Towards a Robust MultiAgent Autonomous Reasoning System (MAARS): An Initial Simulation Study for Satellite Defense

Jun-young Kwak* and Milind Tambe†

University of Southern California, Los Angeles, CA, 90089, United States

Paul Scerri‡

Carnegie Mellon University, Pittsburgh, PA, 15213, United States

Onur Sert§ and Amos Freedy¶

Perceptronics Solutions, Inc., Sherman Oaks, CA, 91423, United States

Multi-agent autonomous reasoning systems have emerged as a promising planning technique for addressing satellite defense problems. The main challenge is to extend and scale up the capabilities of current and emerging reasoning and planning methods to handle the characteristics of the satellite defense problem. This paper focuses on some key critical research issues that need to be addressed in order to perform automated planning and execution fitted to the specific nature of response to ASAT attacks, and provides MAARS, a new autonomous reasoning framework for satellite defense. As the core of MAARS, we present MODERN, a new execution-centric method for DEC-POMDPs explicitly motivated by model uncertainty. There are two key innovative features in MODERN: (i) it maintains an exponentially smaller model of other agents' beliefs and actions than in previous work and then further reduces the computation-time and space expense of this model via bounded pruning; and (ii) it reduces execution-time computation by exploiting BDI theories of teamwork, and limits communication reasoning to key trigger points. We demonstrate a proof of concept of MAARS in the simplified ASAT mitigation scenario. We then show initial evaluation results of MAARS in ASAT domains that are critical in advancing the state-of-the-art in providing autonomous reasoning to delve into unperceived models as well as deal with exponential explosion of the computational complexity of current algorithms.

Nomenclature

t	Time step
$\mathbf{b}$	Belief states of the team
$\mathbf{a}$	Joint action
s	State
U_{C}	Expected utility when to communicate
U_{NC}	Expected utility when not to communicate
V	Value function
$\mathbf{a}^*$	Most likely action
Θ	Most likely observation sequence
π	Given policy
σ	Communication cost
α	Degree of model uncertainty

Subscript

i	Agent index

*PhD Student, Computer Science Department, junyounk@usc.edu, AIAA Student Member.

†Professor, Computer Science Department, tambe@usc.edu.

‡Associate Research Professor, The Robotics Institute, pscerri@cs.cmu.edu.

§Lead Electrical Engineer, onurs@percsolutions.com.

¶President, afreedy@percsolutions.com.

$-i$ Set of agent indices except i

I. Introduction

Multi-agent autonomous reasoning systems have emerged as a promising planning technique for addressing satellite defense problems.[1–4] In such domains, researchers have been focused on rapid response planning algorithms for on-board autonomous systems in the real-time satellite defense environment.[5–7] Despite their NEXP-complete policy generation complexity,[8] *Distributed Partially Observable Markov Decision Problems* (DEC-POMDPs) have been used to tackle real-world multiagent collaborative planning problems under transition and observation uncertainty.[9–14] DEC-POMDPs are able to quantitatively express action and observational uncertainty, and yet optimally plan communications and domain actions.

Our approach focuses on some key critical research issues that need to be addressed in order to perform automated planning and execution fitted to the specific nature of response to ASAT attacks. The main challenge is to extend and scale up the capabilities of current and emerging reasoning and planning methods to handle the characteristics of the satellite defense problem. The critical aspect of reacting to anti-satellite (ASAT) attacks includes uncertainty over the model caused by lack of information, varied defense/offense strategies (which leads to large search space during reasoning), and robust communication reasoning. In particular, we often only have an approximate model of agent observation or transition functions, and thus computing optimal plans whose promised quality may not be realized in practice is not well-justified given model uncertainty.

Unfortunately, past work[9–14] has two major weaknesses to handle realistic ASAT domain problems. First, they assume full accuracy of the provided models, such as the transition and observation probabilities; but, such models are rarely accurate in real-world settings. Specifically, in the multi-counter ASAT domain, obtaining such accurate models for agent teamwork and coordinated action that we focus on is extremely difficult. As such, it is difficult to justify pouring computational resources into such careful planning over the entire set of the team's joint belief states, as the resulting policy is definitely not guaranteed to be optimal or even near-optimal given that our models are inaccurate. Second, they maintain the entire set of the team's joint belief states, a costly undertaking that is not well-justified given model uncertainty. And they reason about the right action and communication before each decision step at execution-time relying on the entire team beliefs, leading to inefficient computation.

This paper provides a new autonomous reasoning framework for satellite defense called MAARS (MultiAgent Autonomous Reasoning System). As the core of MAARS, we present a new execution-centric framework for DEC-POMDPs called MODERN (MOdel uncertainty in Dec-pomdp Execution-time ReasoNing). MODERN is the first execution-centric framework for DEC-POMDPs explicitly motivated by model uncertainty. It is based on two key ideas. First, MODERN reasons with an exponentially smaller model of other agents' beliefs and actions than the entire set of joint beliefs as done in previous work;[15,16] then it further reduces the computation time and space expense of this model via bounded pruning. Second, MODERN reduces execution-time computation by: (i) engaging in decision-theoretic reasoning about communication only at *Trigger Points* instead of every agent reasoning about communication at every step, only agents encountering trigger points perform such reasoning; and (ii) utilizing a pre-planned policy for actions that do not involve interactions, avoiding on-line planning at every step. Our approach has significant advantages in domains with interaction-sparseness.

We first demonstrate a proof of concept of MAARS in the simplified ASAT mitigation scenario to understand dimensions and limitations of problem. We then experimentally evaluate a new algorithm for DEC-POMDPs with model uncertainties and illustrate its initial algorithmic advances over previous work. It is critical in advancing the state-of-the-art in distributed autonomous reasoning to delve into unperceived models as well as deal with exponential explosion of the computational complexity of current algorithms.

II. Anti-SATellite (ASAT) Domain

The globalization of space technology as well as widespread knowledge of space systems expose space systems and satellites to foreign attacks. Anti-SATellite (ASAT) weapons are of varied complexity in design and build. They range from kinetic and chemical interceptors, conventional guns, and low power lasers, to more complex nuclear and radio frequency weapons, and finally to sophisticated high-energy lasers and particle beam weapons. The effects of even a limited deployment of ASAT weapons on essential satellites can be quite broad, as seen in case of the test of an ASAT weapon in January 2007.[17] Although there are no simple or complete solutions to the satellite vulnerability problem, there are courses of action and available technologies that could be used to counter these threats to space capabilities.

A major challenge in the engagement of counter measures is the rapid configuration of an optimal response plan and real time execution of the defense mission. Figure 1 provides a layout of the domain that MAARS system will operate in.

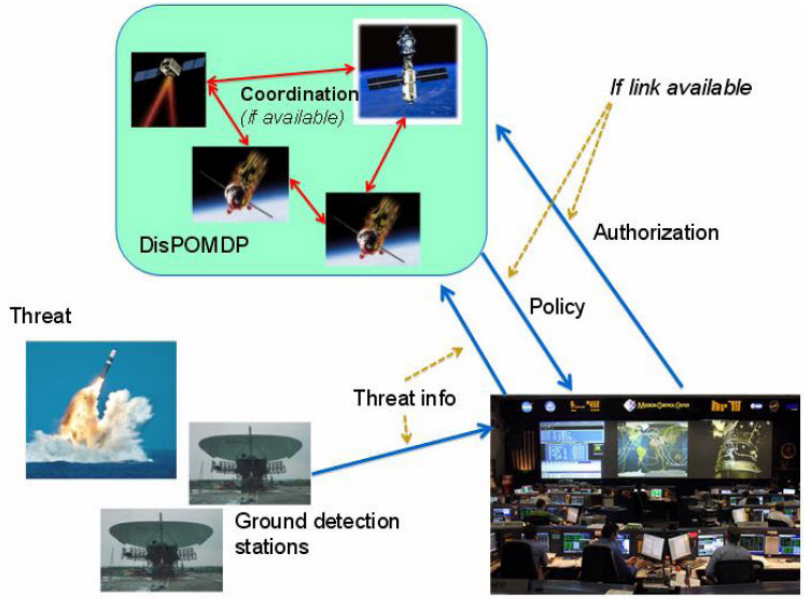


Figure 1. MAARS Operation Domain

Our proof of concept is centered on a scenario where MAARS is responsible for defending an orbiting asset from an anti-satellite missile attack. We describe more details about the scenario below.

