A Compact Mathematical Formulation For Problems With Structured Agent Interactions

Hala Mostafa
Computer Science Department
University of Massachusetts, Amherst
hmostafa@cs.umass.edu

Victor Lesser
Computer Science Department
University of Massachusetts, Amherst
lesser@cs.umass.edu

ABSTRACT

The general problem of calculating policies in decentralized POMDPs is known to be NEXP-complete. One way of dealing with this prohibitive complexity is identifying a sub-class of the general problem that is more tractable to solve, but still of practical interest. One such sub-class consists of problems exhibiting structured transition and reward interactions among agents, and is modeled using Event-Driven Interactions with Complex Rewards (EDI-CR). In this paper, we propose a Mixed Integer Linear Program formulation of EDI-CR instances. The key insight we use is that from one agent's perspective, most action sequences of another agent have the same effect, thereby allowing us to treat them similarly and use fewer variables in the formulation. Experimental results show that our formulation is more compact, and leads to faster solution times, than formulations ignoring the structure of interactions.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Coherence and coordination

General Terms

Algorithms, Performance

Keywords

Coordination, Optimization

1. INTRODUCTION

Decision-theoretic models have been widely used to model multi-agent decision making problems. One of the most general models is Decentralized Partially Observable Markov Decision Process (DEC-POMDP) which can capture situations where the agents do not have access to the true world state, but receive observations correlated with it.

But with this expressive power comes computational intractability; calculating agent policies in DEC-POMDP is known to be NEXP-Complete [5]. One way of dealing with this prohibitive complexity is identifying sub-classes of the general problem that are more tractable to solve, but still of practical interest. Among the sub-classes that have been proposed in the literature are DEC-MDPs, which assume that pooling observations of all agents uniquely determines the underlying world state. A special case of DEC-MDPs is where the state space factors into a state per agent and a world state that is not under any agent's control. Different models make different assumptions about how each agent's transitions and rewards are affected by other agents (e.g. Transition Independent DEC-MDPs [4], Networked Distributed POMDPs [9]).

The sub-class we address in this paper consists of problems exhibiting structured transition and reward interactions among agents. In other words, an agent's rewards and transition probabilities cannot be said to be completely independent of what other agents do, but the interactions among agents are relatively few compared to all possible ways in which one agent can affect the others, and involve a well-defined set of states and actions. For example, consider a robotic team dealing with a building on fire. One agent is in charge of putting out the fire, another locates and evacuates survivors and a third delivers first aid to the injured. The actions of each agent mainly affect only itself; the first agent's decision of how to attack the fire and what kind of extinguisher to use mainly affect its own progress in fire fighting. Likewise, the paramedic agent's choice of the kind of first aid care to give to the injured mainly affects its own progress towards getting them out of critical conditions. However, the fire-fighting agent's decision of when to secure a given area affects how easy it will be for the rescue agent to locate survivors in that area.

The kind of interaction discussed above can of course be modeled using DEC-MDPs with factored state space. However, this representation does not exploit the fact that the agent's sub-problems are largely independent, and therefore cannot achieve any representational savings out it. More importantly, a DEC-MDP would obscure all structure, making it difficult for a solution algorithm to exploit it and achieve computational savings. To deal with these issues, the model Event-Driven Interactions with Complex Rewards (EDI-CR) was developed for problems with structured transition and reward interactions. For cooperative agents, EDI-CR is a much more compact representation than general DEC-MDPs [7]. For the case where agents are self-interested, EDI-CR has been shown to be much more compact than general extensive form games [8]. EDI-CR has the same expressive power as DEC-MDPs with local observability (cooperative case) and general extensive form games (selfish case).

In this paper, we exploit structured interactions to obtain a compact formulation of EDI-CR as a Mixed Integer Linear

The Sixth Annual Workshop on Multiagent Sequential Decision-Making in Uncertain Domains (*MSDM-2011*), held in conjunction with *AAMAS-2011* on May 3, 2011 in Taipei, Taiwan.

Program (MILP) and show the representational and computational savings achieved by doing so. The next section contains background material on EDI-CR and the sequence form representation of a policy. Section 3 reviews an existing approach to formulating DEC-MDP as MILP. In section 4, we present our MILP formulation and show how we used the special structure of our problem to reduce the number of variables in it. Section 5 experimentally shows the savings obtained from our formulation compared to one that does not exploit structure. A survey of related work is given in Section 6, followed by conclusions and directions for future work.

2. BACKGROUND

In this section, we give background material about the decision-theoretic model that will be used in the rest of this paper. We also explain the policy representation that we will use in our mathematical formulation.

