Towards Realistic Decentralized Modeling for Use in a Real-World Personal Assistant Agent Scenario

Christopher Amato, Nathan Schurr, and Paul Picciano

Aptima, Inc.
12 Gill Street, Suite 1400
Woburn, MA 01801
{camato,nschurr,ppicciano}@aptima.com

Abstract. Many approaches have been introduced for representing and solving multiagent coordination problems. Unfortunately, these methods make assumptions that limit their usefulness when combined with human operators and real-life hardware and software. In this paper, we discuss the problem of using agents in conjunction with human operators to improve coordination as well as possible models that could be used in these problems. Our approach – Space Collaboration via an Agent Network (SCAN) – enables proxy agents to represent each of the stakeholder agencies in a space setting and shows how the SCAN agent network could facilitate collaboration by identifying opportunities and methods of collaboration. We discuss this approach as well as the challenges in extending models to 1) take advantage of human input, 2) deal with the limited and uncertain information that will be present and 3) combat the scalability issues in solution methods for a large number of decentralized agents. As a step toward dealing with limited information, we propose the *shared MDP* model, which allows private and shared information to be expressed separately.

1 Introduction

In today's world, people are increasingly connected to many other people and many sources of information, providing more opportunities than ever for tasks such as working more efficiently with others and discovering helpful information. Unfortunately, the vast scale of information can be overwhelming. As computers become more powerful and pervasive, software in the form of personal assistant agents can provide support in many situations.

For example, with the increased deployment and use of space assets, a number of interesting and challenging problems have come to the fore. The persistent nature of space surveillance (i.e., 24/7 operations), the mass of data, the varied data sources, and the heterogeneous needs of consumers and producers throughout the community all point to a pressing need for an enhanced Space Situational Awareness (SSA) picture, one that can only be achieved by a coordinated, collaborative approach to the collection, dissemination and sharing of information.

For the foreseeable future, the challenges in the U.S. National Space Policy will demand human-in-the-loop participation. This is particularly relevant given the amount and importance of data and knowledge not held in any database, or streaming in bits

through the ionosphere, but that resides in the minds of individuals or the institutional knowledge of a team. The criticality of an enhanced situational awareness and its subsequent deployment suggests that we integrate these methods within the existing workflow, obviating the need to add additional tools.

The solution to this problem requires a coherent, integrated approach, viewing the space domain as a socio-technical system, within which the human plays an integral part. Several diverse agencies are stakeholders and produce and/or consume information related to space. Each agency has its own organizational structure and protocols, so the solution must be versatile enough to allow for effective inter-agency collaboration while still maintaining the standard practices of each. Further, the human operator network alone is not sufficient to create adequate SSA. There is an enormous amount of data recorded from sensors on satellites, ground stations and other periodic collectors. The human operator is overloaded by the transmission, capture, cataloging, and sense making of these data. The optimal solution for enhanced SSA is a synergistic, collaborative effort between networks of humans and automated agents.

The Space Collaboration via an Agent Network (SCAN) approach enables proxy agents to represent each of the stakeholder agencies and facilitate collaboration by identifying opportunities and methods for collaboration. This allows humans to have access to large amounts of data through their agents, while personalizing them given the users specific preferences and capabilities. These agents can then communicate with each other with varying levels of input from the user for tasks ranging from retrieving information, securing the services of another agency or team formation for complex tasks.

Many models have been developed for solving decentralized problems requiring cooperation. For SCAN, we assume a sequential problem in which a series of decisions need to be made, each affecting the next. Limiting these models to those based on decision theory, a large number of models can still be used [2, 4, 5, 10, 14]. Nevertheless, each of these models makes assumptions that cause it to be inappropriate for a domain such as SCAN. These assumptions include: a full model of the problem with the value other agents derive from all actions or the results of these actions, perfect knowledge of the current state of the system (or a common estimate), a shared objective function, a centralized solution method. An ideal model for our domain would relax these assumptions to permit only local knowledge and solution methods, self-interested agents, model uncertainty and independence between groups of agents.

The remainder of this paper is organized as follows. We first discuss the SCAN domain in more detail as well as the progress made to date. We also discuss several models that could be used to represent this problem and their shortcomings. Finally, we discuss key assumptions that we are interested in relaxing in these approaches and the challenges involved in doing so. The goal in this project is to produce a system of personal assistant agents with accompanying solution methods that maintain tractability and high solution quality.

