PROJECT

Implement a Planning Search

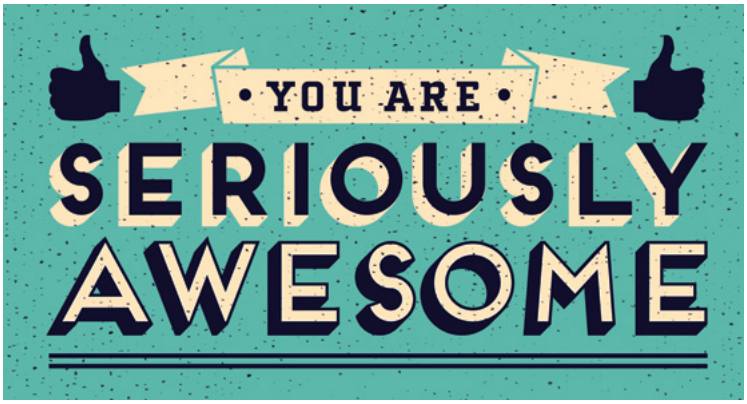
A part of the Artificial Intelligence Nanodegree and Specializations Program

PROJECT REVIEW | CODE REVIEW 13 | NOTES

Meets Specifications

SHARE YOUR ACCOMPLISHMENT

Terrific project! Not a single word of suggestion 😉



Planning Problem Representation

The problems and class methods in the `my_air_cargo_problems.py` module are correctly represented.

```
**********************************************************************
                        Test Result Summary
**********************************************************************

air_cargo_p1 returns the correct initial fluents:                     .
air_cargo_p1 returns the correct goal fluents:                        .
air_cargo_p1 returns an object of type problem:                       .
air_cargo_p1 returns the correct initial values:                      .
air_cargo_p2 returns the correct initial fluents:                     .
air_cargo_p2 returns the correct goal fluents:                        .
air_cargo_p2 returns an object of type problem:                       .
air_cargo_p2 returns the correct initial values:                      .
air_cargo_p3 returns the correct initial fluents:                     .
air_cargo_p3 returns the correct goal fluents:                        .
air_cargo_p3 returns an object of type problem:                       .
air_cargo_p3 returns the correct initial values:                      .
AirCargoProblem correctly lists possible actions in a given state:    .
AirCargoProblem correctly constructs all possible actions:            .
AirCargoProblem correctly updates state for a given action:           .
AirCargoProblem yields a correct solution when input to breadth_first: .

----------------------------------------------------------------------
           . - Test Passed    F - Test Failed    E - Error
```

An optimal sequence of actions is identified for each problem in the written report.

👌 Perfect optimal plans with an additional line specifying the goal achieved

Automated Heuristics

Automated heuristics “ignore-preconditions” and “level-sum” (planning graph) are correctly implemented.

```
**********************************************************************
                        Test Result Summary
**********************************************************************

AirCargoProblem implements the ignore preconditions heuristic:        .
Action levels have the correct number of actions:                     .
Literal levels have the correct number of literals:                   .
competing_needs_mutex behaves correctly:                              .
inconsistent_effects_mutex behaves correctly:                         .
inconsistent_support_mutex behaves correctly:                         .
interference_mutex behaves correctly:                                 .
negation_mutex behaves correctly:                                     .
Serialization of mutexes is correct:                                  .
levelsum heuristic behaves correctly:                                 .

----------------------------------------------------------------------
           . - Test Passed    F - Test Failed    E - Error
```

Performance Comparison

At least three uninformed planning algorithms (including breadth- and depth-first search) are compared on all three problems, and at least two automatic heuristics are used with A* search for planning on all three problems including “ignore-preconditions” and “level-sum” from the Planning Graph.

👏 Perfect code! Don't forget to check out code review.

Nice selection of uninformed planning algorithms!

A brief report lists (using a table and any appropriate visualizations) and verbally describes the performance of the algorithms on the problems compared, including the optimality of the solutions, time elapsed, and the number of node expansions required.

👍 Great tables! All the information needed is here!

It's nice that you used pypy, your running times are amazingly short!

Good conclusions, in general: Breadth-first and A* heuristic searches should provide optimal results; depth-first does NOT guarantee optimality. Level-sum should have the least number of expansions, with ignore-preconditions as the next best for this metric.

The report explains the reason for the observed results using at least one appropriate justification from the video lessons or from outside resources (e.g., Norvig and Russell’s textbook).

👍 Not one buy many justifications

- Good explanation of ignore preconditions
- `levelsum` is actually admissible if goals are independent.
- Good job about remarking the quality of a good heuristic. Keep in mind that we may add some Domain Knowledge specific to the type of problem we're handling.

Research Review

The report includes a summary of at least three key developments in the field of AI

planning and search.

👍 Nice write up focusing on the difference between FF vs HSP!

Nice key terms: Relaxing Graphplan, Hill climbing, pruning the search space, etc.

DOWNLOAD PROJECT

13 CODE REVIEW COMMENTS ›

RETURN TO PATH

Student FAQ