CS 182: Artificial Intelligence
Final Projects

1 Important Deadlines

The final project will consist of the following four aspects:

1. Project Proposal
2. Status Update
3. Final Paper
4. Presentation, either:
    - Oral presentation in class
    - Poster presentation at computer science fair

No late days for any course project deadline. If more students prefer oral presentations than can be accommodated during the last lecture period, talk slots will be filled by lottery or we will consider adding a second presentation day on 11/29 during the normal lecture time.

These deliverables will be due at the following deadlines:

Aspect	Deadline
Project Proposal	10/28, 5:00 PM
Status Update	11/23, 5:00 PM
Oral Presentations	12/1, During Lecture
Posters to Printer	Before 12/6 !!
Poster Presentations	12/9, 12:00-2:00 PM
Final Project Report	12/12, 11:59 PM

2 Goals and Scope

The CS 182 final project provides an opportunity for you to apply and extend the foundational concepts you have learned in the course. It is intended to encourage you to integrate ideas from different course components and allow you to delve deeper into areas of interest. Projects may be implementation, testing, and analysis of algorithms mentioned in lecture or described in the text but not covered in assignments or design and implementation of an intelligent system that combines algorithms and representations from several course components (e.g., a more complex game-playing program or an intelligent advising system). Pick something that interests you. Students are expected to design and carry out final projects working in teams of 2-3 students. Note that we expect all students to demonstrate a roughly equal amount of work, so teams of 3 should be sure to tackle appropriately sized problems. You can use the Piazza forum to find partners. **Special Note:** If you want to undertake a theoretical project or work alone, please talk to Scott or one of the teaching fellows before beginning your proposal.

A list of possible course projects has been given below. Their content and scope are meant to be suggestive, not definitive. The teaching staff would be delighted to talk with you about possibilities. After the proposal, each final group will be assigned a course mentor who will be the best point of contact throughout the project.

Final projects that fall outside the specifics but within the spirit of these suggestions are encouraged, as are projects that draw on other computer science courses you have taken. It is, however, essential that the project make clear connections with topics covered in the course. If the idea you have for a project varies significantly from the suggestions, or you are unsure of its suitability, please contact the teaching staff enough in advance of the project proposal due date that our discussions with you can help shape your proposal. Some places beyond the text and course readings to look for inspiration: Proceedings of the major AI conferences (AAAI, IJCAI, ECAI, ICAPS, AAMAS, HCOMP, UAI) and journals (Artificial Intelligence, JAIR: J. of AI Research, JAAMAS: J. of Autonomous Agents and Multi-Agent Systems). Most CS papers are available free online either through the ArXiV or through search on resources like Google Scholar.

If your project involves building a system that includes designing a GUI or other significant software components, make sure that you focus most of your effort on the AI aspects of the project. We will evaluate your project on the concepts it investigates and the results and on how well it demonstrates your comprehension of the concepts, techniques and issues we have covered in the class. The project grade will incorporate evaluation of the proposal, presentation (oral or poster) and final paper quality. As there is no final exam for CS 182, your final project is the major integrative element of coursework.

3 Deliverables

3.1 Proposal

To ensure that you choose an appropriate project, you are required to turn in a 1–2 page project proposal by the date indicated above. The proposal should begin with a clear, unambiguous statement of your topic, and include all of the following:

1. a brief discussion of the problem and algorithms you intend to investigate and the system you intend to build in doing so,
2. identification of specific related course topics (e.g. heuristic search, MDPs, CSPs, etc.).
3. examples of expected behavior of the system or the types of problems the algorithms you investigate are intended to handle,
4. the issues you expect to focus on,

5. and a list of papers or other resources you intend to use inform your project effort. This list will form the core of your project report reference list. If your project includes anything unusual (such as having significant systems demands), please state this as well.

This document will be worth 5% of your grade for the final project. Use it to demonstrate to us that you have completed some background work on your chosen topic. The proposal should resemble a CS 182 problem set in the sense that if one were to give your proposal to another student, s/he should be able to complete the project as well. Proposals should identify all members of the team and indicate how you intend to divide work on the project. We will review your proposal and return it to you with comments and suggested modifications.

