Project: Tactical Path Planning for Reconnaissance

### ECS170, Spring 2018

# Group Members

Please place all group member names, email addresses, and student IDs.

Jiehong Jiang, [jjhjiang@ucdavis.edu](mailto:jjhjiang@ucdavis.edu), 914974552

# Problem and Contribution Statement

In this section, you will describe the problem you want to solve. You should answer the following questions:

We are going to predict the army formation given what’s seen, including the amount of the construction units, buildings, and current army, by scouting. We call it dynamic army prediction.

* Why is this problem important to solve?

Winning if the ultimate goal of each Starcraft match. To win the game, players need to choose their own strategies, which can vary throughout the entire match. If the player can correctly predict what’s coming for him(the opponents’ army), he/she can make an better decision of army formation to fight against.

* Does there already exist a solution to this problem?

As far as I know, there are some predictions of strategies out there like but none is specifically aiming for dynamic army prediction. Here is a list of sc2 AI (https://github.com/SKTBrain/awesome-starcraftAI)

* Why is this problem technically interesting?

The algorithm can be ultimately utilized by other gameplay bots to make prediction in real game player. It should also be easily trained by different replays with different factions, because the observation of features should only base on the general concept of SC rather than, for example, knowing the tech tree of one faction first.

* What possible AI approaches are there to this problem?

Machine Learning method, Decision Tree, should be a good starting point of the research.

* What is your solution and contribution to the solution of this problem?
* Why did you choose your solution given the possible alternatives?

# Design and Technical Approach

Here is where you detail your solution. Here are some items to address:

* What AI techniques are you proposing?
* How do the techniques address the problem? Connect the levels of abstraction between problems space and technical solution. Architectural diagrams help.
* What is your technology stack?
* Why is your technology stack the appropriate choice?
* What programming environments are you using?
* How will you use GitHub?
* What code quality assurance tools are you using?

# Scope and Timeline

Provide a justification for the feasibility and scope of your project. Provide a timeline covering the major deliverables and technical problems.

# Documentation and Access

How are you sharing code? What about a readme? A website is recommended. A Github page or project is recommended.

# Evaluation

Describe your criteria for success. How will you gather data to evaluate your solution? How will it be analyzed?

# Plan for Deliverables

How will you share your GitHub repo? Will you submit your code to tournaments or write a conference paper?

# Separation of Tasks for Team

Environment Setup and Level Creation: Elyhaym & Rydia

Basic A\* implementation: Nasir & Rydia

Hierarchical A\*: Nasir & Yang

IDA\*: Elyhaym

Testing Harness: Elyhaym & Nasir

Evaluation Masters: Cid & Nasir

Inline Documentation: Rydia & Cecil

Deliverable Reports: Elyhaym