A. Variable Characteristics

Our intent is to analyze the collection of instances of scenarios with varying characteristics given our overall scenario description. The overall scenario consists of several variable characteristics, each of which is fixed for any instance of the scenario. These include communication interruptions aiding the ASAT attack (either links with the ground or between defending satellites), the number of attacking missiles, margin maintenance requirements (e.g., we might need to ensure that we can intercept a subsequent missile with a certain amount of confidence), and what defensive assets are available (e.g., counter-ASAT missiles and guidance/navigation interference technologies).

B. Assumptions

To focus our efforts on our scope of work, we are making several assumptions. We assume that ASAT launch detection is not in scope, but response to the detection is in scope. We do not concern ourselves with the specific counter-ASAT technology, instead we focus on being able to achieve appropriate configurations and use abstract models of the effectiveness and cost of use of the ASAT technology. In the same vein, observation and detection of the effectiveness of counter-ASAT is out of scope, while reasoning about the statistics concerning observation of the effectiveness of counter-ASAT is in scope.

C. Communication

Communication is the central issue to counter-ASAT response. It is assumed that many ASAT threats will include some capability to destroy or degrade communications between satellites in space and communications between satellites in space and controllers on the ground. It is also assumed that if there are adequate time and communication bandwidth, the ideal thing to do would be to have ground control make a decision about what action to take. While it is conceivable that good autonomous reasoning could result in better decisions than ground control might make, for legal, ethical and organizational reasons we believe it is best to assume that ground controllers should make decisions where possible.

However, we are also assuming that in the majority of situations of interest for this work, there will be limited ability to get ground control input and the satellites must be able to act without ground control input.

Our approach is to leverage ground control where possible and provide automated control when necessary. Leveraging ground control is not simply waiting for commands while the ground thinks, rather it is starting response immediately along the lines that the ground would normally command. This buys time for the ground and for the space system should we lose communications during the interceding interval between ASAT detection and ground response.

When communications with the ground are lost, MAARS coordinates response for each team member to achieve counter-ASAT effectiveness while accommodating necessary orbital changes, expected or known team-member behavior, and likely team member effectiveness.

D. Numbers and Timing of ASAT

To demonstrate flexibility, MAARS will be evaluated against varying numbers of ASAT attacks and varying required margin that must be maintained after an attack to support response to likely subsequent ASAT attacks.

E. Defensive Assets

There are numerous different types of defensive assets that could be brought to bear in the counter-ASAT mission. These include counter-ASAT missiles that require the orbital elements (launch platforms) to be optimized for response effectiveness, interference avoidance (both from friendly and foe systems), and margin, all of which are taken into consideration for appropriate pointing and stand-off attitude. Other defensive assets include guidance and navigation interference systems using laser or EMF technologies. Controlling any of these defensive systems requires we achieve configurations in time to deliver coordinated response.

In general, our approach is to avoid modeling specific counter-ASAT assets and, instead, to model the abstract notions of effectiveness, cost to configure, and margin maintenance.

III. Related Work

Following sections summarize two earlier research tracks for planning and execution related to teamwork.

A. Beliefs Desires Intentions (BDI)

Early research in multi-agent teamwork (in the 1980s) focused on beliefs-desires-intentions (BDI) heuristic approaches. In the 1990s, new foundations were developed for these BDI approaches.[19–21] Specifically, it became clear that relying on pre-planned coordination without any reasoning at execution time led to significant failures. In other words, if agents simply trusted the plans provided to them at planning time for team coordination, they experienced teamwork failures. Therefore, a new approach based on "teamwork models" became the more standard approach. The key ideas were to simplify coordination planning at planning time, and instead focus efforts on execution time coordination reasoning. Agents were each provided a teamwork model that allowed them to understand how to act in a team. By using this teamwork model, agents could reason about how they should act in case of a team coordination failure and recover from such failures.

The BDI approach to teamwork model had the following key strengths. First, it led to more robust teamwork as agents were themselves able to reason about teamwork failures and recover from them. Second, the teamwork models were reusable across domains, and thus programming effort was significantly reduced. In essence, at planning time, users were required to provide high level "team-oriented programs" that abstracted out from very detailed low level coordination questions such as which agents should communicate with which other and when. Instead, agents automatically used the teamwork models to reason about such questions at execution time.

The key weakness of this BDI approach to teamwork was that it lacked explicit reasoning about costs or rewards, and about uncertainties. As such, it was difficult within these teamwork models to reason about certain courses of action being more optimal (leading to higher expected rewards) than others. For example, communication often has a cost in terms of energy consumed or time wasted in communicating with others, and communication may also have a chance of failure. However, it was very difficult within the BDI framework to reason about whether such communication cost is worth it in the case of teamwork, given its success/failure probabilities.

B. Distributed Partially Observable Markov Decision Problems (DEC-POMDPs)

Given the weaknesses of the BDI approach to teamwork, DEC-POMDPs became extremely popular in the last decade.[9–14] The key strength of this approach was that it allowed detailed reasoning about all the uncertainties and costs, and thus allowed an agent team to explicitly reason about optimality of its actions and it would allow very detailed computation of expected rewards. More specifically, in DEC-POMDPs, we can represent agents' action uncertainty (the outcomes of actions are uncertain), observation uncertainty (agents' observations of the world are uncertain) and costs/rewards for actions. These uncertainties and rewards are represented over the team's joint actions, allowing for an extremely expressive framework to represent teamwork. As such for the past 10 years, research on planning with DEC-POMDPs has gathered significant momentum and many new algorithms have been developed that exploit problem structure to provide efficient solutions.

While DEC-POMDPs have taken the field by storm, there are two key weaknesses in DEC-POMDPs: (i) assumption of accurate world model and (ii) limitation to scalability. Planning over these provided models is doubly exponential in complexity, but the key defense is that such reasoning ultimately leads to an optimal plan that agents then execute without any execution-time reasoning. These approaches appear to be exactly contrary to our approach that is inspired by the lessons learned in the BDI approach:[21,22] our work shifted part of key reasoning to execution-time, while past work has very much focused on planning-time reasoning.

IV. MAARS System Design

The system[a] is targeted at protecting a satellite from multiple attacks. The satellite system (including body guard satellites) will attempt to finish in a configuration that maximizes their ability to defend against another attack and to allow the targeted satellite to continue to function as normally as possible. For example, the bodyguard satellites should not finish in a position that impacts ground communication with the targeted satellite.

Detailed and accurate information about a threat is given to be generated by an external system and received by MAARS between three and twenty minutes before the threat will impact the targeted satellite. This assumption is based on the estimate of when a threat would be detected and the length of time between when a threat might be first detected and the time until impact. In this section, we describe the system design to effectively protect satellites from various types of attacks.

A. System Architecture Overview

A DEC-POMDP planner takes a detailed model of the environment, possible atomic actions and a model of how agents observe the environment and determines an optimal course of action. The course of action, a policy, maps every possible belief the agent could have to the best thing to do in that state. Unfortunately, saying that the policy is optimal is only a theoretical concept and typically the policy will not be optimal in the real world. The model will typically not be perfectly accurate or may change over time or be incomplete. Due to these facts, the expense of computing the optimal policy may be infeasible. Furthermore, discretization of the state space may introduce sub-optimality and so on. Despite this, DEC-POMDP planners are an attractive option because they take into account many sources of uncertainty and find a course of action that takes that uncertainty into account.

Our overall architecture takes the limitations of existing DEC-POMDP planners into account in several ways. First, a sensitivity analysis aids in finding a policy that is robust to small errors in the model. Second, an adjustable autonomy component attempts to get human input when possible. Third, an execution manager monitors the execution of the policy to determine when it is no longer a good policy and fills in parts of the policy, which were not feasible to compute initially, during execution. Together, these components allow us to take advantage of the strengths of DEC-POMDPs while avoiding their practical limitations. The function and operation of each of these components is described in more detail below.

Notice that it is not envisioned that each of the functional components would be discrete software components. For example, the sensitivity analysis and adjustable autonomy reasoning will be integrated into the DEC-POMDP planner. Figure 2 outlines the MAARS system and its components.

[a]In this section, we describe the envisioned system architecture that we aim at (i.e., focus on illustrating the overall design concept that is not fully covered in this paper) to handle general ASAT domain problems.

