Introduction to AI - assignment 3

Oded Yechiel

27/11/18

Exercise 1 Agents and environment simulators

1.1 An agent that plays Whist

Whist is a zero-sum adversarial game. The environment is static and discrete, non observed and non deterministic. There are many possible goal states, and the agent should maximize its own performance while minimizing the performance of the adversarial agent. Therefore, a **utility-based agent** will best assess the situation and provide the ultimate action.

1.2 An agent that can solve Sokoban problems

Sokoban is a single player puzzle-like game with a single goal state. The environment is static, discrete, fully observable and deterministic. Therefore, a **goal-based agent** will be most suitable than all other agents.

1.3 An autonomous humanoid robot that can win the DARPA robotics challenge

The DRC's task require the robot to perform a series of tasks. Although points are distributed per task, such as, go through the gate, enter the car, drive the car and exit the car, these are merely subgoals and are sequential. Therefore, it would be most logical of using a **goal-based agent**. The environment is static, continuous, discrete and partially observable.

1.4 An internet shopping agent specializing in trip planning

An internet shopping agent has many goals, with various optimizing criteria, such as time, cost and comfort. The environment **may be** considered as a deterministic, fully observable (after query the agent has all of the information) single operating agent (assuming tickets are not running out). Since there is no one perfect goal, the agent should be **utility-based agent.**

1.5 An agent that can play Go

Go is a fully observable, non-deterministic, static adversarial game. Although there is only one winner and a definite goal state, it is unlikely for any agent to calculate so deeply into the game and find the optimal action. Therefore, for a given action it is required to assess the situation and decide what action is sub-optimal for the problem. Therefore, a **utility-based agent** is advised.

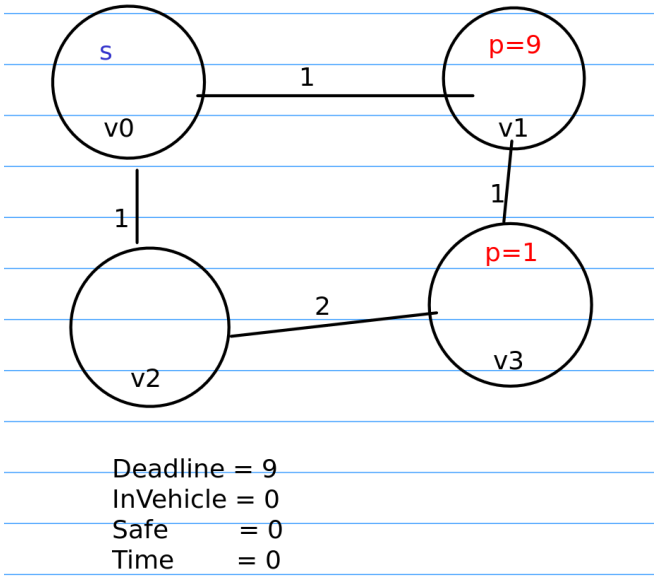


Figure 1: The hurricane environment.

Exercise 2 Search

In the hurricane environment, shown in Fig. 1, there are many different ways in which it is possible to calculate the heuristics of an action. The half-star heuristic, h' calculates the performance by locating the vertex with most un-savable people and multiplying them by a large number. In this exercise we shall multiply each person by 100 for ease of calculation and distinction, although multiplying by the "Deadline value" is sufficient.

We can examine an example of calculating the heuristics for the starting location. The agent is at V_0 and it is possible to go to V_1 or V_2. The heuristic for going to V_1 will be 100, since we have picked up all the 9 people in V_1 and the only town with people is V_3 with 1 person. Going to V_2 will result in a heuristic of 900, since in V_1 there are 9 people which are un-savable.

2.1 Greedy agent

From the example above, the greedy agent will obviously prefer going to V_1, even-though it is obvious that the pick-up will be his final move. Fig. 2 shows the decision making of the greedy agent: $V_0 \rightarrow V_1 \rightarrow V_3$.

The greedy agent relies on an extremely optimistic heuristic that assumes that whoever in the vehicle will get to safety, and does not take those people (the ones in the vehicle) into account in the calculation. Therefore, the result of the greedy agent is not optimal.

