# Exploiting Structure To Efficiently Solve Loosely Coupled Stochastic Games

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

This paper is concerned with sequential decision making by self-interested agents when their decision processes are largely independent. This situation can be formulated as a stochastic game which would traditionally be represented in extensive form (a single game tree), a representation that fails to exploit the loose coupling in the game. We propose a new representation for 2-agent loosely coupled stochastic games that allows exploiting the sparsity and structure of agent interactions while still being able to capture a general stochastic game. We provide analytical and experimental results to show the representational and computational savings we obtain compared to extensive form in settings with different degrees of coupling. Our second contribution is a compact formulation of our problem as a Multi-Agent Influence Diagram, a first step towards the goal of solving problems with more than two agents. Finally, we investigate the challenges that need to be resolved to meet this goal.

## Categories and Subject Descriptors

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

## General Terms

Performance, Economics

## Keywords

Game theory, Structured game representations

## 1. INTRODUCTION

In this section, we introduce the notion of loosely coupled stochastic games, provide an example to motivate it and provide a general characterization of this class of games. We also introduce the running example that will be used throughout this chapter.

### 1.1 Loosely coupled stochastic games

A *stochastic game* describes a situation where agents are self-interested and interact over a number of stages. Each stage begins with the game at some state. Agents take actions simultaneously and, in general, receive rewards based on the actions taken by all agents and the particular state the game was in. The game then probabilistically transitions to a new state based on the previous state and joint actions. The agents are therefore very tightly coupled; each action of each agent affects the rewards and next games of all others.

In contrast to this tight coupling, we introduce *loosely coupled stochastic games* where the agents are largely independent. Each agent has its local state and its decisions only affect its own reward and next state. In our games, an agent generally does not know about the other agent's states and actions and, typically, does not care. What ties the decision processes of the two agents, however, is that there is some interaction between the agents; some actions of an agent affect the others. As the number of interactions increases, we move from completely independent decision processes to the tightly coupled stochastic games discussed in the literature.

Loose interactions arise in many situations. Consider a set of cleaning robots owned by different companies which, between them, manage the cleanliness of a building. Each robot is responsible for a set of halls, but corridors are joint responsibility and the robots can get extra reward if they correctly coordinate their cleaning of this shared space. Other interactions stem from sharing the waste bins and potentially getting into each other's way in shared areas like corridors and elevators. Non-situated agents can also have loose interactions. Consider a set of self-intersted computational servers that offer their computational resources for a fee. Each server has its set of incoming computational tasks, resources, and fee policy. However, some tasks require more resources than is available to any single server, in which case the server receiving the task can propose to some of the others to complete the task for a fee. In addition to affecting each other's rewards, the servers can affect each other's state. Collaborating on certain tasks can result in a server gaining more experience on these tasks, causing it to transition to states where future tasks of this kind are executed more quickly.

Traditionally, the above situations would be represented in extensive form, which requires specifying an agent's rewards and next state for each of its actions *and* the other agents' actions. Clearly, this representations is overly verbose, since in most cases, the rewards and new states are independent of the other agents. Ignoring this fact results in game trees that are much larger than they need to be. Besides being representationally inefficient, a single game tree obscures the structured interaction among agents, making it hard to exploit to efficiently find a Nash equilibrium.

### 1.2 Characteristics of loosely coupled games

AAMAS 2010 Workshop on Multi-agent Sequential Decision-Making in Uncertain Domains, May 11, 2010, Toronto, Canada.

The above examples share these general characteristics:

- Each agent has its local states and actions (e.g. robots have different locations and grabbing actions). Most of an agent's action outcomes and rewards are independent of the other agent.
- Agents are generally unaware of each other's states and actions, unless some form of communication is specified.
- Few interactions among agents, compared to the number of actions they can take. They are also structured, meaning that an action can affect the other agent in a specific way. In a *reward* (resp. *transition*) *interaction*, an action of one agent affects the reward (resp. outcome probabilities) of certain actions of the others.
- Interactions are not only between actions taken at the same time. An action can be affected by something that happened in the past (e.g. a robot's dumping a large object in a waste bin will affect the outcome of using the bin at any later point). The fact that the affecting action happened is not necessarily encoded in an agent's state.

### Self-interested Mars rovers

To ground future discussions, we use the following simple example that captures the above characteristics.

We use a self-interested variant of the Mars Rovers scenario [1]. Consider a situation where two countries send autonomous rovers to collect data from the surface of Mars. Each rover has a list of samples that it wants to gather. The rovers need to accomplish their missions within a limited time, so they cannot gather all the samples on their list. A rover's set of actions is the set of sites it has not visited yet, and each action has two outcomes, fast and slow, probabilistically chosen by nature. An agent's decision problem is therefore which sites to gather data from and in what order so as to maximize its expected reward. Even though the rovers are generally unaware of each other's actions and whereabouts, their decision processes are not totally independent. We define a *shared site* to be a site where one agent's visit can affect the reward or the outcome probability of the other agent when/if it visits that same site. For example, if both rovers want to gather data from the same site, they may help each other, thus getting a higher reward for the site, or they may get in each other's way, thus reducing one or both rover's reward.

One solution concept for the above situation is the Nash equilibrium; a pair of policies such that no agent is motivated to deviate from its policy. At an equilibrium, i.e., each agent's policy is a best response to the other's.

