COMP3702/7702 Artificial Intelligence
Semester 2, 2020
Tutorial 5 - Sample Solutions

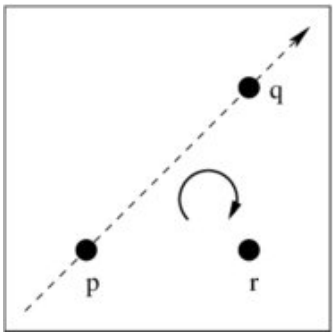
Exercise 5.1

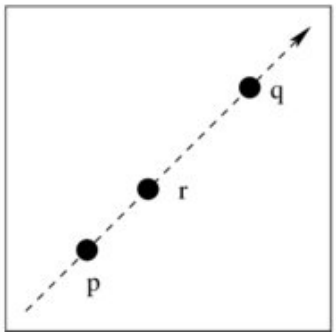
a) To check for collision between two line segments, ab and cd:

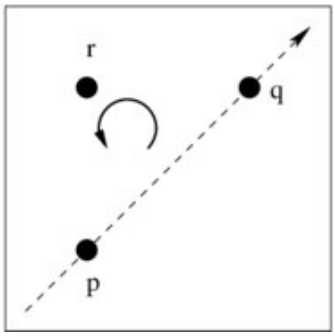
(i) First do a counterclockwise test, to check if the orientations are the same or not for the pairs of sequences of 3 points, abc and abd, and cda and cbd, by taking the cross product of the 3 points

(ii) Check if the orientations are different for both pairs of sequences of 3 points

For any three points p, q, r in a plane, their orientations can visually checked using the counterclockwise test, where the first has orientation down (-ve), and the third, with a counterclockwise circuit, has orientation up (+ve):







Computationally, the formula for checking orientation using cross-product for 3 2D points (a_x, a_y), (b_x, b_y), $(c_x, c_y) = (b_x - a_x) * (c_y - a_y) - (c_x - a_x) * (b_y - a_y)$

For abc and abd we have:

$Area(abc) = (3-0) * (4-2) - (1-0) * (5-2) = 3$
$Area(abd) = (3-0) * (3-2) - (4-0) * (5-2) = -9$

$Area(abc) > 0$, so orientation is counterclockwise
$Area(abd) < 0$, so orientation is clockwise

For cda and cbd we have:

$Area(cda) = (4-1) * (2-4) - (0-1) * (3-4) = -7$
$Area(cdb) = (4-1) * (5-4) - (3-1) * (3-4) = 5$
$Area(cda) < 0$, so orientation is clockwise
$Area(cdb) > 0$, so orientation is counterclockwise

Since abc and abd have different orientations AND cda and cbd have different orientations, there is a line collision between ab and cd.

$$\begin{vmatrix} 0 & 2 & 1 \\ 3 & 5 & 1 \\ 1 & 4 & 1 \end{vmatrix} = 3 \quad \begin{vmatrix} 0 & 2 & 1 \\ 3 & 5 & 1 \\ 4 & 3 & 1 \end{vmatrix} = -9$$

$$\begin{vmatrix} 1 & 4 & 1 \\ 4 & 3 & 1 \\ 0 & 2 & 1 \end{vmatrix} = -7 \quad \begin{vmatrix} 1 & 4 & 1 \\ 4 & 3 & 1 \\ 3 & 5 & 1 \end{vmatrix} = 5$$

We can repeat the same process for lines ab and ef:

$Area(abe) = -3 \quad < 0$ so clockwise
$Area(abf) = -15 \quad < 0$ so clockwise
$Area(efa) = 1 \quad > 0$ so counterclockwise
$Area(efc) = 13 \quad > 0$ so counterclockwise

Since abe and abf have same orientation AND efa and efc have same orientation. There is no line collision.

$$\begin{vmatrix} 0 & 2 & 1 \\ 3 & 5 & 1 \\ 1 & 2 & 1 \end{vmatrix} = -3 \quad \begin{vmatrix} 0 & 2 & 1 \\ 3 & 5 & 1 \\ 6 & 3 & 1 \end{vmatrix} = -15$$

$$\begin{vmatrix} 1 & 2 & 1 \\ 6 & 3 & 1 \\ 0 & 2 & 1 \end{vmatrix} = 1 \quad \begin{vmatrix} 1 & 2 & 1 \\ 6 & 3 & 1 \\ 3 & 5 & 1 \end{vmatrix} = 13$$

Bound box - collision check:
Two bounding boxes for 3 points (a_x, a_y), (b_x, b_y), (c_x, c_y) are in collision if:
$a_x < c_x < b_x$ AND $a_y < c_y < b_y$

For point a, b and c we have:
$0 < 1 < 3$ AND $2 < 4 < 5$
So bounding boxes collide.

