Comp 424 - Project Specifications

Project TA: Clement Gehring
Instructor: Joelle Pineau

Code and Report due: April 10th 11:59 pm

Goal

The main goal of the project for this course is to give you a chance to play around with some of the AI algorithms discussed in class, in the context of a fun, large problem. This year, we will be working on a team-based (2 vs 2) alternative of a game called Halma.

Clement Gehring is the TA in charge of the project and should be the first contact about any bugs in the provided code. General questions should be posted in the project section in mycouses.

Rules

In this game, you are in control of 13 pieces. Each player starts at the corner of a 16x16 grid. With the help of your ally occupying the opposite corner, you must send all your pieces to his corner. Your ally must also do the same. The first team to completely switch places with his teammate wins.

During your turn, you can move one of your pieces to a neighbouring square in any direction (up, down, left, right, and diagonally) if it is free. Alternatively, you are allowed to hop one piece over another adjacent piece. Similarly to checkers, you can consecutively hop the same piece over as many pieces as you want (as long as you land on a free square and you only hop over one piece at a time).

There are four players in the game: player 0, 1, 2, and 3. Player 0 and 3 are allies and so are player 1 and 2. Player 0 starts the game then player 1, then player 2, and so on.

After 100 turns, you are not allowed to have a piece in any goal zone but your target zone. Failure to do so will cause your team to lose. No more than 5000 turns are allowed, after which the game is considered a draw.

The rules for piece movements and goal zones are the same as the original four player game. They can be found at `http://en.wikipedia.org/wiki/Halma`.

Implementation

Competition Constraints

We will hold a competition between all the programs submitted by students in the class. The top 5 players in this competition will receive bonus points. Even if you are not a top player, you could receive bonus points if your approach is interesting and attempts to combine different AI techniques.

During the competition, your player will be given no more than 1 second per move and no more than 10 seconds to start. Your player will run in its own process and will not be allowed

to exceed 500 mb of ram. The code submission should not be more than 10 mb in size. Going over any of these constraints is considered an automatic loss.

You are free to implement any method you wish as long as your program runs within the constraints and is well documented in both the write up and the code. Commenting code is an important part of coding so we expect well commented code. All implementation must be your own. You are **not** allowed to use non-standard libraries. If you are unsure if a particular library is allowed, feel free to ask on mycourses where the project TA will clarify.

Submission Constraints

We provide code for the board game logic as well as an interface for running and testing your player. You are only required to extend the 'boardgame.Player' class.

We will enforce the following naming/structure constraints:

1. The player class you want us to test **must** be named "XXXXXXXXXPlayer", where XXXXXXXXX is your student number.

2. The player class you want us to test **must** have a default constructor which calls

   `super("XXXXXXXXX");`

   , where XXXXXXXXX is your student number.

3. All your classes **must** be in the same parent package which must be named "sXXXXXXXXX", where XXXXXXXXX is your student number.

   Sub-packages are allowed (e.g., "sXXXXXXXXX.mytools").

4. The player class you want us to test **must** be in the parent package "sXXXXXXXXX".

5. You must submit your source code with the package structure intact.

6. **Do not include the provided code**.

You are free to reuse any provided code in your code (must be documented). We plan on running several thousands games and we cannot afford to change any of your submissions. Any deviations from these requirements will have you disqualified, resulting in part marks. **An example submission is provided.**

You are expected to submit working code to receive a passing grade. If your code does not compile or throws an exception, it is not considered as working code. If your code runs on the trottier machines with the (unaltered) provided code, your code will run without issues in the competition. The provided code includes a README file on how to start a game.

We are willing to allow other programming languages but we won't support them. It will be up to you to write the socket based message passing to communicate with the server. You will have to provide a shell script to launch your player as well as a make file. It will need to compile/run on the trottier machines. If you chose not to use java, please let the project TA know and include a detailed README file explaining how to run your shell script. For non-java languages, which libraries are allowed will be determined on a case by case basis in order to avoid unfair advantages.

Write up

You are required to write a report with a detailed explanation of your approach and reasoning. The report must be typed and grammatically sound. **You must submit a .pdf file.** The minimum suggested length is 4 pages and should be no more than 5 pages (300 words per

page), but the most important part is to produce a report that is clear and concise. The report must include the following required components:

1. An explanation of how your program works, and a motivation for your approach.
2. A brief description of the theoretical basis of the approach (about a half-page, in most cases); references to the text of other documents, such as the textbook, are appropriate but not absolutely necessary. If you use algorithms from other sources, briefly describe the algorithm and give a citation to your source.
3. A summary of the advantages and disadvantages of your approach, expected failure modes, or weaknesses of your program.
4. If you tried other approaches during the course of the project, summarize them briefly and discuss how they compared to your final approach.
5. A brief description (max. half page) of how you would go about improving your player (e.g. by introducing other AI techniques, changing internal representation etc.).

Academic integrity

This is an individual project. The exchange of ideas regarding the game is encouraged, but sharing of code and reports is forbidden and will be treated as cheating. We will be using document and code comparison tools to verify that the submitted materials are the work of the author only. Please see the syllabus and `www.mcgill.ca/integrity` for more information.