PS1: Practice problems

AnswerMinor revisions in solutions 10/12/10 8:00.

Answer Answers are shown in green. These answers have not been proofread by anyone but me, so there is substantial chance of error. Please let me know if you have any doubts. -lpk

1 Discrete optimization

1.1 Constraint satisfaction

From AIMA3E. Give precise formulations for each of the following as constraint satisfaction problems:

1. Rectilinear floor-planning: find non-overlapping places in a large rectangle for a number of smaller rectangles.

   Answer To make this into a traditional constraint satisfaction problem, we will have to discretize the possible locations for the tiles. It's clear that as we discretize more finely, we enable more possible solutions, but increase complexity.

   The most sensible solution is probably to have a variable for each small rectangle, which can take on, as values, possible locations of, say, its lower left corner within the big rectangle. It makes it relatively easy to check constraints, to see if any rectangles are overlapping.

   An alternative formulation is to have a variable for each location in the big rectangle, containing the value *None* or a value indicating one of the small rectangles overlaps that location. In this case, we automatically satisfy the non-overlapping constraint, but we'd have to work hard to be sure we had a coherent assignment (that is, the right number of contiguous locations were all assigned to the same small rectangle). One way to do it would be to assign a whole batch of them at once.

2. Class scheduling: There is a fixed number of professors and classrooms, a list of classes to be offered, and a list of possible time slots for classes. Each professor has a set of classes he or she can teach.

   Answer There are many choices here. Let's say we have K classes, L profs, M possible times and N possible rooms.

   Formulation 1: Have three different variables for each class: which professor, which time, and which room. So, we'd have K variables with domain size L, K with domain size M, and K with domain size N. Constraints would have to be that profs can't be in two classes at the same time; that you can use the same room for two classes at the same time; that only appropriate professors are assigned to classes.

Formulation 2: Have two sets of variables, each of which have as their domain the K possible classes: L $\times$ M variables representing professor time-slots and M $\times$ N variables representing room time slots. Constraints would have to be every class has to be assigned to exactly one room; that every class has to be assigned to exactly one prof; that only appropriate professors are assigned to classes.

One rule of thumb is that it's better to pick formulations in which variables have smaller domains, because the constraints can do more work for you in ruling out choices. Another rule of thumb is that its better to pick formulations that already have some of your constraints built in.

1.2 Optional zebra

Discuss different representations of this problem as a CSP. It's originally from *Life International* magazine, 1962. (One minor error in the original is fixed.)

- There are five houses.
- The Englishman lives in the red house.
- The Spaniard owns the dog.
- Coffee is drunk in the green house.
- The Ukrainian drinks tea.
- The green house is immediately to the right of the ivory house.
- The Old Gold smoker owns snails.
- Kools are smoked in the yellow house.
- Milk is drunk in the middle house.
- The Norwegian lives in the first house.
- The man who smokes Chesterfields lives in the house next to the man with the fox.
- Kools are smoked in a house next to the house where the horse is kept.
- The Lucky Strike smoker drinks orange juice.
- The Japanese smokes Parliaments.
- The Norwegian lives next to the blue house.

Now, who drinks water? Who owns the zebra? In the interest of clarity, it must be added that each of the five houses is painted a different color, and their inhabitants are of different national extractions, own different pets, drink different beverages and smoke different brands of American cigarettes. One other thing: in statement 6, right means your right.

Answer I don't have a lot to say here. The sensible thing is to have, for each house, a set of 5 variables, each representing the nationality, pet, drink, smoke, and color associated with the

house. Each of these 25 variables can take on values from the appropriate domain (the domains are size 5).

There are *alldiff* constraints on each of the sets of variables. That is, we know that all five *pet* variables will each have a different value of pet assigned to them. Then, there are constraints arising from the individual assertions in the problem.

2 Continuous optimization

2.1 One dimension

Consider two functions:

$$f(x) = (x-1)^2$$
$$g(x) = (x-1)^4$$

For each one, assuming that we're starting with an initial value $x_0 = 0$, and search for the minimum.

1. What is the gradient of the function at $x = 0$?

Answer

$$\nabla f(x) = 2(x-1)$$
$$\nabla f(0) = -2$$
$$\nabla g(x) = 4(x-1)^3$$
$$\nabla g(0) = -4$$

2. What is a step-size that will make gradient descent converge?

Answer For f, gradient descent will converge with step size .1. The first few x values, starting at 0, are: $0, 0.2, 0.36, \ldots$.

For g, gradient descent will converge with step size .1. The first few x values, starting at 0, are: $0, 0.4, 0.4864, \ldots$.

3. What is a step-size that will make gradient descent diverge?

Answer For f, gradient descent will diverge with step size 2. The first few x values, starting at 0, are: $0, 4, -8, \ldots$.

For g, gradient descent will converge with step size 1. The first few x values, starting at 0, are: $0, 4, -104, \ldots$.

4. What are the first two steps that would be taken by Newton's method?

Answer For f, the Newton update is

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)} = x_n - \frac{2(x_n - 1)}{2} = 1$$

so we go immediately to $x = 1$ and terminate.