2.1 Factored DEC-MDP

A *factored* DEC-MDP is where the world state can be expressed as a tuple of locals states, one per agent. *Local observability* means that instead of only receiving a noisy observation regarding its local state, an agent can actually observe its own state with certainty (but not necessarily those of other agents). Formally, an n-agent DEC-MDP with factored state and local observability is a tuple $< S, A, P, R, T >$ where

- $S = S_1 \times S_2 \times ... \times S_n$ is the set of factored world states
- $A = A_1 \times A_2 \times ... \times A_n$ is the set of joint actions
- $P : S \times A \times S \rightarrow [0, 1]$ is the transition function $P(< s'_1, s'_2, ..., s'_n > \mid < s_1, s_2, ..., s_n >, < a_1...a_n >)$ is the probability of transitioning to new local states $< s'_1, s'_2, ..., s'_n >$ when the joint action $< a_1...a_n >$ is taken in local states $< s_1, s_2, ..., s_n >$
- $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward function. $R(< s_1, s_2, ..., s_n >, (a_1...a_n), < s'_1, s'_2, ..., s'_n >)$ is the reward obtained from taking joint action $< a_1...a_n >$ when the agents local states are $< s_1, s_2, ..., s_n >$ and transitioning to new local states $< s'_1, s'_2, ..., s'_n >$
- T is the time horizon of the problem

It is important to note that in spite of the state space being factored, and of the fact that each agent can observe its local state, transitions and rewards depend on *joint* actions and states. So even though an agent observes its state, its rewards and next local state are affected by all other agents.

2.2 EDI-CR

Event-Driven Interactions with Complex Rewards (EDI-CR) is a model developed for problems with structured transition and reward interactions [7]. It builds on the intuition that for a group of agents that are largely independent of each other, it is more natural to describe each agent's decision problem separately and list the interactions that tie these processes together. Formally, an EDI-CR instance is a tuple $< S, A, P_{1..n}, R_{1..n}, \rho, \tau, T >$ where:

- $S = S_1 \times S_2 \times ... \times S_n$ is the set of factored world states
- $A = A_1 \times A_2 \times ... \times A_n$ is the set of joint actions
- $R_i : S_i \times A_i \rightarrow \mathbb{R}$ is agent i's local reward function
- $P_i : S_i \times A_i \times S_i \rightarrow [0, 1]$ is agent i's local transition function
- $\rho = \{< (s_{k_1}, a_{k_1}), ..., (s_{k_m}, a_{k_m}), r_k >_{k=1..|\rho|}\}$ is the set of reward interactions. Each element is an interaction that lists the state-action pairs and the additional reward/penalty obtained when the specified agents take the specified actions in the specified states. An interaction can involve any subset of agents.
- $\tau = \{< (s_{k_1}, a_{k_1}), (s_{k_2}, a_{k_2}), p_k >_{k=1..|\tau|}\}$ is the set of transition interactions. The k^{th} entry specifies the new transition probability p_k of the state-action pair of agent k_2 when agent k_1 does its specified state-action pair before the affected agent makes its transition.
- T is the time horizon of the problem

The individual states, actions, transitions and rewards describe the dynamics of each agent's decision process, while ρ and τ capture the interactions among them. Previous work showed that EDI-CR's explicit representation of interactions does not diminish its applicability; EDI-CR has the same expressive power as DEC-MDPs with local observability [7]. It is capable of representing problems with arbitrary interactions among agents, in which case there would be entries in ρ and τ for every combination of actions and states to mirror the fact that all actions affect all agents. The benefit of using EDI-CR over DEC-MDP depends on the degree of coupling among agents (the amount of interaction, as given by the sizes of the sets ρ and τ in the definition). The tighter the coupling, the less advantage we obtain from using EDI-CR.

In EDI-CR and DEC-MDP, as in any setting with cooperative agents, the goal is to compute a joint policy that maximizes total cumulative rewards of all agents. The exact way a policy is represented is discussed next.

2.3 Sequence form policy representation

The use of sequence form to represent policies was first suggested by Koller et. al [6] to represent games. It has since been used both in settings with self-interested agents [8] and cooperative agents [2]. In the context of game trees, the idea behind this representation is that a policy can be characterized by the probability distribution it induces over the leaves of the tree. If two policies induce the same distribution, then they result in the same reward.

For models with local observability, a *sequence* of agent k $\{s_1, a_1, s_2, a_2, .., s_m, a_m\}$ is an ordered set of k's actions and local states. Because we will adapt the formulation developed by Aras and Dutech [2], we will follow their nomenclature and use the term *history* instead of sequence. A history containing T (the problem's time horizon) actions is called a *terminal* history. For agent k, the set of all histories is denoted by $\mathcal{H}_k$, the set of terminal histories by $\mathcal{Z}_k$, and the set of non-terminal histories by $\mathcal{N}_k$. A *joint history* $h \in \mathcal{H}$ is a tuple $\{h_1, h_2, ..h_n\}$ containing one history per agent.

An agent's policy induces a probability distribution over its histories. The *realization weight* of a history $\{s_1, a_1, .., s_m, a_m\}$ under a policy is the probability that the policy assigns to taking actions $a_1, .., a_m$ *given* that the states $s_1, .., s_m$ are encountered. In other words, a history's realization weight does not include chance outcome probabilities. Instead, we will have separate terms that reflect these probabilities. The

Table 1: DEC-MDP formulation as an NLP

$$\begin{aligned}
\max \quad & \sum_{h \in \mathcal{Z}} \mathcal{R}(h) \prod_{k \in \mathcal{A}} x_k(h_k) && && (1)\\
\text{s.t.} \quad & \sum_{a \in A_k} x_k(a) = 1 && \forall k \in \mathcal{A} && (2)\\
& \sum_{a \in A_k} x_k(h.s.a) = x_k(h) && \forall k \in \mathcal{A}, s \in S_k, h \in \mathcal{N}_k && (3)\\
& x_k(h) \geq 0 && \forall k \in \mathcal{A}, h \in \mathcal{H}_k && (4)
\end{aligned}$$

vector of realization weights of all sequences of agent k will be denoted as x_k. The realization weight of history $h \in \mathcal{H}_k$ is given by $x_k(h)$.

