# Practice Exam D

## 1 Action, Belief, Configuration

1. Let's do trajectory optimization in continuous spaces! Our problem is set up as follows:

$$\mathcal{S} = \mathbb{R}^2; \;\; \mathcal{A} = \mathbb{R}^2; \;\; T(s,a) = s + a$$

You can imagine that the states are points and actions are offsets. We start in $s_0$ and take $k$ actions to reach $s_1, \ldots, s_k$. We have a target $g \in \mathcal{S}$ and cost $l_f(s_k, g) = \|s_k - g\|^2$ on the final state $s_k$.

The cost of each action is $c(a) = \|a\|$, denoting Euclidean distance (not distance squared!). For example, $c((3,4)) = 5$.

Here is the usual direct transcription objective: Optimize $a_0, \ldots a_{k-1}$, $s_1, \ldots, s_k$ to minimize

$$\mathcal{L}(s,a) = \|s_k - g\|^2 + \sum_{j=0}^{k-1} c(a_j) + \|T(s_j, a_j) - s_{j+1}\|^2$$

(a) (1 point) Is there more than one optimal solution?
$\surd$ **yes** ◯ no

(b) (2 points) Sketch a picture of an optimal solution for $s_0 = (0,0)$, $s_k = (1,1)$, $k = 3$. Use points to denote states and vectors to denote actions.

**Solution:** A solution to this problem is any set of points that makes a straight line from $s_0$ to $s_k$, moving only in the direction of $s_k$. The picture should have a point at $(0,0)$, $(1,1)$, and (at most) 2 points between. The action vectors should be exactly the deltas between successive states.

Now! Let's think about doing this in belief space. The robot is uncertain about its position, and we represent its uncertainty with a round Gaussian with parameter $\sigma^2$. Our belief about the robot's state is encoded with three values, $b = (x, y, \sigma^2)$, signifying that we believe the current state is distributed as $s \sim \mathcal{N}((x,y), \sigma^2\mathbf{I})$.

As before, if we are at an actual location $s = (x,y)$ and take an action $a$, assume there is no error in our actions, so our new $(x', y')$ is the old $(x,y) + a$. We can now make observations with distribution $o|(x',y') \sim \mathcal{N}((x',y'), \|a\|^2\mathbf{I})$. The bigger the step we take, the less accurate the observation.

We want to write a closed-form expression for the robot's posterior belief if we start with belief $b = (x, y, \sigma^2)$ and do action $(a_x, a_y)$.

(c) (2 points) **Transition update.** If we start with belief $b = (x, y, \sigma^2)$ and do action $(a_x, a_y)$, what is the new belief on the robot's next state after the transition function is applied?

> **Solution:** Our transition function is not noisy, so we know that after the transition update,
>
> $$s' \sim N((x + a_x, y + a_y), \sigma^2 \mathbf{I})$$
>
> so
>
> $$b' = (x + a_x, y + a_y, \sigma^2).$$
>
> Either of the above suffices as a good answer.

(d) (2 points) **Observation distribution.** If we have a belief $b = (x, y, \sigma^2)$ and we do action $(a_x, a_y)$, what is the distribution over observations?

> **Solution:** If we have a belief $b = (x, y, \sigma^2)$ and we do action $(a_x, a_y)$, our belief about the current position is $s' \sim N((x + a_x, y + a_y), \sigma^2 \mathbf{I})$.
> Our observation is therefore distributed as
>
> $$o \sim (x + a_x, y + a_y) + N(0, \sigma^2 I_2) + N(0, \|a\|^2 \mathbf{I}) = N((x + a_x, y + a_y), (\sigma^2 + \|a\|^2)\mathbf{I})$$
>
> Note that this is the unconditional distribution, which is what we are asking for!
> Lots of students confused this with the conditional distribution
>
> $$o \mid s' \sim N(s', \|a\|^2 \mathbf{I})$$
>
> .

Let's make the most-likely observation (MLO) approximation.

(e) (2 points) **MLO.** Using your answer to the previous part, if we have a belief $b = (x, y, \sigma^2)$ and we do action $(a_x, a_y)$, what is the most likely observation?

