Planning for a Single Agent in a Multi-Agent Environment Using FOND

Christian Muise, Paolo Felli, Tim Miller, Adrian R. Pearce, Liz Sonenberg
Department of Computing and Information Systems, University of Melbourne
{christian.muise, paolo.felli, tmiller, adrianrp, l.sonenberg}@unimelb.edu.au

Abstract

Single-agent planning in a multi-agent environment is challenging because the actions of other agents can affect our ability to achieve a goal. From a given agent's perspective, actions of others can be viewed as non-deterministic outcomes of that agent's actions. While simple conceptually, this interpretation of planning in a multi-agent environment *as* non-deterministic planning remains challenging, not only due to the non-determinism resulting from others' actions, but because it is not clear how to compactly model the possible actions of others in the environment. In this paper, we cast the problem of planning in a multi-agent environment as one of Fully-Observable Non-Deterministic (FOND) planning. We extend a non-deterministic planner to plan in a multi-agent setting, allowing non-deterministic planning technology to solve a new class of planning problems. To improve the efficiency in domains too large for solving optimally, we propose a technique to use the goals and possible actions of other agents to focus the search on a set of *plausible* actions. We evaluate our approach on existing and new multi-agent benchmarks, demonstrating that modelling the other agents' goals improves the quality of the resulting solutions.

1 Introduction

Synthesizing plans for agents operating in a multi-agent domain is a challenging and increasingly important problem. While there has been recent progress in this area [Brafman *et al.*, 2013; Bolander and Herzig, 2014], most of this work focuses on synthesizing plans to coordinate a team of agents working towards a collaborative goal. However, many problems must instead take a first-person perspective: a single agent planning in a multi-agent environment, in which the agent must consider the contingencies in which other agents perform actions that affect its own.

This first-person view is essential in not only competitive settings, but also collaborative. Take for example a human-agent team: the agent may synthesize a plan that includes particular actions for the human actors in the environment, but human team members may use additional knowledge or intuition to complete goals in unplanned ways. Even without humans, the first-person perspective may be required; for example, if communication becomes unavailable but agents must continue operating. We therefore must consider computing conditional plans in settings where the environment is collaborative, but out of the planning agent's control.

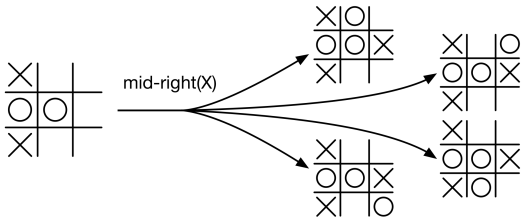


Figure 1: Tic-Tac-Toe example from **X**'s perspective

We approach the first-person multi-agent planning (FP-MAP) problem as a non-deterministic planning problem, where we must synthesize a policy to achieve a goal using actions in which the effects are non-deterministic. The key insight is that actions taken by others can be viewed as non-deterministic outcomes of our own choices. We focus on multi-agent planning problems where the goals and possible actions of all agents are known, and the state of the world is fully observable; restrictions that admit a wide class of multi-agent problems, including many collaborative and competitive games. As an example, consider the partially-played game of Tic-Tac-Toe in Figure 1 where we are player **X** and it is our turn. Playing the right column / middle row can subsequently lead to four possible states where it is our turn again: these are the non-deterministic outcomes of our move. From **X**'s perspective, it is largely irrelevant which action player **O** chooses: it is the resulting state that is important.

In a FP-MAP problem, considering *all* applicable actions for other agents results in an explosively-large search space. To mitigate this, we focus the search on the most *plausible* actions for other agents, which are those that are assessed to be the most valuable to them; e.g., based on the highest heuristic value given the agent's goal. We do not consider how the goals of others are known, but in many scenarios goals are known apriori or can be inferred; e.g., using goal recognition [Ramírez and Geffner, 2010]. This gives us a general means

to handle adversarial settings containing agents with conflicting or opposing goals, and cooperative settings containing agents with same goal as ours (e.g., on the same team); or a spectrum of heterogeneous agents with a variety of conflicting or matching goals. Note that we do not presume authority over our teammates, but instead plan for what they would plausibly do given their goal.

Given the complex nature of characterizing the joint actions that may be possible, using a FOND planner as a black-box is infeasible. We instead modify the state-of-the-art FOND planner PRP [Muise *et al.*, 2012] to handle FP-MAP problems, and evaluate the approach on five new multi-agent domains adapted from existing problems. The evaluation assesses improvements made to the non-deterministic planner for the multi-agent setting, and demonstrates the capability to solve FP-MAP problems with existing planning technology.