A *pure* policy (also known as deterministic policy) is one that chooses a single action with probability 1 at each decision making point. Because in cooperative settings there is at least one optimal pure joint policy, we restrict our attention to pure policies. But even a pure policy will have multiple *terminal* histories with non-zero weights, because it must specify an action to take under each possible chance outcome of each action in a history with non-zero weight. And because a history's weight does not include chance outcome probabilities, the realization weight can only be 0 or 1. The set of terminal histories with non-zero weights under a policy is called its *support set*, denoted by σ_i, and its size is called *support size*.

3. EXISTING MATHEMATICAL FORMULATIONS

In this section, we review existing mathematical formulations of a DEC-MDP with local observability as a Non-Linear Program and as a Mixed Integer Linear Program. We will adapt some of the ideas behind these formulations in the next section.

3.1 DEC-MDP as NLP

The formulation of DEC-MDP with local observability as a Non-Linear Program (NLP) is given in Table 1.

In the objective function, $\mathcal{R}(h)$ is the expected reward of terminal joint history h. For terminal joint history $h = \{s_1, a_1, .., s_T, a_T\}$ consisting of joint states and actions

$$\mathcal{R}(h) = \beta(h) r(h) + \sum_{s_l} P(s_l | s_T, a_T) R(s_T, a_T, s_l) \qquad (5)$$

where s_l iterates over all the terminal states reachable from the probabilistic branching after the last joint action a_T. $\beta(h)$ is the probability of encountering the joint states in h given the actions in h.

$$\beta(h) = \prod_{t=1}^{T-1} P(s_{t+1} | s_t, a_t)$$

In equation (5), $r(h)$ is the sum of rewards of states and actions along the history, up to but not including the last action.

$$r(h) = \sum_{t=1}^{T-1} R(s_t, a_t, s_{t+1})$$

The group of constraints in the NLP are called *policy constraints*. They guarantee that a solution to the NLP represents a legal policy where the sum of an agent's action probabilities in any state is 1.

The problem with the NLP formulation is that it results in a non-concave objective function. There are no methods that guarantee finding a globally optimal solution for this kind of NLPs.

3.2 DEC-MDP as MILP

Aras and Dutech [2] developed a formulation for DEC-POMDPs as a Mixed Integer Linear Program (MILP). The advantage of this formulation is that for MILPs, it is possible to find a global maximum, so a solver like Cplex [1], for example, can find the realization weights representing an optimal joint policy.

We slightly modify the original formulation of Aras and Dutech for the case of DEC-MDP with local observability in order to build on it and adapt it for EDI-CR. For ease of explanation, we give here the formulation of the case with 2 agents i and j. The formulation is given in Table 2.

As in the NLP formulation, $\mathcal{R}(h, h')$ in the objective function (7) already accounts for the transition probabilities of both agents, so realization weights are either 0 or 1.

To change the NLP to a MILP, the objective function must be linearized. To do this, Aras introduces *compound* variables z. A variable $z_{h,h'}$ is created for each pair of terminal histories $h \in \mathcal{Z}_i$ and $h' \in \mathcal{Z}_j$. The variable is related to the existing x variables by the identity

$$z_{h,h'} = x_i(h) x_j(h') \qquad (6)$$

Because a realization weight can only be 0 or 1, the x and z variables are in fact all binary. However, the difficulty of solving a MILP increases as the number of integer variables increases. Aras observed that if we only restrict weights of terminal sequences to be integer, other variables will be guaranteed to take on integer values as well, so integrality constraints are only imposed on weights of terminal sequences.

The question now is how to enforce the identity using a set of linear constraints. To do this, Aras uses combinatorics (knowing the support size of each agent's policy) and treats the z variables as counters. For example, constraint (10) guarantees that if history h is part of agent i's support set, there should be enough compound variables set to 1 to cover j's support set, i.e. enough compound variables involving h should be 1. And if h is not part of the support, then all of the compound variables involving h should be 0. Constraint (12) limits the number of compound variables that can be simultaneously set to 1.

4. PROPOSED MILP FORMULATION

In this section, we present our MILP formulation for EDI-CR instances with 2 agents, with a note on extensions to more than 2 agents at the end. Before we do so, we explain the insight that we used to develop a compact formulation.