2 Overview of SCAN approach

The approach used in SCAN represents each of the stakeholder agencies for space with proxy agents and facilitates collaboration by identifying opportunities and methods for

collaboration. It demonstrates the suitability of our human-agent solution to address the three critical aspects above, that is: 1) Knowledge and mental models of human operators, 2) Collaboration methods and barriers to collaboration and 3) Integrating solutions that conform to preferred workflows.

A variety of factors shape the opportunities for collaboration and the constraints on collaboration. The SCAN Agent architecture incorporates these components into the SCAN Proxy Agent Architecture. The need to collaborate varies based on an interaction of the demands of the mission, the architecture of the team and its resources, and the distribution of expertise within the team. Collaboration becomes necessary when mission-critical expertise is distributed among multiple people, and resources and responsibilities are likewise divided between people.

A team's ability to collaborate depends on a number of factors, including available technology, team members collaboration skills, and team composition. The factors affecting the ability to collaborate directly affect the products of collaboration (e.g., assessments and plans), as well as the apparent coordination of the team as a whole. Collaborative critical thinking is intended to provide active support to teams above and beyond the three factors affecting a team's ability to collaborate. Figure 1 shows a top-level diagram of our proposed solution: Space Collaboration via an Agent Network (SCAN). It shows proxy agents, each of which represents a stakeholder for space. We implemented an initial version of this infrastructure by developing agent interactions to carry out the use case(s), and tested the resultant model by generating results to show dynamic constraints, changing parameters, and making modifications to the use cases(s) that show the benefit of agent use. This solution will fit integrally into the current workflow, without the need for additional tools for users to learn, and will make privacy issues explicit.

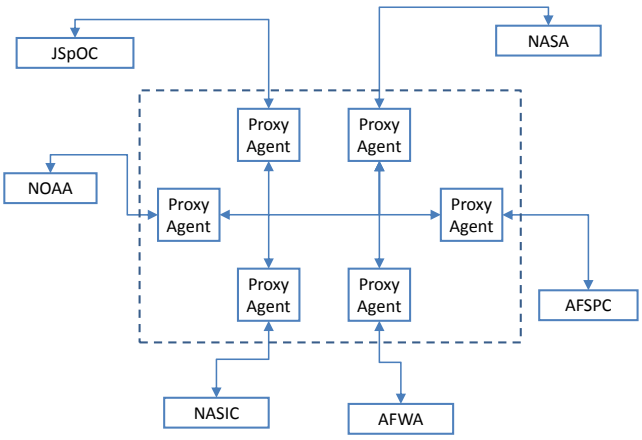

Fig. 1. Proxy agents being used to facilitate collaboration between heterogeneous stakeholders.

Note that our approach to SCAN is to have the human remain in control and drive the collaboration. The proxy agent is designed to facilitate the human operators and point them in the right direction to improve interagency collaboration. Human operators will not be surprised or superseded by the SCAN proxy agent's actions.

3 The SCAN framework

In conceiving an agent-based architecture, we identified the following benefits to individual agencies of agent-based collaboration: 1) Fusion/Integration of agency plans, 2) Locating potential collaboration opportunities, 3) Forewarning: An agency may anticipate future events based on an alert from another agency, 4) Increased Quality/Accuracy of data, 5) Facilitating levels of automated collaboration, 6) Detecting collaboration failures, 7) Adjusting to realtime changes in policy/capabilities, 8) Learning preferences of agencies, 9) Persuading agencies to cooperate through incentives. We selected areas 1), 2), and 7) as areas for further study.

3.1 Use case

In order to study the above, we constructed a working Use Case. The requirements for the Use Case were: Scalability: The initial Use Case should describe a simple problem that can easily scale to more complex problems. This allows us to demonstrate our solution initially in an easily controllable environment, but also set the stage for our solution to handle a larger set of more complex problems. Generalizability: The Use Case should be representative of a general class of collaboration problems, so that the solution will be applicable to the needs of other agencies as well as those mentioned in the Use Case. The Use Case was based on interviews with space weather analysts from the Air Force Weather Agency (AFWA), and their information needs as primary producers of information on environmental effects to agencies such as the National Air and Space Intelligence Center (NASIC), the Joint Space Operations Center (JSpOC), and the Air Operations Center (AOC)