2 Background

2.1 (FOND) Planning Notation

Here, we describe briefly the requisite planning background (cf. Ghallab et al. (2004) for a full treatment). A *Fully Observable Non-Deterministic* (FOND) planning problem consists of a tuple $\langle \mathcal{F}, \mathcal{I}, Goal, Act \rangle$; $\mathcal{F}$ is a set of fluents, and we use $\mathcal{S}$ as the set of all possible states; $\mathcal{I} \subseteq \mathcal{F}$ is the initial state; $Goal \subseteq \mathcal{F}$ characterizes the goal to be achieved; and Act is the set of actions. An action $a \in Act$ is a tuple $\langle \text{Pre}_a, \text{Eff}_a \rangle$ where $\text{Pre}_a \subseteq \mathcal{F}$ is the precondition (i.e., the fluents that must hold for a to be executable) and Eff_a is a set of one or more outcomes. Each $e \in \text{Eff}_a$ contains a set of positive and negative effects that update the state of the world. After executing a, *exactly one* of $e \in \text{Eff}_a$ occurs; the agent does not know which in advance, and so must plan for every contingency.

A solution is in the form of a policy P that maps the state of the world to an action: $P : \mathcal{S} \rightarrow Act$. Executing a non-deterministic action in a state can potentially lead to several states. We say that a state s' is *reachable* from state s by the plan P if it is equal to s or if there exists some other state s'' reachable from s such that s' is a possible successor to the execution of $P(s'')$ in state s''.

P is a *weak plan* if there is some state that is reachable from $\mathcal{I}$ where $Goal$ holds. P is a *strong cyclic plan* if for every state s that is reachable from $\mathcal{I}$, there is another state reachable from s where $Goal$ holds. Finally, P is a *strong plan* if it is a strong cyclic plan and no state s reachable from the initial state is reachable from the possible successors of s (i.e., a reachable cycle is impossible). Finally, the *all-outcomes determinization* of a FOND problem $\langle \mathcal{F}, \mathcal{I}, Goal, Act \rangle$ is the classical planning problem $\langle \mathcal{F}, \mathcal{I}, Goal, Act_{AO} \rangle$ where Act_{AO} is defined as: $Act_{AO} = \{\langle \text{Pre}_a, \{e\} \rangle \mid a \in Act \text{ and } e \in \text{Eff}_a\}$

2.2 A First-person View of Multi-agent Planning

We consider a first-person view of planning in a setting where the world is known fully and is observable to all agents. We assume a general framework where action execution for all agents occurs concurrently. This setting follows the NAPL syntax for multi-agent planning introduced in Jensen and Veloso (2000) and Bowling *et al.* (2003), but we make some notation changes for convenience. Formally, we define the multi-agent setting as follows:

Definition 1. First-person Multi-agent Planning Problem

A first-person multi-agent planning (FP-MAP) problem is a tuple $\langle Ag, \mathcal{F}, \mathcal{I}, \mathcal{G}_{i=0\cdots|Ag|-1}, \mathcal{A}_{i=0\cdots|Ag|-1} \rangle$ where:

- Ag is the set of agents in the world, including the planning agent specially designated as 0;
- $\mathcal{F}$ and $\mathcal{I}$ are as before;
- $\mathcal{G}_i \subseteq \mathcal{F}$ is the goal for agent $i \in Ag$; and
- $\mathcal{A}_i$ is the finite set of actions agent i can execute.

The set of joint actions is the cross product of all individual actions: $\mathcal{A} = \mathcal{A}_0 \times \cdots \times \mathcal{A}_{|Ag|-1}$. We will use lowercase letters to denote individual actions and capital letters to denote joint actions (e.g., $a \in \mathcal{A}_i$ and $A \in \mathcal{A}$). A_i denotes the individual action $a \in \mathcal{A}_i$ that agent i executes in the joint action A. Finally, $\mathcal{A}_i^a \subseteq \mathcal{A}_i$ denotes the set of joint actions that are possible when agent i chooses to execute a.

Joint actions constitute a choice from each agent as to what individual action they perform. Every joint action $A \in \mathcal{A}$ is a tuple $\langle \text{Pre}_A, \text{eff}_A \rangle$ where $\text{Pre}_A \subseteq \mathcal{F}$ dictates what must hold for A to be applicable, and eff_A is a single outcome of positive and negative effects.