4.1 Useful insight

For the 2-agent case, the NLP in Table 1 reduces to a Quadratic Program (QP). Looking at the objective function of this QP, we see that it has the form

$$\mathbf{x}^{\mathsf{T}} Q \mathbf{x}$$

Table 2: DEC-MDP formulation as a MILP

$$
\begin{array}{rll}
 & \text{maximize} & \\
 & \displaystyle\sum_{h\in\mathcal{Z}_i,h'\in\mathcal{Z}_j} \mathcal{R}(h,h')z_{h,h'} & & (7)\\
 & \text{subject to} & \\
 & \displaystyle\sum_{a\in A_k} x_k(a) = 1 & k\in\{i,j\} & (8)\\
 & \displaystyle\sum_{a\in A_k} x_k(h.s.a) = x_k(h) & k\in\{i,j\}, s\in S_k, h\in\mathcal{N}_k & (9)\\
 & \displaystyle\sum_{h'\in\mathcal{Z}_j} z_{h,h'} = x_i(h)\|\sigma_j\| & \forall h\in\mathcal{Z}_i & (10)\\
 & \displaystyle\sum_{h\in\mathcal{Z}_i} z_{h,h'} = x_j(h')\|\sigma_i\| & \forall h'\in\mathcal{Z}_j & (11)\\
 & \displaystyle\sum_{h\in\mathcal{Z}_i,h'\in\mathcal{Z}_j} z_{h,h'} = \|\sigma_i\|\|\sigma_j\| & & (12)\\
 & x_k(h)\geq 0 & k\in\{i,j\}, h\in\mathcal{H}_k & \\
 & x_k(h)\in\{0,1\} & k\in\{i,j\}, h\in\mathcal{Z}_k & \\
 & z\in[0,1] & &
\end{array}
$$

Table 3: EDI-CR formulation as a MILP

$$
\begin{array}{rll}
 & \text{maximize} & \\
 & \displaystyle\sum_{b_h\in B_h} \mathcal{R}_k(h,b_h)z_{h,b_h} & k\in\{i,j\}, h\in\mathcal{Z}_k, & (13)\\
 & \text{subject to} & \\
 & \displaystyle\sum_{a\in A_k} x_k(a) = 1 & k\in\{i,j\} & (14)\\
 & \displaystyle\sum_{a\in A_k} x_k(h.s.a) = x_k(h) & k\in\{i,j\}, s\in S_k, h\in\mathcal{N}_k & (15)\\
 & \displaystyle\sum_{b_h\in B_h} z_{h,b_h} = x_k(h) & k\in\{i,j\}, h\in\mathcal{Z}_k & (16)\\
 & z_{h,b_h} \leq & \displaystyle\sum_{h'\in b_h} \beta(h'|h)x_{-k}(h') & \\
 & & k\in\{i,j\}, h\in\mathcal{Z}_k, b_h\in B_h & (17)\\
 & x_k(h)\geq 0 & k\in\{i,j\}, h\in\mathcal{H}_k & \\
 & x_k(h)\in\{0,1\} & k\in\{i,j\}, h\in\mathcal{Z}_k & \\
 & z\in[0,1] & &
\end{array}
$$

where $\mathbf{x} = [x_i, x_j]$ and Q is the reward matrix. $Q(h,h') = \mathcal{R}(h,h')$ if both h and h' are terminal sequences, otherwise the entry in the matrix is 0. So what the MILP in Table 2 does is that it "flattens" this matrix and multiplies each matrix entry by the corresponding compound variable. We can see that the introduction of compound variables essentially created a variable for each entry in Q associated with terminal histories.

For DEC-MDPs, because agents' decision processes are tightly coupled, the rewards and transitions of one agent strongly depend on the actions taken by another. So for a given history h, $\mathcal{R}(h,h')$ can vary widely, depending on h'. The variation comes from the dependency of encountered states and rewards on the particular actions taken by the other agent in h'. As a result, a given row or column in Q contains many distinct values.

The situation can be very different in the presence of structured interactions. An agent is only affected by the *critical* actions of another agent; those actions involved in reward and transition interactions as given by ρ and τ in the definition of EDI-CR. As a result, for a given history h of one agent, the rewards and transition probabilities of states and actions along h do not depend on the exact actions in the history of another agent. For example, suppose ρ says that agents i and j get a bonus of 10 if they do actions a_1 and a_5, respectively, at any point in time, and τ says that action a_3 of agent j affects the transition probability of a_7 of i. Now suppose history $h\in\mathcal{Z}_i$ involves doing action a_1 at time 4 and a_7 at time 6. In this case, all histories $h'\in\mathcal{Z}_j$ that involve doing a_3 before time 6 and a_5 any time will have the same effect on the transitions and rewards of h.

Let us go back to the matrix view of the objective function. Because in EDI-CR agents have their local reward functions, we can express Q as the sum of reward matrices of the 2 agents $Q_i + Q_j$. Note that this does not assume that rewards are simply additive; each entry in each of these matrices can depend on the histories of both agents, thus representing the complex reward structure ρ. The rows in Q_i will contain many duplicate entries, and the columns in Q_j will also contain many duplicate entries, reflecting the fact that an agent is oblivious to many of the details of the other agent's history.

The above observations clearly indicate that simply using the DEC-MDP MILP formulation for EDI-CR ignores important structure that can be used to avoid duplication and reduce the number of the introduced compound variables.

4.2 2-agent EDI-CR as MILP

In our formulation of EDI-CR with 2 agents as a MILP, we introduce compound variables, but use the insight discussed above to obtain the compact formulation given in Table 3.