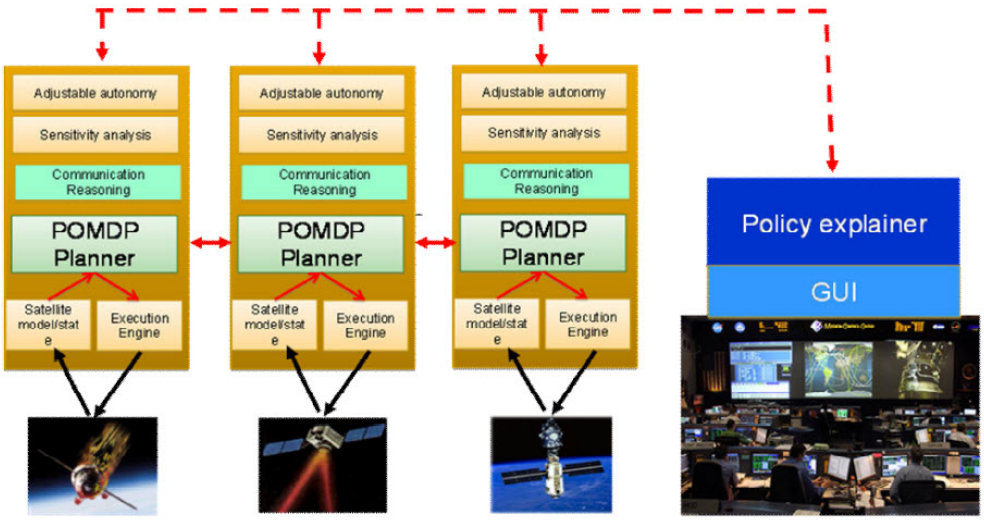


Figure 2. MAARS System Diagram

B. Functional Capabilities

In this section we describe the purpose of each of the functional capabilities of the system, discuss some of the design tradeoffs and provide an overview of the anticipated implementation concept.

1. Adjustable Autonomy

As noted above, it is generally desirable to get user input in how to react to a threat. However, a key component of an attack will be an attempt to disrupt communications between the ground and the satellite, making it critical that the satellite can act autonomously. The adjustable autonomy component, responsible for managing interaction between the satellite and the ground, has two basic functional aspects: (i) making decisions about when to get ground input and when to act autonomously; and (ii) maximizing the ability of the ground to give informed approval (or disapproval) for a plan.

The basic adjustable autonomy reasoning, i.e., whether to act autonomously or wait for user input, will be reasonably straightforward. While the cost of delaying execution of the plan is less than the expected cost of acting autonomously, the satellite should wait for user input. The major difficulty is determining what the expected cost of acting autonomously would be. For example, if the satellite knew for sure that it was always going to get approval then there is no cost to autonomous action. Alternatively, if it is about to execute an expensive plan (e.g., using multiple body guard satellites) that the ground might be able to dramatically reduce the cost of, then the cost of acting autonomously is high. Most previous work makes simple assumptions about the cost of autonomous actions, hence additional research may be required to design/implement appropriate adjustable autonomy.

Maximizing the ability of the ground to provide informed input will involve several ideas. First, the satellite should create plans that leave as much time as possible for user input, i.e., by delaying irreversible actions as long as possible. Second, the satellite could include extra activities into the plan to increase the probability of getting operator input, e.g., actions that delay the need for an irreversible response or actions that increase the probability of being able to communicate with the ground. Third, the satellite could make plans that potentially sacrifice some optimality in favor of being more intelligible or more likely to fit user preferences. For example, one plan option might fit closely with standard operating procedure while another option might appear to have slightly higher expected utility but differ greatly from normal procedures. Choosing the plan that fits with standard operating procedures can make the job of the ground to check the plan dramatically easier. Notice that theoretically each of these ideas reduces the expected utility of the autonomous plan in favor of increasing the probability of getting ground input or approval. If confidence is developed in the planner over time, it may possibly be desirable to try to plan better plans and think less about getting user input.

It is not envisioned that the adjustable autonomy component will be a standalone component that post-processes a DEC-POMDP policy. Instead, it will be a set of techniques that manipulate and augment the DEC-POMDP model

so that the resulting policy has the adjustable autonomy built-in. For example, the DEC-POMDP model would be augmented with actions that buy time for user input and rewards would be modified to encourage plans that match standard operating procedure. This type of augmentation and shaping of models for a specific purpose is a common technique, although its application to adjustable autonomy would be novel.

2. *Sensitivity Analysis*

Planners produce policies that are optimal with respect to the model they have of the environment, the ability to sense the environment and the actions that can be taken in the said environment. However, if the specific numbers used in the model are not exactly correct then the policy produced by the planner will not be optimal either. In fact, relatively little is known about how the expected utility of a policy produced using an incorrect model differs from the expected utility that might be obtained if the model was exactly correct.

The sensitivity analysis component of the overall architecture is responsible for ensuring that the policies produced by the planner are robust to errors in the model. The details of how this is achieved are described in Section VI. Initially, we are assuming that the model details will not be correct, but that we know the types of errors we will reason about (e.g., we assume all actions are represented, but that probabilities etc. are not necessarily right).

3. *DEC-POMDP Planner*

In the proposed architecture, a Distributed Partially Observable Markov Decision Problem (DEC-POMDP) planner performs the core planning. A DEC-POMDP planner was selected since the policies of action that result from this planner would take into account the key sources of uncertainty that the agent team faces. Specifically, the planner reasons about uncertainty in the outcomes of actions and uncertainty in perception of the environment. No other automated planning approach is capable of explicitly taking into account both of these sources of uncertainty.

While the process of planning with a DEC-POMDP model is very complex and algorithms extremely complicated, the underlying basics are rather simple. All possible beliefs that the agents could have about the current state of the environment and themselves are discretized into a set of joint belief states. An observation function tells the agents how they will move between belief states depending on observations from the environment. For example, at $t = 0$, agents may believe that there is a 90% probability that some actuator is fully retracted but after using a noisy sensor to check the state of the actuator it may believe at $t = 1$ that there is a 50% chance the actuator is retracted. For each possible belief state, a reward model tells the agents how good they are to be in that state. For example, if the aim was to retract the actuator, the state where the agents believe there is a 100% probability the actuator is retracted will have highest reward. The agents also know of a set of actions it can take and the probabilities and costs of different outcomes for those actions. For example, issuing a "retract" command might have a 95% probability of retracting the actuator and costs 4 for use of the battery. Using these models, a DEC-POMDP planner works out the best possible joint action to take in each possible belief state. The mapping from all possible belief states to optimal actions is called the policy. For example, the policy might specify that when the probability of the actuator being in a retracted state is 0, it should issue a "retract" command, but when it is 50% another observation should be taken, effectively moving the agents to a state where it is more certain whether they need to issue another retract command. In real-world problems, there can be very large numbers of belief states, therefore very large policies will be created, at high computational cost.

Because they promise to provide optimal policies of action and handle the primary sources of uncertainty agents will face, DEC-PODMPs have been a very active research area in recent years. Basic algorithms to find optimal courses of action have been known for a long time, but they are infeasible for all but the smallest problems. Much recent work has focused on identifying subclasses of DEC-POMDPs, e.g., where uncertainty has a particular form, where specialized and hopefully more efficient solvers can be found. Other work has looked at ways big gains in efficiency can be made by sacrificing optimality guarantees, but hopefully still getting very high quality plans. Researchers, including those working on this project, are attempting to make DEC-POMDP planners more tractable by doing both of the above, i.e., identifying subclasses of DEC-POMDP planners for which efficient, high-quality heuristic solvers can be developed. Such work has led to DEC-POMDPs becoming much more useful for a wider class of real problems.

Other ongoing, relevant research work is looking at how some of the constraints of working with DEC-POMDP planners can be removed. For example, as noted above, DEC-POMDP planners typically require a discretized state space which is not always natural or even feasible; hence some research (including work being done by researchers involved in this project) is looking at how to build DEC-POMDP planners that can reason over continuous state spaces. Other research is looking at how to represent policies compactly, which is important if there are many belief states.

4. Satellite Models and Statistics

The satellite models encapsulate what the system knows about the satellites, the environment and threats. These models are the basis for the DEC-POMDP planning. It is assumed that these models change very rarely, if ever. Dynamically changing information about the specific situation is stored elsewhere.

Any known information about the uncertainty associated with any model values is also stored here. For example, the model may indicate that the probability of a bodyguard successfully intercepting a target is 0.8 if fired, but due to lack of testing and information about opponents, the real probability might be anywhere between 0.6 and 0.95.