For convenience, we assume that Pre_a is defined for each individual action. We also make one other key assumption (following the same reasoning as [Bowling *et al.*, 2003]): *the applicability of an individual action should not depend on the individual actions of other agents*:

$$\{A \mid A \in \mathcal{A} \wedge \text{Pre}_A \subseteq s\} = \prod_{i \in Ag} \{a \mid a \in \mathcal{A}_i \wedge \text{Pre}_a \subseteq s\} \quad (1)$$

In a FP-MAP problem, the world is updated through the execution of a joint action. Like FOND, we use $Prog(s, A)$ to indicate the state of the world that results in executing joint action A in state s. It is not restrictive to assume that A_i is defined for every joint action A and agent i, as we can introduce a 'noop' individual action as required, meaning that the agent does nothing. Every joint action has a single outcome, but from the perspective of a single agent, choosing an action can have potentially many outcomes (or consistent joint actions).

> *From the planning agent* 0*'s perspective, choosing individual action $a \in \mathcal{A}_0$ can result in any one of the applicable joint actions in $\mathcal{A}_0^a$ being executed.*

The planning agent's task in our FP-MAP setting is to synthesize a policy that maps states to actions from $\mathcal{A}_0$ given the uncertainty of what other agents may do. The analogies to FOND solutions are direct, and formally we have:

Definition 2. FP-MAP Solution

Given a FP-MAP problem $\langle Ag, \mathcal{F}, \mathcal{I}, \mathcal{G}_{i=0\cdots|Ag|}, \mathcal{A}_{i=0\cdots|Ag|} \rangle$, a solution for agent $0 \in Ag$ is a partial policy $Pol : \mathcal{S} \rightarrow \mathcal{A}_0$ where $Pol(s)$ is applicable in s whenever $Pol(s)$ is defined.

Similar to FOND problems, we define notions of reachability and guarantees on the FP-MAP solution. We say that s'

Algorithm 1: Generate FP-MAP Strong Cyclic Plan

Input: FP-MAP Problem $\langle Ag, \mathcal{F}, \mathcal{I}, \mathcal{G}_{i=0\cdots|Ag|}, \mathcal{A}_{i=0\cdots|Ag|}\rangle$
Output: Partial policy *Pol*

```
1  Initialize policy Pol
2  while Pol changes do
3      if !ContinuePreviousEpoch() then
4          Open = {⟨I, 0, 0⟩}; Seen = {};
5      while (Open ≠ ∅) ∧ (!NewEpoch()) do
6          ⟨s, dist, plaus⟩ = Open.pop();
7          if G_0 ⊈ s ∧ s ∉ Seen then
8              Seen.add(s);
9              if Pol(s) is undefined then
10                 GenPlanPairs(⟨F, s, G_0, A⟩, Pol);
11             if Pol(s) is defined then
12                 for s' ∈ GenerateMASuccessors(s, Pol(s)) do
13                     Open.add(⟨s', dist + 1, Plaus(s')⟩);
14     ProcessDeadends();
15 return P;
```

is *reachable* from s with *Pol* iff one of the following hold: (1) s is equivalent to s'; (2) a state s'' is reachable from s and s' is in turn reachable from s''; or (3) there exists a joint action A such that $Pol(s) = A_0$ and $s' = Prog(s, A)$. A policy *Pol* is *complete* iff $Pol(\mathcal{I})$ is defined as well as every state reachable from $\mathcal{I}$ with *Pol*. A policy for a FP-MAP problem can be weak, strong, or strong cyclic. Note that the other agents in the environment may inherently be unfair; i.e., they may not execute every possible action with non-zero probability.

3 Approach

3.1 MAP as FOND

Our method of solving MAP problems as FOND is straightforward on the surface, but powerful in practice: *we view the set of possible states that result from other agents' actions as non-deterministic outcomes to the planning agent's own choice of action*. Not all FOND planners are built using the same solving technique, but many approaches (e.g., NDP [Alford *et al.*, 2014], FIP [Fu *et al.*, 2011], and PRP [Muise *et al.*, 2012]) build a policy by exploring the reachable state space and augmenting the policy with new plan fragments (i.e., weak plans) when encountering a new state (see Muise *et al.* (2012) for a deeper discussion). The key changes we make for the FP-MAP setting are to define suitable replacements for creating weak plans (GenPlanPairs) and successor states (GenerateSuccessors). Algorithm 1 shows our algorithm with additional elements. These are the core methods used in the FP-MAP setting (further details are given below):

- GenPlanPairs($\langle \mathcal{F}, s, \mathcal{G}_0, \mathcal{A}\rangle, Pol$): Solves the classical planning problem where the actions correspond to every joint action in $\mathcal{A}$ and the goal used is $\mathcal{G}_0$.
- GenerateMASuccessors(s, a): Returns the set of states possible after executing any of the joint actions possible given that agent 0 executes individual a in state s:

$$\{Prog(s, A) \mid A \in \mathcal{A}_0^a\}$$

- ProcessDeadends(): Records the deadends and creates a *forbidden state-action pair*, so that future iterations of the outer loop can avoid joint actions that have at least one outcome leading to any of the deadends.