4.2.1 Binning histories

The main idea in our formulation is that for a given history h, we group all histories of the other agent that have the same effect on the transitions and rewards in h into a single *bin*. For each history $h\in\mathcal{Z}_k$ of some agent k, the set of bins it induces, B_h, is a partition over the set of terminal histories of the other agent, i.e. $\bigcup_{b_h\in B_h} b_h = \mathcal{Z}_{-k}$ and $b_h^m \cap b_h^n = \varnothing\ \forall b_h^m, b_h^n \in B_h$. Consequently, a history that is not affected at all by the other agent has a single bin containing all histories of the other agent.

Instead of creating a compound variable for every pair of terminal histories, our formulation introduces a compound variable for every history and every one of the bins associated with this history. So for every history h and associated bin b_h, we introduce z_{h,b_h}. In the matrix view of the objective function, instead of creating a variable for each entry in Q, we create a variable for each *distinct* entry in each of Q_i and Q_j. Because these matrices will typically contain many duplicate entries as previously discussed, this idea can significantly reduce the number of compound variables we introduce.

We now turn to the formulation of the objective function (13). We would like to fold into $\mathcal{R}_k(h,b_h)$ all the factors that are common to all histories in b_h. These are quantities of h that are oblivious to which history $h'\in b_h$ is played, namely h's transition probabilities and rewards[1]. We therefore have

$$\mathcal{R}_k(h, b_h) = r_k(h|b_h)\beta(h|b_h)$$

The factors on the right can be calculated using *any* history in the bin. Suppose h' is one such history, in which case

$$r_k(h|b_h) = \sum_{t=1}^{T-1} R_k(s_t, a_t, s_{t+1}) + r_\rho(h, h')/2$$

where R_k is the individual reward function and $r_\rho(h, h')$ represents rewards that depend on actions of both agents, as specified in ρ, and will be zero if the actions in h are not involved in any reward interactions. Dividing by 2 avoids double counting reward interactions.

As for the transition probabilities, they are given by

$$\beta(h|h') = \prod_{t=1}^{T-1} P_\tau(s_{t+1}|s_t, a_t, \{a'_1..a'_t\})$$

where $\{a'_1..a'_t\}$ are the actions in h' done up to time t. P_τ depends on the local transition function P_k and, for transitions involved in τ, previous actions in the other agent's history.

Having folded the factors common to all histories in the bin into $\mathcal{R}_k(h, b_h)$, we fold into z_{h,b_h} quantities that depend on the particular h' in the bin, namely the transition probabilities along h', given history h. The meaning of a compound variable is therefore given by the following identity

$$z_{h,b_h} = x_k(h) \sum_{h' \in b_h} \beta(h'|h) x_{-k}(h') \quad (18)$$

where x_{-k} is the realization weight vector of the other agent. z_{h,b_h} is therefore the probability that h is part of an agent's policy, multiplied by the probability that the other agent plays a history in the bin b_h.

The effect of the number of interactions on the size of the formulation is clear from the meaning of the introduced compound variables. The more interactions, the more details one agent will care about in another agent's history. As we increase the number of ways in which an agent can be affected by another, we need more bins to group histories into, since each bin represents a unique way in which one agent can affect another's rewards and transitions. So the more interactions, the larger the number of bins, and the more compound variables we need. In the extreme case where each history of agent i affects agent j's history in a different way, we need a separate bin for each of i's histories, essentially creating a compound variable for every pair of histories, resulting in a formulation with the same number of variables as in the DEC-MDP MILP.

4.2.2 Enforcing the identity

As Aras did in the DEC-MDP formulation, we need to enforce the identity defining compound variables through a set of linear constraints. However, doing so is more challenging than in the DEC-MDP case. To see why, note that the identity (6) defining the DEC-MDP's compound variables results in variables that are binary, since they are products of binary variables. Therefore the DEC-MDP MILP formulation can use compound variables as counters (in constraints (10), (11),(12)), allowing the use of combinatorics and support set sizes to enforce the identity by linear constraints.

In our formulation, the definition of compound variables involves transition probabilities, so these variables are no longer binary, thereby preventing the use of combinatorics. We must resort to other properties of, and relations among, the variables to derive constraints equivalent to the identity.

Starting with the identity in (18), summing over all bins on both sides gives

$$\begin{aligned}\sum_{b_h \in B_h} z_{h,b_h} &= \sum_{b_h \in B_h} x_k(h) \sum_{h' \in b_h} \beta(h'|h) x_{-k}(h') \\ &= x_k(h) \sum_{b_h \in B_h} \sum_{h' \in b_h} \beta(h'|h) x_{-k}(h')\end{aligned} \quad (19)$$

Since B_h is a partition over $\mathcal{Z}_{-k}$, the double sum on the second line reduces to a sum over all histories of the other agent, giving

$$\sum_{b_h \in B_h} z_{h,b_h} = x_k(h) \sum_{h' \in \mathcal{Z}_{-k}} \beta(h'|h) x_{-k}(h') \quad (20)$$

A property of a legal policy is that it prescribes an action at each state reachable by a non-terminal history with non-zero realization weight. As a result, histories in a policy's support set cover all possible transitions of actions along parents of these histories. This means that the sum of probabilities of transitions along histories in the support set must be 1, i.e. for agent k and any history h of the other agent,

$$\sum_{h' \in \sigma_k} \beta(h'|h) = 1 \quad (21)$$

It follows that

$$\sum_{h' \in \mathcal{Z}_{-k}} \beta(h'|h) x_{-k}(h') = 1 \quad (22)$$

because the x variables act as switches for including the various βs, and only the switches of histories in the support will be 'on', so the left side only contains their corresponding βs and, by (21), is 1.