## 2. REPRESENTING AND SOLVING 2-PLAYER LOOSELY COUPLED GAMES

### 2.1 Existing representation: Extensive form

A stochastic game can be represented as an extensive form game (EFG), which is a tree capturing the order in which agents take actions, what they know when they take each action, and the probabilistic nature of actions. An EFG is a tuple $< I, V, E, P, H, u, p >$ where:

- $I$ is the set of $n$ players
- $(V, E)$ is a finite directed tree with nodes $V$ and edges $E$ and $Z$ is the set of terminal nodes
- $Player : V \setminus Z \rightarrow I$ determines which player moves at each decision node.
- $H = \{H_0, ..., H_n\}$ is the set of information sets, one for each player. Each $H_i$ is a partition of $Player_i$.
- $A_i(h)$: the set of actions available at information set $h$
- $u : Z \rightarrow \mathbb{R}$ is the utility function defined over the set of terminal nodes. For $x \in Z$, $u_i(x)$ is the payoff to player $i$ if the game ends at node $x$
- $p$ is the transition probability of chance moves

In a game with imperfect information, an agent does not know exactly the state of the other agent (and thus the game played by the agents at any particular stage), but does have a probability distribution over it. In such games, an information set can contain more than one node, which the agent cannot tell apart. A policy should therefore make the same decision across all nodes belonging to the same information set. It is a mapping from information sets to probability distributions over actions.

### 2.2 Proposed representation: Self-Interested EDI-CR

In loosely coupled games, the rule is that agents' actions are independent and the exceptions are the reward and transition interactions among them. We would therefore like to represent the decision processes of the two agents separately and enumerate the (relatively few) interactions. This is exactly what our model Event-Driven Interactions with Complex Rewards (EDI-CR) does. In previous work [14], we used EDI-CR to capture structured interactions among cooperative agents. For self-interested agents, we slightly modify the original definition of EDI-CR to allow agents to have different reward functions. A 2-agent EDI-CR instance is therefore a tuple $< S_{1,2}, A_{1,2}, P_{1,2}, R_{1,2}, \rho, \tau, T >$ where:

- $S_i$, $A_i$, $R_i : S_i \times A_i \rightarrow \mathbb{R}$, $P_i : S_i \times A_i \times S_i \rightarrow [0, 1]$ are the local state space, action space, reward function and transition function of agent $i$. These elements specify the separate decision processes of the agents.
- $\rho = \{< (s^k_{k_1}, a^k_{k_1}), ..., (s^k_{k_2}, a^k_{k_2}), r^k >_{k=1..|\rho|}\}$ specifies reward interactions. The $k^{th}$ entry specifies $r^k$, the additional reward/penalty obtained when each agent does its specified action in its specified state at any point during its execution.
- $\tau = \{< (s^k_{k_1}, a^k_{k_1}), (s^k_{k_2}, a^k_{k_2}), p^k >_{k=1..|\tau|}\}$ specifies 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 for the problem.

The individual states, actions and rewards describe the dynamics of each agent's decision process, while $\rho$ and $\tau$ capture the interactions among them. We stress that the model can be used for general stochastic games; it would have entries in $\rho$ and $\tau$ for every combination of actions and states to mirror the fact that all actions affect all agents.

Representing a game as an EFG is more compact than EDI-CR when modeling tightly coupled games. With 2 agents, each having 2 states and 2 actions, the joint representation has 32 entries (16 in each of the reward and

transition functions). If all actions participate in determining the game's next state and individual rewards, EDI-CR has 4 entries per local transition and reward function, in addition to 16 reward and 16 transition interactions, clearly less compact than EFG. At the other extreme, if the agents do not interact, then instead of having a *single* tree in which rewards and transitions depend on the *joint* states and actions of agents, EDI-CR has *separate* trees with transition and reward functions defined over *local* states and actions. In this case, EDI-CR only has entries in the local functions, for a total of 8 entries compared to EFG's 32. Between these two extremes, the extent to which EDI-CR's representation is more intuitive depends on the number of interactions (e.g., for 3 interactions, EDI-CR would have 8+3=11 entries, compared to EFG's 32).

## 2.3 Solution Method

In this section, we review an existing formulation of a stochastic game as a bilinear program (BLP) and discuss how this formulation is affected by having nearly separable decision processes as opposed to a single game tree. The original formulation of extensive form games was derived by Petrik and Zilberstein [15] and is as follows:

$$\begin{array}{ll} \text{Maximize} & x^\mathsf{T} r_1 + x^\mathsf{T}(C_1 + C_2)y + y^\mathsf{T} r_2 - \lambda_1^\mathsf{T} b_1 - \lambda_2^\mathsf{T} b_2 \\ \text{subject to} & A_1 x = b_1 \quad A_2 y = b_2 \\ & r_1 + C_1 y - A_1^\mathsf{T}\lambda_1 \le 0 \\ & r_2 + x^\mathsf{T} C_2 - A_2^\mathsf{T}\lambda_2 \le 0 \quad x, y \ge 0 \end{array}$$

We use the sequence form [10] to represent the agents' policies. $x$ and $y$ are vectors of realization weights of agent $i$ and $j$'s action sequences, respectively. $r_1$ (resp. $r_2$) is a vector representing the individual rewards of agent $i$ (resp. $j$); those rewards that do not depend on what the other agent does. Each entry in $r_i$ contains the expected reward of a sequence of player $i$. $C_1$ and $C_2$ are matrices with a row for each sequence of $i$ and a column for each sequence of $j$. These matrices represent rewards of $i$ and $j$ whose attainment depends on what *both* agents did. $A_1$, $A_2$, $b_1$ and $b_2$ form constraints that guarantee that a solution indeed represents the realization weights of a legal policy; i.e. probabilities of actions at each state add up to 1 (for more details on the constraints over a policy in sequence form, see [10]). $\lambda_1$ and $\lambda_2$ are the variables of each agent's dual optimization problem. Their presence in the objective function reflects our interest not in a solution that maximizes the sum of rewards, but in one that is an equilibrium.