As an example, consider our Tic-Tac-Toe scenario in Figure 1. One omnipotent strategy is to play in the top-middle spot in lieu of middle-right, as it brings us close to winning with the top row. However, this leaves player **O** with a winning move at middle-right (a deadend). Our forbidden state-action procedure creates rules that forbid every move other than playing the one shown that blocks **O** from winning. In fact, with the deadend generalization already in PRP, our modified forbidden state-action pairs will create a template of the form shown in Figure 2a – no matter what the rest of the board is like ($\star$ indicates any value of the spot), if player **O** is about to win on the middle row then we block.

3.2 Focusing the Policy Search

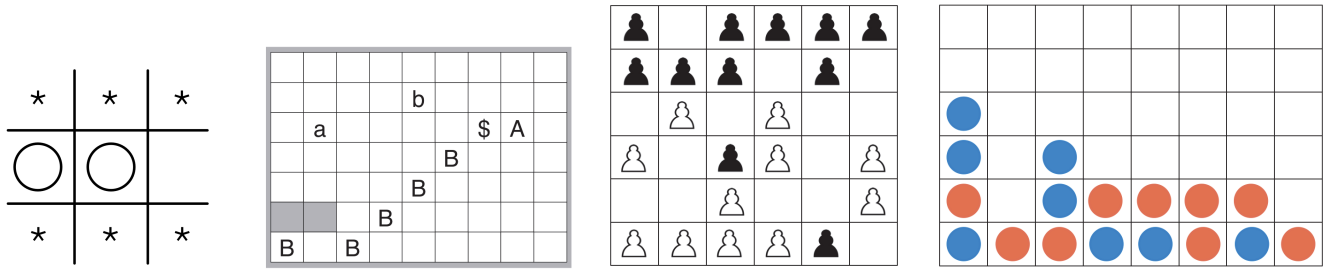
Thusfar, we only use the goal for agent 0, which is acceptable when we can derive a strong plan for agent 0. In practice though, this is often not the case. Here we consider how to focus the search for a policy in two complementary ways when a guaranteed solution is not possible: (1) address states that we are likely to encounter early; and (2) address states that are more likely to be reached. We achieve both by modifying how states are selected (cf. Alg 1 / line 6).

Traditionally, PRP uses a stack for *Open*. Here, we use a priority queue that selects the most *plausible* states, and then selects one among those that minimizes the distance from the initial state (breaking ties arbitrarily). The second criterion involves simple bookkeeping when we add new states to the open list (cf. line 13), and is useful when a complete solution cannot be fully computed. We focus our attention on the criterion of plausibility. Note that one option for further exploration is to be more aggressive in our handling of plausible states and *prune* those states that are not plausible.

There are many ways to define plausibility, and in this work we investigate one of these: *goal-based plausibility*. When the planner determines the set of possible successors using GenerateMASuccessors(s, a), these successors are ranked from the most to least plausible by reasoning *as if* we were the other agents in the domain (i.e., using their goal). Formally, the plausibility score of each successor is computed as follows (where lower is more plausible):

$$Plaus(s) = \sum_{i \in Ag \setminus \{0\}} h(s, \mathcal{G}_i)$$

The heuristic function $h(s, \mathcal{G}_i)$ can be any admissible or inadmissible classical heuristic in the omnipotent equivalent of the FP-MAP problem; that is, for computing $h(s, \mathcal{G}_i)$ we assume that $\mathcal{A}_i$ coincides with $\mathcal{A}$. In practice we use the common h_{FF} heuristic [Hoffmann and Nebel, 2001] as it is computationally efficient, and leave investigating other options as future work. Once we have determined $Plaus(s')$ for all $s' \in$ GenerateMASuccessors(s, a), we re-map the values from 1 to |GenerateMASuccessors(s, a)|. Then, $Plaus(s) = k$ indicates that s was the k^{th} most plausible outcome of an action agent 0 decided to execute. Note that in the calculation



(a) Tic-Tac-Toe Deadend (b) Sokoban (c) Breakthrough (d) Connect4

Figure 2: Example configurations for the Tic-Tac-Toe (partial state), Sokoban, Breakthrough and Connect4 domains.

of *Plaus*, we do not include the heuristic for agent 0, as the individual action for agent 0 will already be chosen.

Combining the above, line 6 selects states that first minimize the remapped *Plaus*(s) score, and then minimize the distance that the planner has gone from $\mathcal{I}$ to reach s in the search for a policy. As a result, all states that are considered most plausible will be investigated first (ties broken on those states we are likely to see earlier on during execution).

