Computer Science 2542 March 1, 2019
St. George Campus University of Toronto

Homework Assignment #1: Planning Formalisms

Due: — by 11:59 PM

Silent Policy: *A silent policy will take effect 24 hours before this assignment is due, i.e. no question about this assignment will be answered, whether it is asked on the discussion board, via email or in person.*

Total Marks: This part of the assignment represents 10% of the course grade.

Handing in this Assignment

What to hand in on paper: Nothing.

What to hand in electronically: You must submit your assignment electronically. Download the assignment files from the course web page. Modify `river.fond.pddl` and `river.prob.pddl` appropriately so that they address the issues specified in this document. **In addition to your modified** `river.fond.pddl` **and** `river.prob.pddl`**, submit all your written answers in a single PDF document.**

How to submit: Submit your assignment by email to `csc2542tas@cs.toronto.edu`. Your email's subject line should read "[CSC2542 - Assignment 1] ⟨YOUR NAME⟩". Please submit all three submission files in a single zip or tgz attachment.

Extra Information

Installation Instructions: Installation instructions for the required software are available in the appendix. They have been tested to work on a fresh Ubuntu 64bit installation. The amount of disk space required for the installation is less than 2GB.

Clarification Page: Important corrections (hopefully few or none) and clarifications to the assignment will be posted on the Assignment 1 Clarification page.[1] You are responsible for monitoring the A1 Clarification page.

Questions: Questions about the assignment should be asked on Piazza.[2] If you have a question of a personal nature, please email the TAs (`csc2542tas@cs.toronto.edu`) or the instructor (`csc2542prof@cs.toronto.edu`) placing 2542 and A1 in the subject line of your message.

---

[1] `https://www.cs.toronto.edu/~sheila/2542/w19/Assignments/A1/a1_faq.html`.
[2] `https://piazza.com/utoronto.ca/winter2019/csc2542/`.

Introduction

In this assignment we will look at how to model and solve planning problems. The main objective of the assignment is for you to get your hands dirty and get to use actual, state-of-the-art planners. In so doing, you will also see the impact of using different modeling formalisms and producing different solution descriptions. To that end, we will take a relatively simple problem as a running example, and we will model it as a deterministic, non-deterministic, and probabilistic planning problem.

We will be using the *River* problem, which is based on the following situation:

> You are on one side of a river, and want to get to the other side. There are some rocks that look like they could be traversed. They are slippery though, so there is a 75% chance you would slip and fall. If that happened, there would be a 1 in 3 chance that you would drown in the current, but you would probably be able to make it to a small island in the middle of the river. As an alternative, there is a place further down the river where you might be able to swim across. The current is strong though, so you give yourself an even chance of making it. If you could get to the island you would have a better chance, around 80%. There is no way of swimming there directly, though; the current is just too strong. What do you do to maximise your chance of getting to the other side without drowning?[3]

Part 1: Deterministic Planning (6 marks)

An example of the *River* problem has been given to you in PDDL. Recall that in PDDL problems are divided in two files. The *domain* file contains a description of the dynamics of the environment (an *action theory* for the environment). The *instance* (also referred to as *problem*) file contains the initial and goal conditions, and sometimes predicates specific to the instance that you want to model and solve.

The PDDL domain and instance files for the *River* problem modeled with deterministic actions are available in:

```
models/river.pddl
models/instance.pddl
```

Now, follow these instructions carefully:

1. Describe—in your own words—the dynamics, the initial state, and the goal of the *River* problem as specified by the PDDL files. Keep it to three sentences, approx. **(3 marks)**
2. Use an automated planner to get a plan for the River problem. We will use a planner that is accessible on the cloud:
   (a) Go to `http://editor.planning.domains/`.
   (b) Copy or upload the domain and instance PDDL files.
   (c) Hit *Solve* and let the magic happen. You found a plan!
3. Reflect on the plan, and why the agent chose it. Answer the following questions:

[3] Little, Iain, and Sylvie Thiébaux. "Probabilistic planning vs replanning." In *ICAPS Workshop on IPC: Past, Present and Future*. 2007.

(a) What is the plan found by the planner? i.e., the sequence of actions that take the agent from the initial state to the goal state. **(1 mark)**

(b) Is the plan optimal? i.e., does it have the minimum number of steps among all plans that solve the problem? **(1 mark)**

(c) Why is the shorter plan [`(swim-river)`] not a solution to the problem? **(1 mark)**

Part 2: Non-Deterministic Planning (9 marks)

We will follow up with the *River* example—where the agent wants to cross the river. Modeling the problem as a deterministic planning problem is perhaps too simplistic. The kind of dynamics that we want to model is for the agent to be able to cross the river, but not with complete certainty of success—we know that crossing the river in real life may be dangerous. This time we will take a slightly more complex formulation of the problem, as a *fully observable non-deterministic* (FOND) planning problem. In a FOND planning problem we do not have probability distributions over the action outcomes; instead, we have *possibilities*.

