CSE 571 Fall 2022 Team Project Description

General guideline (please read carefully)

- Team projects are open-minded—you have a lot more freedom than individual projects here; at the same time, this means more researching: more reading, exploration, potential for errors, and debugging of course. Deadlines is tentatively set at **Dec 5**. You have about 1.5 months to complete the project.
- **Team registration step 1 is due Oct 14.**
- **You will receive your final project team assignment after step 1.**
- **Each team must select a leader by Oct 18** (I will leave some time after class for the team to meet with each other on Oct 18). Leader is responsible for organizing meetings, coordinate team effort, monitor progress to meet schedules, and handle team project submission. ***Only the leader should submit.***
- **Each team must draft a 1-page plan by Oct 21** (must submit on Canvas), listing the following: tentative project milestones, task assignments, meeting schedule. **Failing to submit a plan by the deadline may result in a penalty** (of no more than 5pt from the team score).
- Each team is suggested to meet (remotely or in-person) once in a week and team members (other than the leader) take turns as the scribe for the meetings. You are encouraged to use Google doc to collaborate on the notes.
- Meeting notes **must contain the following**: scribe name, meeting date, attendance, status update from each team member, and may *optionally* include planned activities for the next week, issues with meeting the milestones, plans to address the issues, or any other things discussed.
- The scribe must share the notes with other team members.
- **Notes must be submitted along with the project submission.** Notes are expected to be clear and consistent with the team effectiveness report. Inconsistencies will be checked and **may in a penalty** (of no more than 5pt from the team score).

- You may consult online resources but do not copy (which is less fun and can lead to plagiarism). ***Plagiarism will be checked***. If you are caught, you team project will be 0.
- For your report, use the IEEE transactions template (Choose: Template Selector->Transactions, Journals and Letters->IEEE Transactions on Robotics) [https://journals.ieeeauthorcenter.ieee.org/create-your-ieee-journal-article/authoring-tools-and-templates/ieee-article-templates/templates-for-transactions]
- Report should not exceed 4 pages, *not* counting references. Recommended organization: 1) abstract and introduction: *0.75* page; 2) Technical approach: *1* page; 3) Results, analyses and discussions: *2* pages; 4) Conclusions and discussions: *0.25* page.
- Each team must select a topic from the 5 listed below.

- Each team must also submit a "***team effectiveness report***" in pdf format, with a description ***and a percentage*** of contribution from each team member (please include names). *Every teammate must **SIGN** the team effectiveness report.* Electronic signatures are fine. This report must be included in your submission package, along with the project report, code repository, team meeting nodes, and a README file. ***Only the leader should submit.***
- Grading will be based on 1) Correct implementation (40); 2) Completeness and thoroughness of the results and their analyses (40); 3) Clarify of the report (20).
- ***Grading***: Assume that you have N students on your team. Each team will receive a team score X in [0-100] for the project. Each student will also have a contribution score (in percentage) from your team effectiveness report W in [0-100%] The final score of a student is computed as
  - $\min(100, \min(X * 1.1, X * (W * N)^{1/6}))$ if W >= 1/N
  - X * W * N else

The scoring scheme encourages collaboration and ***equal contribution*** to the project; we *slightly* award students that contribute more *while substantially* penalize students that contribute less. For example, assuming a project team of 4 students, a student that contributes *30% **more** than average* will receive a 4.5% bonus (with bonus capped at 10% of the team score); on the other hand, a student that contributes *30% **less** than average* will receive a 30% penalty.

The instructor reserves the rights to update the grading scheme if necessary.