We identified two important facets to examine in developing the use case: (P1) How does the SCAN agent know when to initiate collaboration? (P2) What is the message structure necessary to handle collaboration? The message structure will be driven by the type of collaboration, therefore (P1) must be considered before (P2). For (P1), the following sequence of steps should take place in the Use Case: (S1) NASIC's communication to the satellite goes down (S2) NASICs SCAN agent attempts to diagnose the reasons (S3) Two causes are identified, a solar flare or a system problem (S4) Collaboration IDs are generated for each of the two causes (S5) NASICs SCAN agent sends a query to the other agents to collaborate (S6) Agents on behalf of the other agencies provide an affirmative response to the collaboration request (S7) The agencies are connected.

This protocol is motivated by Signaling System 7 (SS7) of the telephone network [13] which separates data from service. This property is particularly desirable, as it allows an agency to limit the amount of data it discloses to non-collaborating agencies. In the future, the topology of the SCAN agents will be further refined.

3.2 Basic model

From the perspective of a SCAN agent receiving collaboration requests, this agent is called the Collaboration Facilitator (CF). The requesting agent is called the CR. As stated previously, each SCAN agent will have a model of the other agents and agencies. The model couldl be used to derive such information as an arrival rate of resource requests. Based on the model of the requests coming in, the model couldl also predict a service rate of these requests. The model will also predict the benefits of each collaboration from each organization. Likewise, from the perspective of the CF, it is possible that a collaboration initiated by another agency will require collaboration with a third agency, and that only the CF and not the CR knows about this collaboration. In this case, it will NOT be the role of the CF to request collaboration with the third agency. This could reproduce the well-known "dining philosophers" problem from Operating Systems literature which is essentially that a bottleneck results when multiple processes compete for a limited set or resources. As a consequence of that bottleneck, system performance is degraded. Instead, the CF agent passes the information about the 3rd party back to the CR agent and the CR agent submits the request for collaboration to the third agency.

The motivation behind this model is that a SCAN agent receiving a collaboration request may be able to make decisions like: (1)"Well, technically I could collaborate you; after all I have enough resources to do it. However I know I'm about to get some high priority requests from important agencies soon, so... collaboration request denied." (2)"I can't collaborate with you right now. But my model says I'll be free in a few hours, do you want to book my time in advance?"

4 Multiagent models of interest

To model the type of problems described above, we need to consider the following requirements: sequential decisions, uncertainty regarding outcomes and other agents' decisions and decentralized control. Sequential decisions are needed because agents' decisions at the current time will affect the future outcomes. That is, suggesting to collaborate with one agency rather than another may have consequences in terms of time and resources required, affecting future collaboration possibilities. Uncertainty is present in these scenarios as well considering that information may be unavailable or outside forces could cause the situation to change. Also, users may decline collaboration and the actions of the other agents are often unseen, causing uncertainty about the other agents. The system must also be decentralized for similar reasons. Agents must make choices based solely on local information due to the lack of updates from other agents. This local information may also be uncertain or incomplete information about the human user or other humans and agents in the system.

This type of modeling can be accomplished with decision-theoretic approaches. In general, these representations have probabilities which represent the uncertainty in action outcomes and seek to maximize an objective function in an attempt to optimize the sequence of decisions by the agents. As agents are built for ever more complex environments, methods that consider the uncertainty in the system have strong advantages. Developing effective frameworks for reasoning under uncertainty is a thriving research

area in artificial intelligence and we discuss some of these approaches below. These models are briefly discussed below and summarized in Table 1.

4.1 DEC-POMDPs

A decentralized partially observable Markov decision process (DEC-POMDP) [4] can be defined with the tuple: $\langle I, S, \{A_i\}, P, R, \{\Omega_i\}, O, T\rangle$ with I, a finite set of agents, S, a finite set of states with designated initial state distribution b_0, A_i, a finite set of actions for each agent, i P, a set of state transition probabilities: $P(s'|s, \boldsymbol{a})$, the probability of transitioning from state s to s' when the set of actions $\boldsymbol{a}$ are taken by the agents, R, a reward function: $R(s, \boldsymbol{a})$, the immediate reward for being in state s and taking the set of actions, $\boldsymbol{a}$ Ω_i, a finite set of observations for each agent, i, O, a set of observation probabilities: $O(\boldsymbol{o}|s', \boldsymbol{a})$, the probability of seeing the set of observations $\boldsymbol{o}$ given the set of actions $\boldsymbol{a}$ was taken which results in state s', T, a horizon or number of steps after which the problem terminates.