3.2 Update

To ensure that you are on track with the project and to identify any issues on time, you are required to submit a short (1-page) report with a status update. The report should describe the problem you are working on, the progress you've made so far, and any problems you came across that you would like to get help with. With the status update each group should also identify whether they would prefer to give an oral presentation or present at the poster session.

3.3 Presentation

The oral presentations will be done in class and the poster session will be at the CS poster session on the date indicated above. These presentations are a chance to explain your problem and approach, showcase what you've accomplished, and get advice on surmounting any hurdles you've encountered. Students are expected to attend both presentation sessions, as they provide an opportunity for you to learn from each other.

- Oral presentations will be allocated 10 minutes, and should be focused on key issues. You are encouraged to bring less than 4 slides, because often just one diagram or chart can explain the essence of your idea and save lots of presentation time. You need not prepare fancy graphics; just come prepared to explain your topic and share what you have discovered.
- Posters will be printed through a SEAS allocated fund for student poster printing. A good rule of thumb is to design a poster to act similarly to a presentation as a tool for walking readers through your work. Along with the poster each member of your group should be able to explain the project to a general audience at the fair as well as the TFs and other students in the class.

3.4 Report

You must submit a written report on your project and the complete, well-documented source code for it. The report should be 5-10 pages in length. It should describe the algorithms you implemented and the data on which they were tested and include an analysis of results or system performance (depending on the scope of the project). We recommend GitHub as a method of collaborating and submitting source code. Reports must be formatted using LaTeX. We strongly recommend using the provided template on Canvas to structure your report.

The report must contain all of the following content:

- A description of the purpose, goals, and scope of your system or empirical investigation. You should include references to papers you read on which your project and any algorithms you used are based. Include a discussion of whether you adapted a published algorithm or devised a new one, the range of problems and issues you addressed, and the relation of these problems and issues to the techniques and ideas covered in the course.
- A clear specification of the algorithm(s) you used and a description of the main data structures in the implementation. Include a discussion of any details of the algorithm that were not in the published paper(s) that formed the basis of your implementation. A reader should be able to reconstruct and verify your work from reading your paper.
- Analysis, evaluation, and critique of the algorithm and your implementation. Include a description of the testing data you used and a discussion of examples that illustrate major features of your system. Testing is a critical part of system construction, and the scope of your testing will be an important component in our evaluation. Discuss what you learned from the implementation.
- For algorithm-comparison projects: a section reporting empirical comparison results preferably presented graphically.

In addition, the report should include two appendices:

- Appendix 1 – A clear description of how to use your system and how to generate the output you discussed in the write-up. *The teaching staff must be able to run your system.*
- Appendix 2 – A list of each project participant and that participant's contributions to the project. If the division of work varies significantly from the project proposal, provide a brief explanation.

Your code should be clearly documented. Submit your code along with the project document to Canvas (or link directly to a public repository in your report). This link should appear in Appendix 2. If you have any questions about these specifications, please ask the teaching staff.

4 Suggestive Project Ideas

Below is a list of ideas that is intended to inspire your own project ideas, not define them!

1. (Chuck) Baseball season is almost over, so it's time to create a schedule for next season! Model the season as a CSP incorporating the conference/division requirements for teams. Try to be as realistic as possible: teams typically play each other for a 3-4 game series in a row, certain holidays do or don't have games. Bonus: try to optimize rest/traveling for players (having a variable n and m for n games in m days and a soft constraint to minimize this, maybe also keep track of distance traveled for a team and have a soft constraint to minimize that as well).
2. (Chuck) Uber Satisfaction Problem - given a set of drivers and passengers spread out in various locations/desired destinations, optimize the routes for the drivers. Try to implement a pooling mechanism to save the passengers money!

3. (Chuck) An extension of MDP's is the case where the current state is not completely observable: partially observable markov decision processes. There are variations of value iteration for POMDP's and it might be cool to try to modify the given pacman framework to run in a more "realistic" environment and make comparisons to the full information case.

4. (Ankit) It's election season, which means that political rhetoric is as heated as ever. Explore relationships between different congresspeople using the models from this course. Consider using clustering algorithms to find congresspeople with similar voting patterns. Or Hidden Markov Models to analyze their speeches or bills.