> **Solution:** From the previous part, our observation is distributed as
>
> $$o|s' \sim N((x + a_x, y + a_y), (\sigma^2 + \|a\|^2)\mathbf{I})$$
>
> The most-likely observation is the mean of the distribution:
>
> $$(x + a_x, y + a_y)$$

(f) (4 points) **Observation update.** If we start with belief $b = (x, y, \sigma^2)$, do action $(a_x, a_y)$, and get the most likely observation, what is the new belief on the robot's state?
You may find it helpful that if we have two distributions

$$p(A) \sim \mathcal{N}(\mu_a, \Sigma_a) \quad p(B|A) \sim \mathcal{N}(A, \Sigma_b)$$

then $p(A|B = b)$ has mean $\left(\Sigma_a^{-1} + \Sigma_b^{-1}\right)^{-1}\left(\Sigma_a^{-1}\mu_a + \Sigma_b^{-1}b\right)$ and variance $\left(\Sigma_a^{-1} + \Sigma_b^{-1}\right)^{-1}$.

**Solution:**
From the previous part, our prior after the transition update is:

$$s' \sim N((x + a_x, y + a_y), \sigma^2\mathbf{I}).$$

Our observations are distributed as

$$p(o|s') \sim \mathcal{N}(s', \|a\|^2\mathbf{I}).$$

We compute the posterior $p(s'|o = o_{ml})$ where $o_{ml}$ is the most likely observation using the hint formulas above.
Note that when $\mu_a = b$, (which is the case for us), the posterior mean from the hint above becomes

$$\left(\Sigma_a^{-1} + \Sigma_b^{-1}\right)^{-1}\left(\Sigma_a^{-1}\mu_a + \Sigma_b^{-1}\mu_a\right) = \mu_a.$$

Therefore, the posterior mean is

$$\boxed{(x + a_x, y + a_y)}.$$

The posterior variance is

$$\left(\Sigma_a^{-1} + \Sigma_b^{-1}\right)^{-1} = \left(((\sigma^2\mathbf{I})^{-1} + (\|a\|^2\mathbf{I})^{-1}\right)^{-1} = \boxed{(\frac{1}{\sigma^2} + \frac{1}{\|a\|^2})^{-1}\mathbf{I}}$$

Another solution:
Applying the Kalman observation update formula from lecture:

$$\sigma^2_{new} = \sigma^2 - \sigma^2(\sigma^2 + \|a\|^2)^{-1}\sigma^2$$
$$\mu_{new} = (x + a_x, y + a_y)$$

Note that the mean doesn't change because the observation matches exactly with our expectation. The variance always decreases when we get an observation. Therefore, the new belief is

$$b_{new} = (x + a_x, y + a_y, \sigma^2 - \frac{\sigma^4}{\sigma^2 + \|a\|^2})$$
$$b_{new} = (x + a_x, y + a_y, \frac{\sigma^2\|a\|^2}{\sigma^2 + \|a\|^2})$$

(g) (2 points) Let's optimize! Imagine now that we want to arrive near our goal with high certainty. We might look for a plan using a problem setting like this. Let $b_i = (s_i[x], s_i[y], \sigma_i^2)$ where $s_i$ is a location. Then we might want to find $s_1, \ldots, s_k$ and $a_0, \ldots, a_{k-1}$ to minimize

$$\mathcal{L}(s, a) = \mathcal{L}_\sigma(a_0, \ldots, a_{k-1}) + \|s_k - g\|^2 + \sum_{j=0}^{k-1} c(a_j) + \|T(s_j, a_j) - s_{j+1}\|^2.$$

Come up with an $\mathcal{L}_\sigma(a_0, \ldots, a_{k-1})$ that penalizes being uncertain at the end of the trajectory, under the most-likely observation (MLO) assumption. You may find it helpful that at step $j$, the expression for the posterior variance on the robot's location (under MLO), as a function of $\sigma_0^2$ and the action sequence $a_1, \ldots, a_j$ is:

$$\mathbf{I}\left(\frac{1}{\sigma_0^2} + \sum_{i=1}^{j} \frac{1}{\|a_i\|^2}\right)^{-1}$$

**Solution:** We can take the determinant of the posterior variance matrix at $s_k$:

$$\mathcal{L}_\sigma(a_0, \ldots, a_{k-1}) = det\left[\mathbf{I}\left(\frac{1}{\sigma_0^2} + \sum_{i=1}^{k} \frac{1}{\|a_i\|^2}\right)^{-1}\right]$$

Alternatively,

$$\left(\frac{1}{\sigma_0^2} + \sum_{i=1}^{k} \frac{1}{\|a_i\|^2}\right)^{-1}$$

is also acceptable.