The PDDL files for FOND planning problems are similar to the PDDL files for deterministic planning problems. Two main exceptions occur:

- The `:non-deterministic` requirement flag appears in the header of the PDDL domain file. This is to tell the planner that the domain includes non-deterministic actions.
- Non-deterministic actions may have non-deterministic effects. These are modeled with `(oneof` e_1 e_2 `...` e_n`)` statements, that tell that the result of an action is given by only one of the effects e_1, e_2, $\ldots$, e_n.

An example of the *River* problem has been given to you in PDDL, modeled as FOND. The PDDL files for the *River* problem, modeled as a FOND planning problem, are available in:

```
models/river.fond.pddl
models/instance.pddl
```

1. Describe—in your own words—the dynamics, the initial state, and the goal of the *River* problem. Keep it to three sentences, approx., and make sure that the differences between this model and the deterministic planning model from the previous question are made clear. **(3 marks)**

2. Is there a policy that guarantees the agent will achieve the goal, in spite of the non-determinism? Why? **(2 marks)**

3. Now we will search for a strong-cyclic policy for the River problem[4]. There is no FOND planner in the cloud, so we will install one locally.

   (a) Look at the installation instructions in this document's appendix, and install PRP.

   (b) Run PRP on the non-deterministic model of the *River* problem:

   ```
   ./PRP/src/prp models/river.fond.pddl models/instance.pddl \\
                               --trials 100000 --dump-policy 2;
   ```

[4] A strong cyclic policy for a FOND planning problem is a policy that guarantees eventual achievement of the goal, prescribed that all the effects of an action will eventually occur if the agent tries infinitely often.

(c) Translate the policy file (`output`) generated by PRP to a human-readable policy.

```
python2 ./PRP/prp-scripts/translate_policy.py
```

Warning: PRP outputs a policy file `output` in the working directory. That file is then read by `translate_policy.py`. Make sure you run the two commands above sequentially (in other words: don't mess up by translating the policy you've computed to another problem.)

Observe that:

- PRP returns 'Plan found, but not strong cyclic.'. That means that there is no way to guarantee achievement of the goal, but the planner returned the best policy it found during search.
- PRP reports 'Policy Score: 0.507' (or a similar number). This number has no probabilistic meaning. However, it means that among all the simulations that PRP performed on the best policy, only a fraction of 0.507 achieved the goal—i.e not all of them.

4. What are the consequences of the `(swim-river)` action if the first non-deterministic effect is the one that occurs? Which actions would the agent be able to perform afterwards? **(1 mark)**

5. Create a new PDDL domain file that modifies the given PDDL FOND model. In the new file, to modify the first non-deterministic effect of the action `(swim-river)` in a way that strong cyclic solutions exist. You cannot incorporate new actions or declare new predicates.

   As a sanity check:

   - PRP has to report that a 'Strong cyclic policy was found.'
   - PRP has to report 'Policy Score: 1.'

   Describe the changes that you made in the PDDL domain file, and explain what would be a strong cyclic solution. Is it also strong?[5]Attach the modified PDDL file as part of your submission. **(3 marks)**

Part 3: Probabilistic Planning (13 marks)

We will follow up with the *River* example. Recall that the model as a FOND planning problem had no strong cyclic solution, meaning that no policy existed that guaranteed achievement of the goal. We are now going to incorporate probabilities into the model, and search for policies that attempt to maximize the probability to achieve the goal.

The first step is to model the problem. For this, we will use PPDDL, the probabilistic version of PDDL. Alternative and possibly more interesting formalisms—such as RDDL[6], where a lifted propositional language is used to model MDPs—are too complicated for the scope of this assignment. PPDDL is very similar to the FOND version of PDDL:

- The `:probabilistic-effects` requirement flag appears in the header of the domain file, instead of the `:non-deterministic` flag. This tells the planner that the domain includes probabilistic actions.

[5]A strong solution is one that guarantees eventual achievement of the goal, without making any assumptions about how the environment chooses what non-deterministic effects take place.

[6]`http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf`

- Probabilistic actions may have probabilistic effects. Instead of modeling these with `oneof` predicates, we use `probabilistic` predicates: (`probabilistic` p_1 e_1 p_2 e_2 ... p_n e_n) statements, that tell that the result of an action is given by effect e_i with probability p_i.