When dealing with loosely coupled games, there can potentially be many sequences of one agent's actions that neither affect, nor are affected by, the other agent. These sequence will have their own local rewards, but will have zero entries in the agent's $C$ matrix. Using this observation, we can have much fewer rows and columns in $C_1$ and $C_2$ if we exclude such sequences. Denoting by $\bar{x}$ and $\bar{y}$ those elements of $x$ and $y$ that affect, or are affected by, actions of the other agent, we get the following program

$$\begin{array}{ll} \text{Maximize} & x^\mathsf{T} r_1 + \bar{x}^\mathsf{T}(C_1 + C_2)\bar{y} + y^\mathsf{T} r_2 - \lambda_1^\mathsf{T} b_1 - \lambda_2^\mathsf{T} b_2 \\ \text{subject to} & A_1 x = b_1 \quad A_2 y = b_2 \\ & r_1 + [C_1\bar{y}; \bar{0}] - A_1^\mathsf{T}\lambda_1 \le 0 \\ & r_2 + [\bar{x}^\mathsf{T} C_2; \bar{0}] - A_2^\mathsf{T}\lambda_2 \le 0 \quad x, y \ge 0 \end{array}$$

where $[v; \bar{0}]$ is vector $v$ padded with enough zeros to make it of the desired length. The details of calculating $C_1$ and $C_2$ are similar to those of calculating the team's reward matrix $C$ in the formulation of the cooperative case [14]. The resulting bilinear program is solved using an existing algorithm [15] to find a Nash equilibrium.

# 3. COMMUNICATION SCHEMES IN EDI-CR AND EXTENSIVE FORM

In this section, we investigate how the degree of coupling affects the relative savings of using EFG and EDI-CR. One simple way to change the degree of coupling is by introducing different amounts of communication among agents[1]. The more the agents communicate, the more they affect each other; sending a message affects what the receiver observes.

## Analytical and experimental setup

We present three communication schemes; no, mandatory and optional communication. For each, we analyze the effect on the size of an instance when represented using EDI-CR, EFG and the MAID which will be discussed in the next section. For EFG and EDI-CR, we measure size as the number of states in the joint game tree and in each agent's decision process, respectively. For MAIDs, we look at the total size of the CPDs of all nodes. We express these quantities in terms of $A$ actions, $O$ outcomes per action, $T$ time steps, $k$ reward interactions and $m$ transition interactions, with $k + m \le A$. The variables $k$ and $m$ allow us to investigate how the number of interactions and their nature affect the size of a representation. We stress that our analysis is not specific to the Mars rover example. It applies to any loosely coupled game that fits the characterization we give in Section 1.2. To simplify the analysis, we assume that an action takes one time unit and that actions can repeat.

We also look at the effect of communication on the time to find the first Nash equilibrium[2]. EFGs are solved using the `logit` solver in the game theoretic package Gambit [13] and reported as "Gambit" (we report results of `logit` because it performed better than `lcp`). EDI-CR is solved as a bilinear program reported as "BLP". We time out a solver after 30 minutes and report "N/A".

We present experimental results from 8 instances of the Mars rovers domain with $T \in [6, 8]$ and number of actions is 5 or 6 (unlike the analytical analysis, an action here can take more than one time unit). To avoid generating very large games that would not fit in memory regardless of the representation, we can specify restrictions over actions by having earliest start times before which they cannot proceed. We obtained results from a larger set of data which showed the same trend as in the results we report, so we omit them.

## 3.1 No communication

We first look at the case where communication is not allowed. An agent makes decisions based only on its local state, which keeps track of what actions have been done so far and the outcomes obtained for them. With EFG, each stage consists of actions and outcomes for both agents. The number of nodes is therefore $\mathcal{O}(A^{2T}O^{2T})$.

[1]Communication is a special kind of transition interaction; sending a message makes the recipient transition to a state where the message is observed, thereby affecting its transition probability. As such, communication can be handled by our solution method.

[2]Because it is hard to compare solution qualities in selfish settings, we are concerned with finding *any* equilibrium

**Table 1: Size and performance comparison for the no-communication case (times in seconds)**

| EFG infosets | EFG size | EDI-CR size | % reduction | Gambit time | BLP time |
|---|---|---|---|---|---|
| 49 | 1301 | 132 | 89.85% | 9 | 2 |
| 68 | 1618 | 140 | 91.35% | 18 | 9 |
| 100 | 3195 | 216 | 93.24% | 63 | 2.4 |
| 151 | 7987 | 303 | 96.21% | 306 | 2.5 |
| 173 | 11.2k | 348 | 96.89% | 1080 | 3 |
| 296 | 25.1k | 610 | 97.57% | N/A | 2.5 |
| 333 | 44.4k | 695 | 98.43% | N/A | 2.6 |
| 841 | 473k | 2079 | 99.56% | N/A | 8 |

**Table 2: Size and performance comparison for the mandatory communication case (times in seconds)**

| EFG infosets | EFG size | EDI-CR size | % reduction | Gambit time | BLP time |
|---|---|---|---|---|---|
| 82 | 1301 | 1107 | 14.91% | 21 | 2.7 |
| 83 | 1618 | 377 | 76.70% | 29 | 6 |
| 135 | 3195 | 600 | 81.22% | 120 | 6.5 |
| 204 | 7987 | 1481 | 81.46% | 460 | 3 |
| 201 | 11.2k | 2600 | 76.80% | 900 | 4 |
| 574 | 25.1k | 3475 | 86.17% | N/A | 5 |
| 630 | 44.4k | 3000 | 93.24% | N/A | 14.3 |
| 1438 | 473k | 7823 | 98.35% | N/A | 5.6 |

In EDI-CR, each stage in an agent's decision process consists of an action and outcome for this agent only, resulting in $\mathcal{O}(A^T O^T)$ states per agent. Because there are transition interactions, an agent needs to remember the outcome of affected actions, but our state representation remembers all outcomes, so states space size is independent of $m$.

