CS404 Fall 2021- Assignment 3

Dilara Tekinoglu - 27868

December 26, 2021

1 Crazy Frog Puzzle: Task Planning

There are n^2 cells in a nxn grid.We can use $loc - x_i - y_j$ to denote the cell at row x and column y of the grid. The frog will be on any of these cells in each step. We use the following predicates:

- blocked(X): Cell X is already visited or it has an obstacle.
- connected(X,Y): Cell Y is directly jumpable from cell X. (I.e. It is either in the same row or in the same column.)
- at-robot(X): Frog is on cell X.

Only action will be move(X,Y): jump from cell X to eat insect at cell Y.

- Preconditions = [on(X), connected(X,Y), (not(blocked(Y))]
- Add = [on(Y), blocked(Y)]
- Delete = [on(X)]

Start configuration: The cells that have obstacle and the cell that the frog is on will be blocked initially. So, for each such cell X, blocked(X) will be in the start state. For cell C that the frog is on, at-robot(C) will also be in the start configuration. Moreover, for all pair of cells X, Y that are either in the same row or same column connected(X,Y) and connected(Y,X) will be in the start configuration.

Goal configuration: For each cell X, blocked(X) will be in goal configuration. I.e. All cells will be blocked in the end.It is also to okay add pairs of cells that are jumpable from each other to the goal configuration but it is not necessary.

INSTANCE 1:
Input Farm:

0 0 0
0 F 1
0 0 1

Output of FastDownward:

(move loc-x1-y1 loc-x1-y0)
(move loc-x1-y0 loc-x0-y0)
(move loc-x0-y0 loc-x0-y2)
(move loc-x0-y2 loc-x0-y1)
(move loc-x0-y1 loc-x2-y1)
(move loc-x2-y1 loc-x2-y0)
; cost = 6 (unit cost)

INSTANCE 2:
Input Farm:

F 1 0
0 1 0
0 1 1

Output of FastDownward:

(move loc-x0-y0 loc-x0-y2)
(move loc-x0-y2 loc-x1-y2)
(move loc-x1-y2 loc-x1-y0)
(move loc-x1-y0 loc-x2-y0)
; cost = 4 (unit cost)

2 Motion Planning

1. What is motion planning?

Motion planning is planning the movements of a robot to go from start to goal without any collisions and while respecting all robot constraints. It is a hard problem. It has geometric constraints as well as dynamics constraints.

Four motion planning approaches that are mentioned in the talk are as follows: (i)Sampling-based planning, (ii) heuristic discrete search methods, (iii) optimization-based approaches, (iv) control-theoretic approaches.

2. How is motion planning different from task planning?

Task planning requires discrete reasoning. Motion planning requires continuous reasoning.
Motion planning is a subset of Task & Motion planning. Task planning is for computing the path you need to follow for completing task, while motion planning is the process by which you define the set of actions (movements of robot) you need to execute to follow the path you planned.

3. Describe three sample real-world applications of motion planning mentioned in the talks.

- **Cargo Transfer Bag Retrieval at NASA:** Cargo on the International Space Station is stored in Cargo Transfer Bags. A robot could retrieve bags for logistic purposes. Since there is no gravity at ISS hands of the robot always need to be held on a platform. The problem that requires a motion planning solution can be described with the following steps: (i) Approach area safely, (ii) Hold the bag, (iii) transport the cargo.
- **Autonomous underwater vehicles:** Researchers used an online multilayered motion planning approach for planning the movements of an underwater vehicle. There are many constraints in this application. First of all, the vehicle used has nonlinear dynamics. Secondly, underwater domain is an unknown and unpredictable environment. Besides, they had limited computational resources (only has scanning profiling sonar).
- **Applications to Cancer Immunotherapy:** Motion planning is also used for cancer immunotherapy applications as described in the talk. It agent that applies motion planning sticks to some cells and performs necessary actions to heal cancer.

3 PDDL Files for FastDownward

In order to obtain plan for my instances, I used the below command with domain.pddl and instance.pddl files:

./fast-downward.py domain.pddl instance.pddl –search "astar(lmcut())"

The contents of .pddl files are as follows:

domain.pddl

```
(define (domain grid-visit-all)
(:requirements :typing)
(:types place - object)
(:predicates (connected ?x ?y - place)
        (at-robot ?x - place)
        (blocked ?x - place)
)

(:action move
        :parameters (?curpos ?nextpos - place)
        :precondition (and (at-robot ?curpos) (connected ?curpos ?nextpos) (not (blocked ?nextpos)))
        :effect (and (at-robot ?nextpos) (not (at-robot ?curpos)) (blocked ?nextpos))
)
)
```