A DEC-POMDP involves multiple agents that operate under uncertainty based on different streams of observations. At each step, every agent chooses an action based on their local observation histories, resulting in a global immediate reward and a local observation for each agent. Note that because the state is not directly observed, it may be beneficial for the agent to remember its observation history. A *local policy* for an agent is a mapping from local observation histories to actions while a *joint policy* is a set of local policies, one for each agent in the problem. The goal is to maximize the total cumulative reward until the horizon is reached, beginning at some initial distribution over states. In the infinite-horizon problem, T is infinity and the decision making process unfolds over an infinite sequence of steps. In order to maintain a finite sum over the infinite-horizon, in these cases a discount factor, $0 \leq \gamma < 1$, is employed.

The DEC-POMDP model is very general, but has a very high complexity (NEXP-complete[1]). Algorithms for solving DEC-POMDPs also typically assume that all model parameters are known in advance and the solution is found in a centralized manner, providing policies that each agent can implement in a decentralized way. This model also assumes no communication (except as part of the observations) and full cooperation between the agents. The high complexity and large number of assumptions make the DEC-POMDP model currently inappropriate for the scenarios we are interested in for SCAN with a large number of agents with limited knowledge of the full problem model.

Communication and learning have also been studied in the DEC-POMDP model. Recent work has examined using communication for online planning in DEC-POMDPs [18]. This type of work could be useful in our context because communication is used when inconsistencies are detected between the agents. Unfortunately, it still requires knowledge of the full model and a DEC-POMDP to be solved for some number of steps. Rather than assuming the model is known, it can also be learned or policies can be learned directly [7, 16]. Direct policy learning may be an appropriate approach in that it does not require a model, but in order to calculate the gradient the model must be sampled. This sampling requires many instances and still is not guaranteed to converge to an equilibrium.

[1] This results in doubly exponential complexity as long as P!=NP.

4.2 DEC-MDPs and MMDPs

We can restrict DEC-POMDPs in various ways to simplify solution calculations. For instance, we can assume that each agent can fully observe its own local state, but not the global state. This is a form of the DEC-MDP model [2], which has been shown to have more efficient solutions in some cases, but in general has the same complexity as the general model (NEXP-complete) [4]. If we consider problems in which the agents have independent transitions and observations and a structured reward model (IT-IO DEC-MDPs), the complexity drops to NP-complete [3]. Recent work has shown that subclasses of DEC-POMDPs which have independent rewards, but dependent observations and transitions as well as those with certain structured actions retain the same complexity as the general problem [1]. Some other modeling assumptions and the resulting complexity are studied in [9], but none of these seems appropriate for our case as they continue to assume centralized solution methods and knowledge of the model.

DEC-MDPs can be further simplified by assuming that the state of the problem is fully observed by each agent, resulting in a multiagent MDP (MMDP) [5]. There has been work on efficient planning solutions for these problems (modeled as factored MDPs) [10], which allow algorithms to exploit independence between agents. While this is a useful quality (which is discussed in more detail in the context of ND-POMDPs), centralized solution methods are used and full model knowledge is required, limiting the applicability of these models to our scenario.

Model	Pros	Cons
DEC-POMDPs	Very rich model of cooperative agents	Assumes known model parameters, fully cooperative, high complexity
IO-IT DEC-MDPs	Somewhat rich, less complex than full model	Limiting model assumptions (interact only through rewards), Assumes known model parameters fully cooperative
MMDPs	Low complexity	Limiting model assumptions (centralized knowledge), Assumes known model parameters, fully cooperative
ND-POMDPs	Exploits locality, distributed solution, less complex than full model	Limiting model assumptions (very limited interaction), Assumes some model parameters fully cooperative
Graphical games	Allows self interest, exploits locality	Assumes some model parameters, not sequential
I-POMDPs	Allows self interest, exploits locality	Assumes known model parameters, high complexity

Table 1. Model pros and cons for distributed personal assistant domains.