3.3 Planning in Epochs

Rather than backtrack, PRP records deadend information during the search and then uses that information in subsequent iterations of the outer loop to avoid falling into the same traps. While effective in many FOND domains, this strategy can be damaging in problems that are too difficult to solve fully.

> *If a single pass of the outer loop exhausts the planner resources, then little deadend information will be used during the search.*

This situation is particularly problematic where a decision early in the search can be detrimental to performance (e.g., making a bad move early in a 2-player game). To remedy this, we introduce *epochs* for the planning process. Given a time limit of T seconds and a bound of k epochs, we allow a single pass of the loop in Algorithm 1 on line 2 to run for T/k seconds before we process the deadends on line 14 and complete another pass. We must be cautious, however, and avoid handicapping the planner if it has the ability to fully solve the problem. We thus use the following strategy:

- If no deadends are found during the previous pass, the search resumes where it left off (cf. line 3).
- If one epoch finishes early, the next epoch is allotted the remaining time.
- If all k epochs have occurred and time remains, the planner continues as normal.
- Half of the total time given to the planner is allocated to 5 epochs (a value that proved most effective across the range of benchmarks). The remainder performs one final pass of the incumbent policy with deadend avoidance disabled, allowing a "best effort" response for any state that the policy does not handle (i.e., do *something* that could succeed, even if it may fail).

The combination of the above finds complete solutions for problems that are small enough to do so, while developing effective strategies for problems that are prohibitively large.

4 Evaluation

We modified the FOND planner PRP as described above to create MA-PRP, and enabled it to parse custom PDDL that specifies the agents and their goals. To simplify the expression of joint actions, the domains enforce a round-robin execution of the agents. This setup is similar to the round robin games specified in the Game Description Logic [Love *et al.*, 2008], and allows us to adhere to Equation (1) effectively.

As our approach opens planning to a new class of problems, there are no publicly available FP-MAP benchmarks to evaluate on as far as we are aware. Instead, we provide a suite of new benchmark problems for five domains: Blocksworld, Sokoban, Tic-Tac-Toe, Breakthrough, and Connect4. We use these to evaluate our proposed strategies for mitigating nondeterminism, and the general ability of the planner to solve fully-observable FP-MAP problems.

We ran several planner configurations to generate the policies for a single planning agent, and we tested the generated policies using 100 simulated trials. Moves for other agents were selected by taking the best applicable action measured using 500 monte-carlo roll-outs per action in the current state, with the stopping condition set to the agent's goal.

The general average success rate for each of the problems and a representative selection of the planner configurations is shown in Figure 3. Some setups did not show anything interesting in most cases, so are omitted for readability; although some are presented later in Table 1. Every run was limited to 2Gb memory and 30min time limit (unless otherwise specified). If the planner did not finish in the time provided, the best incumbent policy computed thus-far was used. The following planner configurations were considered: (**30min**) MA-PRP with 5 epochs and smart priority list (cf. Sec 3.2); (**[3min|30sec]**) reduced time limit; (**1-Epoch**) same as 30min with only one epoch; (**[Plausible|Distance|Stack] OL**) same as 30min with only action plausibility (respectively distance from initial state and original LIFO) used for the open list.

Blocksworld The Blocksworld domain includes the first 10 problems from the benchmark in the 2008 International Planning Competition (IPC), with roughly 5 blocks and three new agents included in addition to the planning agent to evaluate

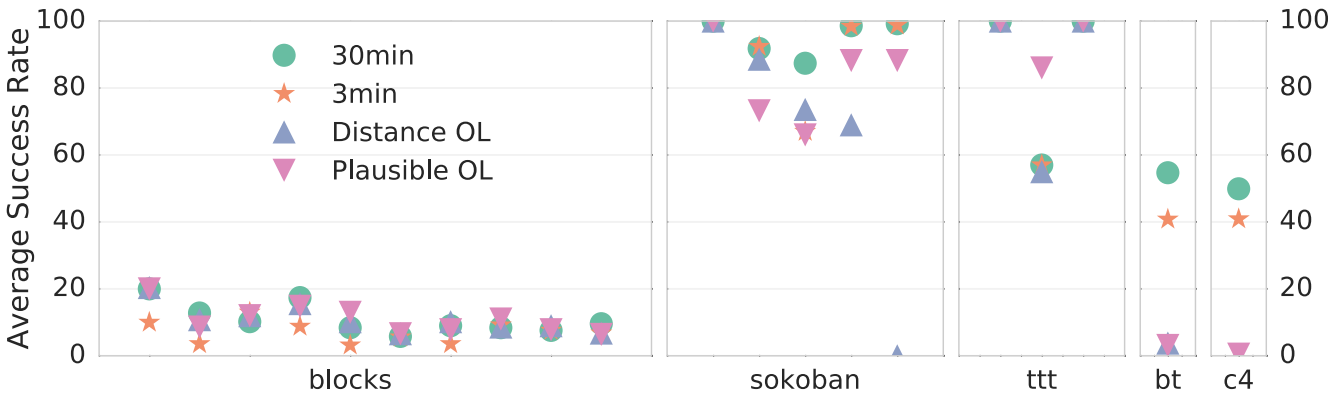


Figure 3: Average number of successful runs for all five domains for four representative planner configurations.