While theoretically the sizes of EFG and EDI-CR are both exponential in the time horizon $T$, Table 1 shows that in practice, doubling the exponent results in game trees that are too large to build and/or solve.

## 3.2 Mandatory communication

We now model situations where communication is inherently part of the setting, rather than a conscious decision on the part of the agents. An agent $i$ doing its part of a reward or transition interaction *involuntarily leaves a trace* that it has done this action. Consequently, the other agent $j$ will see this trace upon doing *its* part of the action. An agent does not suffer a cost for this implicit communication, but cannot avoid it either. To allow an agent to make decisions based on the traces it sees, an agent's state keeps track of a flag denoting whether a trace was seen upon doing each reward or transition interaction.

In EFG, even though the state now stores the actions, outcomes and $k+m$ flags of each agent, the number of states is *not* $\mathcal{O}(A^{2T}O^{2T}2^{2k+2m})$. The reason is that the values of an agent's flags are fully determined by earlier actions of the other agent, so when an agent does an ineteraction, there is no branching over whether it will see a trace there. In fact, there is no more branching in this communication scheme than in the case without communication, and the number of nodes in the EFG tree is still $\mathcal{O}(A^{2T}O^{2T})$.

Even though they are of the same size, the EFG representation of the no communication case and the mandatory case are not the same. To see why, note that because of the additional flags, an agent has more information available to make its decisions when there is communication. This translates into the game tree having more information sets per agent; nodes that were indistinguishable in the absence of communication can now be told apart. Comparing Tables 1 and 2 shows how much the number of information sets in an EFG increased. Since a policy specifies a distribution over actions for each information set, mandatory communication increases the size of the policy space and makes finding a Nash equilibrium more difficult. Table 2 indeed shows that even though the size of EFG did not change, the solution time generally increased.

As for EDI-CR with mandatory communication, there *is* probabilistic branching in an agent's decision process over whether or not it sees a trace upon doing an interaction, since that depends on what the other agent has done. The size of each agent's process is therefore $\mathcal{O}(A^T O^T 2^{k+m})$.

It is interesting to note the effect of mandatory communication on the size gap between EFG and EDI-CR. Compared to no-communication, mandatory communication results in EDI-CR achieving less reduction in size. The increased coupling introduced by communication makes EFG less inadequate, compared to EDI-CR. If we increase the frequency and language of communication, at some point the decision processes will be so tightly coupled that EDI-CR's advantage of representing them separately will be lost.

## 3.3 Optional communication

We now look at optional communication where an agent can choose whether to leave a trace upon doing its part of an interaction. Even though communication here does have a cost, an agent may still decide to communicate if it knows that communication will cause the other agent to do something beneficial to it. For example, in the Mars rovers scenario, if rover $j$ knows from $i$'s policy that if $i$ visits site $s_1$, then $i$ will visit $s_2$, and if $s_2$ has a much higher value if visited by both rovers, then rover $i$ will choose to leave a trace at $s_1$ as an inducement for $j$ to go there too.

To represent optional communication in EFG, in addition to actions and outcomes for each agent, there is an action node with two branches (leave trace or not) after every decision to do part of an interaction. A state keeps track of the actions and outcomes of both agents, as well as at most $k+m$ binary communication decisions per agent, for a total of $\mathcal{O}(A^{2T}O^{2T}2^{2(k+m)})$ states. Note that even though in this scheme an agent can potentially have the same information to make its decisions as in the mandatory case, the number of decisions itself is much larger, because of the communication decisions, resulting in a larger number of information sets.

In EDI-CR, again, there are communication decision nodes, in addition to branching over whether an agent will see a marker upon visiting a site. The number of states is $\mathcal{O}(A^T O^T 2^{2(k+m)})$.

Table 3 shows that having communication decisions results in huge EFG trees, making it impossible for Gambit to solve them within a reasonable amount of time. However, the $4^{th}$ instance shows that solution time and size are not always correlated, which can be explained by the fact that we are searching for the *first* equilibrium we can find, and the time this takes depends on both the size of the problem and the structure of the search space.

**Table 3: Size and performance comparison for the optional communication case (times in seconds)**

| EFG infosets | EFG size | EDI-CR size | % reduction | Gambit time | BLP time |
|---|---|---|---|---|---|
| 547 | 21.1k | 6213 | 70.58% | N/A | 8.6 |
| 136 | 3777 | 671 | 82.23% | N/A | 3 |
| 190 | 7511 | 1093 | 85.45% | N/A | 2.8 |
| 602 | 51.6k | 5651 | 89.06% | N/A | 214 |
| 589 | 68.3k | 5766 | 91.57% | N/A | 11 |
| 2668 | 295k | 13.9k | 95.29% | N/A | 35 |
| 2004 | 316k | 10.2k | 96.76% | N/A | 32 |
| N/A | 2200k | 21.8k | 99.01% | N/A | 195 |

# 4. COMPACTNESS OF MAIDS FOR LOOSELY COUPLED GAMES

EFG is a representation that does not exploit any structure in a game. In this section, we investigate a more structured representation and its suitability for loosely coupled games. We give a brief background on Multi-Agent Influence Diagrams (MAIDs) and discuss how we can represent our communication schemes using it.

