University of Alberta
Computing Science Department
CMPUT 366 - Fall 2022

Project (Optional)
Due December 09 at 23.59
Replaces One Assignment

Search & Planning in AI (CMPUT 366)

Introduction

In the project you will have the opportunity to work independently on a topic related to search & planning in AI. You should see the project as an assignment that you will design from scratch. You have to come up with the idea of the assignment and write your own "starter code". The project is optional and it will replace the marks of one assignment, whichever assignment that maximizes your grade. You should feel free to choose your own topic, but remember, it must be related to the course. Also, it is advisable to discuss your topic with the instructors, to ensure that your topic is not too difficult nor too easy.

I provide starting points for the project in the sections below, but it is your responsibility to research what is needed for the project. The instructors are available to help you with your project, but it is unlikely they will be able to provide you step-by-step instructions of what needs to be done.

Deliverables

You will write a report (in pdf format) and upload it on eClass by the deadline of the project. Your report should contain the following sections and have no more than 6 pages, including references (single column and font size 11).

- **Introduction** - Describe the problem you are solving and briefly explain why the problem is important.
- **Methods** - Explain how you solved the problem. Here you should provide enough information so others can reproduce your solution.
- **Evaluation** - Present the results you obtained. For example, if you decided to implement IDA* to solve puzzles, then you could present the number of nodes IDA* needs to expand to solve a set of puzzles.

You also need to include a link to your Github repository with the code for your project. Alternatively, you can compress all the files needed to run your implementation in a zip file.

Evaluation

You will be evaluated on the quality of your report and implementation. The report needs to clearly explain what was done and the results that you obtained. It is important to choose carefully how to present the results. Will you use a table or a plot? How to design the table to tell the story you observed in your

experiments? If you believe that a plot will more easily convey the story you observed, then which kind of plot will you use? These are the questions you should ask yourself when deciding how to present the results. If you have time, experiment with different ways of presenting the results, ask your friends to read your report. Do they understand the story you are trying to tell?

Your implementation will be mostly evaluated through your report. So make sure you explain what was implemented in the report. For example, if you decide to implement IDA* to solve puzzles, you need to explain the puzzle you are solving and the heuristic function used to solve the puzzles .

I will use the guidelines described below to mark the projects.

1. **Excellent Performance** (90-100% of marks). Complete implementation, handles corner cases (when applicable); implemented common and uncommon enhancements encountered in the literature; report is clear; results are presented clearly and succinctly.
2. **Competent Performance** (70-90% of marks). Complete implementation; implemented common enhancements encountered in the literature; report is clear; results are presented clearly and succinctly.
3. **Acceptable Performance** (50-70% of marks). Complete implementation; report is mostly clear; results are presented clearly.
4. **Could Improve** (0-50% of marks). Incomplete implementation; report is mostly clear and some results are presented.

Project Topics

Here are a few suggestions of project topics. Some of the topics listed below will be covered in class after the reading week.

1. Implement IDA* for solving the 3×3 Sliding-Tile puzzle.
2. Implement a CSP solver for problems other than Sudoku.
3. Implement other enhancements for solving Sudoku (e.g., learning no-goods).
4. Implement a local-based synthesizer for solving simple math problems.
5. Implement a problem using PDDL and use a classical planner to solve the problem.
6. Implement Levin tree search to solve a pathfinding problem.