with a range of heterogeneous agents: (1) The **friend** agent has the same goal as us. (2) The **opponent** agent has a conflicting goal. (3) The **stranger** agent has non-interacting goal.

In general the success rate ranged from 10% to 20% for the 30min configuration, with the other configurations not far behind in performance (cf. Figure 3). The interesting aspect of this domain is what happens with the *unsuccessful* trials: a deadlock situation occurred in the vast majority of trials, and the trials ended only after a set limit of actions was reached.

Recall that we build plans under the assumption of "fairness" [Cimatti *et al.*, 2003]: every outcome has some chance of happening. In some domains, this optimistic assumption does not hurt (e.g., in games that we rarely or never see the same state twice). The Blocksworld domain, however, is inherently "unfair". This would be common across adversarial settings. Even if all of the agents shared the same goal, deadlocks may still occur; e.g., if everyone expects someone else to pick up the next block, all agents would idle indefinitely.

Sokoban We extended the Sokoban domain so that the planning agent and a competitive opponent must push a single box each to a different location. Across the five problems in the benchmark, the opponent agent starts at decreasing distances from both our starting location and the block (i.e., in problem p1 the other agent cannot interfere, while in problem p5 the other agent can interfere to a large extent). Figure 2b shows the layout of the Sokoban instances. The planning agent is player **A** pushing the block **$** to location **a**, while the opponent starts in one of the **B** locations and must push the same block to location **b**.

We found an interesting transition for the problem difficulty when the opponent agent was directly below their goal location; demonstrated by the dip in success rate seen in Figure 3 for problem p3. In the later problems, the opponent's monte-carlo simulation concludes that it is best if **A** pushes the block part of the way (note that **B**'s monte-carlo roll-outs do not consider **A**'s goal). However in problem p3, the opponent proceeds under the assumption that they should move the block themselves to **b**, thus causing trouble for agent **A**.

As for the planner configurations, we found that the 1-Epoch performed as well as the 30min configuration, and outperformed it in a few problems. This is due to a good policy computed initially not being recognized as having the best chance of success in the 30min configuration. The 1-Epoch configuration never finishes a single pass of the outer loop in Algorithm 1, and so uses the first and only policy computed. In the Sokoban domain, there is no way to guarantee that the goal is achieved, and the deadend handling discards the initial policy in lieu of later ones that avoid recognized deadends. This motivates the need for better assessment of policy quality in the multi-agent setting, which currently uses a naive approach of random roll-outs using the planning agent's goal.

Tic-Tac-Toe For Tic-Tac-Toe, we considered three variations on the common setup:

1. We play second, the board starts open, and the goal is to not lose the game (i.e., do not let the opponent win).
2. Standard setup with us first trying to win.
3. We play **X**, the board starts with **X**/**O** in the left / middle cells on the bottom row, and the goal is to win.

The opponent's goal is always for them to win. A complete solution (i.e., one that guarantees the goal is achieved) was found for the first and third problems in every planner configuration. The success rate varies for the second problem, but these rates are largely uninformative as the game is too small to adequately distinguish between good and bad policies; an arbitrary choice in the first move yields wildly different results. The standard configuration poses an interesting challenge, as there is no play that can guarantee a win. There is no distinction between losing by a wide margin and drawing a game: if we cannot win in problem 2, then the state is a deadend. Tic-Tac-Toe falls under a generalization of goal achievement where we would like to satisfy a hierarchy of objectives: e.g., I would like to win, and otherwise draw.

Breakthrough The Breakthrough domain is a two-player game from the general game playing contest [Love *et al.*, 2008] that involves moving chess pawns from one end of the board to the other. The winner is the first player to get a pawn to the other side of the board (see an example in Figure 2c where black wins). The starting configuration is two rows of pawns, and the standard chess mechanics apply.