Project topic 1. Bi-directional search

a) Implement bi-directional search describe in the following paper: "*Bidirectional Search That Is Guaranteed to Meet in the Middle*", Robert C. Holte, Ariel Felner, Guni Sharon, Nathan R. Sturtevant, AAAI 2016, and ***integrate*** into the Pacman domain for path-finding problems (from a start to a goal location) in your individual project 1 (http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/download/12320/12109)

b) Compare the algorithm's performance with other search methods (e.g., BFS, DFS and A*) in terms of their search behaviors like what we have seen in class below (e.g., which nodes are expanded first and last):

Greedy Uniform Cost A*

c) Also, compare them in environments *of **different tasks, sizes and complexities***. Provide ***statistical analyses*** *such as using student's T-test or ANOVA test. Choose the test to run and explain why you choose the test or a specific configuration of the test*

*(e.g., t-test could be paired or unpaired) f*or your comparison results with different tasks (i.e., start and goal position pairs), sizes, and complexities. You will need to create a script for creating different environments. You will need to collect results from many different runs for the different approaches and choose a test to ***meaningfully*** compare them. Explain and analyze your comparisons clearly (e.g., why you perform the comparisons and what the conclusions are for each comparison).

d) Submit a written report with your findings, which should include *at least the following*: 0. Abstract; 1. Introduction (motivation and your achievements in this project); 2) Technical approach (a brief technical discussion of the bi-direction search method you implemented); 3) Results (***Results that CLEARLY illustrate the strengths of your approach compared to the others in a statistically meaningful way***, for example, you could compare the number of nodes expanded, computational time, etc.). 4) Conclusions (any observations and discussions).
e) Submit project report along with the code repository ***and instructions*** (as a README file) to run it. Coding comments are always welcome.

Project topic 2. Life-long planning

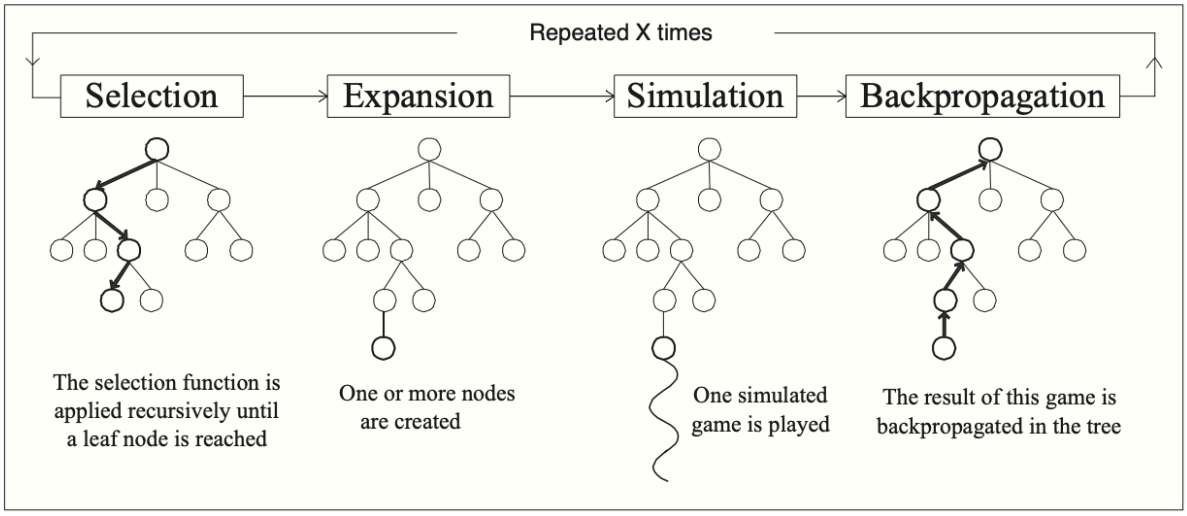
a) Implement lifelong A* search (D* lite) described in the following paper: "*D* Lite*", Sven Koenig and Maxim Likhachev, AAAI 2002, and ***integrate*** into the Pacman domain for path-finding problems (from a start to a goal location) in your individual project 1 (http://www.aaai.org/Papers/AAAI/2002/AAAI02-072.pdf). 1) Assume that Pacman only knows about the size of the (i.e., a M X N grid-world) environment initially, and can ***only observe local environment*** surrounding itself (you must define a local observation range as a variable, e.g., 3 blocks from the current pacman location). 2) Also, assume that the Pacman always knows where it is in the environment (i.e., it can localize). 3) Once the Pacman observes something, it is able to ***keep it in its mind*** (i.e., it maintains the knowledge that there is an obstacle in a given location once that is observed). **You should maintain such a live-map as a structure on which planning is based. Every time new obstacles are detected, the live-map may be updated and plan changed. Once a new plan is made, the pacman will execute the plan until a new obstacle is observed that blocks the path and replanning will be triggered (via D* lite).**
b) Analyze D* in terms of its search behavior, similar to what we discussed in class as follows, except that showing ***the order of nodes expanded at different replanning steps.***