From (20) and (22), we have the following set of constraints, one per terminal history of each agent

$$\sum_{b_h \in B_h} z_{h,b_h} = x_k(h) \quad (23)$$

This constraint makes intuitive sense because it guarantees that if h is not part of an agent's support set, then all of the compound variables involving h and all of its bins should be 0. If h is part of the support, it guarantees there is enough contribution from the compound variables associated with all bins of h.

The problem with the above constraint is that on the left hand side, it does not prevent one compound variable from taking too high a value at the expense of another compound variable. So having this kind of constraint alone is not enough to enforce the identity in (18). To limit the contributions of individual compound variables on the left hand side, we need to impose upper bounds on them. A simple source of such bounds is the identity itself. Because in (18)

[1]For ease of exposition, we do not include here the reward and transition of the last action in h. These can be included using terms that refer to all terminal states reachable by the last action, as done in Equation (5).

$x_k(h)$ is either 0 or 1, we can form the following constraints

$$z_{h,b_h} \leq \sum_{h' \in b_h} \beta(h'|h) x_{-k}(h') \qquad (24)$$

Together, the sets of constraints in (23) and (24) enforce the identity. One advantage of our constraints over the combinatorics-based constraints in the DEC-MDP formulation is that ours do not involve the size of the support set. Aras assumes that this support size can be calculated using a simple formula involving the time horizon and the size of an agent's observation set (for DEC-POMDP). So the implicit assumption is that an agent's branching factor after each action at each time step is the same. But this is not necessarily true in settings where a state-action transition can end up in any number of next states, depending on the particular action taken. In such settings, determining the support size requires carefully looking at an agent's decision tree and the paths that can be taken in it, which is non-trivial for large problems.

A final remark on our formulation concerns the number of constraints. The set of constraints in (23) has the same size as the set of constraints in the DEC-MDP MILP. The set in (24), however, is larger, because it has a constraint for each bin of each history of each agent. But as will be seen in the next section, this does not prevent us from obtaining computational advantage over the DEC-MDP formulation.

A note on 3 or more agents.

For problems with $n > 2$ agents, we can still use the idea of grouping histories that have the same influence on a given agent's history. But with n agents, a bin does not contain histories of just 1 agent. Instead, it contains history tuples, where each tuple consists of histories of the $n-1$ agents. We denote a tuple of histories of everyone except agent k as h_{-k}, and denote an individual history within this tuple by a superscript indicating the agent it belongs to. For a history $h \in \mathcal{Z}_k$ of agent k and one of its bins b_h, the associated compound variable is given by the identity

$$z_{h,b_h} = x_k(h) \sum_{h_{-k} \in b_h} \prod_{h^q \in h_{-k}} \beta(h^q|h, h_{-k}) x_q(h^q)$$

Similar to the 2-agent case, the set of bins associated with history h, B_h, is a partition over $\bigcup_{q \neq k} \mathcal{Z}_q$, so we can use constraint (23). The problem is that if we use the idea behind constraint (24) to bound the values of the compound variables, the result will not be a linear constraints, but a constraint of order $n-1$. We are still trying to find a set of linear constraints that imposes the necessary upper bounds.

The existing general DEC-MDP formulations extend to n agents. The NLP formulation in Table 1 naturally extends to any number of agents because it simply multiplies realization weights of history tuples in the objective function. The DEC-MDP MILP formulation given by Aras also extends beyond 2 agents, but the number of compound variables grows exponentially with the number of agents, since a variable is created for each history tuple.

5. EXPERIMENTAL RESULTS

In this section, we present experimental results from comparing 3 formulations of EDI-CR instances; 1) QP formulation given in Table 1, but restricted to 2 agents, 2) MILP formulation developed for general DEC-MDPs ignoring structure given in Table 2 and 3) MILP formulation for EDI-CR given in Table 3. All 3 kinds of mathematical programs were solved using IBM ILOG Cplex [1]; the first using Cplex Mixed Integer QP solver, and the other two using Cplex MILP solver. In all 3 cases, we used the default Cplex parameters.

We experimented with 22 instances of the modified cooperative Mars rovers problem used in [7]. The number of interactions ranges from 4 to 7.

Before discussing the performance of the solver in solving each of the 3 formulations, we note that the time to *generate* these formulations is almost the same. Specifically, constructing the bins and objective function in the EDI-CR MILP is not more expensive than constructing the reward matrix in the QP or the objective function in the DEC-MDP MILP. In all 3 cases, we need to iterate over every joint history, calculate its expected reward and calculate the probability of the chance outcomes along the history.