## 4.1 Background on MAIDs

Multi-agent influence diagrams (MAIDs) [11, 3] are representations that have their origins in influence diagrams [7]. Like all alternatives to the extensive form representation, MAIDs try to explicitly capture a structural property of a game that would otherwise be obscured in extensive form. In the case of MAIDs, this property is that not all decision variables in a game are inter-dependent.

A MAID defines a directed acyclic graph in which nodes correspond to random variables of three types. For each agent $i$, there is a set of 1) decision variables, $\mathcal{D}_i$, whose domains are available actions and are represented as rectangles; 2) chance variables, $\chi_i$, whose values are chosen by nature and are represented as ovals; and 3) utility variables, $\mathcal{U}_i$, which represent the agent's payoffs and are drawn as diamonds. A conditional probability distribution (CPD) specifies the conditional probability of the node's variable given an instantiation of its parents, $P(x|Pa_x)$.

A strategy profile for agent $i$ is a set of decision rules, one for each node in $\mathcal{D}_i$. A decision rule specifies the probability of making a certain decision given values of its parents. To represent *perfect recall* (an agent does not "forget" decisions it made in the past), all earlier decisions and their parents are among the parents of a later decision node.

## 4.2 No communication MAID

At each stage, each agent $i$ has a decision node $D_i$, a chance outcome node $O_i$ (e.g., a Boolean representing the slow or fast outcome of visiting a site) and a utility node. To guarantee perfect recall, an agent's decision node depends on all its previous decisions and outcomes. The per-agent utility nodes $U_i$ at each stage represent rewards from individual actions. Addition utility nodes $U_i^\rho$ represent payoffs from reward interactions.

In a naiive representation, $O_i$ and $U_i^\rho$ nodes depend on *all* $j$'s past decision nodes to account for the dependence of $i$'s transitions and the shared rewards on $j$'s actions.

To avoid the blow up in CPDs size that this results in, we introduce *helper nodes* that act as "memory" or storage, allowing us to break some of the dependencies on previous decision nodes and replace them with dependencies on helper nodes at the previous stage only. A helper node is a chance node representing a Boolean variable whose value deterministically depends on that of the corresponding helper node at the previous stage and on the decision node at the current stage. The helper node remembers whether a certain action was done in the past.

For the no-communication scheme, we add, for each agent, a helper node at each stage for each of the $k+m$ interacting actions. The variable of agent $i$'s helper node at stage $t$ indicates whether $i$ did the associated action at or before stage $t$. The node for agent $i$'s $x^{th}$ action that is involved in a reward (resp. transition) interaction is denoted $r_i^x$ (resp. $t_i^x$) and is referred to as *reward indicator* (resp. *transition indicator*). An indicator has the value True if it was True at the previous stage or the associated action was taken at the current stage. Figure 1 shows a no-communication Mars rovers instance with $A = 4, k = m = 2, T = 3$ represented as a MAID.

We now calculate the size of the MAID representation of the no-communication case.

- Decision nodes: because of perfect recall, an agent's decision node has as its parents all its own previous decisions and outcomes. The CPD of a decision node at level $t$ has $A^{t-1}O^{t-1}$ instantiations of parents, for each of which it specifies the probability of $A$ values, for a CPD size of $\mathcal{O}(A^T O^T)$ per agent.
- Because it has a decision node and a Boolean as its parents, a transition or reward indicator's CPD is of size $2A$.
- Outcome nodes: if an action is part of a transition interaction, its outcome probability depends on whether the affecting action was done by the other agent. An outcome node therefore depends on the current level decision node and the other agent's $m$ transition indicators from the previous level. The CPD then specifies the probability of $AO$ values for each of $A \times 2^m$ instantiations of its parents.[3]
- Utility nodes: the individual utility nodes $U_i$ (labeled $u_{i/j}^x$ in the figure), specify a reward for each outcome of each action, resulting in $A^2O^2$ entries for each CPD. For shared reward nodes $U_i^\rho$ (labeled $b_x$ in the figure to denote bonus from the $x^{th}$ interaction), the reward depends on whether each agent has done its part of the interaction as summarized in the last level reward indicators of the $x^{th}$ interaction. Each CPD therefore has 2 Boolean parents, for a total size of $4k$ per agent for all reward interactions.

Even though the size of MAID's decision CPDs is the same order as EDI-CR, the MAID is overall larger because of the CPDs of the other kinds of nodes.

### *Problems with MAID representation*

The above mapping highlights some problems MAIDs have in representing loosely coupled games. First, as can be seen in the figure, the structure in our loosely coupled games (the independence of most actions' rewards and transitions) is obscured because a decision node does not branch over

[3]Note that the outcome node needs the value of all $m$ transition indicators because we do not know which of the $m$ affected actions, if any, this agent will take.