4.3 ND-POMDPs

Another way of simplifying DEC-POMDPs is to assume agents interact in a limited way based on locality. This assumption often considers agents that can only interact with their neighbors and thus is more scalable in the number of agents in the problem. This has been studied using the networked distributed POMDP (ND-POMDP) model for the finite-horizon case [14] as well as a more general factored models [15, 17].

The ND-POMDP model is similar to the DEC-POMDP model except for the following: states can be factored $S = \times S_i \times S_u$ for each set of states for each agent i and an unaffected state set, actions and observations can be factored similarly with $A = \times A_i$ and $\Omega = \times \Omega_i$ for each agent, transition independence where the unaffected state and each agent transition independently $P(s'|s,a) = P(s'_u|s_u) \prod_i P(s'_i|s_i,a_i)$ and observation independence where $O(o|s',a) = \prod_i O_i(o'_i|s_i,a_i)$. Also, rewards can be broken up based on neighboring agents and summed as $R(s,a) = \sum_l (s_{l1},\ldots,s_{lk},s_u,\langle a_{l1},\ldots,a_{lk}\rangle)$ where l represents a group of $k = |l|$ neighboring agents. The ND-POMDP model also assumes that a the belief state (the state distribution estimate) is known by all agents.

A solution to an ND-POMDP can be computed in a distributed manner using message passing to converge to a local optimum. This approach can be very efficient in cases where the agents are only loosely connected. Nevertheless, strong assumptions are still made such as knowing the model in the local neighborhood and additive rewards across neighborhoods.

4.4 Game theoretic representations

If we assume that agents may have different interests that may not necessarily align perfectly (which is likely to be the case in many real-world scenarios), we could use a game theoretic approach. A model that retains the idea of agents loosely connect into neighborhoods is the graphical game [12]. This approach combats the tabular (and thus exponential) representation of n-agent games by using an undirected graph where two agents are connected if they can affect each others' payoffs. Assumptions in this model are that each agent can calculate a best response, which means that it knows the payoffs for the different actions given the other agents' choices. This is also not a sequential model, so all possible horizon T policies would have to be considered as actions, resulting in an exponential increase in action size and intractability to solve.

Sequential game theoretic models have also been studied. A generalization of DEC-POMDPs to the case where agents may have different rewards results in a partially observable stochastic game (POSG). Thus, if we assume POSGs with common payoffs for all agents, it is equivalent to the DEC-POMDP model [11]. This model relaxes the assumption that all agents are fully cooperative, but retains the intractability and knowledge of the model assumed by the DEC-POMDP. Another game theoretic view of multiagent POMDPs from the Bayesian perspective is the interactive POMDP (I-POMDP) model [8]. An extension of the I-POMDP model to graphical representations, allowing agent independence to be exploited is the interactive dynamic influence diagram (I-DIDs) [6]. While both of these models are interesting in that they can represent problems with estimates of other agent behavior based on observations that are seen, a full model is again assumed and complexity is at least as high as the corresponding DEC-POMDP models.

5 Dealing with private information

One weakness of the decision-theoretic models discussed in the previous section is the assumption that agents will have full knowledge of each others models. This is somewhat mitigated when only local agents are considered (as in an ND-POMDP), but even these local agents must share their information before a decentralized solution can be found. In a real-world scenario, such as SCAN, we cannot assume the agents will have knowledge of the other agent's model: states, actions, observations, let alone their transition and observation probabilities or rewards values.

There may be privacy or security reasons for not sharing this information. For instance, individuals may not want to disclose their full schedules or preferences and organizations do want to share proprietary information. More specifically, Chris may have a preference for meeting with Paul over Nathan which he would rather not divulge (especially to Nathan!). Also, there may be several other projects that a person is working on that have no direct relationship to a possible collaboration between entities, but they be affected in some way. For example, I may not want to share my schedule for all projects that I am working on, but I would consider changing my schedule to collaborate on a particularly important or interesting project. This could change depending on who is working on the project, when it is being scheduled, different travel locations etc. Nevertheless, planning must be accomplished based on this limited model information. To represent these problems, we describe a new model, the *shared MDP*, which is an initial step towards dealing with shared and private information.