For g, the Newton update is

$$x_{n+1} = x_n - \frac{g'(x_n)}{g''(x_n)} = x_n - \frac{4(x_n - 1)^3}{12(x_n - 1)^2} = x_n - \frac{1}{3}(x_n - 1)$$

so the first few x values, starting from 0 are: $0, 1/3, 5/9, 19/27, \ldots$. This function is very flat near the minimum, so it's taking a while to get there.

2.2 Two dimensions

What if you wanted to minimize $f(x, y) = (x + 4)^2 + (y + 4)^2 + xy$?

1. What is the gradient at $(10, 7)$?

Answer

$$\nabla f(x, y) = (2(x + 4) + (y + 4)^2 + y, (x + 4)^2 + 2(y + 4) + x)$$
$$\nabla f(10, 7) = (35, 32)$$

2. Roughly what would conjugate gradient do on a problem like this? (Draw a picture or explain in words).

Answer Conjugate gradient would first move in the gradient direction, doing a line search to find a near-optimal step length. Then it would move in a conjugate (kind of like orthogonal, but respecting the local shape of the function) direction, again doing a line search to select the step length. Because this f is quadratic, it will terminate at the optimum after two steps.

3. What if you wanted to optimize subject to the constraint that $y > 0$? Explain at least one strategy for handling it. Consider implementing it to see what happens.

Answer The easiest way to handle a constraint is to add a penalty term, so that the objective becomes

$$o(x, y) = (x + 4)^2 + (y + 4)^2 + xy + \mu([y]^-)^2$$

where $[y]^- = \max(-y, 0)$. This works, but you have to make μ surprisingly high to be sure the constraint isn't violated.

3 Path search

From AIMA3E. Generally, the problems in chapter 3 are good practice.

3.1 Two friends

Suppose two friends live in different cities on a map. On every turn, we can simultaneously move each friend to a neighboring city on the map. The amount of time needed to move from city i to neighbor j is equal to the road distance $d(i, j)$ between the cities, but on each turn the friend that arrives first must wait until the other one arrives, before the next turn can begin. We want the two friends to meet as quickly as possible.

1. Write a detailed formulation of this search problem. Describe the state space, the action space, the transition model, the goal test, the path cost function. Assume you know the set of cities, their connectivity, and the distance function.

Answer Assume that we have *successor*, *actions*, and *arcCost* defined on the original graph. Note that the problem above should have asked for **arc cost** not **path cost**. Now, in the new problem, we have:

- States: $\langle l_1, l_2 \rangle$ where l_1 and l_2 are cities in the graph, and the pair represents the positions of the two friends.
- Actions: The action function in this problem is defined as

  $$\text{actions}(\langle l_1, l_2 \rangle) = \{\langle a_1, a_2 \rangle\}$$

  for all $a_1 \in \text{actions}(l_1)$ and $a_2 \in \text{actions}(l_2)$
- Successor:

  $$\text{successor}(\langle l_1, l_2 \rangle, \langle a_1, a_2 \rangle) = \langle \text{successor}(l_1, a_1), \text{successor}(l_2, a_2) \rangle$$

- Arc cost:

  $$\text{arcCost}(\langle l_1, l_2 \rangle, \langle a_1, a_2 \rangle) = max(\text{arcCost}(l_1, a_1), \text{arcCost}(l_2, a_2))$$

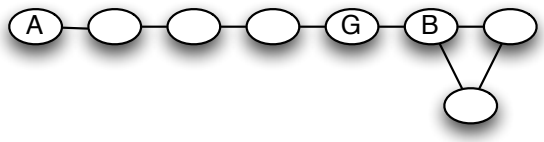
- Goal test:

  $$\text{goalTest}(\langle l_1, l_2 \rangle) = (l_1 == l_2)$$

2. Specify a good admissible heuristic. **Answer** Euclidean distance between the two friends is an admissible heuristic.

3. Are there completely connected maps for which no solution exists? **Answer** If the friends are not allowed to stay at the same city, then yes. Consider a map with just two cities, with the friends starting in different cities. They are cursed to move back and forth, never meeting...

4. Are there maps in which all solutions require one friend to visit the same city twice? **Answer** If the friends are not allowed to stay at the same city, then yes. If the friends start at locations A and B, they can meet at G, and that requires B to go once around the loop and back to his original location.



3.2 Grid

Consider an unbounded regular 2D grid. The start state is at the origin $(0, 0)$ and the goal state is at (x, y).

