Planning and Approximate Reasoning

PART 1: Planning

November, 9th, 2022

Time: 2 h. You cannot use any printed or electronic materials. Please give clear and detailed answers (with examples, if appropriate). All the answers must be properly justified to be valid. You should show your work and the necessary calculations for the full credit.

---

1. Classical Planning and High Level Representation

(1 point) Explain briefly the idea of the problem solver of Means-Ends analysis (MEA).
(1 point) Non-linear Planning (N-LP) uses a goal set instead of a goal stack and it includes in the search space all possible subgoal orderings. Explain in details the advantages and the disadvantages of N-LP, and give an example of N-LP algorithms.
(1 point) With state-space planning, we generate plans by searching through state space using Forward or Backward searching. Explain the difference between the two types of searching.

2. PDDL Implementation

Gripper task with four balls: There is a robot *x* that can move between two rooms, r_1 and r_2 and pick up or drop balls with his two arms. Initially, all balls and the robot are in the first room r_1. We want the balls to be in the second room, r_2. The robot *x* can hold two balls at one time. So,

- ***Initial state***: All balls and the robot are in the first room. All robot arms are empty.
- ***Goal***: All balls must be in the second room.
- ***Actions/Operators***: The robot can move between rooms and pick up or drop the ball.

Objects*:*

Rooms: rooma, roomb
Balls: ball1, ball2, ball3, ball4
Robot arms: left, right

Predicates:

ROOM(r) – true iff r is a room
BALL(b) – true iff b is a ball
GRIPPER(x) – true iff x is a gripper (robot arm)
at-robby(r) – true iff r is a room and the robot is in r
at-ball(b, r) – true iff b is a ball, r is a room, and b is in r
free(x) – true iff x is a gripper and x does not hold a ball
carry(x, b) – true iff x is a gripper, b is a ball, and x holds b

The predicates in PDDL to solve the Gripper task are:

```
(:predicates
    (ROOM ?r) (BALL ?b) (GRIPPER ?x)
    (at-robby ?r) (at-ball ?b ?r)
    (free ?x) (carry ?x ?b)
)
```

The three actions, you need to select from them to update the world state, are.

Action	Usage
move (?r1, ?r2)	Action "**move**" for the robot to move from room *r1* (source) to room *r2* (destination).
pick-up (?b ?r ?x)	Action "**pick-up**" for the robot-arm "*gripper" x* can pick up *b* in a room *r*.
drop (?b ?r ?x)	Action "drop" for the robot-arm "*gripper" x* can drop *b* in a room *r*.

(2 points) Based on the problem described above, write the declaration of actions ***move*** and ***drop*** in PDDL showing the parameters, preconditions, and effects, including add and delete lists.

3. GraphPlan

Suppose you want to prepare dinner as a surprise for your friend (who is asleep). You can use GraphPlan as a planner to deal with the tasks and domain of this problem. If we assume your world has five predicates:

cleanHands – *true iff your hand is clean*
quiet – *true iff the house is quiet*
garbage – *true iff you have a garbage*
dinner – *true iff the dinner was already prepared*
present – *true iff the dinner was wrapped*

In addition, there are three actions available

cook – *to prepare the food,*
wrap – *to prepare the present,*
remove – *to remove the garbage from the house*

where

Action	**Preconditions**	**Effects**
cook	cleanHands	dinner, garbage
wrap	quiet, dinner	present
remove	garbage	~garbage, ~cleanHands, ~quiet

Initial State: {cleanHands, quiet, garbage}
Goal: {dinner, present, ~garbage}

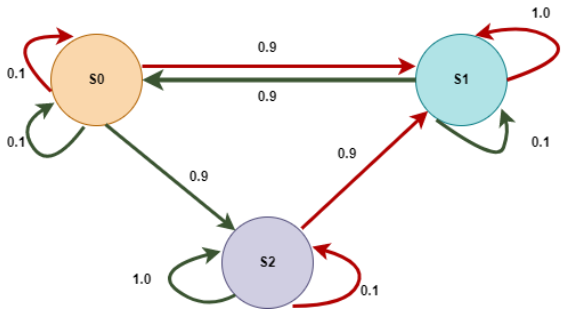
(1 point) Construct the graph structure of GraphPlan (without mutex), adding the literals of propositions and actions, including no-operation that can be achieved in the first step of the graph.
(1 point) Show the list of the independent actions in the constructed graph.

4. Markov Decision Process (MDP)

Consider the Markov decision process (***MDP***) with three states $s \in \{S0, S1, S2\}$, two actions $a \in \{$ *sleeping, working*$\}$, discount factor $\lambda=0.9$, and the reward, $R(S)$, of each state is shown below:

S	*S0*	*S1*	*S2*
R(S)	-100	+100	+5

And the probability of transition between each pair of states for the *sleeping* and *working* actions is shown in the following figure:



where *red* connections are for the *working* action and *green ones* for the *sleeping* action.

(1 point) (a) consider the policy π that chooses the action a = *sleeping* in each state. For this policy, solve the linear system of Bellman equations (by hand) to compute the state-value function $V^{\pi}(S)$ for $S \in \{S0, S1, S2\}$. <u>You should fill in the following table and show the necessary calculations for full credit.</u>

S	*S0*	*S1*	*S2*
$\pi(S)$	*sleeping*	*sleeping*	*sleeping*
$V^{\pi}(S)$			

(1 point) (b) consider the policy π that chooses the action $a = working$ in each state. For this policy, solve the linear system of Bellman equations (by hand) to compute the state-value function $V^{\pi}(s)$ for $s \in \{S0, S1, S2\}$. You should fill in the following table and show the necessary calculations for full credit.

S	$S0$	$S1$	$S2$
$\pi(s)$	$working$	$working$	$working$
$V^{\pi}(s)$			

(1 point) (c) Update the policy $\pi(s)$ with respect to the state-value function $V^{\pi}(s)$ computed in part (a) and part (b). You should fill in the following table and show the necessary calculations for full credit.

S	$S0$	$S1$	$S2$
$\pi(s)$			

A goal without a plan is just a wish.