. $g_s \leftarrow \underset{g \in (\hat{G}_i - g_{achievable})}{argmax} \hat{P}(g, s_{cc})$ *// Select goals with maximum priority*
   // From the goals with the same priority,
5. $g_s \leftarrow \underset{g \in g_s}{argmin}\ \hat{R}(g, s_{ci})$ *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*

4.1 Agent Selection

In a multi-agent dynamic environment, 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 these landmarks. Landmarks are the states that exist in every plan to achieve the goal.

Table 2 represents the algorithm that the current agent $agent_c$ applies to select an $agent_i$ to delegate its goals g_d. *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 function 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 $L[1..m]$ 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, $agent_c$ delegates its goals to $agent_i$.

Table 2. 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*
2. $agent\_index \leftarrow \underset{j \in CA}{argmin}\ \text{cost}(\hat{\Pi}(\Sigma_j, s_{cj}, L[1])$ *// Select agent with minimum cost to achieve first landmark*
3. $agent_i \leftarrow CA\ [agent\_index]$
4. $return\ agent_i$

4.2 Knowledge Sharing

Given the selected agent $agent_i$ from section 4.1 and the goals to delegate g_d 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 require. However, $agent_c$ should reason using its knowledge to determine the information required by $agent_i$ to achieve the delegated goals.

Table 3 represents the algorithm that 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 $agent_c$ 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 the 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).

Table 3. 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. $\quad s_{ei} \leftarrow s_{ei} \cup pre(a) \cup a^+ - a^-$ *// Expectations of* $agent_i$ *stemming from the results of action* a
6. $\quad s_r \leftarrow AbstractRelatedStates(s_{ei}, s_{cc})$ *// Related states of* $agent_c$ *to expectations*
7. $\quad s_{share} \leftarrow s_{share} \cup (s_r - s_{ci})$ *// Add related states to share with* $agent_i$
8. $s_{ci} \leftarrow s_{ci} \cup s_{share}$ *// Update knowledge of* $agent_i$
9. $return\ s_{share}$

4.3 Goal Acceptance and Goal Rejection

An autonomous agent in a multi-agent system cannot accept all the delegated goals. It is often limited by the resources and must reason about its goals while rejecting or delegating the remaining goals.

Table 5 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. $agent_i$ 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 returns the list of goals $agent_i$ can achieve (line 4) while rejecting the remaining goals.

Table 4. Goal Acceptance and Goal Rejection

Table 5. 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.	$g_{unachievable} \leftarrow MonitorDelegation(\hat{G}_i \cup g_d, s_{ci})$ *// 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.	$return\ \hat{G}_i$

5. The Marine Life Survey Domain

Consider the problem of time-limited surveys of marine environments with *autonomous underwater vehicles (AUV*s). Typical missions are to take various readings such as salinity, temperature, and pressure throughout the water column and to investigate key aspects of marine life. An important feature within a marine ecosystem is the presence of *hot spots* or regions of high fish density. These areas and the aquatic pathways between them that fish transit represent areas of ecological sensitivity. Thus, discovering the location of major hot spots, especially for endangered species, is an important application of GDMA. However, many barriers exist in such environments that make mission success difficult. Sea creatures may attach themselves to platforms and slow progress. Tides and currents exist that also impede progress and obstacles may appear requiring course change.

Our research team regularly deploys AUVs such as Slocum gliders and custom robotic fish for the purpose of making such marine surveys and to test and evaluate new platforms. During missions, the platforms surface to communicate on regular schedules or in response to forced interrupts. We use such vessels to explore a region dedicated to research in Gray's Reef National Marine Sanctuary located on the inner shelf of the South Atlantic Bight off the coast of Savannah, GA (see Figure 2). The restricted area contains fish previously tagged with acoustic transmitters that send an acoustic signal or 'ping' at a pre-determined frequency depending on the application, typically 5 minutes for short experiments or every 30-180 minutes for longer-term tracking. AUVs can detect these fish if the ping falls within a platform's acoustic detection radius and can hear multiple pings from the same fish (i.e., it can classify unique pings).