5.1 Shared MDP model

We present the shared MDP model, but this could be extended to the POMDP case. In a shared MDP, each agent, i, consists of an MDP with S_i, A_i, P_i, R_i and T_i, the states, actions, transitions, rewards and horizon for agent i's model. There may be some set of states which are *shared* by a set of agents. For agent i, we denote the set of shared states as S_i^s. In this case, the transitions and rewards for the shared states depend on all agents that are in the shared states at that time.

More formally, we consider $S = \cup_i S_i$ the full set of states for all agents, $D(s) = \{i | s \in S_i\}$ the possible set of decision-makers at state s as well as joint reward $R(s, \boldsymbol{a_{D(s)}})$ and transitions for each agent i $P_i(s'|s, \boldsymbol{a_{D(s)}})$. We define a *private state* as an agent's state that is not shared by any other agent. That is, state s is private for agent i if $D(s) = i$. Note that $D(s)$ represents all agents that have state s in their MDP, but some or all of these agents may be in a private state at the given time and therefore do not affect the rewards and transitions of the other agents. In these cases, we can replace the action of the agents in private states with a dummy action or omit them altogether.

Proposition 1. *A shared MDP with no shared states is equivalent to a set of independent MDPs.*

Proof is straightforward and thus omitted.

Proposition 2. *A shared MDP with no private states and common transition functions is equivalent to an MMDP.*

Proof is straightforward and again omitted. In general, agents will have full knowledge of the (common) state of the system as well as the joint transitions and rewards. This is also the case if agents' MDPs are subsets of other agents' MDPs.

The interesting (and realistic) case occurs when each agent has a set of private states that are not shared with the others. A shared MDP could also be solved centrally as one large MDP, but this requires sharing private and secure information with a central agency (by communicating the details of the private model as well as the shared one). A central approach would also introduce a single point of failure and make it more difficult to add or remove agents over time. As a result, we explore fully decentralized solution methods.

5.2 Example shared MDP

An example shared MDP with two agents is shown in Figure 2. Here, $S_1 = \{s^1, s^2\}$ and $S_2 = \{s^2, s^3\}$, resulting in shared state s^2 and private states s^1 and s^3 for agents 1 and 2 respectively. We also assume $A_1 = A_2 = \{a^1, a^2\}$.

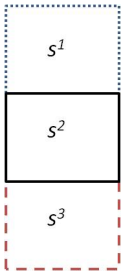

Fig. 2. Shared MDP with two agents. Agent 1 contains states 1 and 2, while agent 2 contains states 2 and 3. State 2 is a shared state.

The dynamics for agent 1 in the private state will be that a^1 causes the agent to stay in the private state, $P(s^1|s^1, a^1) = 1$, while a^2 causes the agent to transition to the shared state $P(s^2|s^1, a^2) = 1$. Agent 2 has the same dynamics $P(s^3|s^3, a^1) = 1$ and $P(s^2|s^3, a^2) = 1$. Note the agent subscript is omitted for private states. In the shared state, if both agents take the same action, each agent transitions back to its private state. That is, $P_1(s^1|s^2, a_1^1, a_2^1) = 1$, $P_1(s^2|s^2, a_1^1, a_2^2) = 1$, $P_1(s^2|s^2, a_1^2, a_2^1) = 1$, $P_1(s^1|s^2, a_1^2, a_2^2) = 1$. The same can be written for agent 2's P_2. When only one agent is in the shared state, its dynamics are the same as those for the private state, $P_1(s^1|s^2, a^1) = 1$, $P_1(s^2|s^2, a^2) = 1$, $P_2(s^3|s^2, a^1) = 1$ and $P_2(s^2|s^2, a^2) = 1$.

If $R_1(s^1, a^1) = R_1(s^1, a^2) = 10$, $R_2(s^3, a^1) = R_2(s^3, a^2) = -10$ and for both agents, $R(s^2, a_1^1, a_2^1) = R(s^2, a_1^1, a_2^2) = R(s^2, a_1^2, a_2^1) = R(s^2, a_1^2, a_2^2) = 0$, and when only one agent is in s^2 $R(s^2, a^1) = R(s^2, a^2) = -2$ then there is no cooperative solution to this problem in which both agents attain their highest values. This is because agent 1 will gain higher reward for being in its private state and thus will attempt to