Greedy Uniform Cost A*

c) Also, compare D* with other search baseline methods (e.g., BFS, DFS and A*) in environments *of **different tasks, sizes and complexities***. Assume that replanning will be handled as a separate planning task for the baselines based on the live-map. Provide ***statistical analyses*** *such as using student's T-test or ANOVA test. Choose the test to run and explain why you choose the test or a specific configuration of the test (e.g., t-test could be paired or unpaired) f*or your comparison results with different tasks (i.e., start and goal position pairs), sizes, and complexities. You will need to create a script for creating different environments. You will need to collect results from many different runs for the different approaches and choose a test to ***meaningfully*** compare them. Explain and analyze your comparisons clearly (e.g., why you perform the comparisons and what the conclusions are for each comparison).
d) Submit a written report with your findings, which should include: 0. Abstract; 1. Introduction (motivation and your achievements in this project); 2) Technical approach (a brief technical discussion of the life-long planning method you implemented); 3) Results (***Results that CLEARLY illustrate the strengths of your approach compared to the others in a statistically meaningful way***, for example, you could compare the number of nodes expanded, computational time, etc.). 4) Conclusions (any observations and discussions).
e) Submit project report along with the code repository ***and instructions*** (as a README file) to run it. Coding comments are always welcome.

Project topic 3: Games

a) Implement Monte-Carlo tree search as described in this paper: https://www.aaai.org/Papers/AIIDE/2008/AIIDE08-036.pdf for our Pacman Project 2



b) Compare the performance of Monte-Carlo tree search with our project 2 solution that uses evaluation function in environments *of* ***different sizes, complexities and difficulties (e.g., varying number of ghosts)***. Implement different selection functions with different exploration and exploitation strategies and re-evaluate. Provide ***statistical analyses*** *such as using student's T-test or ANOVA test. Choose the test to run and explain why you choose the test or a specific configuration of the test (e.g., t-test could be paired or unpaired) f*or your comparison results with different sizes, complexities and difficulties. You will need to create a script for creating different environments. You will need to collect results from many different runs for the different approaches and choose a test to ***meaningfully*** compare them. Explain and analyze your comparisons clearly (e.g., why you perform the comparisons and what the conclusions are for each comparison).

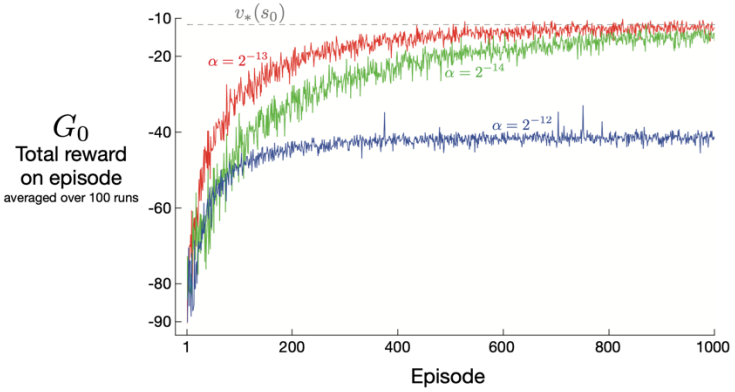
c) Submit a written report with your findings, which should include: 0. Abstract; 1. Introduction (motivation and your achievements in this project); 2) Technical approach that discusses details of your approach (e.g., how you choose exploration and exploitation in MC tree search; any heuristics you used); 3) Results (***Results that CLEARLY support your conclusions in a statistically meaningful way***. For example, you could compare the computational time, winning rate, etc.). 4) Conclusions (any observations and discussions).

d) Submit project report along with the code repository ***and instructions*** (as a README file) to run it. Coding comments are always welcome.