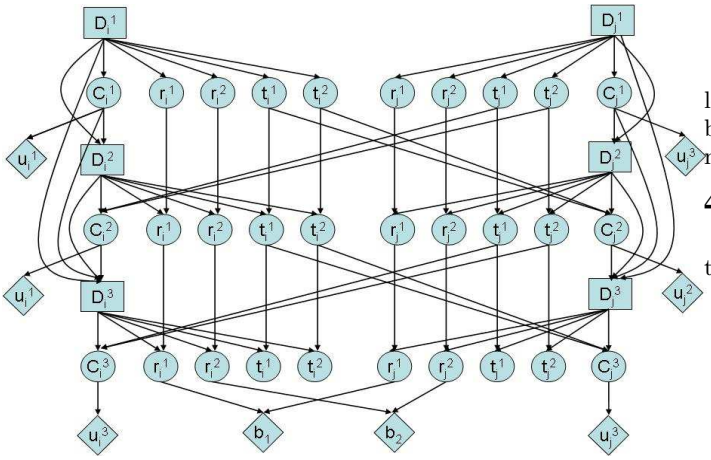

**Figure 1: MAID representation of a no-communication Mars rovers instance**

the possible decisions, so we cannot isolate a single action and represent its dependence on another. Second, MAIDs do not naturally capture dependencies that are temporally non-localized, forcing us to resort to constructs that "remember" actions done in the past and allow them to affect future actions without having the latter depend on all previous decisions. A MAID representation is essentially stateless, and trying to capture a game in which agents have local state that is affected by previous actions and affects the choice of future actions is problematic.

We also note that for simplicity, we assumed that actions can repeat. MAIDs are not good at representing domains where the set of available actions is context-sensitive, which is what would be needed. To disallow repeated actions, we would need, for every pair of an agent's decision nodes, a utility node that imposes a large penalty if the actions taken at these nodes are the same.

## 4.3 Mandatory communication MAID

Representing mandatory communication requires making the following modifications to the no-communication MAID.

- At each stage, we need an indicator for whether an agent sent a message (left a trace) when it did its part of an interaction. This information is already contained in the transition and reward indicators.
- For each agent, each stage, and each of the $k+m$ interactions, we add a helper node called $RCV$ that indicates whether a message regarding the corresponding interaction was received. A $RCV$ is True if it was True at the previous level, or if the agent did its part of an interaction that was also done by the other agent. The parents of a $RCV$ at level $t$ are the $RCV$ from level $t-1$, the other agent's indicator for the interaction from level $t-1$ and this agent's decision node at level $t$, for a size of $4A$.
- We change each decision node at each level $t$ so that in addition to all previous decision and outcome nodes, its parents also include $RCV$ nodes from level $t-1$, allowing an agent to base decision on what messages it received. The size of a decision node at level $t$ is $A^t O^t 2^{k+m}$, resulting total size of decision nodes being $\mathcal{O}(T 2^{k+m} A^T O^T)$.

Mandatory communication exacerbates MAID's main problems because 1) receiving messages depends on actions done by the other agent arbitrarily long ago and 2) decision nodes now have even more information feeding into them.

## 4.4 Optional communication MAID

To represent optional communication, we need to make the following changes to the mandatory communication MAID:

- We add a *SND* node after each decision node. *SND* is a Boolean decision node representing the choice of whether to leave a trace or not. To represent perfect recall, a *SND* has all previous decision, *SND* and $RCV$ nodes as parents. The total size of *SND* CPDs is therefore $\mathcal{O}((4AO(k+m))^{T+1})$.
- Decision nodes have the same set of parents as *SND* nodes, so their size is comparable.
- A $RCV$ node at level $t$ now depends on the $RCV$ at level $t-1$, the decision node at level $t$, and the other agent's *SND* node at level $t-1$.

Experimentally comparing MAID to EFG and EDI-CR was not possible because of the simplifying assumptions we made (actions have unit durations and can repeat) in order to get a reasonable MAID representation, assumptions that place MAIDs and other representations on unequal footings. Without these assumptions, we would get even bigger MAIDs, but even with them, the MAIDs were too large to solve.

# 5. RELATED WORK: STRUCTURED GAME REPRESENTATIONS

**MAID:** As introduced earlier, Multi-Agent Influence Diagrams [11, 3] (MAID) is a representation for sequential games that is suitable for capturing independence among variables, rather than among actions of different agents. Initial work on MAIDs used this representation to decompose a game into interacting fragments, and provided an algorithm that finds equilibria for these smaller games in a way that is guaranteed to produce a global equilibrium for the entire game [11]. Later work addresses the issue that most realistic games are not decomposable in this way. Blum et. al. address this by exploiting finer-grained structure in MAIDs to improve the efficiency of a certain family of algorithms called continuation algorithm [3]. In Section 6, we discuss the possibility of exploiting the structure in loosely coupled games to improve continuation algorithm.

**TAGG:** Temporal action graph games (TAGG) [8] is a graphical representation of imperfect-information extensive form games that can be much more compact than MAIDs; a TAGG can be exponentially more compact than a *naiive* MAID representation. However, a carefully constructed MAID is only polynomial in the size of the TAGG. TAGGs represent games with anonymity (a player's payoffs depend on how many players took a certain action, rather than exactly who they are) and context-specific utility independence. TAGGs are an extension of AGGs to represent games taking place over multiple stages. Because TAGGs are specifically geared towards games with anonymity, we cannot use them to represent our games.

**Succinct EFG:** For some games, the game trees expressed in extensive form are too large to be stored in memory explicitly. To overcome this, Dudik and Gordon propose an implicit representation called *succinct EFG* [5]. A representation is succinct if it has enough information to support certain queries that make it possible to simulate play in a game through sampling. As such, MAIDs are themselves examples of succinct EFGs. However, MAIDs cannot represent context-specific independence (e.g. allowing different decision nodes to have different available actions), a drawback addressed by succinct EFG. For loosely coupled games, however, succinct EFG does not capture the large degree of independence that agents have, and still represents their interaction in a single game tree.

**I-DIDs** [4] model multi-agent interactions extending over time. An agent maintains and updates models of other agents as part of its belief update. We believe this explicit modeling and the maintenance of the models can get expensive. For loosely coupled games, agents may not need to construct and maintain such accurate models of each other.