coordinate with agent 2 to choose the same action to transition accordingly. On the contrary, agent 2 will gain higher value for staying in the shared state and thus will attempt to choose a different action than agent 1. These differences can be seen in the optimal policies for each agent's MDP. Assuming an infinite horizon problem with discount of 0.9, agent 1 produces a value of 100 starting in s^1 by always choosing action a^1, while if both agents start in s_2 a value of 90 can be achieved if both agents choose the same action on the first step and then agent 1 chooses a^1 thereafter. For each step that the same action is not chosen, value is lost by agent 1. This is in contrast to agent 2, who achieves at most -9 after starting in s^2 and cooperating with agent 1 to choose the same action, while it could attain a value of 0 if both agents always chose different actions. Starting in s^3, agent 2 can attain a value of -10 by choosing a^2 and choosing the opposite action as agent 1 if that agent is in the shared state s^2. If agent 1 is not in the shared state for any other step, a value of -28 can be attained by always choosing a^2.

5.3 Solutions for shared MDPs

It can be shown that in the shared MDP example above, starting from s^2, agent 1 can ensure that it receives at least 80 by randomly choosing either action in s^2. A similar worst case value can be found for agent 2, which again would randomly choose actions in s^2 when agent 1 is present to stay in that state as long as possible. This solution can be found by transforming a shared MDP into a competitive problem. This is appropriate since the presence of private states may cause each agent's value to be different for taking the same set of actions in the same state (since the value of an action in a state depends on on all subsequent rewards that can be obtained, shared or private).

To produce a solution to any shared MDP, it is necessary to combine solutions for the private and shared aspects of each agent's MDP. Algorithm 1 shows how a policy for agent i can be found. This algorithm constructs and solves a game (for the Nash equilibria, NE) from solving MDPs that consider all possible policies for all agents for shared states of the shared MDP. That is, it first constructs a PolicySet, which consists of all mappings from shared states S_i^s of agent i to actions for all agents which also contain that shared state. For each agent and each of these policy mappings, an MDP is solved which consists of the private states of the agent's MDP and the fixed policy given by PolicySet for the shared states. These MDP values are then used to determine a Nash equilibrium (if we assume a unique one is found) which serves as the policy for agent i.

In the case that multiple equilibria are found, coordination mechanisms or equilibria refinements can be used. In order to coordinate, the agents could share the equilibria found for the shared states to determine a single one that can be played (ensuring that policies for the shared states are known to all agents). Also, more efficient ways for finding policies in Algorithm 1 can be utilized. This could include not considering all possible policies for the other agents in PolicySet, but rather some subset, which could be based on heuristic value. If matching equilibria cannot be found, the algorithm could be repeated (with different parameters) until a solution is found.

In practice, we are interested in finding a pure strategy Nash equilibrium as a deterministic solution is preferred by our users. This will often allow us to determine a

Algorithm 1 SOLVESHAREDMDP(i)

```
PolicySet ← {}
for all s, ∈ S_i^s do
    Policy(s) ← null
    for all j, ∈ D(s) do
        for all a, ∈ A_j do
            PolicySet.Add(s_j,a)
for all p, ∈ PolicySet do
    Value(S_i^s,p) ← SolvePrivateMDP(S_i^s, p)
Policy ← SolveForNE(Value)
return Policy
```

solution to the problem much more quickly and only resort to using Algorithm 1 when a pure strategy NE cannot be found. We will attempt to discover a NE with iterated best response as shown in Algorithm 2. That is, each agent begins with a random policy (or one previously found) for the shared states and, while keeping the other agents' policies fixed, an agent determines a best response for its MDP. This new best response policy for the shared states is now fixed, along with other agents' policies, while the next agent determines a best response. This continues until no agent changes its policy, resulting in a Nash equilibrium. Unfortunately, it might be the case that there are no pure strategy NE in the given problem, causing the agent policies to continue changing forever. Currently, the algorithm checks for these oscillations by making sure the total number of changes by the agents is less than the total number of possible policies (m^n where m is the number of strategies and n is the number of agents). More sophisticated analysis of oscillation could also be used to determine when Algorithm 1 should be used to find a mixed strategy. In practice, it will often be the case that several pure strategy NE exist, allowing a solution to be found quickly.