2.2 A^*

Opposed to the greedy agent, the A^* agent will provide an optimal solution since the heuristic is admissible. Since, the heuristic is extremely optimistic the A^* agent will perform much more expansions than he would have needed if the heuristic was better.

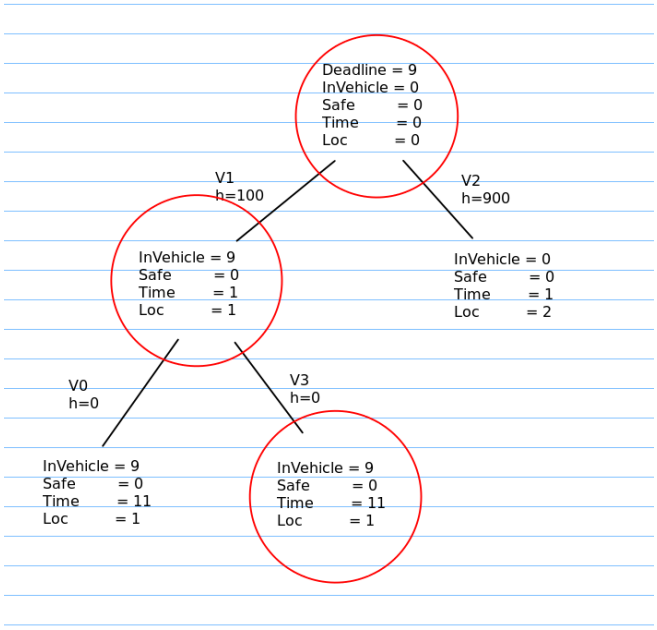


Figure 2: Greedy agent expansions

The A^* agent takes into account the time took to get to this state, and if a goal state is reached it adds the number of unsaved people.

$$g_{v_i \to v_j} = \begin{cases} g[v_i] + w_{v_i v_j}, & t + w_{v_i v_j} < T_{deadline} \\ g[v_i] + w_{v_i v_j} + 100 \times UnsavedPeople, & else \end{cases} \tag{1}$$

The agent adds the g value with the h value to decide which node to expand.

Fig. 3 shows the expansion tree of the A^* agent. It can be seen that the agent has 11[1] expansions to find the optimal path, which is $V_0 \to V_2 \to V_3 \to V_2 \to V_0$.

2.3 Real-time A^*

This agent is similar to the A^* agent, however, it stops after 2 expansions and makes a decision. Fig 4a shows the first two expansions of the agent. It can be seen that in this case, the same result state is achieved as with the the A^* agent. After the agent reached V_3 the agent expanded up to two more nodes, as shown in Fig. 4b.

2.4 $h = 2 * h'$

Since h' multiplied the number of people by a large number, it does not matter if this large number is X or $2 * X$. Therefore, all of the results will remain the same, and the heuristic is still admissible.

[1]The state V_0 with f=902 was not expanded since it is assumed there is a checking for loops to avoid expanding already visited states.