We report the results for every planner configuration in the left column of Table 1. We see a marked improvement when we enable the concepts introduced in Section 3.2. 30sec leads

Configuration	Breakthrough	Connect4
30min	54.7	49.9
3min	40.8	40.9
30sec	25.9	1.8
1-Epoch	53.8	3.4
Plausible OL	3.1	0.5
Distance OL	3.8	0
Stack OL	*mem*	0

Table 1: Average success rates for the various configurations.

to only 25.9% of the trials succeeding compared to 54.7% in the 30min configuration. The 1-Epoch configuration does well, as there are no shallow deadends in this domain. The alternate forms of open list perform poorly: the Plausible-only and Distance-only versions have less than 4% success while the original Stack open list runs out of memory.

Connect4 The final domain is another from the general game playing contest: Connect4. Starting with an empty board, players take turns dropping tokens in a column with the aim of getting four in a row (vertical, horizontal, or diagonal), such as in Figure 2d. Table 1 outlines the results.

Again, the alternate open list types perform poorly. However, note that the performance of 1-Epoch and 30sec configurations have dropped substantially as well. The reason is that there are a number of deadends early in the game. Even with 5 epochs, 30 seconds is not enough time to detect these, but given more time they are properly identified and avoided in future epochs. The Connect4 domain shows the full potential of the methods we introduce, as it is difficult for the planner to finish even a single epoch in a reasonable amount of time. Both the Breakthrough and Connect4 domains have the property of never reaching the same state twice, and so fairness is not an issue. Further, unlike the Sokoban domain, policies from future epochs are always better to use in these domains, as they mitigate bad moves early in the game.

5 Related Work

A similar thread of research for multi-agent planning can be found in the probabilistic planning literature. In particular, Decentralized Partially Observable Markov Decision Processes (DecPOMDPs) aim to model distributed multi-agent planning [Bernstein *et al.*, 2002], and subsequent related formalisms have been introduced to model the qualitative case with QDecPOMDPs [Brafman *et al.*, 2013] as well as the iterative case with I-POMDPs that captures the expected behaviour of the other agents [Gmytrasiewicz and Doshi, 2005]. While similar in spirit, the realization of the framework, and the techniques used to solve the problems, are quite different from what we present here.

The FP-MAP formalism is most related to the work of Jensen and Veloso (2000), although the solution techniques are vastly different. In a fashion similar to FP-MAP, planning is conducted from the perspective of a single agent in a multi-agent environment, including the possibility of non-deternministic actions. On the other hand, many multi-agent planning frameworks assume a form of omnipotence in the planning process: the planning agent has full control over the other agents in the domain (e.g., [Brenner, 2003; Brafman and Domshlak, 2008; Kovacs, 2012]). Under this umbrella, various avenues have been explored to plan with privacy-based constraints [Bonisoli *et al.*, 2014; Brafman, 2015] or plan with explicit representation of the agents belief [Kominis and Geffner, 2015; Muise *et al.*, 2015]. However, the assumption of omnipotence is a common thread between many multi-agent formalisms.

This work is in contrast with treating the environment and the other agents as an explicit adversary, which is the idea behind game structures used in verification [Piterman *et al.*, 2006]. This is in turn at the base of ATL interpretation structures [Alur *et al.*, 2002], in which a successor state is selected depending on the action that each agent performs. The latter approach is the one captured by the notion of joint state-action pairs in Bowling et al. (2003). Although conceptually different, these two approaches allow one to model actions whose effects are not completely determined by the state of the world. The high level concepts we use (e.g., treating agent actions as non-determinism) are also found in game theory formalisms such as extensive form games [Aumann and Hart, 1992]. However, the game theory techniques could not be practically used on the domains we are interested in, as the states are too many to enumerate.

Finally, the work of Bowling *et al.* (2003) considers a setting similar to ours where the goals of the other agents are known. The distinction, however, is that they use the model of agents' goals to devise a game-theoretic notion of equilibria for the agents, whereas we use the information to improve the efficiency of reasoning.

6 Summary

In this work, we presented a novel application of non-deterministic planning in multi-agent environments based on the intuition that actions taken by others can be viewed as non-deterministic outcomes of our own choices. The main contribution of our work is the realization of this intuition in order to leverage the recent advances in non-deterministic planning for solving problems in a multi-agent setting. A second key contribution is a means to focus the search on actions for other agents that are *plausible* given their goal. We demonstrated the ability of our approach to solve multi-agent problems on a new suite of benchmarks that include both collaborative and competitive tasks.

The connection that we make between multi-agent planning and FOND planning presents an exciting initial step towards a range of more sophisticated techniques. The generality of a plausibility function opens the door to techniques ranging from nested simulations to online learning methods that model other agents. It also provides an avenue to incorporate UCT and sample-based approaches (e.g., the PROST planner [Keller and Eyerich, 2012]) with the more systematic search used by determinization-based planners such as PRP. One such opportunity is to use UCT as a mechanism for determining plausibility, and another is to use the plans produced by MA-PRP as guidance for UCT.

