Brandon Percin, Darla Drenckhahn

CSC 440 – Intro To Artificial Intelligence

Final Project Proposal

***Project Description***

For our final project, we will be solving Minesweeper. Minesweeper is a single player puzzle (typically) computer game. The goal of Minesweeper is to clear a rectangular board containing hidden bombs (“mines”) without making any of the bombs go off. This is done with clues about the number of neighboring mines in each field. The problem of solving this is proven to be NP-Complete. Minesweeper is like Sudoku because your success can be greatly dependent on being able to eliminate possible answers until only one remains. The difference comes in the form of how information is given, where not enough information is given to solve the game from just its initial state: the program must interact with the game after choosing a candidate position by clicking on a square in order to obtain more information, almost like a combination of an online search and adversarial search.

***Steps to Investigate Problem***

One thing we need to decide on is what difficulty levels we can implement (if not all of them). There are 3 general difficulty levels for Minesweeper which include beginner, intermediate and expert. The beginner level has 10 mines and is either 8x8, 9x9 or 10x10. So beginner only deals with square playing fields. Intermediate has 40 mines and could be 13x15 or 16x16. Expert has 99 mines and is always 16x30. We also could have an option to set custom game parameters (number of mines, grid).

Because solving Minesweeper is a difficult problem, we would like to attempt to solve Minesweeper both as a constraint-satisfaction problem and as an adversarial search problem in order to determine which is more efficient of a solution. This is important because "score" in Minesweeper is determined solely by the amount of time it takes to clear all non-mine squares from the grid. The largest issue comes from the fact that there is a random element to Minesweeper, and in some cases, it is impossible to know for sure which position of two a mine is in, meaning that even a perfect solver can fail under the right circumstances. Another issue comes from the online-search-like elements of the game, where by a tile must be clicked on in order to obtain more information about the board that is required to solve the problem. To combat this, the goal of the searches will be to determine positions that are definitely not mines, after which the program will click on each of these locations, take another screenshot of the game, and restart the search with a new initial state (with more information than the last time). The game is solved when all spaces that are not mines are clicked on.

***Team Dynamic***

Team members include Brandon Percin (section 001) and Darla Drenckhahn (section 801). Since one of us is an online student and the other is in class – we will need to be diligent with our communication to ensure we complete the project on time and adequately. As it is harder to communicate with the professor as an online student with questions – if we are stuck on anything and need input, Brandon will probably take charge in asking the instructor after class one day or whenever most convenient.

**Team Roles**:

* **Darla**:
  + code for minesweeper solver/project write up
  + Planning formal write-ups
  + Primarily focusing on solving as a CSP
* **Brandon**:
  + code for minesweeper solver/project write up
  + Converting screenshots of the game to initial states
  + Primarily focusing on solving as an Adversarial Search

***Steps to Complete Project***

Steps to completing the project include:

1. Figure out how we are taking in input of the minesweeper grids, decide what levels of difficulty we want to support (all of them?) and if we want to support custom game parameters. Create example grids that we can test on.
   1. *Date to be completed:* Monday November 11th
2. Develop the minesweeper class that can utilize both search strategies we plan to implement. Develop programs to the point where tiles that definitely are not bombs can be returned, given an initial state.
   1. *Date to be completed*: Monday November 25th
3. Implement heuristics which could help decide what cell to look at next and start comparing heuristics/algorithms on the run time and performance of the solver.
   1. *Date to be completed*: Monday December 9th
4. Do any remaining work and clean up and write final project report based on findings.
   1. *Date to be completed:* Tuesday December 17th