We implemented a simulator for this domain to test search techniques prior to actual deployment and to empirically evaluate the mechanisms discussed in this paper. The lower right of Figure 2 shows the portion of the research area modeled by the simulator and split into 25 cells. One cell is shown in the lower left of the figure. The red dots depict 1000 fish (currently assumed to be static) that emit a ping every 17 time steps. In this cell, a hot-spot is located near the co-ordinate (6,14). The blue dot represents a simulated AUV controlled by an agent, and the blue circle represents the receiver detection radius. At Gray's Reef, the detection radius varies with environmental conditions,

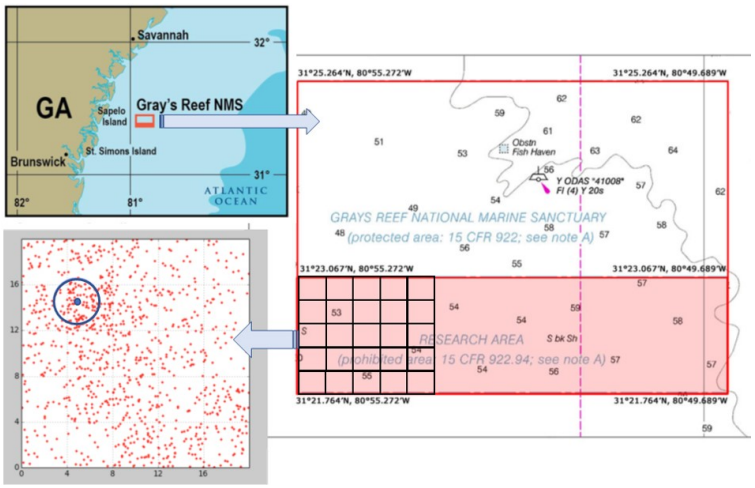

Figure 2. Gray's Reef National Marine Sanctuary is located off the coast of Georgia and contains a research area shown in the insert shaded in pink. Within this, we represent a 5x5 subsection. This grid contains fish hot-spots that are of interest to marine scientists, here within the cell at location (6,14). The agent (indicated by the blue dot) is also in this cell. The circle around the agent indicates the sensor range for detecting acoustic fish tags (the red dots).

but currently the simulator assumes it to be 2 co-ordinate units. As mentioned, an agent can identify hot-spots based on the number of pings.

6. Experimental Design

As previously mentioned, agents in our multi-agent system identify when to delegate goals, select agent(s) to delegate goals, share required knowledge with the selected agent, and finally decide on goal acceptance or rejection. In this paper, we refer to this multi-agent system as the SMART.

We introduced two naturally occurring discrepancies in the Marine Life Survey Domain: Remora attacks and Flow attacks. Remora attacks hinder the agents' movement, which acts randomly with a specific rate of occurrence. These Remora attacks negatively affect an agent's speed, and enough attacks could altogether disable the movement of an individual agent. An agent can successfully respond to a Remora attack by formulating a goal to glide backward. Flow attacks occur at a specific location in the Marine Life Survey Domain. These Flow attacks disable the agent and push them out of the region. In such a case, an agent can only delegate its goals and asks the operator for help to initiate a rescue operation.

Figure 3 shows the 5x5 region of the Marine Life Survey Domain in which SMART operates. There are three agents in this multi-agent system, namely Grace, Franklin, and Remus. All these 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 each to survey the green and red regions. Furthermore, as shown in the figure, two specific flow-affected cells are at (2,0) and (0,2) in the region. When

agents survey these flow-affected regions, they are pushed far from the region and are disabled. However, they can delegate goals and call for help. Similarly, there are randomly occurring Remora attacks, as mentioned above.