Acknowledgements
This research is partially funded by Australian Research Council Discovery Grant DP130102825, *Foundations of Human-Agent Collaboration: Situation-Relevant Information Sharing*.

References

[Alford *et al.*, 2014] Ron Alford, Ugur Kuter, Dana Nau, and Robert P Goldman. Plan aggregation for strong cyclic planning in nondeterministic domains. *Artificial Intelligence*, 216:206–232, 2014.

[Alur *et al.*, 2002] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, September 2002.

[Aumann and Hart, 1992] Robert J Aumann and Sergiu Hart. *Handbook of game theory with economic applications*, volume 2. Elsevier, 1992.

[Bernstein *et al.*, 2002] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Math. Oper. Res.*, 27(4):819–840, 2002.

[Bolander and Herzig, 2014] Thomas Bolander and Andreas Herzig. Group attitudes and multi-agent planning: overview and perspectives. Technical report, 2014.

[Bonisoli *et al.*, 2014] Andrea Bonisoli, Alfonso Emilio Gerevini, Alessandro Saetti, and Ivan Serina. A privacy-preserving model for the multi-agent propositional planning problem. In *21st European Conference on Artificial Intelligence*, pages 973–974, 2014.

[Bowling *et al.*, 2003] Michael H. Bowling, Rune M. Jensen, and Manuela M. Veloso. A formalization of equilibria for multiagent planning. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 1460–1462, 2003.

[Brafman and Domshlak, 2008] Ronen I. Brafman and Carmel Domshlak. From one to many: Planning for loosely coupled multi-agent systems. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling*, pages 28–35, 2008.

[Brafman *et al.*, 2013] Ronen I. Brafman, Guy Shani, and Shlomo Zilberstein. Qualitative planning under partial observability in multi-agent domains. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[Brafman, 2015] Ronen I. Brafman. A privacy preserving algorithm for multi-agent planning and search. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 1530–1536, 2015.

[Brenner, 2003] Michael Brenner. A multiagent planning language. In *Proceedings of the ICAPS Workshop on PDDL*, volume 3, pages 33–38, 2003.

[Cimatti *et al.*, 2003] A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, 147(1):35–84, 2003.

[Fu *et al.*, 2011] Jicheng Fu, Vincent Ng, Farokh B. Bastani, and I-Ling Yen. Simple and fast strong cyclic planning for fully-observable nondeterministic planning problems. In *Proceedings of the 22nd International Joint Conference On Artificial Intelligence*, pages 1949–1954, 2011.

[Ghallab *et al.*, 2004] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated planning: theory & practice*. Elsevier, 2004.

[Gmytrasiewicz and Doshi, 2005] Piotr J. Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005.

[Hoffmann and Nebel, 2001] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[Jensen and Veloso, 2000] Rune M. Jensen and Manuela M. Veloso. OBDD-based universal planning for synchronized agents in non-deterministic domains. *Journal of Artificial Intelligence Research*, 13:189–226, 2000.

[Keller and Eyerich, 2012] Thomas Keller and Patrick Eyerich. PROST: Probabilistic Planning Based on UCT. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling*, pages 119–127. AAAI Press, June 2012.

[Kominis and Geffner, 2015] Filippos Kominis and Hector Geffner. Beliefs in multiagent planning: From one agent to many. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling*, pages 147–155, 2015.

[Kovacs, 2012] Daniel L Kovacs. A multi-agent extension of pddl3.1. In *Proceedings of the Workshop on the International Planning Competition*, page 19, 2012.

[Love *et al.*, 2008] Nathaniel Love, Timothy Hinrichs, David Haley, Eric Schkufza, and Michael Genesereth. General game playing: Game description language specification, 2008.

[Muise *et al.*, 2012] Christian Muise, Sheila A. McIlraith, and J. Christopher Beck. Improved Non-deterministic Planning by Exploiting State Relevance. In *The 22nd International Conference on Automated Planning and Scheduling*, 2012.

[Muise *et al.*, 2015] Christian Muise, Vaishak Belle, Paolo Felli, Sheila A. McIlraith, Tim Miller, Adrian R. Pearce, and Liz Sonenberg. Planning over multi-agent epistemic states: A classical planning approach. In *The 29th AAAI Conference on Artificial Intelligence*, 2015.

[Piterman *et al.*, 2006] N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive(1) designs. In *VMCAI*, pages 364–380, 2006.

[Ramírez and Geffner, 2010] Miquel Ramírez and Hector Geffner. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.