**Other representations** Unlike the dearth of representations for sequential games, a number of representations have been proposed for 1-stage games with special structure. For example, graphical games [9], Game nets (G-nets) [12] and action-graph games [2] address games whose special structure is the locality of interactions where an agent only interacts with a subset of other agents whose size is small relative to the total number of agents.

The work on poker (e.g. [6]) tries to exploit structure in sequential games to scale to larger games and provides automatic abstractions that produce much smaller games whose solutions can be converted to solutions of the original games. The problem is that with the assumptions they make, it is not clear that these techniques are of general use.

# 6. CHALLENGES OF HAVING MORE THAN TWO AGENTS

Even though we pointed out some weaknesses in using MAIDs to represent loosely coupled games, MAIDs still have one important advantage over our bilinear program formulation: using MAIDs and algorithms developed for MAIDs, we can represent and solve games with more than two agents. In this section, we briefly overview the state-of-the art algorithm for solving MAIDs [3], investigate whether we can exploit the structure in loosely coupled games to make this algorithm more efficient, and highlight the challenges involved in doing so.

## 6.1 Continuation method for MAIDs

Continuation methods work by perturbing a problem into a simpler problem that can be easily solved. The solution is then traced to that of the original problem by decreasing the magnitude of the perturbation. When the perturbation is zero, we have a solution to the original problem.

This approach was used by Blum et. al to solve MAIDs [3]. A large pertubation is applied to the rewards in the form of a bonus vector that rewards an agent for its actions regardless of anything else that happens in the game. If large enough, these bonuses dominate the original game rewards and simply determine what the optimal strategies are.

When applied to MAIDs, tracing the solution of the perturbed problem to that of the original problem requires finding the Jacobian (the first order derivative) of the vector function $V^G(\sigma)$. Each entry $V_a^G(\sigma)$ in this function maps the profile $\sigma$ to the payoffs obtained by the agent playing $a$ for deviating from $\sigma$ and playing $a$ all the time. Using the sequence form representation, if there is a total of $n$ sequences for all agents, then $\sigma$ is a profile of length $n$, $V^G(\sigma)$ is a vector of length $n$ and $\nabla V^G(\sigma)$ is an $n \times n$ matrix.

In an unstructured game, we would need to fill an entry in the Jacobian for each pair of sequences. In a MAID, however, Blum et. al decompose this task into computing a joint marginal distribution for every pair of agents $i$ and $j$, and every utility node $U_i$ of agent $i$ over $Pa_{U_i}$, $D_i$ and $D_j$, where $Pa_{U_i}$ is the set of parents of $U_i$ and $D_x$ is the set of decision nodes of agent $x$. For node $U_i$, the calculation is

$$\sum_{Pa_{U_i}, D_i, D_j} \frac{Util(U_i) * P(Pa_{U_i}, D_i, D_j)}{\sigma_i(D_i)\sigma_j(D_j)}$$

where $\sigma_x(D_x)$ is the realization probability of decisions in $D_x$ as dictated by $x$'s part of the profile $\sigma$, and $Util_i(U_i)$ is $i$'s utility from $U_i$ under a given assignment of the variables in $Pa_{U_i}$, $D_i$ and $D_j$. Note that the above expression is an expectation $E(\frac{Util(U_i)}{\sigma_i(D_i)\sigma_j(D_j)})$ taken over all values of the variables in $Pa_{U_i} \cup D_i \cup D_j$.

Instead of doing naiive inference on the induced Bayesian Network of the MAID[4] , Blum et. al use the clique tree algorithm to compute and cache factors over pairs and triplets of cliques which are later used to calculate the desired marginals. So the joint probability $P(Pa_{U_i}, D_i, D_j)$ would be obtained from a triple factor over the union of variables in the 3 cliques containing these 3 sets of variables. In what follows, we denote the clique containing a set of variables $V$ by $Q(V)$.

## 6.2 Computational challenges

We tried to use Blum et. al's implementation of their continuation method for MAIDs [3] to solve 2-agent instances of Mars rovers[5]. We faced the following computational challenges. The first is typical of most implementations of numerical algorithms, while the second is a more inherent concern.

**Sensitivity to the initial random seed**: The continuation method starts with certain random parameters, which, because tracing the path is not 100% exact, can affect whether a run will find an equilibrium. For our examples, we found it difficult to hit upon a random seed that results in a solution.

**Large size of Jacobian**: In our instances, a profile can easily have 500 elements (with $A = 4$ and $T = 3$, $\sigma$ has 512 elements), so the sheer size of the matrix is very large. Manipulating the matrix, and even constructing it, quickly becomes infeasible.

We next discuss how the second problem can be addressed.

## 6.3 Exploiting loose coupling

The construction of the Jacbobian as described in [3] does not use the fact that not all of an agent's variables affect another agent's reward. Our idea for making the calculation of the Jacobian more efficient is to exploit the structure in loosely coupled games to come up with reduced versions

[4]The induced BN of a MAID under a profile $\sigma$ is obtained by replacing decision nodes in the MAID with random variables whose CPDs are dictated by the decision rules in $\sigma$

[5]We are very grateful to Prof. Christian Shelton of UC Riverside for the code and related discussions.

of the marginal utilities that abstract away details of one agent that are irrelevant to another agent's reward, leading to smaller factors and a speed up in calculation.