First, we look at the behavior of the 3 formulations with respect to optimality. As mentioned before, the QP formulation is not guaranteed to result in an optimal solution because it is non-concave, so Cplex may report a locally optimal solution and quit. But even after obtaining a solution that we know is optimal (by comparing to another solution known to be optimal), Cplex may spend a very long time until it verifies optimality. We therefore have 5 possible outcomes of a run: 1) optimal solution found and verified, 2) optimal solution found but not verified within reasonable time[2] , 3) Locally optimal solution found, 4) Optimal solution could not be found within reasonable time, but a suboptimal solution was found, 5) No solution at all found within reasonable time. Of our 22 instances, Table 4 compares how many fall in the different categories for each formulation. Because a solver would never report a locally optimal solution for a MILP, the corresponding entries are marked by '-'. As can be seen, our formulation resulted in a provably optimal solution in 17/22 instances. In the remaining instances, we obtained higher rewards than the other formulations, but cannot say with certainty that our solution was optimal because there is nothing to compare it to, so each of the remaining 5 instances falls into into category 2 or 4. QP and DEC-MDP MILP were equally good at finding optimal solutions, although DEC-MDP MILP was better at verifying optimality. The table shows that the non-concavity of the QP can often lead the solver to just report a locally optimal solution. It also shows that in some cases, the number of compound variables introduced in the DEC-MILP is too large to allow the solver to find *any* solution within reasonable time.

Next, we look at the compactness of the MILP formulation with and without exploiting structured interactions. We group our 22 instances into 3 groups of small, medium and large instances containing 5, 9 and 8 instances, respectively. Table 5 shows the number of terminal sequences for each agent, and the number of compound variables, z, introduced in the DEC-MDP formulation and our EDI-CR formulation. Results were averaged over instances in each group. Clearly, the DEC-MDP formulation introduces many more compound variables than our formulation. The difference in the number of variables becomes more pronounced

[2]Reasonable time for most scenarios is taken to be 60 seconds. For larger scenarios, we time out a run after 600 seconds.

Table 4: Comparison of formulations with respect to optimality

	QP	DEC-MDP MILP	EDI-CR MILP
1) Optimal, Verified	5	9	17
2) Optimal, Not verified	9	5	x
3) Local optimal	5	-	-
4) Suboptimal	3	6	5-x
5) No solution	0	2	0

Table 5: Comparison of formulations with respect to compactness

	$\|\mathcal{Z}_i\|$	$\|\mathcal{Z}_j\|$	EDI-CR MILP $\|z\|$	DEC-MDP MILP $\|z\|$
Small	81	46	254	3,762
Medium	162	112	608	18,062
Large	941	781	3,793	596,950

as the problem size increases. Whereas the DEC-MDP formulation creates a compound variable for each pair of terminal histories, our formulation will only create as many variables as needed to distinguish between bins induced by a given history. Although our formulation has more constraints than the DEC-MDP MILP, we next show that the increased number of constraints is offset by the large reduction in the number of variables, resulting in MILPs that are overall easier to solve.

We now look at the solution time for each formulation. Table 6 shows the results of comparing both the time needed to find the optimal solution (reported as 'Find'), and the time needed to verify that the solution is indeed optimal (reported as 'Verify'). The times are in seconds, averaged over instances in each group. For groups where some solutions were not found/verified within reasonable time, the number of instances over which the averaging was done is indicated in brackets. In general, solving the EDI-CR MILP formulation is significantly faster than solving the other 2 formulations. There is also a large difference in the time needed to verify optimality. In the Small group, only 3 instances could be solved provably optimally within 60 seconds using the DEC-MDP MILP and QP formulations. In the Medium group, the differences in time to verify optimality among the different formulations is even more pronounced. In the Large group, Cplex found solutions for all our problems, but could not verify optimality. A solution with the same quality could not be found with any of the other formulations.

6. RELATED WORK

Formulating decision problems as mathematical programs has been done by other researchers, with the aim of making use of available industrial-grade solvers like Cplex. Aras and Dutech proposed two MILP formulations for general DEC-POMDPs [2]. One of these formulations was given in Table 2. The other uses game-theoretic concepts to linearize the objective function. This latter formulation is outperformed by the one we reviewed and built upon in this paper.

Petrik and Zilberstein [10] developed formulations of decision problems of cooperative and self-interested agents as separable bilinear programs and presented an algorithm for solving this class of programs. The QP discussed in this paper is itself a bilinear program, because realization weights of one agent are only multiplied by weights of the other agent, so the objective function is linear if the weights of one agent are fixed. Previous work on the EDI-CR model [7] used the bilinear formulation and solution algorithm.

Aras et. al [3] give a mathematical formulation for a special case of DEC-POMDP called Network Distributed POMDP [9] (ND-POMDP) where agents have independent transition and observation functions, but have reward interactions. In ND-POMDP, we can decompose the set of agents into subsets, where an agent's reward only depends on agents belonging to its subset(s). Because they only consider problems where each subset contains 2 agents (i.e. binary interactions), Aras et. al were able to formulate this problem as a QP. They present a linearization of the QP to a compact MILP that avoids having a compound variable for each joint terminal history. However, they report that the compactness of their formulation does not translate to savings in the time needed to solve the resulting MILP, compared to a simple formulation with one variable per joint history. One explanation they provide is that the compact MILP has a constraint matrix that is not sparse, making it hard for Cplex to deal with it efficiently.