5. Execution Engine

In most previous work, DEC-POMDP planners plan an action in advance for every possible belief state the agents might find themselves in. When there are multiple cooperative agents, joint actions (one action per agent) are planned for every possible joint belief state (what the agents collectively believe). The agent, then, executes the computed policy without any additional reasoning; it simply computes its new belief state and looks up the action to take in that state. However, in general, this is computationally very expensive and often unnecessary if the DEC-POMDP model is not exactly correct.

The execution engine reduces the requirements on the DEC-POMDP planner by performing some of the reasoning at runtime. Using inspiration from Belief-Desire-Intention (BDI) agent architectures, where a sophisticated online reasoning engine executes abstract, vague plans; the execution engine will take DEC-POMDP policies and execute them flexibly. This achieves two major objectives.

First, the DEC-POMDP planning can be made simpler. Specifically, we intend to leave reasoning about when to communicate (with other satellites, etc.) to the online engine, which can leverage previous research and information about available communication channels to decide what to do. Including this reasoning in the DEC-POMDP planner would dramatically increase the computational expense of that planning, since all possible communication failures would have to be considered in advance, while only a small cost in terms of optimality is paid for doing the reasoning online.

Second, the execution engine reduces the requirements on the DEC-POMDP planner by monitoring execution to determine when the plan is not performing acceptably or as expected. If something is missing from the model or parameters in the model were far from correct values, the execution engine can determine that progress is not proceeding according to expectations and initiate some remedial action. Because this safeguard is in place, it is less important to make sure that the DEC-POMDP model captures every detail of the environment perfectly. The execution monitoring is actually made easier because the plan being monitored was created with a DEC-POMDP planner. The planning process both uses and generates expectations of how the environment will evolve and what the expected utility will be at a particular point in time. In the case of environment expectations, the execution engine can look for statistically important differences in the rates at which events will occur according to the DEC-POMDP model and the rates that are actually occurring. Sometimes, luck will just be against the agent or team and it will find itself in a situation where nothing good can happen. The execution engine can detect this by noting that the expected utility of following the optimal course of action from the current state on is low. This is computed as a normal part of DEC-POMDP planning.

Notice that the general philosophy of the execution engine is to use techniques that are best suited to the problem, rather than attempting to force a particular technique to work in all cases. This general idea, which we refer to as creating hybrid approaches, is recurrent throughout the proposed architecture.

6. Policy Explainer

A DEC-POMDP policy describes the optimal joint action for every possible state agents might find themselves in. Typically, there are millions or more possible belief states for which actions are chosen. The policy does not have any explicit abstract strategy. Thus, in a raw form, the policy is very difficult for a human to understand it is simply millions of atomic actions taken in millions of belief states. The difficulty in understanding the policy becomes even more difficult if multiple agents have interacting policies that must be understood together.

Although some previous work has looked at DEC-POMDP policy explainability, little progress has been made and it is generally acknowledged as a very difficult challenge. Broadly speaking, it is one of the strengths of DEC-POMDPs that makes it particularly difficult to explain the resulting policy. Specifically, interesting DEC-POMDP policies will generally not have any particular high-level strategy that is consistent across the whole policy. Reasoning at the level of atomic actions per state effectively allows the DEC-POMDPs to switch between high-level strategies

depending on the situation or follow a path that allows multiple alternatives to be considered later and so on. For example, if a policy prescribed that an agent take a left road at some particular point, it may be (effectively) so that it keeps multiple possible strategies open for later on or it may be that this represents a change in strategy or it may be a natural next step in the basic strategy. Determining what the strategy in the policy will be extremely difficult. Moreover, even if a high-level strategy could be discerned, as noted above, that might be only one of many that the policy will take on under subtly different circumstances.

Explaining to a user why a policy has chosen a particular course of action over another course is another extremely difficult problem. A DEC-POMDP planner chooses one course of action over another if the expected utility of that sequence of actions is higher than the other. The expected utility of each action was individually computed to be better than any other available option in that state once the best possible tree of optimal actions given all possible outcomes was considered. The planner does not explicitly have any rationale for choosing a particular sequence of actions over another; hence in any explanation it would be necessary to infer some set of particular probabilities or outcomes that most strongly influence the sequence of actions taken.

V. MAARS Proof of Concept Demonstration

The aim of the proof-of-concept development was to verify that DEC-POMDPs were a reasonable tool to apply to the problem of ASAT mitigation. The process of modeling a specific scenario in detail, mapping the features of the scenario into a DEC-POMDP, computing policies with state-of-the-art solvers and examining the resulting policies serves to show that DEC-POMDPs are an appropriate tool for the ASAT mitigation domain. The aim was not to build a complete solution, only to increase confidence that a complete solution could be developed.

A secondary aim of the proof-of-concept was to clarify the key research issues that must be addressed for DEC-POMDPs to be a broadly applicable, practical approach to ASAT mitigation domain. The modeling and computation process exposed aspects of DEC-POMDPs that are not available or sufficiently mature for practical use, providing our team with future research directions.

The proof-of-concept was not intended to be a complete, working system. The DEC-POMDP policies (assuming full and free communication among agents at plan-time) were evaluated as policies, without executing in any high-fidelity simulation environment.

A. Scenario

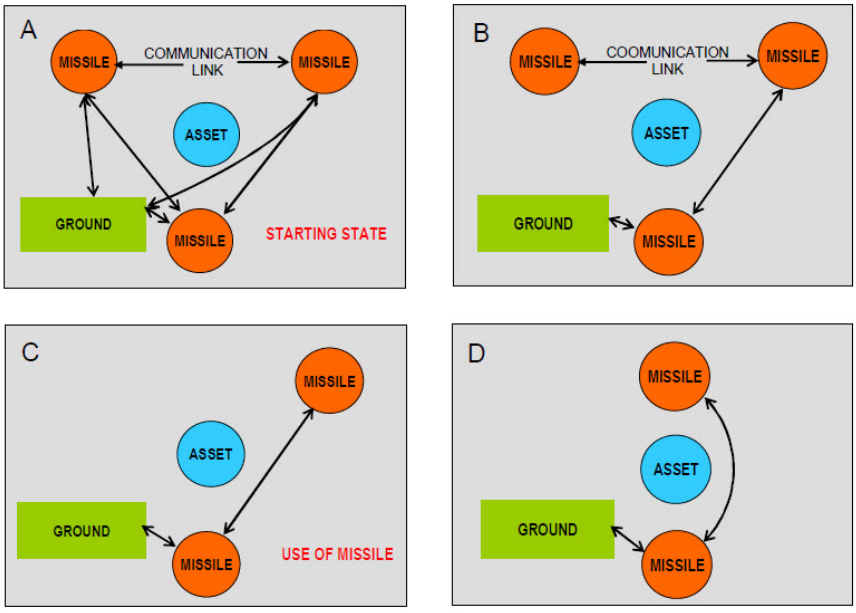


Figure 3. Proof of Concept Demonstration Domain

The scenario used for the proof-of-concept was designed by JPL (see Figure 3). A critical satellite is protected by a number of bodyguard satellites. The bodyguard satellites are informed of an incoming threat and must plan and execute a response. The bodyguards can stop the threat by firing on it, but they destroy themselves as a result. The bodyguard may fail to fire or it may fail to hit. The bodyguard can reliably detect that it fails to fire, but cannot reliably detect when it fails to hit. The bodyguards can observe the threat, but are unreliable at detecting whether the threat has been destroyed.

The satellites are able to move at a small cost. We are using Two-Line Element set (TLE) representation for satellite position, which allows us to reason about moving the satellite along only one dimension. There is an optimal location to fire from which the bodyguard knows and can choose to move to, albeit with some cost and with some chance of failure.

The location of the threat and the location of the bodyguards are continuous values in the real world, and must be discretized to be captured by the DEC-POMDP planner. The discretization parameter is experimentally set. Any level of discretization generates approximate solutions, however, in practice, it does not harm the overall solution quality in this domain as long as the approximate solution quality is higher than the given threshold.

Notice that working out by hand what the right thing to do is very hard. Whether one or more bodyguards should fire, whether to trust observations or fire first and whether to move first or simply fire are decisions that depend very on the details of the scenario.