## 2 SDDL

2. (10 points) This is the Stata Definition Language! We want to be able to find our way around Stata, so we need a planner. Note: upper case symbols are **constants** and upper case predicates denote **static** facts (they are always true).

   We can represent floor plans as graphs between spaces, such as lobbies, hallways and offices. Assume we have an arbitrary **directed** graph encoded by `(CONN ?x ?y)` facts and a single `(move ?person ?x ?y)` PDDL action which updates a `(loc ?person ?x)` fluent.

```
(:action move
    :parameters (?person ?x ?y)
    :precondition (and (loc ?person ?x) (CONN ?x ?y))
    :effect (and (loc ?person ?y) (not (loc ?person ?x))))
```

   An example goal might be to go from initial state `(loc ME S)` to goal `(loc ME G)`.

   (a) How do the values of $h^{\max}(s)$, $h^{\text{add}}(s)$, and $h^{\text{ff}}(s)$ compare to the optimal path-length in the graph (as found by BFS). Indicate whether the values for each of the heuristics would in general be larger, smaller or equal to the optimal value. Also, indicate whether any of the heuristic values would be equal to each other.

   - $h^{\max}(s)$ has what relation to the optimal value? ◯ larger $\surd$ **equal** ◯ smaller
   - $h^{\text{add}}(s)$ has what relation to the optimal value? ◯ larger $\surd$ **equal** ◯ smaller
   - $h^{\text{ff}}(s)$ has what relation to the optimal value? ◯ larger $\surd$ **equal** ◯ smaller
   - $h^{\max}(s) = h^{\text{ff}}(s)$ $\surd$ **True** ◯ False
   - $h^{\max}(s) = h^{\text{add}}(s)$ $\surd$ **True** ◯ False
   - $h^{\text{add}}(s) = h^{\text{ff}}(s)$ $\surd$ **True** ◯ False

   Justify your answers.

   > **Solution:**
   > $h^{\max}(s) = h^{\text{add}}(s) = h^{\text{ff}}(s)$ and all are equal to the optimal path length. The RPG essentially carries out a BFS, a fact (location) appears at level $i$ if it takes $i$ actions to reach there, and since there is only one goal fact, all these heuristics are equal.

   Moving in and out of Stata is not so easy these days, different connections have different requirements. To go from the STATA-LOBBY to the CSAIL hallway one needs an ATTESTATION and a PHONE, while to get into an office one needs a KEY-111 for office 111. We will extend our representation from the first part of this problem to indicate how many requirements there are on a given connection by adding one more argument to `CONN`: `(CONN ?x ?y ?nr)` where `?nr` is 0, 1 or 2, indicating how many requirements there are.

```
(CONN STATA-HALLWAY STATA-111 1)
(CONN STATA-LOBBY CSAIL-HALLWAY 2)
```

   We can then specify the requirements with some additional static facts:

   - `(REQ1 ?x ?y ?r)`: represents one requirement for going from location `?x` to location `?y`;
   - `(REQ2 ?x ?y ?r)`: represents another requirement, when there is more than one; and
   - `(CONN ?x ?y ?r)`: represents a (directed) arc in the map and indicates how many requirements are needed to make the transition.

   For example:

```
(REQ1 STATA-HALLWAY STATA-111 KEY-111)
(REQ1 STATA-LOBBY STATA-HALLWAY ATTESTATION)
(REQ2 STATA-LOBBY STATA-HALLWAY PHONE)
```

We also introduce a predicate `(has ?person ?r)` which indicates that the person satisfies a requirement, e.g. `(has Alexa PHONE)`.