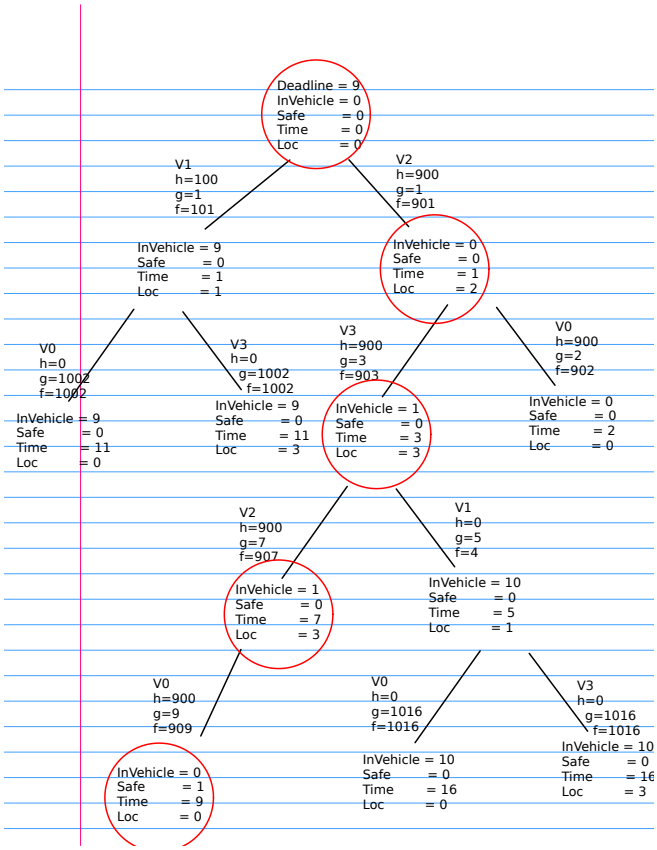
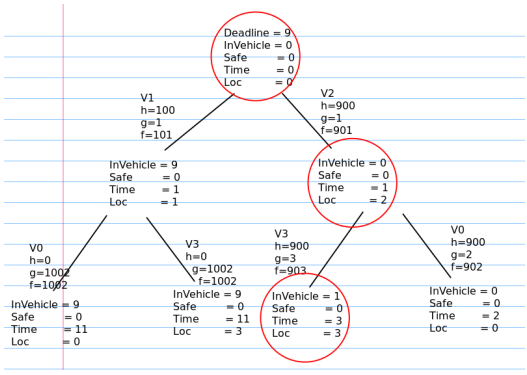
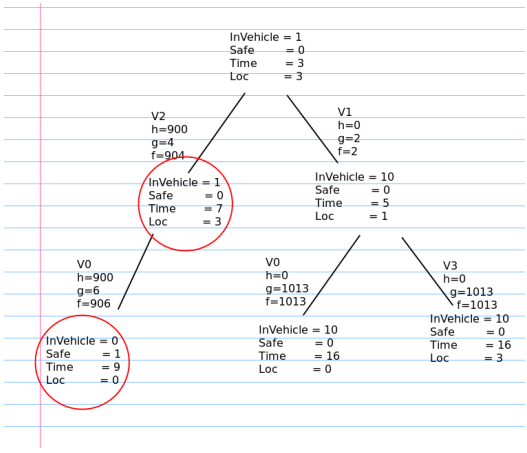


Figure 3: A^* agent expansions



(a) Real-Time A^* agent first 2 expansions.



(b) Real-Time A^* agent second 2 expansions.

Exercise 3 Game trees

Exercise 4 Game-tree search - alpha-beta pruning

Exercise 5 Propositional logic

For validity it will be shown that the sentence is always true.
An example for satisfiability a true model and a false model will be given.
For unsatisfiability it will be shown that the setence is always false.
Number of models equals to: 2^n, where n is the number of symbols.

5.1 $(\neg A \wedge \neg B \wedge \neg C \wedge \neg D \wedge \neg E \wedge F) \vee (A \wedge B \wedge C \wedge D \wedge E)$

True model - A, B, C, D, F are true, therefore,
$True \vee False = True$
False model - A is true, B is false (the rest whatever), therefore,
$False \vee False = False$
Satisfiable.

5.2 $(\neg A \vee \neg B \vee \neg C \vee \neg D \vee \neg E \vee \neg F) \wedge (A \vee B \vee C \vee D \vee E \vee F)$

False model - A, B, C, D, F are true, therefore,
$True \wedge False = False$
True model - A is true, B is false (the rest whatever), therefore,
$True \wedge True = True$
Satisfiable. Number of models is 64.

5.3 $(A \vee B \wedge (D \vee \neg A) \wedge (E \vee A) \Rightarrow (B \vee C \wedge (\neg D \vee E))) \wedge (A \wedge \neg A)$

$\alpha \equiv (A \vee B \wedge (D \vee \neg A) \wedge (E \vee A) \Rightarrow (B \vee C \wedge (\neg D \vee E)))$
$\alpha \wedge (A \wedge \neg A) \rightarrow \alpha \wedge False \rightarrow False$
Unsatisfiable Number of models is 64.