An example of the *River* problem has been given to you in PDDL, modeled as a probabilistic planning problem. The corresponding PDDL files are available in:

```
models/river.prob.pddl
models/instance.pddl
```

1. Describe—in your own words—the dynamics, the initial state, and the goal of the *River* problem. Observe that some action effects change with respect to previous formulations. Keep it to three sentences, approx. **(3 marks)**
2. What is the probability of drowning immediately after the `(traverse-rocks)` action is performed? **(1 mark)**
3. What is the probability of achieving the goal after the `(traverse-rocks)` action is performed? **(1 mark)**
4. What is the policy that maximizes the probability of achieving the goal? What is the probability of achieving the goal? **(1 mark)**
5. Follow the instructions to install ProbPRP, and run it on the probabilistic PDDL files. What is the 'Policy Score'?

```
./ProbPRP/src/prp river.prob.pddl instance.pddl --final-fsap-free-round 1 \\
                                                --optimize-final-policy 1 \\
                                                --trials 100000          \\
                                                --dump-policy 2;
python2 ./ProbPRP/prp-scripts/translate_policy.py
```

6. The probability of reaching `on-far-bank` by action `(swim-river)` is fixed to 50%, and the probability of reaching `on-far-bank` by action `(swim-island)` is fixed to 80%. Suppose furthermore that the probability of reaching `on-far-bank` and `on-island` by action `(traverse-rocks)` is the same. What is the minimum probability of drowning by action `(traverse-rocks)` so that the action `(swim-river)` becomes optimal? **(1 mark)**
7. Make a copy and modify the PDDL so that it reflects the probability distribution computed in the question above. Describe briefly (1 sentence) what you did and attach the PDDL file to your submission. **(1 mark)**
8. Run ProbPRP on the new PDDL files. What is the new 'Policy score'? Did it find the optimal policy? **(1 mark)**
9. The way ProbPRP works is to first make a guess on a *deterministic* plan that has high likelihood to occur. Then, it completes that plan to make it as robust as possible. If the first deterministic plan was misleading, it's possible that the policy found will not be optimal.
   Take a look at the file `river.multiprob.pddl`.

(a) Do the probabilities of success (i.e., achieving `on-far-bank`) and failure of the action `(swim-river)` change with regard to the domain `river.prob.pddl`? **(1 mark)**

(b) Run ProbPRP on the new domain. What is the probability of success? Did it find the optimal policy? Contrast the result with question 8. What do you think happened? **(3 marks)**

HAVE FUN and GOOD LUCK!

Appendix: Installation Instructions

This appendix contains instructions to install the different planners used in this assignment. It has been tested to work on a fresh Ubuntu 64bit installation. The disk space needed for the installation of all planners amounts to less than 2GB. If you run short of space, try deleting the benchmark-associated folders in PRP and ProbPRP. If you have further issues, contact the TAs.

Fully Observable Non-Deterministic Planning (FOND)

We will install a state-of-the-art non-deterministic planner, PRP. PRP is based on aggregation of deterministic plans to create a policy that is robust to the non-deterministic effects of the actions. PRP uses Fast-Downward (FD) as an underlying deterministic planner (the installation is standalone and integrates FD).

```
sudo apt-get update
sudo apt-get install cmake g++ mercurial make python
sudo apt-get install gcc-multilib g++-multilib
hg clone ssh://hg@bitbucket.org/haz/deadend-and-strengthening PRP
cd PRP/src/
./build_all
```

Now the executable is in `PRP/src/prp`. The commands we will use to run PRP and print out the resulting policy are:

```
./PRP/src/prp <domain.pddl> <instance.pddl> --trials 100000 --dump-policy 2;
python2 ./PRP/prp-scripts/translate-policy.py
```

Probabilistic Planning

We will install what until very recently was a state-of-the-art MAXPROB planner, ProbPRP.[7] MAXPROB planners act in stochastic environments, and the goal is to maximise the probability of reaching a goal. ProbPRP is based on PRP, that is, it performs plan aggregation to construct a policy that is robust to the stochasticity of action effects, and gives preference to explore plans that have high likelihood. ProbPRP is a best effort planner: this means that it gives no formal guarantees of optimality.

ProbPRP is in the same repository as PRP, in the branch 'probabilistic'. We highly recommend you do a fresh install of ProbPRP in a separate folder.

```
sudo apt-get update
sudo apt-get install cmake g++ mercurial make python
sudo apt-get install gcc-multilib g++-multilib
hg clone ssh://hg@bitbucket.org/haz/deadend-and-strengthening -b probabilistic ProbPRP
cd ProbPRP/src/
./build_all
```

Now the executable is in `ProbPRP/src/prp`. The commands we will use to run ProbPRP and print out the resulting policy are:

[7] Some recent papers claim it's been outperformed!

```
./PRP/src/prp <domain.pddl> <instance.pddl> --final-fsap-free-round 1 \\
                                            --optimize-final-policy 1 \\
                                            --trials 100000           \\
                                            --dump-policy 2;
python2 ./PRP/prp-scripts/translate-policy.py
```