Besides the fact that ND-POMDP assumes transition independence and EDI-CR does not, the difference between these two models is that in the former, agents belonging to the same subset are assumed to have very tight reward interaction; there is a single reward function per subset, and it is defined over joint actions and states of agents in the subset. We can see this as a coarse-grained kind of independence where agents either have reward interactions involving all their actions or none at all. EDI-CR captures a more fine-grained kind of interaction where *specific* actions affect, or are affected by, what another agent does. As a result of this difference, formulations of ND-POMDP would not be very useful, if at all, when directly applied to EDI-CR, since they cannot capture and exploit fine-grained interactions. The general ideas and techniques for linearizing a high order expression can, however, be useful across models and formulations.

Another MILP formulation of Transition-Independent DEC-MDP [4] is given by Wu and Durfee [11]. Their formulation is approximate and is the result of discretization and piecewise linear approximation of non-linear constraints. This work therefore finds exact, optimal solutions to an inexact model. The errors introduced by the discretization and linearization can be controlled, at the expense of introducing more variables into the formulation.

7. CONCLUSION

This paper presents a compact mathematical formulation of a class of DEC-MDPs where there are structured transition and reward interactions among agents. This class of problems is modeled using Event-Driven Interactions with Complex Rewards. Previously, instances of EDI-CR would either be formulated as Quadratic Programs, in which case a globally optimal policy is not guaranteed because of the non-concavity of the QP, or as a MILP through a formulation that was conceived for general DEC-POMDPs and adapted to DEC-MDPs. The formulation we developed successfully exploits structured interactions, and rests on the observation that when considering a given history of agent

Table 6: Comparison of formulations with respect to solution time

	EDI-CR MILP Find	DEC-MDP MILP Find	QP Find	EDI-CR MILP Verify	DEC-MDP MILP Verify	QP Verify
Small	0.29	8.68	0.57	0.12(3)	3.5(3)	0.58(3)
Medium	0.59	10.72	6.4	0.35(6)	21.6(6)	> 60
Large	83	N/A	N/A	N/A	N/A	N/A

i, many histories of the other agent have the same effect on the rewards and transitions of i. Therefore, in doing the linearization necessary to obtain a MILP, we can avoid creating a compound variable for every pair of joint terminal histories. Instead, we create one variable for every group of histories that have the same effect on a given history. Our experimental results show the compactness of our formulation compared to a MILP that does not exploit structure. We also show that our formulation allows a solver to find an optimal solution and verify its optimality faster than the other formulations discussed in this paper. This allows us to solve larger problems which would otherwise be intractable.

We are currently working on extending our formulation beyond the 2-agent case. The main issue is to find a set of linear constraints that enforce the non-linear identity defining compound variables. One possible starting point is to look at settings with more than two agents, but where any interaction only involves two agents. This is still different from the binary ND-POMDPs discussed in Section 6 because we allow for transition interactions.

Another future direction we are interested in involves approximation. Currently, we group histories into the same bin only if they have exactly the same effect on a given history. One possible approximation is grouping histories whose effects are similar enough. This is equivalent to making an agent indifferent among a larger set of the other's histories, and would further reduce the number of compound variables. We need to study how this can be done in a principled manner.

8. REFERENCES

[1] IBM ILOG Cplex available under the Academic Initiative.

[2] R. Aras and A. Dutech. An investigation into mathematical programming for finite horizon decentralized POMDPs. *Journal of Artificial Intelligence Research*, 37:329–396, 2010.

[3] R. Aras, A. Dutech, and F. Charpillet. Quadratic Programming for Multi-Target Tracking. In *AAMAS 2009 Workshop on Multi-agent Sequential Decision-Making in Uncertain Domains*, pages 4–10, Budapest, Hungary.

[4] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research*, 22:423–455, 2004.

[5] D. Bernstein, R. R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[6] D. Koller, N. Megiddo, and B. von Stengel. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior*, 14(2):247 – 259, 1996.

[7] H. Mostafa and V. Lesser. Offline planning for communication by exploiting structured interactions in decentralized MDPs. In *2009 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 193–200, Italy, 2009.

[8] H. Mostafa and V. Lesser. Exploiting Structure To Efficiently Solve Loosely Coupled Stochastic Games. In *AAMAS 2010 Workshop on Multi-agent Sequential Decision-Making in Uncertain Domains*, pages 46–53, Toronto, Canada, 2010.

[9] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed pomdps: A synthesis of distributed constraint optimization and POMDPs. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, 2005.

[10] M. Petrik and S. Zilberstein. A bilinear programming approach for multiagent planning. *Journal of Artificial Intelligence Research*, 35:235–274, 2009.

[11] J. Wu and E. H. Durfee. Mixed-integer linear programming for transition-independent decentralized MDPs. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006)*, 2006.