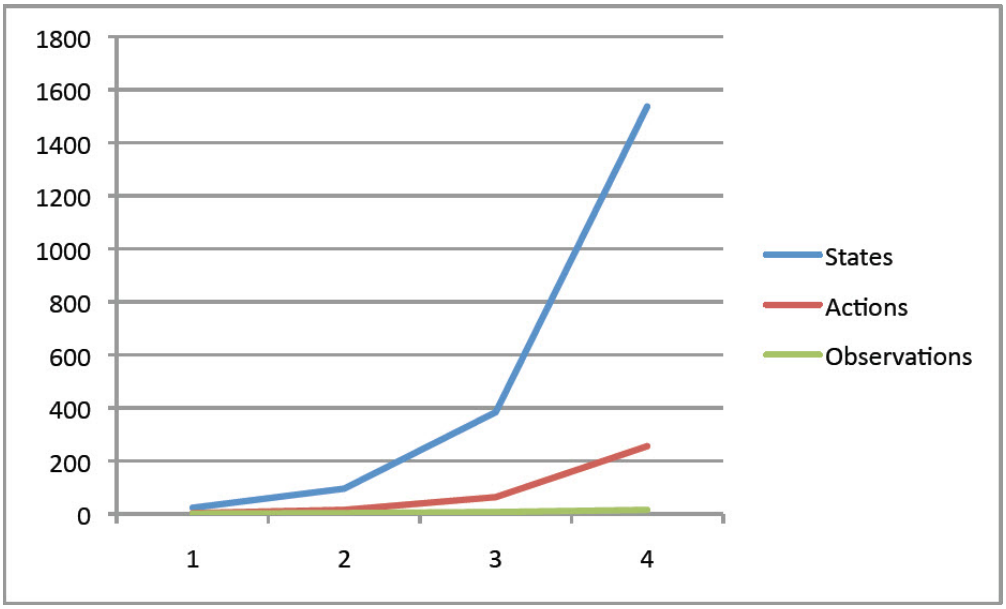


Figure 4. Number of States, Actions and Observations vs. Number of Agents

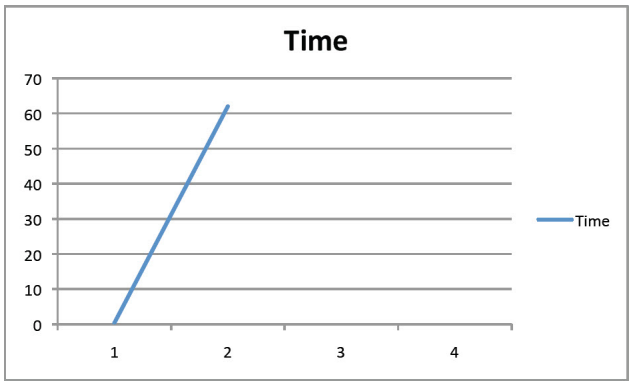
Figure 4 shows the size of the state space, the number of possible joint actions and the number of possible joint observations as the number of bodyguards is increased.

If communication fails, the bodyguard satellites will not be able to perfectly observe one another perfectly, although they will have some ability to try to determine what the other satellite is doing and whether it succeeded in moving or firing.

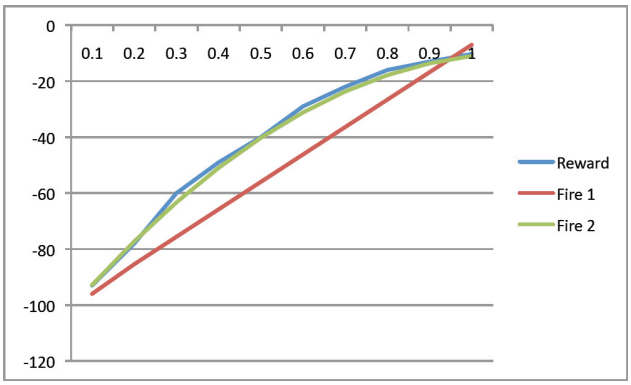
B. Summary of Gathered Results

The proof-of-concept implementation used a completely centralized planning process, which is typical for state-of-the-art DEC-POMDP solvers. Individual policies are created for each of the bodyguard satellites. The policies map from joint belief states to individual actions, i.e., each body guard chooses its action based on the joint beliefs of all the body guards. This implies that there must be full communication of all observations between members of the team. In this scenario, the total amount of communication is very small, since there are only a small number of observations taken. However, if communication were disabled by the opponent, the agents would not be able to execute their policies.

The actual policy computation was performed by a state-of-the-art, but off-the-shelf open source solver.[18] While



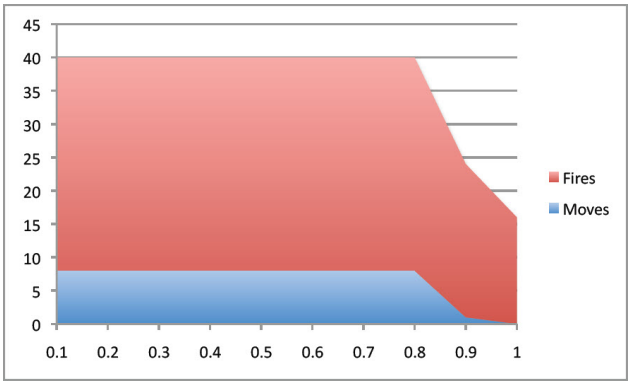
(a) Computation Time (in seconds) vs. Number of Agents



(b) Expected Utility vs. Hit Probability

Figure 5.

(a) Reward vs. Hit Probability



(b) Number of Fires and Number of Moves vs. Hit Probability

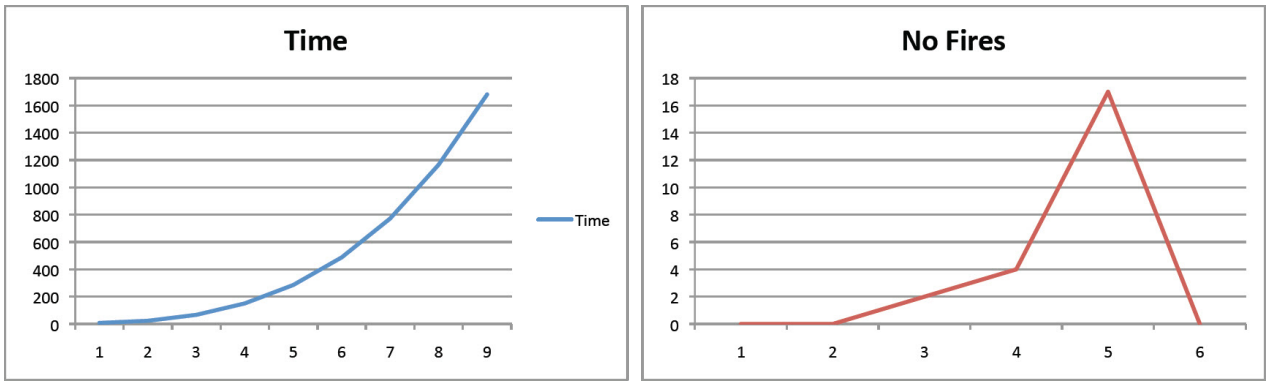
Figure 6. Solution Quality vs. Hit Probability

it may be possible to slightly improve computation time by developing a specialized solver, we believe that the gain would be minimal.

In the first experiment, we looked at the computation time for increasing numbers of bodyguards. Figure 5(a) shows that for any more than two body guards the computation did not finish under one hour, using a standard desktop computer. For two bodyguards, however, the computation time was a very reasonable one minute.

In the second experiment, we compared the expected utility of the policy against two baseline solutions, one where one bodyguard always fired immediately and another where both bodyguards fired immediately. The policies were evaluated for different probabilities of the bodyguard successfully hitting the target when it fired. The quality of the DEC-POMDP policy, as determined by running 1000 simulations was shown to dominate either baseline solution, except in a particular unmodeled edge case. The results are shown in Figure 5(b). When the hit probability was 1.0, the correct thing to do was to fire only one bodyguard. However, the DEC-POMDP still had a finite probability of (incorrectly) observing that the threat was not hit which sometimes caused it to fire a second bodyguard and incur that cost.

In the third experiment, we looked in more detail at what the DEC-POMDP policy was doing as the probability of hitting the target with a bodyguard increased. As shown in Figures 6(a) & 6(b), the reward increased with higher probability of hitting, since the probability of paying the large penalty for losing the protected satellite reduced. More subtly, as the probability of hitting the target approached 1.0, the policy would contain less states where the proscribed action was to fire or to move. The planner was correctly reasoning that if the probability of hitting was very high it was possible to fire one bodyguard and then observe before choosing to fire the other. Moreover, it was correctly reasoning that if the probability of hitting was very high, the cost and risk of moving into an even better position was not worth it.