5.4 $(A \wedge (A \Rightarrow B) \wedge (B \Rightarrow C)) \Rightarrow C$

implication elimination $\neg(A \wedge (\neg A \vee B) \wedge (\neg B \vee C)) \vee C$

Distributivity $\neg((A \wedge \neg A) \vee (A \wedge B)) \wedge (\neg B \vee C)) \vee C$

$\neg(False \vee (A \wedge B)) \wedge (\neg B \vee C)) \vee C$

Associativity $\neg(A \wedge (B \wedge (\neg B \vee C))) \vee C$

Distributivity $\neg(A \wedge ((B \wedge \neg B) \vee (B \wedge C))) \vee C$

$\neg(A \wedge ((False) \vee (B \wedge C))) \vee C$

$\neg(A \wedge (B \wedge C)) \vee C$

De Morgan $\neg A \vee \neg B \vee \neg C \vee C \rightarrow True$

Valid. Number of models is 8.

5.5 $\neg((A \wedge (A \Rightarrow B) \wedge (B \Rightarrow C)) \Rightarrow C)$

implication elimination $\neg(\neg(A \wedge (\neg A \vee B) \wedge (\neg B \wedge C)) \vee C)$

Distributivity $\neg(\neg((A \wedge \neg A) \vee (A \wedge B) \wedge (\neg B \wedge C)) \vee C)$

$\neg(\neg((A \wedge B) \wedge (\neg B \wedge C)) \vee C)$

De Morgan $\neg(\neg A \vee \neg B \vee \neg\neg B \vee \neg C \vee C)$

Double negation elimination $\neg(\neg A \vee \neg B \vee B \vee \neg C \vee C) \rightarrow$

$\neg(\neg A \vee True \vee True) \rightarrow$

$\neg(True) \rightarrow False$

Unsatisfiable. Number of models is 8.

5.6 $(A \Rightarrow \neg A) \vee (\neg A \Rightarrow A)$

implication elimination $(\neg A \vee \neg A) \vee (A \vee A) \rightarrow$

$\neg A \vee A \rightarrow True$

Valid. Number of models is 2.

Exercise 6 FOL and situation calculus

6.1 Axioms of world behavior

6.1.1 Auxiliary predicats

- TimeUp(s) - check if time passed the deadline

  $\forall s, t \quad TimeUp(s) \Leftrightarrow Time(t, s) > Deadline(t)$

6.1.2 Effect axioms

- Moving to edge - update time fluent

  For every action, the new time to the current time the time for performing the action

  $\forall a, s, t \quad Time(t, Result(a, s)) \Leftrightarrow [(a = NoOp \wedge Time(t{+}1, s)) \vee (\exists e \quad a = Traverse(e) \wedge Time(t{+}Weight(e, w), s))]$

- Pickup people

  $\forall v, a, s \quad Loc(v, Result(a, s)) \Rightarrow Carrying(n + At(v, n, s), Result(a, s))$

- Remove picked up people

  For all states and vertices, 0 people are at vertex, v, if loc is v.

  $\forall v, a, s \quad Loc(v, Result(a, s)) \Rightarrow At(v, 0, Result(a, s))$

- Save people

  People in vehicle are safe if vehicle is at a shelter and time is not up

  $\forall s, a, v \quad Loc(v, s) \wedge Shelter(v) \wedge \neg TimeUp(s) \Rightarrow Safe(n + Carrying(n, s), Result(a, s))$

- Remove carried people

  $\forall s, a, v \quad Loc(v, s) \wedge Shelter(v) \wedge \neg TimeUp(s) \Rightarrow Carrying(0, Result(a, s))$

6.1.3 Successor state axioms

- Number of people in towns never change

  $\forall a, v, s \quad At(v, n, Result(a, s)) \Leftrightarrow At(v, n, s) \wedge \neg Loc(v, s)$

- Number of safe people never change

  $\forall a, s \quad Safe(n, Result(a, s)) \Leftrightarrow Safe(n, s) + (\exists v \quad (Loc(v, s) \wedge Shelter(v)) \times Carrying(n, s))$

- Number of people carrying never change

  $\forall s \quad Carrying(n, s + 1) \Leftrightarrow Carrying(n, s) \wedge (\exists v \quad (Loc(v, s) \wedge \neg Shelter(v))$