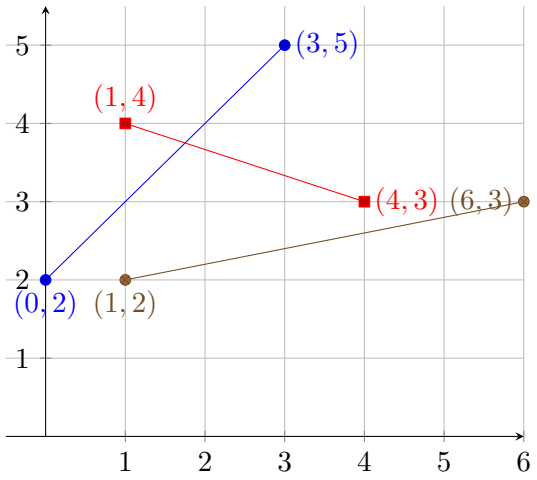
For point a, b and e we have:
$0 < 1 < 3$ AND $2 < 2 < 5$

b) If we construct line segments for each pair of points we will have a total of 15 possible line segments and $15 \times 6 = 90$ possible combinations.

Using bounding boxes will find a large portion of non-collisions before more calculations need to be made. e.g results for combinations AB, CD, EF:

Line segment A	Line segment B	Bounding box collision	Line collision
AB	CD	YES	YES
AB	EF	YES	NO
CD	EF	NO	-

This can be seen in the following diagram:



Exercise 5.2

a) Follow the reasoning from last week's tutorial. Nb. the treat the point of reference on the robot as the lower left hand corner.

For b), c) and d), use rejection sampling. This involves sampling from the whole c-space, and rejecting those samples that do not meet certain criteria. Use the python package `random`, and the method `random.uniform(a, b)` to generate random samples in each dimension.

For b), a simple criterion for testing if point is inside or outside a polygon (polyhedron) is to check how many times a ray starting from the point intersects the edges of the polygon (faces of the polyhedron). If the point is on the outside of the polygon the ray will intersect its edges an even number of times; if it on the inside, it will intersect an odd number of times.

For c), you could uniformly at random generate a point q inside the obstacles, then loop a uniform random sample of the whole c-space (using your answer from a)) until a point within d of q. Alternatively, you could batch the process and generate many points, some withing the obstacles and some outside, and in a second pass, check the distance between the each of those outside the obstacles and all of those inside. For d), you could build a polygon from the nearest edges between the obstacles, then only accept samples from within this area.

Exercise 5.3

Intuitively, problem 1 will be easier to solve than problem 2 using PRM with uniform random sampling. This is because the amount of free space in the C-space for problem 1 is higher than for problem 2. Therefore, for problem 1, a randomly sampled configuration would lie in free space with a higher probability than for problem 2, and thus it would be faster and easier to construct a state graph over the configuration space that allowed an agent to travel from any given initial state to any given goal state.

A more formal answer involves considering the so-called α and β values of the two configuration spaces, and the β-lookout.

In this framework, the β-lookout is measured over the β value, and defines the set of configurations in one region (the observer region) of the C-space that can 'see' β fraction of the free space in another region (the observed region) of the C-space. Here, 'see' means to be able to connect a configuration in the observer region to a configuration in the observed region with a straight line segment, i.e. an edge in a PRM state graph.

The α value in this framework is defined as the fraction of the observer region from which it is possible to observe β fraction of the free space in the observed region. Hence α is dependent on β, meaning α is a function of β. The α value is essentially a measure of the size of the β-lookout set; a large α value indicates the β-lookout set is also large.

A problem is easier to solve with a technique such as PRM when its C-space has large α and β values, as this would mean that a large fraction of free space in one region of the C-space can see a large fraction of free space in another region of the C-space. Thus it is easy to create edges between these two regions when constructing a state graph.

In the case of our two problems, the diagrams that we use to represent the β-lookout of both C-spaces could look like Figure 1.