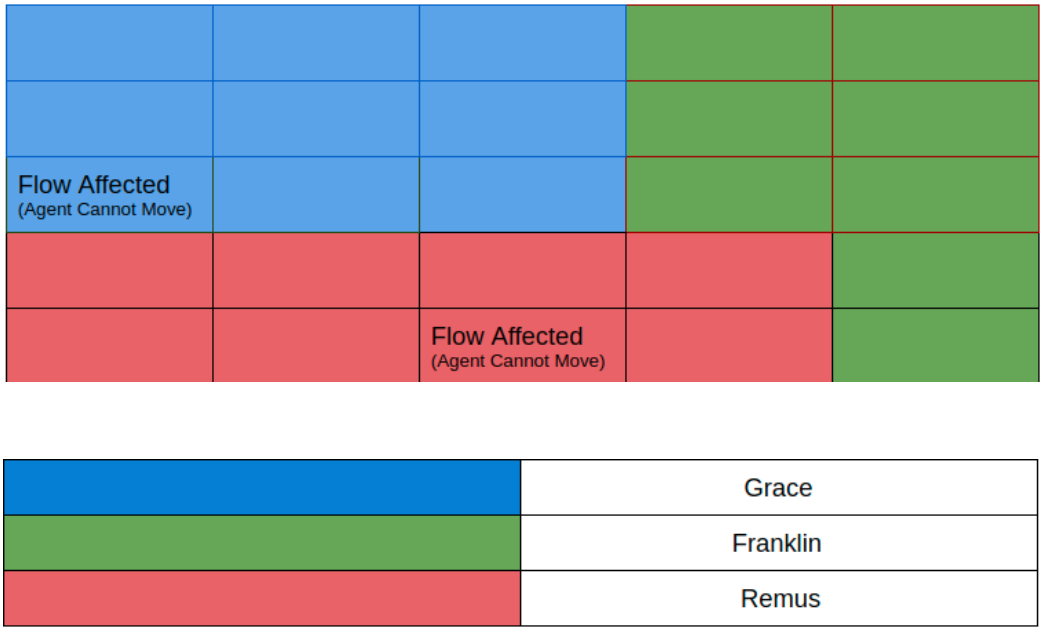

Figure 3. Experimental Design to evaluate a multi-agent system

Each trail in the above-mentioned experimental setting has different initial starting positions for the agents. We have performed fifty trials for each experiment and repeated the experiment three times with different random seed values. This randomness affects the occurrence of Remora attacks, thereby changing the performance of the multi-agent system as a whole. Section 7 presents the results and evaluates the performance of SMART in comparison with other multi-agent systems.

7. Empirical Results

To evaluate SMART in an experimental setting mentioned in the previous section, we introduced another multi-agent system called RANDOM. Although RANDOM and SMART are similar in identifying goals to delegate, they differ in agent selection, knowledge sharing, and goal acceptance/rejection. In RANDOM, the delegating agent randomly selects an agent to delegate its goals. Furthermore, knowledge sharing is non-existent in RANDOM, while the receiving agent always accepts the delegated goals. Furthermore, we also introduced another multi-agent system called IDEAL to see the ideal performance of the agents in the absence of discrepancies and the need for delegation.

Figure 4 depicts the results of SMART, RANDOM, and IDEAL multi-agent systems. The X-axis represents the time taken by the multi-agent systems to achieve their goals, while the Y-axis represents the percentage of goals achieved. As mentioned in the previous section 3, for every multi-

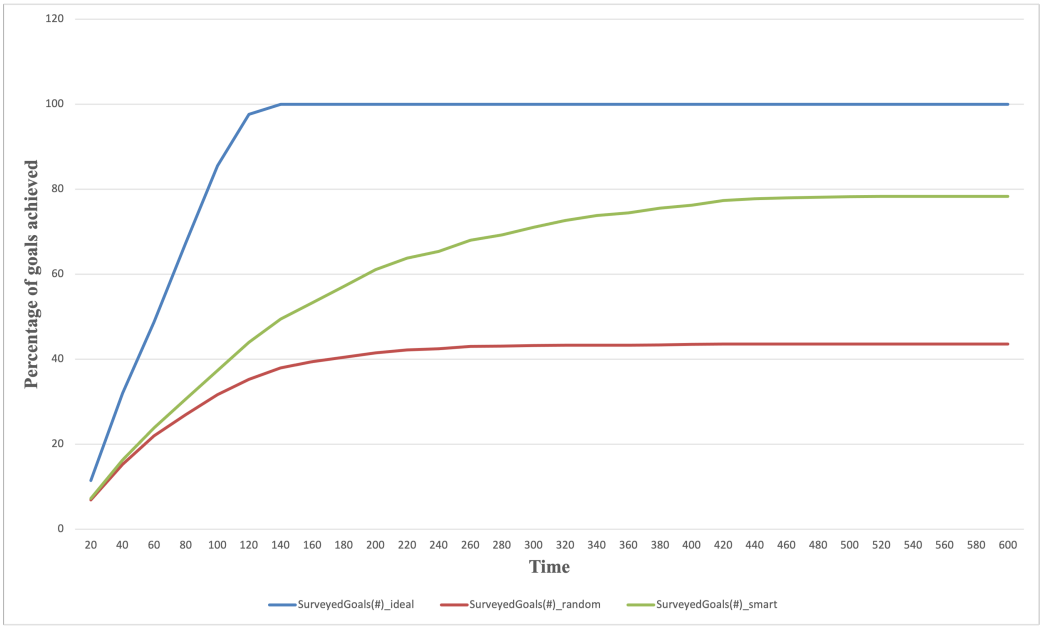

Figure 4. Performance of the SMART and the RANDOM multi-agent systems.

agent system, an experiment is run three times with different seed values. There are fifty trails for each experiment, and each trail contains different initial starting locations of the agents. Therefore, every point in the graph is an average of one hundred and fifty trails. Furthermore, every trial has a simulation time limit of six hundred units.

The results overall show that SMART outperforms RANDOM. Agents in the RANDOM multi-agent system delegate goals to a randomly selected agent, do not share flow-affected regions' knowledge with others, and always accept the delegated goals. However, following this approach, an agent affected by flow delegates its goals to a random agent and makes it vulnerable to the flow-affected region as it does not share its knowledge with the random agent. For this reason, the RANDOM multi-agent system can only achieve forty percent of its goals even if it has enough time to achieve its goals.

