Solutions to Homework 1

Autumn 2012

1 Driving in Manhattan

a

State: suppose $s \in \text{States}$, then $s = (\text{street}, \text{avenue})$, where $1 \leq \text{street} \leq n, 1 \leq \text{avenue} \leq m$.
Action: suppose $a \in \text{Actions}(s)$, then a can be either of the following:

- move_North if street $< m$
- move_South if street > 1
- move_East if avenue > 1
- move_West if avenue $< n$

Successor and **Cost**: suppose $s = (\text{street}, \text{avenue})$, a is an action defined as above, the successor and cost are defined as:

- if a is move_North, $\text{Succ}(s, a) = (\text{street} + 1, \text{avenue})$ and $\text{Cost}(s, a) = 1$
- if a is move_South, $\text{Succ}(s, a) = (\text{street} - 1, \text{avenue})$ and $\text{Cost}(s, a) = 1$
- if a is move_East, $\text{Succ}(s, a) = (\text{street}, \text{avenue} - 1)$ and $\text{Cost}(s, a) = 3$
- if a is move_West, $\text{Succ}(s, a) = (\text{street}, \text{avenue} + 1)$ and $\text{Cost}(s, a) = 3$

Start state: $s_{\text{start}} = (14, 8)$.
Goal test: IsGoal((street, avenue)) is true if and only if street > 96.

b

State: augment each state in (a) by adding one check_bit to indicate whether I have stopped at a grocery store or not. That is, $\forall s \in \text{States}$, $s = (\text{street}, \text{avenue}, \text{check\_bit})$.
Action: action is defined the same as in (a).
Successor and **Cost**: suppose $s = (\text{street}, \text{avenue}, \text{check\_bit})$, $a \in \text{Actions}(s)$, then street/avenue are update by a accordingly as in (a). In addition, we need to update the check_bit. If check_bit is 0 and (new_street, new_avenue) is any of $p_1, \cdots, p_k$, then $\text{Succ}(s, a) = (\text{new\_street}, \text{new\_avenue}, 1)$. Otherwise, $\text{Succ}(s, a) = (\text{new\_street}, \text{new\_avenue}, \text{check\_bit})$. So the check_bit stays 1 once it's 1. The cost of each action is defined the same as in (a).
Start State: $s_{\text{start}} = (14, 8, 0)$.
Goal Test: IsGoal((street, avenue)) is true if and only if street > 96 and check_bit == 1.

c

State: $\forall s \in \text{States}$, $s = (\text{street}, \text{avenue}, \text{check\_bit}_1, \text{check\_bit}_2, \cdots, \text{check\_bit}_k)$. Here we add one check bit per store and set it to 1 and stays 1 when a store is visited.
The number of states is $m \cdot n \cdot (2^k)$.

d

State: $\forall s \in \text{States}$, $s = (\text{street}, \text{avenue}, \text{counter})$. Here the counter is incremented from $i - 1$ to i only when visiting the i-th store. So, when the counter equals to k, I am guaranteed to have visited all stores in the correct pre-defined sequence.
The number of states is $m \cdot n \cdot (k + 1)$, since counter starts from 0.

e

It depends. For example, comparing (c) and (d), the state space gets smaller after we add constraints. On the other hand, comparing (a) and (b), the state space gets larger after we add constraints. Intuitively, either having no constraints or a lot of constraints makes the problem easy in some sense; hard problems are somewhere in between.

2 Package Delivery

a The model

A state can informally be described in terms of the locations of all the delivery trucks, and the locations of all the packages. Since a truck can pick up and drop off packages at will, the ownership of a package is not crucial to model in the state definition.

Formally, a state is a tuple: the first element of which is a tuple of size k taking values $1 \dots |V|$ denoting the locations of the trucks $1 \dots k$. the second element is a tuple of size $|R|$ taking values $1 \dots |V|$ denoting the location of the $|R|$ packages. The state space thus has $|V|^{k+|R|}$ states, corresponding to any combination of locations of the trucks and the packages.

An action is easiest visualized in two parts:

1. An assignment of movements to trucks. A truck at city i may move to any city j such that $(i, j) \in E$, or $i = j$.

2. An assignment of packages to neighboring cities. A package may only move along an edge that a truck has taken.

The cost of an action is a constant cost; e.g., 1.

The initial state assigns each truck to the start city s, and each package to its corresponding order's start city i.