(b) Write an operator for `?person` moving from `?loc1` to `?loc2` when the transition has two requirements `?r1` and `?r2`.

```
:action move2
    :parameters (?person ?loc1 ?loc2 ?r1 ?r2)
    ...)
```

**Solution:**

```
(:action move2
    :parameters (?person ?loc1 ?loc2 ?r1 ?r2)
    :precondition (and (loc ?person ?loc1) (CONN ?loc1 ?loc2 2)
                       (REQ1 ?loc1 ?loc2 ?r1) (REQ2 ?loc1 ?loc2 ?r2)
                       (has ?person ?r1) (has ?person ?r2))
    :effect (and (loc ?person ?loc2) (not (loc ?person ?loc1)))
```

## 3 Rover

3. You are working with a team of rocket scientists to solve a planning problem for a planetary rover. The rover must find a path that satisfies the following requirements:

   A Start and end its mission at the lander.

   B Avoid being inside any craters between 8:00 PM and 8:00 AM, to prevent exposure to freezing temperatures.

   C Visit sites of scientific interest to take a picture. A picture taken at site $i$ provides scientific value $v_i \in \{1, 2, 3, 4, 5\}$ only once (further pictures provide no additional value). The total value of the mission must exceed $V_T$. There are $k$ total sites, but it's possible that only a subset of sites need to be visited.

   Assume a high-resolution planet map is available, which divides the terrain into a grid with 1-meter resolution. For each 1m × 1m cell, the map indicates whether it's in a crater, and if it contains a site of scientific interest, the scientific value of that site. Each cell contains at most one site.

   The rover may move into any 4 of the cells adjacent to the cell it's currently in, or stop and take a picture. The time needed to accomplish all of these actions is known, and the transition function is deterministic.

   Assume your cost function is as follows:

| | |
|---|---|
| move to adjacent cell (normal conditions) | 1 |
| move to adjacent cell (from within a crater between 8:00 PM and 8:00 AM) | 100 |
| take a picture | 1 |

   (a) (4 points) You are asked to find a minimum cost path that satisfies all the constraints. What would be a *minimal* state space you could use to solve this problem? Explain why each part is necessary.

   **Solution:** $(l, t, p)$ where $l$ is a cell in the grid (needed to know the **location** of the rover), $t$ is the current **time** (needed to avoid craters), and $p$ is a binary length vector of size $k$ whose $i$th entry indicates **whether the rover has taken a picture at site** $i$.

(b) Define **grid distance** to be the cost of the shortest path between two cells on the map assuming the cost of moving is always 1.

Define $\mathbf{V_C}$ to be the current scientific value of pictures the rover already has.

Which of the following heuristics are admissible? (For problems with a box, provide an explanation).

i. (1 point) Grid distance between lander and rover's current position
   √ **admissible** ◯ not admissible

ii. (1 point) $(V_T - V_C)/5$.
   √ **admissible** ◯ not admissible

iii. (2 points) Rank sites not yet pictured by their grid distance from the rover's current position, smallest first. Greedily select sites until the total scientific value of selected sites is at least $V_T - V_C$. Sum the grid distances from the rover's current position to each selected site.
   ◯ admissible √ **not admissible**

   **Solution:** Not admissible. Imagine if the remaining sites that need to be visited formed a straight path to the lander. This heuristic would overestimate the length of the path back to the lander.

iv. (2 points) Grid distance to the closest, not yet pictured, scientific site.
   ◯ admissible √ **not admissible**

   **Solution:** In a state where the rover has taken pictures with enough scientific value, this heuristic may overestimate the cost (which is now just the shortest path to the lander).

(c) (2 points) What two properties, in addition to admissibility/consistency, are desirable in a heuristic?

**Solution:** (1) Closeness to the true expected cost to go and (2) efficiency to compute.

Now imagine the rover starts to wear out and its ability to drive between locations becomes erratic. The robot now has some small probability of ending up at a different nearby cell than the one intended after each move action.

(d) (2 points) What would be a good way to solve the problem if we can do a lot of computation in advance, on earth, and we're not sure where the rover is actually going be deployed?

**Solution:** Run value iteration, q-learning, or build and store a huge policy-tree.

(e) (2 points) Imagine we are in a situation where we can't do the computation in advance. In fact, we can only solve planning cheaply by making a most likely outcome determinization. What could cause the rover to incur high costs when it follows the resulting plan? Why does this happen?

**Solution:** It could fall into a crater during the night hours and freeze. Also: deviate from the plan (although this might not incur high cost necessarily, but would require replanning).

