CPSC 481 Project proposal

The course requires a software project. You are to come up with the topic of your project. The final project will be presented in class during the last two weeks of the semester. This document describes the expectations for the project and what to include in a **project proposal**.

The classic student project in an introductory AI course is implementing a program that can play a specific game. However, CPSC 481 is at the end of a long prerequisite chain, and you are close to getting a degree in Computer Science. So, you should consider this project as a way to bring together concepts from other courses - a "capstone" on your undergraduate education. Therefore, you can also think of how AI can be applied to topics covered in other courses (e.g., security). You can also consider the project as a way to learn to use software libraries to complement the theoretical foundation that is the focus of CPSC 481.

Here are some ideas for projects:

- Implement a competitive game-playing agent; the game should be non-trivial so that you can implement your own evaluation function.
    - Deterministic games include:
        - reversi/Othello,
        - Kalah (or other variants of Mancala)
        - Three men's morris/Tapatan (or more complex Nine men's morris)
    - Chess, checkers, Go are complex games but as they have existed for a long time, you can start from an existing piece of code/library.
    - A variant of a popular game such as Tic-Tac-Toe
    - Non-deterministic games are most card games, games using a dice.
    - Simple video games
    - You can also add to the Game of Nim by adding an evaluation function (so that you can play with many rows) *and* a GUI
- Single player games/puzzles
    - Examples include the Rubik's cube, Mastermind, SameGame, Cracker Barrel's peg game or a more complex variant such as Peg solitaire.
- Instead of building an autonomous player, you could think of making a "tutor" or "assistant" for a human player (i.e., it recommends moves to the human player instead of being the opponent)
- Building a simple application that uses an existing AI library such as OpenAI's GPT models, Microsoft Azure Cognitive Services, Google Cloud AI.

    - Note that these commercial libraries are **not free**. So you should plan to use only their trial version or be prepared to pay for the service.
- Applications from another course. For example:
    - Malware analysis (CPSC 458)
    - Natural language for text processing (CPSC 375)
    - A mobile application with intelligence (CPSC 411)
    - Add “intelligence” to a game you developed in CPSC 386

  See the appendix for some other possibilities.

Grading criteria

It will help to go through the grading criteria to decide on the scope of your project. Note that the grading rubric is only approximate.

1. **Contribution (55%)**: you may reuse code from open source projects, but the project must have your own contribution. Typically, you will either apply standard algorithms to a new or under-studied game (in which case your contribution is the application), or you will implement a new algorithm/function for an existing application (e.g., a new evaluation function for chess). You may re-use AI algorithms from the textbook. You can also use openly-available software libraries to build your application programs. You will be evaluated on the appropriateness of the algorithms. For example, a brute-force search will be judged less favorably than an approach that uses heuristics or other techniques.
2. **Performance evaluation (10%):** you should consider how your final software will be objectively evaluated. For example, if you build a game player, you will play against other humans or programs and report its performance. If it is based on machine learning, then you will typically evaluate it with open datasets. Other metrics could include time taken by the program, amount of training data required, etc.
3. **Novelty of the idea (5%)**: For example, Pacman and chess have been studied for a long time; a historic variant of Tic-Tac-Toe is more novel.
4. **Capstone/integration (5%):** How well does your project bring together the concepts and skills from other courses? You could make a list of specific courses that you will build on (programming language courses, data structures?, software engineering, statistics, …).
5. **User interface (5%):** Does it have an appropriate user interface? In most cases, you will demonstrate the software live or video.
6. **Presentation (5%):** How convincing is your presentation? Did it make good use of the allotted time?
7. **Written report (5%):** The report should be

    - Complete: describe all parts of the project and include its performance evaluation
    - Clear: quality of the writing
8. **This proposal (10%)**

Group work

You can work in groups of 1-3.

Note: Graduate students must work individually.

Proposal submission:

Write a **1-2 page paper** (in PDF format) that includes the following:

1. Short project title
2. Group members' names
3. Describe the problem that you want to work on.
4. Programming language
5. Datasets (if required)
6. Is there existing code? If so, what extensions will you add?
7. Algorithm/approach
8. Describe a timeline for how you plan to finish the project.
9. Any special computing platform (e.g., Raspberry Pi, GPU) that you will use.
10. Roles and responsibilities. For example, if the project is on machine learning, at least one person should have ML experience.

Submission and deadline

Team leaders turn in **ONE** .**pdf** file to Canvas by March 10, 2024

Appendix

If you have prior experience in any of these areas, you can propose a project in that area.

- Machine learning (suitable if you have taken CPSC 483):
    - Use machine learning to learn an evaluation function/heuristic for a game/puzzle
    - Games requiring reinforcement learning
- Statistics or logic
    - Write software that recommends actions. Ideally, this will be an application that makes use of your personal experience. For example:

        - recommend what courses to take in a semester given the user's interests, user's past courses, course-prerequisites, degree requirements, time commitment, etc.
        - Physical fitness, diet recommendations
        - Invest money in the stock market
        - Buy a computer, car, …
- Hardware-related projects. You can build a simple embedded application using Raspberry Pi/Arduino + sensors.