(a) Computation Time (in seconds) vs. Number of Target Locations

(b) Number of Fires vs. Number of Target Locations

Figure 7. Results with Various Number of Target Locations

Finally, we performed an experiment where the number of possible target locations was increased. Effectively this reduced the size of the discretization of the target location and bought more time for the bodyguards to take observations and understand whether the target had been destroyed. Figures 7(a) & 7(b) show the computation time and the number of states in the policy in which the proscribed action was fired. A curious quirk of the DEC-POMDP model shows up in the graph, manifested by the policy choosing not to fire if the discretization was too coarse (i.e., one or two possible target locations) or too fine (i.e., 6 target locations). These occurred for different reasons. In the coarse case, the planner found no possibility of the target being hit, because there was insufficient granularity to reason that it could be hit before the target was destroyed. This is a problem with the discretization of the space. In the fine case, the planning horizon was too small to reason about the satellite being hit by the threat, hence the planner saw no value in expending resources firing on the threat. While both these problems are easily addressed with correct modeling, they illustrate the need to take extreme care with the model. When the policy errors are not so obvious, modeling errors may lead to poor policies.

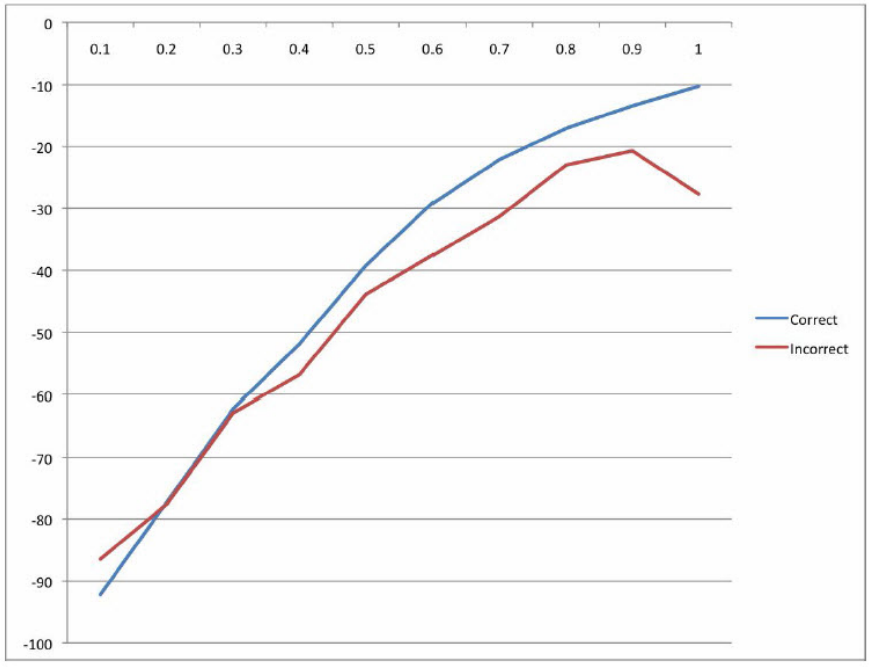


Figure 8. Expected Utility vs. Hit Probability

A key issue with DEC-POMDPs is that the many numbers that make up the model must be known precisely in

order to compute a policy. Unfortunately, often the numbers are not known exactly and must be estimated. In this experiment, we used one set of numbers to compute a policy and another to evaluate the policy. This experiment gives an indication of how robust the policy is to errors in the model. Specifically the experiment focused on errors in the probability that a bodyguard would hit the target when fired. Figure 8 shows the expected value of the policy for various assumed probabilities of hitting the target and the average value of the policy if the actual probability of hitting the target was normally distributed around the assumed value, with variance 0.3. Thus, the experiment represents the case where the modeler correctly knew the average rate at which the bodyguard would hit the target, but the actual value varied quite widely. The real value of the policy is nearly always lower than the expected value of the policy. The exception is when the average probability of hitting is very low and is caused by a statistical quirk — if the assumed hit probability is wrong, it must almost certainly be an underestimate, hence the protected satellite is destroyed less than hoped. The biggest deviation between expected value and real value is when the assumed hit probability is very high because the policy mistakenly often chooses not to shoot but the actual hit probability is low and extra bodyguards should have fired.

The proof-of-concept implementation has shown that there are important ASAT mitigation problems for which DEC-POMDPs are an appropriate tool. The mapping between the provided scenario and the DEC-POMDP formulation was natural and off-the-shelf solvers were capable of finding solutions in reasonable amounts of time. The resulting policies dominated simple rules that might have been used otherwise. The DEC-POMDP policies dominated because they successfully took action and observational uncertainty into account. However, it is also clear that further research is required to make DEC-POMDPs an appropriate tool for many ASAT mitigation scenarios. For example, past work in DEC-POMDPs does not provide a robust solution to errors in the given model, and its computational time exploded as the number of bodyguards and states was increased. We describe a novel approach to address these limitations in past work in the following section.

VI. Addressing Research Challenges

In Section IV, we described the overall system design of MAARS to handle ASAT domain problems. Among the features to strengthen a DEC-POMDP planner, in this paper, we mainly focus on planning and execution in DEC-POMDPs with sensitivity analysis (as described in Sections IV-B-2, 3 & 5).

More specifically, our approach to multi-agent planning is based on DEC-POMDPs, which is explicitly driven by model uncertainty. We do not assume complete model accuracy, thus avoiding a major weakness of previous work in DEC-POMDPs. In particular, we do not assume that the transition and observation probabilities in DEC-POMDPs are fully accurate; only that they are reasonably accurate, i.e., there may be a limited quantifiable error. Indeed, in our domain of multiagent teamwork and coordination in space, it would be very difficult to assume a fully accurate model of all of the action and observation uncertainties encountered by the satellites' actuators and sensors; but we may be able to provide some range over such uncertainties.

This work attempts to address the following research challenges. First, how can we reduce the overall complexity to optimally compute when agents need to communicate with each other assuming various levels of model uncertainties? Since communication has inherent cost, agents individually reason about communication at execution time and selectively communicate to maximize the expected reward. Second, how can we choose critical points of communication to save significant execution-time computation? Third, how can we maintain an approximate model of joint beliefs when the number of possible beliefs grows rapidly over time? Fourth, particularly in domains with interaction-sparseness, how can we utilize some form of pre-planning for individual actions that do not involve inter-agent interaction? This contrasts with previous work in execution reasoning for DEC-POMDPs where every step is planned online regardless of whether it involves interaction.

Our work focuses on robustness via execution-time reasoning about coordination/teamwork: planners create approximate plans that ignore coordination, and execution-time reasoning fills in required coordination in the abstract plans as required. Specifically, this work has the following key contributions: it (i) gets more robust plans by explicitly considering model uncertainty, (ii) reduces planning-time computation by shifting some of the burden to execution-time reasoning, (iii) saves significant execution-time computation by exploiting sparse interactions between agents, and (iv) maintains an exponentially smaller model of agents' beliefs and do not rely on precise online planning too much. This not only provides significant speed-ups, but it further improves robustness to model uncertainty without sacrificing reward performance in the process.

More specifically, we introduce MODERN,[23] which is an execution time coordination/communication reasoning algorithm, to address the above research challenges. Key concepts in MODERN will be "Trigger Point" and "Individual estimate of joint Beliefs (IB)". In order to avoid reasoning about communication at every time step, agents will

use "Trigger Points", heuristically-chosen time steps at which agents should reason about communication, to improve team performance. In addition, IB, which is exponentially smaller belief states than the entire team's beliefs, will be used in MODERN to estimate the most likely actions of other agents and decide whether or not communication would be beneficial. However, the number of possible beliefs in IB grows rapidly, particularly when agents choose not to communicate for long time periods. Hence, we also propose a new pruning algorithm that provides further savings. In particular, it keeps a fixed number of most likely beliefs per time step in IB.