The goal test checks if each package is in its end city j. This amounts to an ordered equality check on the second element of the state space with the tuple denoting each package being in its end state.

b Search algorithm

Since the edge costs are 1, we can use breadth-first search. UCS will also work, but it is slower.

c Average delivery time

Minimizing the average delivery time requires modifying the edge costs in the search graph. Rather than having a cost of 1 for every time step, the cost of an action is equal to the number of packages still awaiting delivery. This will cause the path cost to a state to be the sum of the package transit times, which is proportional to the average transit time. A goal state's path cost is therefore proportional to the average delivery time.

Note that an undelivered package which is not moving still contributes to this sum cost – a package waiting at a city is still contributiong to the sum delivery time. Also note that the number of trucks which move is not necessarily correlated with the number of packages awaiting delivery. A single truck can carry many packages, or many trucks can move without carrying packages.

The actions taken are identical to those described in part (a), except that any action which would move a package out of its destination is prohibited (since that would lead to a suboptimal solution anyway).

d Average delivery time algorithm

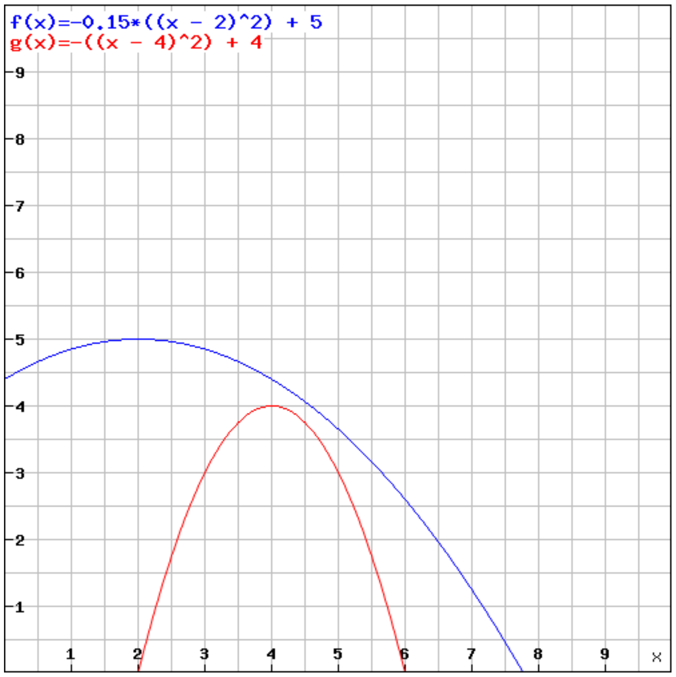
Uniform cost search returns the minimum cost path.

3 Agents

a Utility Functions

Counterexample 1: Imagine just two actions, a_1 and a_2. Let $U(a_1) = 1$ and $U(a_2) = 0$. Let $U'(a_1) = 2$ and $U'(a_2) = 3$. If the agent maximizes with respect to U', the optimal action is a_2. However, a_1 is the better action to optimize U.

Counterexample 2: Below is an example of two utility curves where one is strictly greater than the other, but their maxima are in different locations (here the actions are real numbers). The x-axis represents actions and y-axis represents utilities for those actions. In the question, the blue curve is U' and the red curve is U.



Maximizing with respect to the blue curve will give us an action of 2. If the agent were to use 2, however, it would get 0 utility according to the red curve, which is not optimal.

b Rational Agents (Part I)

Any action will work. If the agent's action is choosing heads, we have $ExpectedUtility = P(heads) * 1 + P(tails) * 0 = 0.5$.

In fact, call Z the probability that the agent picks heads. So if the agent always picks heads, $Z = 1$, and if the agent always picks tails, $Z = 0$. Then we have $ExpectedUtility = P(heads) * Z + P(tails) * (1 - Z) = 0.5 * Z + 0.5 * (1 - Z) = 1$. So any strategy of picking actions will give us the same expected utility of 1.

c Rational Agents (Part II)

- In the case when $p > 0.5$, q must be greater than 0.5 as well. If $q > 0.5$, then a rational agent always chooses heads, which is the best action for any $p > 0.5$.

- In the case when $p < 0.5$, q must be less than 0.5 as well. If $q < 0.5$, then a rational agent always chooses tails, which is the best action for any $p < 0.5$.

- If $p = 0.5$, then this problem is the same as 3.b.

d Reflex Agents

The answer is A.

If the agent began at B, it will go West along the first corridor. It will reach a wall, turn right, turn right again, and then go back to its starting location. Then it will turn right, turn right again, turn right again, and repeat all the previous steps.

C will quickly get caught in the same infinite loop as B.