Presentation Plan
FF-Replan: & RFF: Exploiting classical AI planning for uncertain and probabilistic domains

I.C.T.M Speek, A.C. Stolwijk

Seminar Algorithms, Embedded Software, Msc Embedded Systems,
Delft University of Technology
{i.c.t.m.speek,a.c.stolwijk}@student.tudelft.nl

1 Summary of given papers

1.1 FF-Replan: A Baseline for Probabilistic Planning

At the first international probabilistic planning competition IPPC-04, most entries were Markov Decision Processes. The winner, FF-Replan, however was based on deterministic planning techniques.

FF-Replan is an action selection algorithm for online planning in probabilistic domains. Given a domain description, a goal and an initial state as a probabilistic planning problem, FF-Replan generates a deterministic planning problem. It determinizes the input domain at the start of the planning into a deterministic different domain. They consider two methods of determinization:

1. *Single-outcome determinization*: selects one outcome for each probabilistic construct using a variety of heuristics. The performance strongly depends on the methods for choosing the outcomes.
2. *All-outcomes determinization*: considers every probabilistic outcome as a dinstinct deterministic action. For each possible outcome FF-Replan generates one seperate action for each effect. For nested probabilities, the all-outcomes procedure is applied recursively. This implies that FF-Replan will ceise selecting actions whenever it enters a dead state. However when given no information, all probabilistic effects will be considered as equal which is a potential weakness.

After determinization FF-Replan then uses the deterministic FF planner to compute an ordered plan for the generated deterministic problem. This process is repeated when countering an unexpected state untill a goal state is reached. It maintains a partial state-action mapping using a hash-table which is initially empty. Whenever FF-Replan encounters a state which is not in the table, it determinizes the problem and synthesizes a plan using FF. It then simulates the plan according to the deterministic action definitions resulting in a state-action sequence whose pairs are put in the hash table. The first action of the plan is then executed in the environment, returning a new state. This partial policy

is this produces in an online fashion. It would be valuable to look into incremental planning approaches that don't create a deterministic plan from scratch whenever an unexpected state is encountered or reusing previous planning effort.

FF-Replan has some obvious shortcomings as it first determinizes the input domain, removing all probabilistic information from the problem and then synthesizes a plan. If an unexpected state should occur whilst executing, the planner replans in the same determinization. It also does not consider multiple potential effects if an action.

By relying on search, FF-Replan could function with enormous speed and efficiency. However when dealing with difficult probabilistic problems attempting to span the spectrum of complexity ranging from deterministic problems to highly stochastic problems, FF-Replan is expected to come short. Because FF-Replan is developed hoping to inspire extensions and insight into the approach and planning domains, the authors are hoping to achieve this.

1.2 Incremental Plan Aggregation for Generating Policies in MDPs

Markov Decision Processes (MDPs) is the best known formalism for planning under uncertainty. It can however be computationally intensive if large parts of the policies are tried to optimize at once. Most recent MDP planning techniques are using the following steps as a basis approach:

1. Take MDP problem
2. "Determinize" it into a classical planning problem
3. Generate sequence of actions that represent a possible execution path
4. MDP planner executes plan:
   - If goal is reached, reports success
   - Otherwise algorithm generates a new plan from failed state of the world

There are some issues with this approach:

- It stops after the first computed path to a goal which is likely to fail
- May result in generating the same path over and over again
- Only generates an execution path, not a policy

The authors identified four aspects of policy generation for MDP planning problems using a classical planning algorithm:

1. How to determinize the MDP planning problem
2. which plan (initial state, goal) to ask for
3. How to incrementally aggregate the plans by the classical planner into a solution policy
4. How to combine this coherently

The paper presents the Robust FF (`RFF`) planner, which should solve the issues. `RFF` first generates an initial plan from the initial state to some goal using `FF` on a determination of the MDP. There are several strategies to do this.

MostProbableOutcome (MPO) only uses the action from a state with the highest probability. AllOutcomes (AO) uses all actions from a state. Then for each state in the plan, the successors of that state are added too. This is expanding the execution structure. For the new state the probability to reach this state should be higher than a certain threshold and if `FF` does not return a failure for this state and a certain goal. The goals can be chosen in three different ways, using ProblemGoal, which uses the goals of the MDP, RandomGoal, which randomly picks some states from the policy graph and BestGoal which picks the best states from the policy graph. The calculation of the probability to a state from the initial state can be costly. To optimize this in `RFF` a Monte-Carlo (MC) simulation is used. `RFF` terminates if either the global probability is lower than the threshold, or there are no new terminal states to be added.

Three theorems, one lemma and one corollary about MDP problems using `RFF` are given:

Lemma 1 For every MDP planning problem RFF terminates in finite time

Theorem 1 (Soundness of $\texttt{RFF}_{\texttt{MPO}}$). For every MDP planning problem, every solution that $\texttt{RFF}_{\texttt{MPO}}$ finds is correct

Theorem 2 (Completeness of $\texttt{RFF}_{\texttt{AO}}$). For every MDP planning problem, if a solution exists, it is found by $\texttt{RFF}_{\texttt{AO}}$ or returns a failure

Corollary 1 (Soundness of $\texttt{RFF}_{\texttt{AO}}$). For every MDP planning problem, every solution that $\texttt{RFF}_{\texttt{AO}}$ finds is correct

Theorem 3 If there are no unsolvable states in the MDP, then the probability of success of any solution found by `RFF` is higher than $1 - \rho$ where ρ is the threshold probability

For the IPC-08 the authors have implemented `RFF` in C++ to perform experiments. It performed experiments on the "fully-observable probabilistic (FOP) planning track". At IPC-08 it performed better than other competitors. The experiments confirmed Theorem 3, a higher threshold mean that `RFF` will generate policies that are more prone to replanning.

In the experiments different Goal-Selection strategies were tested. ProblemGoal (PG) performed the worst. RandomGoal performed better and BestGoal generally produced higher quality goals, but also took more time.

The different ways of determinizing MDPs was also tested. It turned out that AllOutcomes solve much larger problems. However MostProbableOutcome generally solve more problems in less time.

2 Criticism on the contribution of the paper

3 Relevant literature on the subject

– Teichteil-Koenigsbuch, Florent, Guillaume Infantes, and Ugur Kuter. "RFF: A robust, FF-based MDP planning algorithm for generating policies with low probability of failure." *Sixth International Planning Competition at ICAPS* 8 (2008).

– Weld, Andrey Kolobov Mausam Daniel S. “Determinize, Solve, and Generalize: Classical Planning for MDP Heuristics.”

4 Presentation setup

– Introduction
– Brief introduction to prior knowledge
– Summary of the papers
– Advantages and disadvantages
– Comparison with ideas of other papers
– Some initial ideas of improvements