Project topic 4. Reinforcement Learning agent

a) ***Implement True online Sarsa(λ) (page 300, Chapter 12) with linear function approximation from Reinforcement Learning: An Introduction, 2nd by Richard Sutton (http://incompleteideas.net/book/RLbook2020.pdf)*** to control your agent in the Pacman domain. This project is based on our project 4 (to be released soon) on Reinforcement learning.

e) ***Compare the performance of your agent with the Q-learning agent (with linear function approximation) in project 4. You must compare their converging behavior for a given environment (see an illustration below). Furthermore,*** run comparisons in different environments with different ***sizes and complexities***. Provide ***statistical analyses*** *using student's T-test or ANOVA test. Run the test and explain why you choose the test or a specific configuration of the test (e.g., t-test could be paired or unpaired) f*or your comparison results with different sizes and complexities. You will need to create a script for creating different environments. You will need to collect results from many different runs for the different approaches and choose a test to ***meaningfully*** compare them in terms of convergence speed (i.e., how fast it converges to the optimal policy) in different environments. Explain and analyze your comparisons clearly (e.g., why you perform the comparisons and what the conclusions are for each comparison).



b) Submit a written report with your findings, which should include: 0. Abstract; 1. Introduction (motivation and your achievements in this project); 2) Technical approach (a brief technical discussion of ***True online Sarsa(λ);*** 3) Results (***Results that CLEARLY support your conclusions in a statistically meaningful way***.). 4) Conclusions (any observations and discussions).

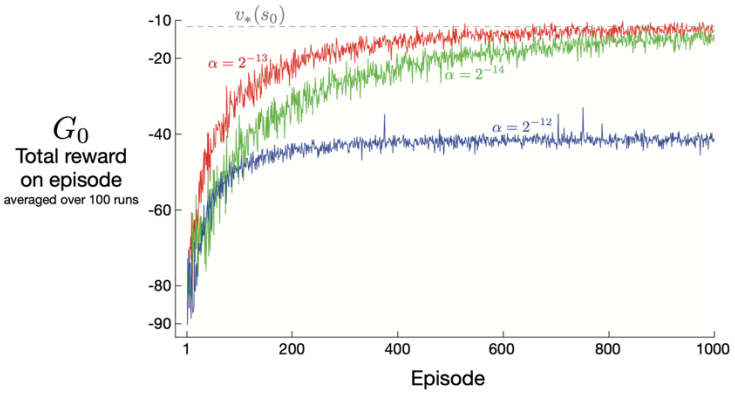
c) Submit project report along with the code repository ***and instructions*** (as a README file) to run it. Coding comments are always welcome.

Project topic 5. Reinforcement Learning agent

d) ***Implement REINFORCE with baseline (page 330, Chapter 13) with linear function approximation from Reinforcement Learning: An Introduction, 2nd by Richard Sutton (http://incompleteideas.net/book/RLbook2020.pdf)*** to control your agent in the Pacman domain. This project is based on our project 4 (to be released soon) on Reinforcement learning.

f) ***Compare the performance of your agent with the Q-learning agent (with linear function approximation) in project 4. You must compare their converging behavior for a given environment (see an illustration below). Furthermore,*** run comparisons in different environments with different ***sizes and complexities***. Provide ***statistical analyses*** *using student's T-test or ANOVA test. Run the test and explain why you*

*choose the test or a specific configuration of the test (e.g., t-test could be paired or unpaired) f*or your comparison results with different sizes and complexities. You will need to create a script for creating different environments. You will need to collect results from many different runs for the different approaches and choose a test to ***meaningfully*** compare them in terms of convergence speed (i.e., how fast it converges to the optimal policy) in different environments. Explain and analyze your comparisons clearly (e.g., why you perform the comparisons and what the conclusions are for each comparison).



e) Submit a written report with your findings, which should include: 0. Abstract; 1. Introduction (motivation and your achievements in this project); 2) Technical approach (a brief technical discussion of ***REINFORCE with baseline;*** 3) Results (***Results that CLEARLY support your conclusions in a statistically meaningful way***.). 4) Conclusions (any observations and discussions).

f) Submit project report along with the code repository ***and instructions*** (as a README file) to run it. Coding comments are always welcome.