The purpose of this research project is to implement semi-supervised learning to the game of CTF (capture-the-flag). Current research (http://kghost.de/cig_proc/full/paper_52.pdf, Ivanovic et al., 2014) (https://www.aaai.org/Papers/AIIDE/2008/AIIDE08-030.pdf, Hefny et al., 2008) has only applied reinforcement learning to the problem, with a heavily structured “reward” system that must be clearly defined as an input beforehand. My goal is to push towards a more semi-supervised approach, with the eventual goal being to make a good CTF team with a reward system consisting of 1 for a win and -1 for a loss. According to the Ivanovic paper, generalist strategies such as my proposed one have not done very well, so there is much to improve upon.

All simulations would be done on AISandbox, based on a Capture the Flag mod downloaded from AISandbox.com, which supports a wide variety of strategies, such as choosing how many units can be controlled as a “team”. I will start with a simple game of CTF, with the goal being to bring the opponent’s flag (at a known location) back to your base. Tagging an opponent holding your flag returns the flag to its original location. I will vary the environments (a variable that Ivanovic believes is necessary to prove the efficacy of a generalist algorithm), and eventually may add more rules, such as having a “jail” for tagged players. Also, my logic will be turn-based, with a recalculation after one step.

First, I will explore (and currently am exploring) the task of programming a learner for a one-on-one battle against an opponent of differing high-level strategy. The opponents would probably be fitted with similar paradigms to those in the Hefny paper (for instance, one offensive opponent, one defensive opponent, one probing opponent, one opponent who has a mix of all three, etc.) If all works out, I expand to two-on-two, where there are added elements of seeing what two other players do.
My approach would differ in that I won’t have a rigid “reward system” for the AI to learn off. Instead, during the training phase, I will let the AI figure out the weights for events for itself, with all values starting at 0. There are multiple methods with which to accomplish this.

First, I feel that even though I won’t be putting priors on events such as tagging an opponent, a proper algorithm will still “notice” these events and take them into account. The events I am taking into account for 1 on 1 (besides winning and losing) are: gaining, losing and keeping sight of an opponent, capturing an opponent’s flag, running into a wall, running in a circle (have I been in this position and orientation in the last 12 or so moves?), seeing an opponent capture a flag, moving a step towards an opponent’s flag, “juking” an opponent, and getting tagged.

My goal is to have my program develop, by itself, two levels of weights. The first level is assigning each of the above events a weight, with a higher weight meaning a higher chance of victory. This is what constitutes “high-level strategy”, as the learner will strive towards the higher-weighted events. The second level is weighting each possible move (up, down, left, right) to an estimated probability of getting higher first-level weights.

My program will likely have 3 levels, similar to those in Hefny’s paper; a high-level strategy recalculated at each turn (in this case, trying to get the highest first-level weight), a randomized pathing level (ridge regression or generative models are very promising here), and a low level that tells the controller which way to go.

A problem arises when considering the first levels of training. Since our robots will have no strategy whatsoever, it will have an infinitesimal chance of beating a well-programmed AI, and draws (when the maximum amount of turns is reached) will likely be the fault of the AI rather than the doing of the learner. Despite this latter note, I believe that a logistic-regression style algorithm has promise.
My general idea is as follows: when an event happens, the learner evaluates the previous events and moves to update the second-level weights (does tagging an opponent lead to a good event?). At the end of the game, the algorithm updates the first-level weights by performing a modified logistic regression with the events as an input (with and output of 0 being losing in the minimum number of moves, where drawing the game out leads to a output close to 0.5). After this, the weights are normalized to have a mean of 0, and the next game is played.

**Goal**

I’m not sure if such a slow learner could ever play up to a human-programmed AI, which is why I’ll test the learner (and hopefully learners) against a variety of opponents. I think the most promising aspect would be to see which strategies the learner has developed after many games. If I’m getting wonky results, maybe I can adjust my paradigm to include more events, or move away from the linear weights, or perhaps put more emphasis on combinations of events.

**“Deliverables”**

I would be willing to meet with you or submit written reports every week, and talk about how my project has gone. I should have some simulations done halfway through the term, so that could be a good sign of progress. Also, once I figure out how well the semi-supervised learning is doing, I will have a clearer picture of what observations to “pull out” of this project. Finally, I’ll have the entire report done by the end of reading period.