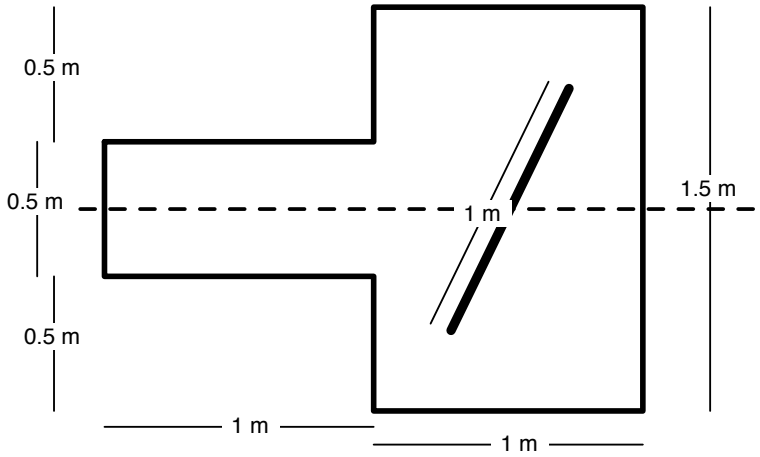
1. What is the branching factor b in this state space? **Answer** 4
2. How many distinct states are there at depth k (for $k > 0$)? **Answer** $2k(k+1)+1$
3. What is the maximum number of nodes expanded by breadth-first tree search? (without dynamic programming) **Answer** It's b^d (branching factor to the depth of the solution), which in this case is 4^{x+y}.
4. What is the maximum number of nodes expanded by breadth-first graph search? (with dynamic programming) **Answer** $2(x+y)(x+y+1)+1$ because that's the number of possible states at that level, and with dynamic programming, we won't visit any one of them more than once.
5. Is $h = |u - x| + |v - y|$ an admissible heuristic for state (u, v)? Explain. **Answer** Yes. The robot can only move in the grid directions.
6. How many nodes are expanded by A* graph search using h? **Answer** $x + y$, because the heuristic is perfect.
7. Is h admissible if some links are removed from the grid? **Answer** Yes; removing links will increase actual path lengths, and the heuristic will be an underestimate.
8. Is h admissible if some links to non-adjacent states are added to the grid? **Answer** No; adding new links will decrease some actual path lengths and the heuristic will no longer be an underestimate.

4 Geometric spaces

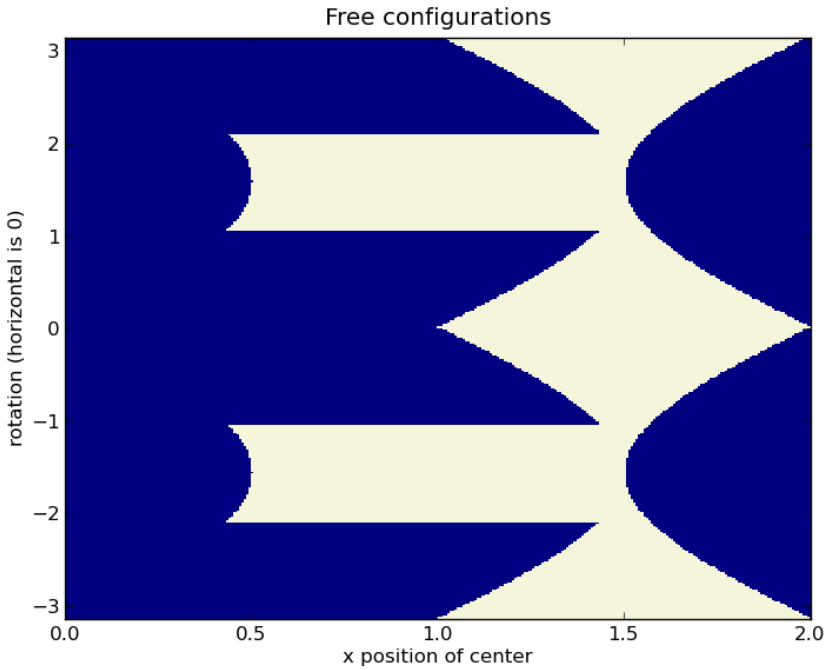
4.1 Free configurations

Consider a robot that is a 1 meter rod. It has two degrees of freedom: it can translate in x (along the dotted line) and it can rotate freely about its center point. But its cen-

ter point is always constrained to be at $y = 0$. It is in the environment shown below.



Sketch the space of free configurations of the robot.



Answer

4.2 Path planning in configuration space

1. Consider a robot trying to move from configuration a to configuration b when there are no obstacles on the straight line in configuration space between those two points. What path

would be constructed by a bidirectional RRT planner? **Answer** It would pick a random point and connect it to a; then the search from b could connect directly to that point as well.

2. M and M is a mobile robot with two arms. It has 18 degrees of freedom. Would it work to uniformly discretize the space and apply A* for planning its trajectories?

Answer Even dividing each dimension into 4 cells gives us 4^{18} states; just allowing the robot to move a single joint at a time, we'd have a branching factor of 36. Unless the heuristic was miraculously good, we would have to visit a huge number of states.

3. Why do we think an RRT might be better?

Answer

- RRTs try to find any path, not an optimal one (making it more like best-first or hill-climbing search than A*).
- The RRT takes steps whose lengths are tuned to the open space available.
- The bi-directional RRT has two advantages: it attempts to connect between trees, which is a kind of flexible heuristic that drives the search toward the goal; also, being bi-directional, the hope is that you get two trees of size $b^{d/2}$ rather than one of size b^d, which is a big win, if it happens.