MODERN is presented in detail in Algorithm 1. The joint policy π is provided as an input to MODERN. On line 1, the initial distribution of possible beliefs, IB^0, is composed of a single node at belief $\mathbf{b}^0$ (the starting belief of the team), which has probability 1, an empty observation history, and a joint action, $\mathbf{a}^0$, which is described by the root of π. In line 8, IB is updated by the standard Bayes update rule. Then, MODERN decides whether or not a trigger point exists on the current time step (line 9). If a trigger point is detected, MODERN reasons whether the expected utility gain caused by communication justifies communication cost (lines 11–22). Specifically, when a trigger point is detected, agent i communicates when the expected utility gain by communication is higher than a given communication cost $\sigma : (U_{\mathrm{C}}(i) - U_{\mathrm{NC}}(i)) > \sigma$.[b] Otherwise, the agent simply selects and executes its action from π. When agents decide to communicate, they share their local histories (i.e., synchronize their histories). Agents can then know the actual joint observation histories and execute the joint action given by the policy (line 20). Otherwise, they execute the estimated best joint action (line 22). If trigger points are not detected, agents take their individual actions from the given policy π (line 24). During communication reasoning, computation of $U_{\mathrm{C}}(i) - U_{\mathrm{NC}}(i)$ is performed as following:

$$U_{\mathrm{C}}(i) = \sum_{\theta \in \mathrm{IB}_i} p(\theta) \cdot V\left(\mathbf{b}(\theta), \mathbf{a}(\theta)\right), \; U_{\mathrm{NC}}(i) = \max_{a_i \in A_i} U_{\mathrm{IB}_i}(\langle a_i, a^*_{-i} \rangle), \tag{1}$$

$$a^*_{-i} = a_{-i}(\theta^*) \;\text{ s.t. }\; \theta^* \in \Theta, \; U_{\mathrm{IB}_i}(\mathbf{a}) = \sum_{\theta \in \mathrm{IB}_i} p(\theta) \cdot V(\mathbf{b}(\theta), \mathbf{a}), \tag{2}$$

where $V(\mathbf{b}, \mathbf{a})$ is the expected utility when an action $\mathbf{a}$ is taken at belief state $\mathbf{b}$. a^*_{-i} is agent i's estimate of the most likely action of all other agents. This is greedily selected using the most likely sequence, Θ, at every time step, where agent i optimistically assumes that all other agents obtain the most likely observations.

VII. Empirical Evaluation

In this section, we show some preliminary results of our execution-time algorithm, MODERN, which successfully plans for DEC-POMDPs with model uncertainty. As we will show in the following results, we provide a novel method to handle the previously unaddressed problem of uncertainty in world models. We evaluate the performance of MODERN on a 2×3 grid domain, which is fundamentally similar to our first phase target domain (the ASAT mitigation domain), and compare it with two previous techniques: ACE-PJB-COMM (APC)[15] and MAOP-COMM (MAOP).[16] As APC and MAOP have only limited abilities to scale up to large domains, we first show results in a small domain (for comparison with APC and MAOP) and then scale up MODERN using pruning and show results for a larger domain. The experiments were run on Intel Core2 Quadcore 2.4GHz CPU with 3GB main memory. All techniques were evaluated for 600 independent trials throughout this section. We report the average rewards.

A. Small Grid Domain

The smaller domain we used for evaluation was a 2×3 grid. In this domain, there are two agents trying to perform one joint task. There are 72 joint states, 25 joint actions and 4 joint observations in this configuration. The time horizon is set to 3, as this is sufficient to complete the joint task.

Using this domain, we compared the average rewards achieved by all algorithms for three different communication costs (σ). The communication costs are selected proportional to the expected value of the policies: 5%, 20%, and 50%. In Table 1, σ in column 1 displays the different communication cost and α in column 2 represents the level of model error given by the Dirichlet distribution. We varied α from 10 (i.e., model error is high) to 10000 (i.e., model error is low). Columns 3–5 display the average reward achieved by each algorithm in the 2×3 grid domain. We performed experiments with a belief bound of 10 nodes per time-step for our algorithm.

As shown in Table 1, MODERN received much higher reward than both APC and MAOP in the grid domain. The results also show that as communication cost increases, the reward obtained by all three algorithms decreases since

[b] $U_{\mathrm{C}}(i)$ is calculated by considering two-way synchronization, which emphasizes the benefits from communication. $U_{\mathrm{NC}}(i)$ is computed based on the individual evaluation of heuristically estimated actions of other agents.

Algorithm 1 MODERN(JOINTPOLICY π, AGENTINDEX i)

Initialize individual estimate IB_i^0
$\mathbf{a}^0 \leftarrow \pi(\mathrm{IB}_i^0)$
Execute an action a_i^0
$\tau \leftarrow false$
for $t = 1, \ldots, T-1$ **do**
  $o_i^t \leftarrow$ Get the observation from the environment
  $h_i^t \leftarrow$ Update agent i's own local history with o_i^t
  $\mathrm{IB}_i^t \leftarrow$ EXPAND($\mathrm{IB}_i^{t-1}, o_i^t$) {Expand an individual belief based on an agent's local observation using the Bayes update rule}
  $\tau \leftarrow$ DETECTTRIGGERPOINT(π, IB_i^t)
  if $\tau = true$ **then**
    $U_{\mathrm{C}} \leftarrow \sum_{\theta \in \mathrm{IB}_i^t} p(\theta) \cdot V(\mathbf{b}(\theta), \mathbf{a}(\theta))$
    $A_i^t \leftarrow \{\pi_i(\mathbf{h}^t(\theta)) | \theta \in \mathrm{IB}_i^t\}$ {Get set of possible actions from IB_i^t}
    $a_{-i}^* = a_{-i}(\theta^*)$ {Estimate most likely actions for other agents via Θ}.
    $a_i^* \leftarrow \arg\max_{a_i \in A_i^t} \sum_{\theta \in \mathrm{IB}_i^t} p(\theta) V(\mathbf{b}(\theta), \langle a_i, a_{-i}^* \rangle)$ {Get optimal individual action of agent i, assuming other agents take the most likely action a_i^*}
    $U_{\mathrm{NC}} \leftarrow \sum_{\theta \in \mathrm{IB}_i^t} p(\theta) \cdot V(\mathbf{b}(\theta), \langle a_i^*, a_{-i}^* \rangle)$
    if $(U_{\mathrm{C}} - U_{\mathrm{NC}}) > \sigma$ **then**
      Sync h_i^t with other agents
      $\tau \leftarrow false$
      $\mathrm{IB}_i^t \leftarrow$ Update belief via communicated joint history $\mathbf{h}^t$
      $a_i^t \leftarrow \pi_i(\mathrm{IB}_i^t)$
    else
      $a_i^t \leftarrow a_i^*$
  else
    $a_i^t \leftarrow \pi_i(\mathrm{IB}_i^t)$
  Execute the action a_i^t

agents have to pay more to avoid miscoordination or face higher chance of miscoordination with less communication, both leading to lower solution quality.

Another trend in the results is that the solution quality generally increases with a lower α. For example, when communication cost was low (5%), as model uncertainty increased (i.e., α decreased from 10000 (row 6) to 10 (row 3)), the average reward increased. Inclusion of model uncertainty into communication reasoning allows for better use of communication to reduce miscoordination. Thus, when model uncertainty increases, MODERN can get better results.

We also compared the average (execution) runtime of the algorithms. In this domain, MODERN and APC took similar amounts of time (about 20 seconds), but MAOP took about 1.3 times more than MODERN.

B. Large Grid Domain

A more interesting question is whether we could scale up the algorithms to run in even larger domains or with longer time horizons to handle real-world problems. An extended 2×3 grid is used for evaluation. In this domain, there are two agents trying to perform one joint task and two individual tasks. The number of joint states is 288, the number of joint actions is 49, and the number of joint observations is 9. With this domain, we look at scalability in terms of time horizon.

We ran experiments in the larger grid domain with increased time horizons. Figure 9(a) shows the runtime on the y-axis and the time horizon on the x-axis. We tested the algorithm under two different communication costs: 5% (low) and 50% (high). MAOP (with belief merging) and APC (with 1 particle) could not solve the problem within the given time limit (1,800 seconds) for even the shortest time horizon — while MODERN took significantly less time than other algorithms. As the time horizon increased, MODERN obtained higher rewards (from 9.8 to 15.7), since there was more time for agents to recover from any failed actions. With σ=50%, MODERN took more time than with σ=5%, although still scaling linearly with time horizon.

We then evaluated trigger points in MODERN. Figure 9(b) shows the runtime of MODERN with and without selective reasoning: the x-axis is the time horizon and the y-axis is runtime in seconds. As shown in the result, MODERN can speedup runtime by over 300% using trigger points. In particular, the average number of trigger points for T=8 was about 2.6. This means MODERN only reasons about communication for about 1/3 of the total time steps,

Table 1. Comparison MODERN with APC and MAOP: Average Performance in the Small Grid Domain

		2×3 Grid		
σ	α	MODERN	APC	MAOP
5%	10	5.28	-2.25	-0.36
	50	5.28	-2.04	-0.68
	100	5.02	-1.85	-0.63
	10000	4.62	-1.80	-0.78
20%	10	4.62	-1.20	-1.47
	50	4.62	-1.20	-1.72
	100	4.36	-1.20	-1.68
	10000	3.96	-1.20	-1.86
50%	10	3.30	-1.20	-3.69
	50	3.30	-1.20	-3.80
	100	3.04	-1.20	-3.79
	10000	2.64	-1.20	-4.01

which leads to roughly three-fold improvement in runtime.

