Homework 3

Due: September 30, 2025 at 11:59pm

Instructions

- See https://rpmml.github.io/homework_instructions/ for instructions on accessing the accompanying code and submitting homework via GradeScope.
- See https://rpmml.github.io/policies for additional policies regarding collaboration, LLMs, and late submissions.
- Use Piazza or attend office hours to ask questions about the homework.

Part 1: Written Exercises

Practice with Heuristic Search (20 points)

Consider the following classical planning problem:

- $\mathcal{S} = \{-1, 0, 1\}^3$. In other words, $s \in \mathcal{S}$ is a tuple (x, y, z) where x, y, z are each -1, 0, or 1.
- $\mathcal{A} = \{\texttt{inc-x}, \texttt{dec-x}, \texttt{inc-y}, \texttt{dec-y}, \texttt{inc-z}, \texttt{dec-z}\}$.
- $I(s, a)$: incrementing or decrementing the respective entry would not go "out of bounds."
- $F(s, a)$ increments or decrements the respective entry.
- $C(s, a, s') = 1$ if $(x \leq y \leq z)$ where $s = (x, y, z)$, and otherwise, $C(s, a, s') = 10$.
- $s_0 = (0, 1, 0)$.
- $G(s)$: $s = (-1, 0, -1)$.

Consider also a "goal count" heuristic:

$$V_{\text{GC}}((x, y, z)) = 1[x \neq -1] + 1[y \neq 0] + 1[z \neq -1]$$

where $1[\textit{condition}]$ is equal to 1 if *condition* is True and 0 otherwise.

Perform A* search by hand using the goal count heuristic. Break ties lexicographically.[1] Report (1) the final plan; (2) the cost of the final plan; (3) the number of nodes expanded before the plan is returned. Show your work to receive partial credit in case of errors.

Pause to Ponder (No Points & No Submission Required)

Was the goal-count heuristic admissible in this problem?

[1] If in doubt, use Python to check, for example, whether $(-1, 0, 0) < (0, -1, 0)$. For the actions, use alphabetic order based on their names.

Practice with Motion Planning (15 points)

Let's play a game! I will give you a *discretization number* $n = 1, 2, \ldots$. You will give me back a motion planning problem in 2D:

- $\mathcal{X} = [0, 1]^2$
- $x_0 = (0, 0)$
- $x_g = (1, 1)$
- $f : \mathcal{X} \to \{\mathrm{T}, \mathrm{F}\}$ *to be determined!*

where:

1. A solution exists.
2. Motion planning by discretization with n fails to find any solution.
3. Motion planning by discretization with $n + 1$ succeeds.

The specific discretization technique that we will consider here works as follows:

- Given n, create a classical planning problem with states $\{\frac{0}{n}, \frac{1}{n}, \ldots, \frac{n}{n}\}^2$.
- Actions are left, down, up, and right, connecting vertically and horizontal adjacent states (with cost 1).
- An action that connects x to x' can be initiated if *all* states between x and x' pass the feasibility check f.

Describe a strategy to win the game. You do not need to prove that your strategy is correct, and you may describe the strategy informally (e.g., by drawing pictures).

Pause to Ponder (No Points & No Submission Required)

How could you generalize your strategy to work in higher dimensions?

Part 2: Coding Exercises

Comparing Search Algorithms (15 points)

Implement a *classical planning problem* and a *heuristic* where the plan returned by A* has lower cost than the one returned by GBFS, but the number of nodes expanded by A* is at least 100 more than GBFS. Examine the unit tests in `test_comparing_search.py` to make sure you understand the question. No written answer required.

Pause to Ponder (No Points & No Submission Required)

How would your code change if we required 1000 more instead of 100?

Fun with PDDL (5 points)

Create any (original) PDDL domain and problem that GBFS with the hFF heuristic can solve. Examine the unit tests in `test_custom_pddl.py` to make sure you understand the question. No written answer required.

Pause to Ponder (No Points & No Submission Required)

What changes could you make to your domain and problem to make it easier or harder for GBFS with hFF?

Centralized Multi-Robot Motion Planning (30 points)

The starter code provides examples of motion planning with a single `Geom2D` (rectangular or circular) robot. Your job is to implement a motion planner that coordinates *multiple* `Geom2D` robots at once. Examine the unit tests in `test_multi_robot_motion_planning.py` to make sure you understand the question. No written answer required.

Pause to Ponder (No Points & No Submission Required)

What if the robots were not centrally coordinated? How would each determine its own motion?

Part 3: Your Turn to Teach

Create a Question (10-15 points)

Create a new question of any kind (writing, coding, etc.) that pertains to the material covered in this problem set. To receive full credit, you only need to write the question. To receive up to 5 points of extra credit, you should also provide a solution. Extra credit points will also be awarded based on creativity and effort. Please indicate in your answer (1) whether your question can be used (with modifications) in future versions of this course; and (2) whether you would like to be credited by name if your question is used.

Part 4: Feedback

General Course Feedback (5 points)

How can we improve the course in future years? Any and all feedback is welcome. Please comment on this problem set, lectures, and anything else. Some feedback is required for full credit. If you prefer to additionally submit anonymous feedback, please do so through the course website.