# 4 Flying

4. (9 points) Now, let's assume that you are a drone, flying above the terrain, but staying at a fixed altitude.

   - You can still go 1 mile per minute, your battery still holds $m$ units, and the chargers and charging time remain the same (this is a *big* drone)!
   - Drones never get bored.
   - There are some high peaks that prevent you from moving within some regions of the space at your fixed altitude, essentially forming obstacles in a planar problem.
   - This drone has no problem hovering and is insensitive to wind, so you can assume that forces and velocities are handled by the controller and you only need to worry about paths and charging.
   - Drone motion is holonomic; it can move in any direction.
   - The chargers are in the same place, and your start and goal locations will be towns, but the road network is irrelevant.

   You can do long-distance induction charging (!). You are given a function $charge - delta(l_i, l_j)$ that returns the aggregate change in charge (which may be positive or negative) due to flying from $l_i$ to $l_j$ in a straight line. Your battery has over-charge protection, so that it will stop accumulating charges when it is at max-capacity.

   You decided to use trajectory optimization to approach this problem.

   The overall cost of the trajectory is the sum of the costs of $n$ individual linear segments.

   $$J(((l_0, b_0), (l_1, b_1), \ldots, (l_n, b_n)) = \sum_i C((l_i, b_i), (l_{i+1}, b_{i+1}))$$

   where $l_0 =$ `start`, $b_0$ is the initial battery charge, and $l_n =$ `goal`. The cost for each segment includes several terms, including the segment length, the degree to which the segment penetrates an obstacle, and the degree to which the charging dynamics are respected ($\alpha$ and $\beta$ are constants that trade off the relative importance of the terms):

   $$C((l_i, b_i), (l_j, b_j)) = |l_i - l_j|^2 + \alpha \cdot penetration((l_i, l_j, obstacles)) + \beta \cdot charge_{i,j}$$

   (a) (2 points) Which of the following terms captures the degree to which charging dynamics are respected ($charge_{i,j}$)?

   ◯ $|b_i - b_j|^2$
   ◯ $|b_i + b_j|^2$
   ◯ $|b_i + charge - delta(l_i, l_j) + b_j|^2$
   √ $|b_i + charge - delta(l_i, l_j) - b_j|^2$

   (b) (3 points) Explain your choice from above.

   **Solution:** The change in charge between the endpoints has to match the charge-delta.

   (c) (2 points) However, we're missing a term! What is it?

   **Solution:** A penalty for $c$ going negative.

   (d) (2 points) Of these four terms (the three we listed, plus the one you provided), which ones, if any, need to be driven to 0 to obtain a legal trajectory?

   ◯ length √ **penetration** √ **charge-delta** √ **missing**

# 5 Searching for Answers

5. Are each of the following claims true or false? Provide a short justification.

(a) (2 points) A* search is equivalent to greedy best-first search if the heuristic is 0 everywhere.

◯ True √ **False**

**Solution:** In this case, A* will become equivalent to UCS.

(b) (2 points) If a solution to a motion planning problem exists, then RRT will eventually find it.

√ **True** ◯ False

**Solution:** It is probabilistically complete, and the sampling guarantees that eventually it can find any path.

(c) (2 points) If a solution to a motion planning problem exists, then path optimization will eventually find it.

◯ True √ **False**

**Solution:** As a gradient-based method, it can get stuck in local optima.

(d) (2 points) On a deterministic min-cost path problem, Monte-Carlo Tree Search (MCTS) has the same worst-case runtime as breadth-first search.

◯ True √ **False**

**Solution:**
The computational complexity of MCTS is hard to analyze, but it will generally do a lot of repeated work, in general.

(e) (2 points) In a large discrete search problem with a high branching factor, MCTS can be a better choice than A* search, even if we use a good heuristic for A* search.

√ **True** ◯ False

**Solution:**
MCTS can take advantage of locality properties in the problem to get additional hints from previous iterations about where to concentrate its search effort.

(f) (2 points) Suppose that for any two states in a discrete search problem, there is exactly one path from the first state to the second. This property alone does not tell us anything about which search algorithms might be better or worse to use.

◯ True √ **False**

**Solution:** In a problem like this, the dynamic programming aspects of search algorithms won't help. So, MCTS or even BFS with no visited list could be good.