Algorithm 2 SOLVESHAREDMDPDETERMINISTIC

```
Policies ← RandomPols()
Converged ← 0
Oscillate ← false
while Converged < n − 1 or Oscillate < possSolutions do
    for all i ∈ I do
        Policies(i) ← BestResponse(Policies(−i))
        if IsChanged(Policies(i)) then
            Converged ← 0
        else
            Converged++
        Oscillate++
if Converged=n-1 then
    return Policies
else
    return false
```

6 Conclusion

In this paper, we discussed a real-world domain for facilitating collaboration between organizations and people, the SCAN proxy agents. We described the characteristics of this domain which require sequential and decentralized decision-making. Various decision-theoretic models for representing these problems are presented along with their shortcomings in this domain. In order to begin to address these shortcomings, we present one approach for representing and solving problems with private and shared information as a shared MDP. This is an initial step towards providing decentralized solutions to sequential collaboration problems with private information. In the future, we are interested in further extending decision-theoretic models so they can be applied in this context. This work will include other model assumptions that better fit the real-world data that is generated during this project. These algorithms will be implemented and tested in the SCAN domain.

References

1. Allen, M., Zilberstein, S.: Complexity of decentralized control: Special cases. In: Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C.K.I., Culotta, A. (eds.) Advances in Neural Information Processing Systems, pp. 19–27. 22 (2009)
2. Becker, R., Lesser, V., Zilberstein, S.: Decentralized Markov Decision Processes with Event-Driven Interactions. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems. pp. 302–309. New York, NY (2004)
3. Becker, R., Zilberstein, S., Lesser, V., Goldman, C.V.: Solving transition-independent decentralized Markov decision processes. Journal of AI Research 22, 423–455 (2004)
4. Bernstein, D.S., Givan, R., Immerman, N., Zilberstein, S.: The complexity of decentralized control of Markov decision processes. Mathematics of Operations Research 27(4), 819–840 (2002)
5. Boutilier, C.: Sequential optimality and coordination in multiagent systems. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence. pp. 478–485. Stockholm, Sweden (1999)
6. Doshi, P., Zeng, Y., Chen, Q.: Graphical models for interactive POMDPs: Representations and solutions. Journal of Autonomous Agents and Multi-Agent Systems 18 (2009)
7. Dutech, A., Buffet, O., Charpillet, F.: Multi-agent systems by incremental gradient reinforcement learning. In: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence. pp. 833–838 (2001)
8. Gmytrasiewicz, P.J., Doshi, P.: A framework for sequential planning in multi-agent settings. Journal of Artificial Intelligence Research 24, 24–49 (2005)
9. Goldman, C.V., Zilberstein, S.: Decentralized control of cooperative systems: Categorization and complexity analysis. Journal of Artificial Intelligence Research 22, 143–174 (2004)
10. Guestrin, C., Koller, D., Parr, R.: Multiagent planning with factored MDPs. In: Advances in Neural Information Processing Systems, pp. 1523–1530. 15 (2001)
11. Hansen, E.A., Bernstein, D.S., Zilberstein, S.: Dynamic programming for partially observable stochastic games. In: Proceedings of the Nineteenth National Conference on Artificial Intelligence. pp. 709–715. San Jose, CA (2004)
12. Kearns, M., Littman, M.L., Singh, S.: Graphical models for game theory. In: Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (2001)

13. Modarressi, A., Skoog, R.: Signalling system no. 7: A tutorial. IEEE Communications Magazine July (1990)
14. Nair, R., Varakantham, P., Tambe, M., Yokoo, M.: Networked distributed POMDPs: a synthesis of distributed constraint optimization and POMDPs. In: Proceedings of the Twentieth National Conference on Artificial Intelligence (2005)
15. Oliehoek, F.A., Spaan, M.T.J., Whiteson, S., Vlassis, N.: Exploiting locality of interaction in factored Dec-POMDPs. In: Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems. Estoril, Portugal (2008)
16. Peshkin, L., Kim, K.E., Meuleau, N., Kaelbling, L.P.: Learning to cooperate via policy search. In: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence. pp. 489–496 (2000)
17. Witwicki, S.J., Durfee, E.H.: Influence-based policy abstraction for weakly-coupled Dec-POMDPs. In: Proceedings of the Twentieth International Conference on Automated Planning and Scheduling. Toronto, Canada (2010)
18. Wu, F., Zilberstein, S., Chen, X.: Multi-agent online planning with communication. In: Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling. Thessaloniki, Greece (2009)