In our loosely coupled game, consider calculating the expectation of $i$'s individual utility node at time 2, which only depends on $i$'s decision at time 2 ($d_i^2$) and its probabilistic outcome ($ch_i^2$), so $Pa_{U_i} = \{d_i^2, ch_i^2\}$. $Q(Pa_{U_i})$, however, can potentially include many more variables; decisions of $i$ and their outcomes, as well as reward and transition indicators of $j$. Similarly, a set $D_x$ contains decision variables of agent $x$, but the clique $Q(D_x)$ contains these variables in addition to all but the last outcome variable of $x$. The number of parents of a utility node is therefore much smaller than the number of variables in the union of the 3 concerned cliques.

If we just wanted to calculate $E(Util(U_i))$, we could get rid of all variables except $\{d_i^2, ch_i^2\}$. But because of the terms $\sigma_i(D_i)\sigma_j(D_j)$, we can only get rid of some of these variables. To see how this can be done, we expand the expectation to

$$\sum_{D_i}\sum_{D_j}\sum_{CH_i}\sum_{CH_j}\sum_{T_j}\frac{Util(U_i) * P(D_i, D_j, CH_i, CH_j, T_j)}{\sigma_i(D_i)\sigma_j(D_j)}$$

where $CH_x$ are $x$'s chance outcome variables and $T_x$ are its transition indicators. By pushing terms outward as far as the summations allow, we get

$$\sum_{d_i^2}\sum_{ch_i^2} Util(U_i) \sum_{D_i \setminus d_i^2} \frac{1}{\sigma_i(D_i)} \sum_{D_j} \frac{1}{\sigma_j(D_j)}$$
$$\sum_{CH_i \setminus ch_i^2}\sum_{CH_j}\sum_{T_j} P(D_i, D_j, CH_i, CH_j, T_j)$$

Clearly, the last 3 summations can be eliminated to give

$$\sum_{d_i^2}\sum_{ch_i^2} Util(U_i) \sum_{D_i \setminus d_i^2} \frac{1}{\sigma_i(D_i)} \sum_{D_j} \frac{1}{\sigma_j(D_j)} P(D_i, D_j, ch_i^2)$$

For a utility node representing $i$'s reward from a shared task, the node's parents will be the reward indicator variables at the last level, which indicate whether each agent has done its part of the shared task. Again, our 3 cliques will contain variables that are irrelevant to the expectation, so we marginalize out all variables $CH_i$, $CH_j$ and all reward indicators except the parents of the utility node.

The question we have not resolved yet is how to calculate these smaller joint distributions. Blum et. al calculate joint distributions by manipulating potentials computed in the calibration step of the clique tree algorithm. But since we want distributions over parts of cliques, we cannot do this.

# 7. CONCLUSION

In this paper, we addressed a special kind of stochastic games where the agents are largely independent except for a relatively small number of interactions among them. We characterized this kind of games and proposed a representation that separates the agents' decision processes and enumerates their interactions. Using this representation, we can formulate the problem of finding a Nash equilibrium as a bilinear program. We also discussed the suitability of two existing representations (extensive form games and multi-agent influence diagrams) for loosely coupled games. We introduce different kinds of communication as a way of varying the degree of coupling among agents. We investigate how changing this degree affects the compactness of the three representations we study, both analytically and experimentally. Finally, we looked at the potential of exploiting the special structure in our games to make algorithms for multi-agent influence diagrams more efficient, which would open the way to solving games with more than two agents.

One important future direction of our work is investigating the effect of having communication on the quality, in terms of social welfare, of the Nash equilibria we find. In order to do this, we need to be able to find (a bounded approximation of) the socially optimal equilibrium; the one with the highest total reward. We are currently trying to make the objective function of the bilinear program reflect this requirement.

# 8. REFERENCES


[1] 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.

[2] N. A. R. Bhat and K. Leyton-Brown. Computing nash equilibria of action-graph games. In *Proceedings of the 20th Conference in Uncertainty in Artificial Intelligence*. University of Toronto, July 2004.

[3] B. Blum, C. R. Shelton, and D. Koller. A continuation method for nash equilibria in structured games. *Journal of Artificial Intelligence Research*, 2006.

[4] P. Doshi, Y. Zeng, and Q. Chen. Graphical models for online solutions to interactive pomdps. In *AAMAS 2007*, 2007.

[5] M. Dudík and G. Gordon. A sampling-based approach to computing equilibria in succinct extensive-form games. In *Proc. 25th Conference on Uncertainty in Artificial Intelligence*, Montreal, Canada, 2009.

[6] A. Gilpin and T. Sandholm. Finding equilibria in large sequential games of imperfect information. In *ACM Conference On Electronic Commerce*, pages 160–169, 2005.

[7] R. A. Howard and J. E. Matheson. Influence diagrams. *Readings on the Principles and Applications of Decision Analysis*, pages 721–762, 1984.

[8] A. Jiang, K. Leyton-Brown, and A. Pfeffer. Temporal action-graph games: A new representation for dynamic games. In *Proc. 25th Conference on Uncertainty in Artificial Intelligence*, pages 268–276, Montreal, Canada, 2009.

[9] M. Kearns, M. Littman, and S. Singh. Graphical models for game theory. In *Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence*, CA, USA, 2001.

[10] D. Koller, N. Megiddo, and B. von Stengel. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior*, 14, 1996.

[11] D. Koller and B. Milch. Multi-agent influence diagrams for representing and solving games. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 1027–1034, 2001.

[12] P. La Mura. Game networks. In *Proceedings of the 16th Annual Conference on Uncertainty in Artificial Intelligence*, pages 335–343, San Francisco, CA, 2000.

[13] R. D. McKelvey, A. M. McLennan, and T. Turocy. Gambit: Software tools for game theory, 2007.

[14] 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.

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