instance1.pddl

```
(define (problem puzzle)
(:domain grid-visit-all)
(:objects
        loc-x0-y0
        loc-x0-y1
        loc-x0-y2
        loc-x1-y0
        loc-x1-y1
        loc-x1-y2
        loc-x2-y0
        loc-x2-y1
        loc-x2-y2
- place
)

    (:init
        (at-robot loc-x1-y1)
        (blocked loc-x1-y1)
        (blocked loc-x1-y2)
        (blocked loc-x2-y2)
        (connected loc-x0-y0 loc-x1-y0)
        (connected loc-x0-y0 loc-x0-y1)
        (connected loc-x0-y0 loc-x2-y0)
        (connected loc-x0-y0 loc-x0-y2)
        (connected loc-x0-y1 loc-x0-y0)
        (connected loc-x0-y1 loc-x0-y2)
        (connected loc-x0-y1 loc-x1-y1)
        (connected loc-x0-y1 loc-x2-y1)
        (connected loc-x0-y2 loc-x0-y0)
        (connected loc-x0-y2 loc-x0-y1)
```

```
        (connected loc-x0-y2 loc-x1-y2)
        (connected loc-x0-y1 loc-x2-y2)
        (connected loc-x1-y0 loc-x0-y0)
        (connected loc-x1-y0 loc-x2-y0)
        (connected loc-x1-y0 loc-x1-y1)
        (connected loc-x1-y0 loc-x1-y2)
        (connected loc-x1-y1 loc-x0-y1)
        (connected loc-x1-y1 loc-x2-y1)
        (connected loc-x1-y1 loc-x1-y0)
        (connected loc-x1-y1 loc-x1-y2)
        (connected loc-x1-y2 loc-x0-y2)
        (connected loc-x1-y2 loc-x2-y2)
        (connected loc-x1-y2 loc-x1-y1)
        (connected loc-x1-y2 loc-x1-y0)
        (connected loc-x2-y0 loc-x0-y0)
        (connected loc-x2-y0 loc-x1-y0)
        (connected loc-x2-y0 loc-x2-y1)
        (connected loc-x2-y0 loc-x2-y2)
        (connected loc-x2-y1 loc-x0-y1)
        (connected loc-x2-y1 loc-x1-y1)
        (connected loc-x2-y1 loc-x2-y0)
        (connected loc-x2-y1 loc-x2-y2)
        (connected loc-x2-y2 loc-x0-y2)
        (connected loc-x2-y2 loc-x1-y2)
        (connected loc-x2-y2 loc-x2-y1)
        (connected loc-x2-y2 loc-x2-y0)
)
(:goal
(and
        (blocked loc-x0-y0)
        (blocked loc-x0-y1)
        (blocked loc-x0-y2)
        (blocked loc-x1-y0)
        (blocked loc-x1-y1)
        (blocked loc-x1-y2)
        (blocked loc-x2-y0)
        (blocked loc-x2-y1)
        (blocked loc-x2-y2)
)
)
)
```

instance2.pddl

```
(define (problem puzzle)
(:domain grid-visit-all)
(:objects
        loc-x0-y0
        loc-x0-y1
        loc-x0-y2
        loc-x1-y0
        loc-x1-y1
        loc-x1-y2
        loc-x2-y0
        loc-x2-y1
        loc-x2-y2
```

- place
)

(:init
(at-robot loc-x0-y0)
(blocked loc-x1-y1)
(blocked loc-x0-y1)
(blocked loc-x2-y2)
(blocked loc-x2-y1)
(blocked loc-x0-y0)
(connected loc-x0-y0 loc-x1-y0)
(connected loc-x0-y0 loc-x0-y1)
(connected loc-x0-y0 loc-x2-y0)
(connected loc-x0-y0 loc-x0-y2)
(connected loc-x0-y1 loc-x0-y0)
(connected loc-x0-y1 loc-x0-y2)
(connected loc-x0-y1 loc-x1-y1)
(connected loc-x0-y1 loc-x2-y1)
(connected loc-x0-y2 loc-x0-y0)
(connected loc-x0-y2 loc-x0-y1)
(connected loc-x0-y2 loc-x1-y2)
(connected loc-x0-y1 loc-x2-y2)
(connected loc-x1-y0 loc-x0-y0)
(connected loc-x1-y0 loc-x2-y0)
(connected loc-x1-y0 loc-x1-y1)
(connected loc-x1-y0 loc-x1-y2)
(connected loc-x1-y1 loc-x0-y1)
(connected loc-x1-y1 loc-x2-y1)
(connected loc-x1-y1 loc-x1-y0)
(connected loc-x1-y1 loc-x1-y2)
(connected loc-x1-y2 loc-x0-y2)
(connected loc-x1-y2 loc-x2-y2)
(connected loc-x1-y2 loc-x1-y1)
(connected loc-x1-y2 loc-x1-y0)
(connected loc-x2-y0 loc-x0-y0)
(connected loc-x2-y0 loc-x1-y0)
(connected loc-x2-y0 loc-x2-y1)
(connected loc-x2-y0 loc-x2-y2)
(connected loc-x2-y1 loc-x0-y1)
(connected loc-x2-y1 loc-x1-y1)
(connected loc-x2-y1 loc-x2-y0)
(connected loc-x2-y1 loc-x2-y2)
(connected loc-x2-y2 loc-x0-y2)
(connected loc-x2-y2 loc-x1-y2)
(connected loc-x2-y2 loc-x2-y1)
(connected loc-x2-y2 loc-x2-y0)
)
(:goal
(and
(blocked loc-x0-y0)
(blocked loc-x0-y1)
(blocked loc-x0-y2)
(blocked loc-x1-y0)
(blocked loc-x1-y1)
(blocked loc-x1-y2)
(blocked loc-x2-y0)

(blocked loc-x2-y1)
(blocked loc-x2-y2)
)
)
)