5. (Ankit) Use the OpenAI Gym, a python-based platform for reinforcement learning tasks, to build an RL algorithm to solve a popular Atari game (`https://gym.openai.com/envs#atari`). Want to reimplement AlphaGo but don't have a million dollars worth of GPU compute lying around? Try your hand at OpenAI's smaller 9x9 Go board with reinforcement learning: `https://gym.openai.com/envs/Go9x9-v0`. If you are feeling ambitious, considering using deep reinforcement learning with a neural network library like Torch or TensorFlow.

6. (Aman) Who doesn't want to make money? Take data from equity markets and use reinforcement learning to construct an agent that can predict with some accuracy whether or not you should buy or sell a stock.

7. (Aman) Create an agent that is a competent "general game player." Check out: `http://www.gvgai.net`. Other game ideas might include Utimate Tic Tac Toe or games from OpenAI. Make sure if you pick a game that it is sufficiently challenging to constitute a final project.

8. (Matt) Pick your favorite board game or card game and model it as a formalized game. Then, build a few agents to play the game using different game-playing algorithms. Modeling a board game or card game can introduce cool extensions to the minimax game-playing we saw in class, such as imperfect or incomplete information, multiplayer games, and combinatorial games. Additional research can be done on heuristic-based game-playing algorithms like Monte Carlo Tree Search. You can also compare reinforcement learning techniques to game-playing algorithms like minimax.

9. (Matt) There are lots of AI applications to sports. For example, model baseball, basketball, and/or football scheduling as constraint satisfaction problems; as Chuck mentioned, these schedules have certain constraints regarding in-division and in-conference games, and you can have soft constraints for distance traveled, number of games on no rest, and having better games in primetime slots (such as Monday Night Football or Sunday Night Baseball). Another example is modeling fantasy football drafting as an MDP, or selecting a daily fantasy sports lineup as a search problem. The vast amount of data in sports makes finding applications of course material not too difficult; for example, HMMs can be used to track the underlying state of a sports game as play-by-play data is observed.

10. (Qi) Track one or more than one objects in a video. You may build a Hidden Markov Model to do it. There are many off-the-shelf datasets for tracking, e.g., the motchallenge dataset. They usually provide you noisy observations of objects in each video frame, and your job is to find the motion trajectory of those objects as accurate as possible across frames.

11. (Shai) Write an AI player for Hive - representing the state will involve some thought, but once you've done that there is lots of strategy for your algorithms to discover and a manageable branching factor

12. (Shai) Map the connectome - turn imaged slices of a brain into a 3-D model by optimally matching the cross-sections of neurons. Note: this is a very challenging open research problem, so you will likely have to scale it back to something achievable.

13. (Charles) Write a program that learns how to write/speak in the style of a certain person (i.e. Shakespeare, Yoda, Wikipedia, Trump) from a corpus of text by building a statistical language model.

14. (Charles) Write a program that recommends news articles to read. Some potential directions this could go: article content tagging, machine learning recommendations, topic clustering, etc.

15. (Anna) Write a CSP solving program that can decompose sentences into their component parts of speech.

16. (Anna) Develop software that intelligently identifies trolls in online forums (reddit, Yahoo Answers, etc.)

17. (Scott) Deep-Pacman: Think back to the first time you played Pacman. You looked at your TV screen started smashing buttons and, in a very short amount of time, figured out what the goal of the game was, what are the relevant state features, and how your actions changed the game state. In this project, you will build an RL agent that only has access to only pacman board pixels (no hand-crafted MDP state or features) and uses Deep Q-Learning come up with good policy.

18. (Scott) Realtime bike routes: Often following Google maps for bike directions in Boston takes you down undesirable (bumpy, narrow, dangerous, etc.) roadways. Design a bike directions program for the Boston area that quickly returns the best bike routes between two points by taking into account information that can be mined from the web (e.g., bike accident frequency, road construction, dedicated bike paths, etc.). Users should be able to adjust their preferences to tradeoff speed with comfort/safety.

19. (Scott) Learning to ride a bike: Download or implement a simulator for a bicycle model, define an MDP representation, and develop a RL algorithm that learns to balance and steer the bicycle.