All these preliminary results highlight:

- MODERN's optimality vs. compared algorithms
- MODERN's scalability to larger domains

VIII. Conclusion

This paper aims to open a new area of research for DEC-POMDPs: in many real-world space domains, we will not have a perfect model of the world, and hence DEC-POMDPs must provide a robust solution considering varied defense/offense strategies under such model uncertainty. To combat those challenges in ASAT domains, we presented a new framework called MAARS, which is based on a new execution-centric approach for DEC-POMDPs explicitly motivated by model uncertainty. We justified our design decisions in MAARS through an initial empirical evaluation and showed that MAARS can provide solutions much faster than existing algorithms while achieving significantly superior solution quality. Further, our proof-of-concept efforts strengthen our confidence in applicability of DEC-POMDPs to ASAT mitigation problems and also provides us with future research directions for ensuring a potential application to a realistic domain. With the resolution of issues such as scalability and robustness and better domain modeling, we can provide a suite of algorithms and technologies that potential users would benefit in critical applications that require the probabilistic reasoning, adjustable autonomy, policy explanation and other capabilities we aim to integrate into the MAARS system.

Acknowledgments

This work was supported in part by Defense Advanced Research Projects Agency (DOD) contract number N10PC20110, and Perceptronics Solutions, Inc.

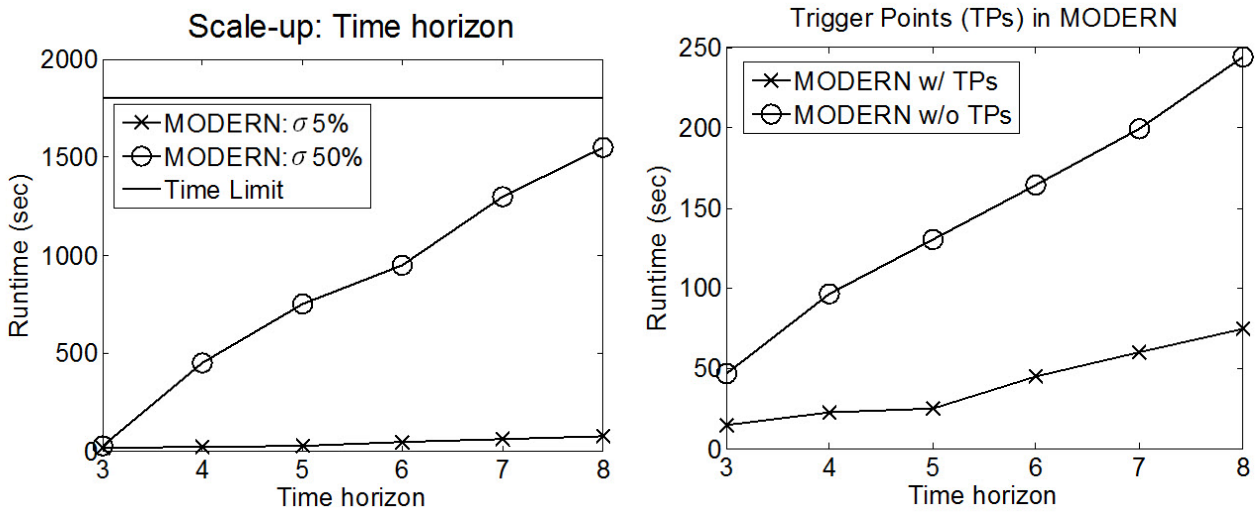
References

[1]Gregory, N. M., Dorais, G. A., Fry, C., Levinson, R., and Plaunt, C., "IDEA: Planning at the Core of Autonomous Reactive Agents," *International NASA Workshop on Planning and Scheduling for Space*, 2002.

[2]Clement, B. J. and Barrett, A. C., "Continual coordination through shared activities," *AAMAS*, 2003.

[3]Chien, S., Sherwood, R., Tran, D., Cichy, B., Rabideau, G., Castano, R., Davies, A., Lee, R., M, D., Frye, S., Trout, B., Hengemihle, J., Shulman, S., Ungar, S., and Brakke, T., "The EO-1 Autonomous Science Agent," *AAMAS*, 2004.

[4]Chien, S., Sherwood, R., Tran, D., Cichy, B., Rabideau, G., Castaño, R., Davies, A., Mandl, D., Frye, S., Trout, B., D'Agostino, J., Shulman, S., Boyer, D., Hayden, S., Sweet, A., and Christa, S., "Lessons learned from autonomous sciencecraft experiment," *AAMAS*, 2005.



(a) Scale-up: T

(b) Runtime improvement by selective reasoning

Figure 9. Evaluation of MODERN

[5]Rabideau, G., Knight, R., Chien, S., Fukunaga, A., and Govindjee, A., "Iterative Repair Planning for Spacecraft Operations Using The ASPEN System," *International Symposium on Artificial Intelligence Robotics and Automation in Space (i-SAIRAS)*, 1999.

[6]Chien, S., Knight, R., Stechert, A., Sherwood, R., and Rabideau, G., "Using Iterative Repair to Improve Responsiveness of Planning and Scheduling," *International Conference on Artificial Intelligence Planning and Scheduling*, 2000.

[7]Chien, S., Engelhardt, B., Knight, R., Rabideau, G., Sherwood, R., Hansen, E., Ortiviz, A., Wilklow, C., and Wichman, S., "Onboard Autonomy on the Three Corner Sat Mission," *International Symposium on Artificial Intelligence, Robotics and Automation for Space (i-SAIRAS)*, 2001.

[8]Bernstein, D. S., Zilberstein, S., and Immerman, N., "The complexity of decentralized control of markov decision processes," *UAI*, 2000.

[9]Goldman, C. V. and Zilberstein, S., "Optimizing Information Exchange in Cooperative Multi-agent Systems," *AAMAS*, 2003.

[10]Nair, R., Yokoo, M., Roth, M., and Tambe, M., "Communication for Improving Policy Computation in Distributed POMDPs," *AAMAS*, 2004.

[11]Pynadath, D. V. and Tambe, M., "The communicative multiagent team decision problem: Analyzing teamwork theories and models," *JAIR*, Vol. 16, 2002, pp. 389–423.

[12]Seuken, S. and Zilberstein, S., "Memory-Bounded Dynamic Programming for DEC-POMDPs," *IJCAI*, 2007.

[13]Seuken, S. and Zilberstein, S., "Formal Models and Algorithms for Decentralized Decision Making under Uncertainty," *AAMAS*, 2008.

[14]Spaan, M. T. J., Oliehoek, F. A., and Vlassis, N., "Multiagent Planning under Uncertainty with Stochastic Communication Delays," *ICAPS*, 2008.

[15]Roth, M., Simmons, R., and Veloso, M., "Reasoning about joint beliefs for execution-time communication decisions," *AAMAS*, 2005.

[16]Wu, F., Zilberstein, S., and Chen, X., "Multi-Agent Online Planning with Communication," *ICAPS*, 2009.

[17]Krepon, M. and Black, S., *Space Security or Anti-satellite Weapons?*, The Henry L Stimson Center, 2009.

[18]Kaelbling, L., Littman, M., and Cassandra, A., "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, Vol. 101, 1998, pp. 99–134.

[19]Levesque, H. J., Cohen, P. R., and Nunes, J. H. T., "On Acting Together," *AAAI*, 1990.

[20]Grosz, B. and Kraus, S., "Collaborative plans for complex group actions," *Artificial lntelligence*, Vol. 86, 1996, pp. 269–358.

[21]Tambe, M., "Towards Flexible Teamwork," *JAIR*, Vol. 7, 1997, pp. 83–124.

[22]Kaminka, G. A. and Frenkel, I., "Integration of Coordination Mechanisms in the BITE Multi-Robot Architecture," *ICRA*, 2007.

[23]Kwak, J., Yang, R., Yin, Z., Taylor, M. E., and Tambe, M., "Teamwork in Distributed POMDPs: Execution-time Coordination Under Model Uncertainty," *AAMAS*, 2011.