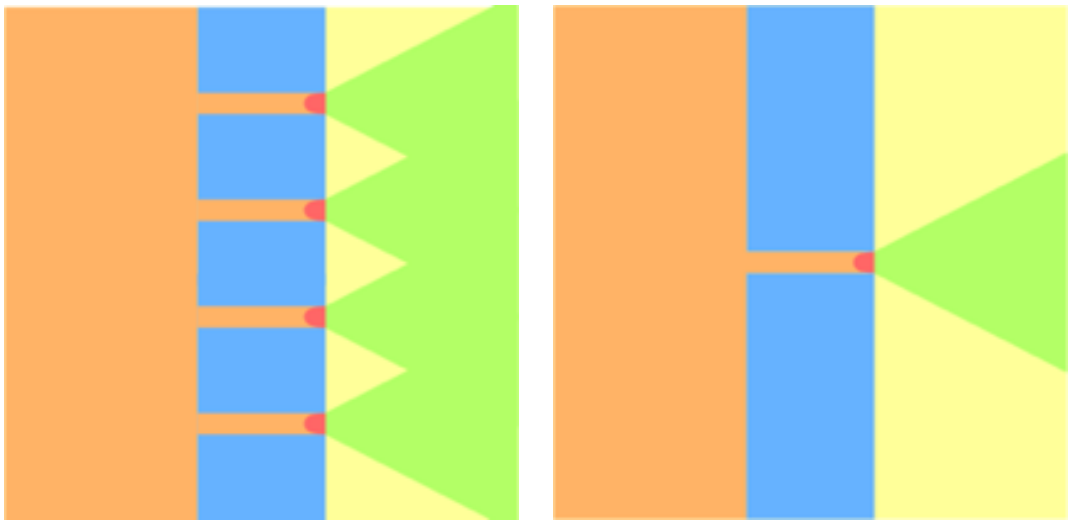


Figure 1: Visualising the α and β lookouts

Where, in both C-spaces, the left part of the free space (observer region) is coloured orange, and the right part of the free space (observed region) is coloured yellow. Additionally, the α fraction of the observer region is represented by the red semi-circles, and the β fraction

of the observed region is represented by the green triangles. α and β values are calculated as follows:

$$\alpha = \frac{\text{Total red area}}{\text{Orange area}}$$

$$\beta = \frac{\text{Total green area}}{\text{Yellow area}}$$

In the case of problem 1 (left diagram), there are actually four distinct lookout points, and hence there are four red semicircles and four green triangles (with some overlap occurring amongst the triangles).

From these diagrams, it is easy to see that the C-space of problem 1 has a larger β value, as a larger proportion of the right hand side free space is covered by the green triangles when compared to problem 2. Problem 1 has a smaller α value than problem 2, as the inclusion of three additional passageways in the middle of the configuration space decreases the ratio of total red area to orange area (even though three more red areas have been added). However, this slightly smaller α value is outweighed by a much larger β value, thus making problem 1 easier to solve than problem 2, if we use PRM with uniform random sampling.

Exercise 5.4

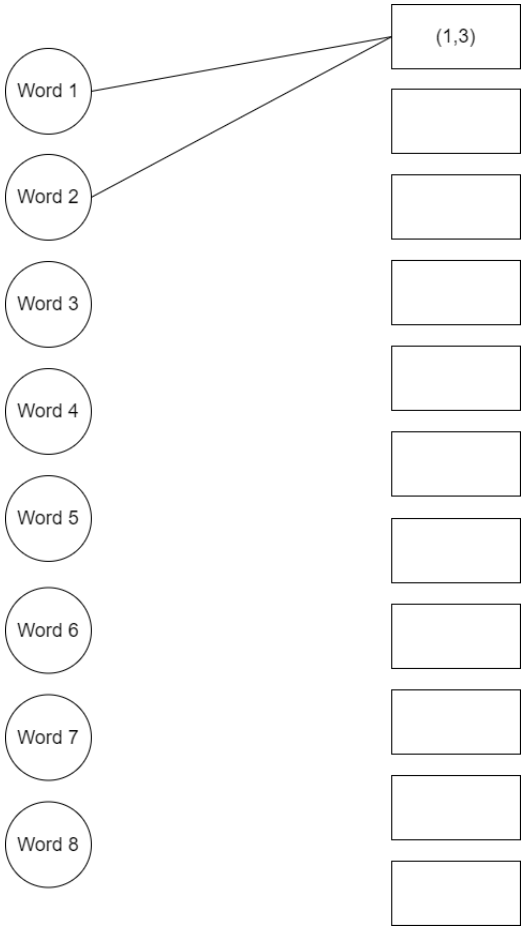
a) List the variables, their domains, and the constraints in the CSP representation of this crossword puzzle.

Variables: Words 1, 2, 3, 4, 5, 6, 7 and 8

Domains: For all variables the list of possible answers AFT, ALE, EEL, HEEL, HIKE, HOSES, KEEL, KNOT, LASER, LEE, LINE, SAILS, SHEET, STEER, TIE.

Constraints: Letters used in two between words. Nb. all constraints are binary

b) Draw a constraint graph for the CSP (you could use a tool like diagrams.net). Use a bipartite graph. A partial solution is given below, where the coordinates refer to the grid in row-column from the top left corner:



c) Apply domain consistency to this CSP, and restate any variable domains that change. Does the constraint graph change?

Domains are reduced to those with answers with word lengths that are consistent with the number of blank spaces, so all variable domains are changed. However, the constraint graph does not change structure.