Similarly, if an agent in the SMART multi-agent system is affected by the flow, it delegates its goals to an agent with minimum planning cost, shares the affected region's knowledge, and decides to achieve or reject the goal. For this specific reason, the receiving agents will be aware of the flow affected region and completely avoids it. For these specific reasons, the SMART multi-agent system can perform around eighty percent of goals within 450 units of time. Furthermore, given unlimited time, the SMART agent cannot achieve a hundred percent of goals because of our experimental settings. For example, the agents can never survey the two flow-affected regions. Moreover, since the agents' initial starting positions are random, two or more agents start in a flow-affected region.

The IDEAL multi-agent system is devoid of discrepancies (Flow and Remora attacks). Its agents can all achieve their respective goals in a span of three hundred and forty units of time without the

need for delegation. Since ideal environmental conditions do not commonly exist in the real world, our multi-agent system performs well.

8. 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 represent (Wooldridge & Jennings, 1999) 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. To recognize and achieve a multi-agent problem, this approach follows a four-step process. Namely: Recognition, to recognize if an agent cannot solve a problem individually; Team formation, to select a group of agents that can be expected to solve the problem; Plan formation, to come up with a plan that is agreed between the agents to solve the problem (involves negotiation for agreement between agents); and Team action, to perform the actions by the agents. 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) and later on the work was extended (Paisner et al., 2014) into arsonist domain. 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.

Similarly Dannenhauer et al. (2019) introduced the idea of goal monitors. An agent creates rules as preconditions to monitor goals. If the preconditions are satisfied then the agent switches its goal or drops the goal. This paper did not consider the problem of selecting goals when there are multiple goals to achieve. Moreover, the preconditions are mostly rule based rather than any kind of functional estimation, which is often a problem when the agent has very limited resources at hand. Kondrakunta (2017) also presents an initial implementation of goal change using predicate transformations. The authors presented the idea to implement other goal operations like change and formulation. A formalism of how to implement the two operations was outlined in the paper.

Although we leverage the implementation of goal operations from the above mentioned works, our prime focus in this paper is to address goal delegation. GDA is also implemented in a other architectures, one such cognitive architecture is ARTUE (Klenk et al., 2013), which presents the performance variation of the ARTUE architecture with both benefits and limitations.

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) propose that an agent can better recognize goals of other agents by externally observing their actions using an implementation of Analogical Theory of Mind. This implementation involves retrieving mapped structures of internal knowledge about the observable actions, thereby predicting the goal using the retrieved structures. The internal knowledge is therefore trained.

9. Conclusions and Future Research

In this paper, we presented a distributed multi-agent system. The agents in the multi-agent 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 two discrepancies (i.e., Remora and Flow). The data supports our claims that a multi-agent system following our approach outperforms a random multi-agent system in a dynamic environment.

We intend to extend this research to include concepts of explanations, Usurpation, and goal sharing. Explanations help the delegating agent to justify the reasons behind the delegated goals to the selected agent. Such reasoning will help the selected agent understand the priority of the delegated goals. Furthermore, 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.

In this paper, we only talked about uncertain events that negatively affect agent's resources. However, real-world opportunities can occur, which positively affects the agent's resources. In such

a case, an agent that can quickly achieve its goals can also ask to achieve other agents' goals, thereby improving the multi-agent system's performance. Such a process is called Usurpation.

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

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

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.

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

Dannenhauer, Z., Molineaux, M., & Cox, M. T. (2019). Explanation-based goal monitors for autonomous agents. *Advances in Cognitive Systems* (pp. 1–6). Cognitive Systems Foundation.

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

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

Gogineni, V. R., Kondrakunta, S., Molineaux, M., & Cox, M. T. (2020). Case-based explanations and goal specific resource estimations. *The Thirty-Third International Flairs Conference*.

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).

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

Klenk, M., Molineaux, M., & Aha, D. W. (2013). Goal-driven autonomy for responding to unexpected events in strategy simulations. *Computational Intelligence*, *29*, 187–206.

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., Cox, M. T., Coleman, D., Tan, X., Lin, T., Hou, M., Zhang, F., McQuarrie, F., & Edwards, C. R. (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* (p. in press). Cognitive Systems Foundation.

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

Lin, T. X., Hou, M., Edwards, C. R., Cox, M., & Zhang, F. (2020). Bounded cost htn planning for marine autonomy. *Proceedings of Global Oceans 2020: Singapore – U.S. Gulf Coast* (pp. 1–6). Los Alamitos, CA: IEEE.

Paisner, M., Cox, M. T., Maynord, M., & Perlis, D. (2014). Goal-driven autonomy for cognitive systems. *Proceedings of the Annual Meeting of the Cognitive Science Society* (pp. 2085–2090).

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.

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.

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