

Undergraduate Final Project

Due: May 8, 2003

1 The Awari Game

In this project, you will write a game-playing program for the two player, turn-based, zero sum, perfect information game Awari. Awari is an ancient counting game, of which there are a myriad of variants. Some of you may have encountered the game Mancala, which is a more complex variant along the same lines. Note that Awari *has* been solved exactly, albeit by use of more intensive computational resources than most of you have available (51 hours of CPU time on a distributed supercomputer with 72 1GHz PIII nodes, each with 1Gb RAM). The official word is that optimal play by both players in this game results in a draw; brownie points to you if your program can always bring the online Awari player applet to a draw (when set to level “Perfect!”, of course):

<http://awari.cs.vu.nl/>

2 The Awari Rules

There are a number of variant rule sets available on the web. To ensure that we’re all playing by the *same* set of rules, please use those at

<http://awari.cs.vu.nl/rules.html>

That rule set should be the same as those used by the Awari applet on the awari.cs.vu.nl.

For a nice example game demonstrating the Awari play, please look at:

<http://www.cs.ualberta.ca/~awari/rules.html>

This is played under a slightly different ruleset than we’ll be using, though it should give the right idea. (The two different rulesets are very close, but differ in some minor details – e.g., whether a single repetition of board state or a triple repetition is required to end the game).

If you have any questions on the rules, please forward them to the class mail list early, so that we can be sure that everybody is playing under the same rule set.

3 Your Program

Your task is to write a game-playing program for the Awari game. Your program will play one side of the game, while the other side should accept human input. You must provide a toggle at the beginning of the game to set whether the computer plays North or South (i.e., whether the computer or the human plays first). You need not provide a spiffy GUI interface to it (though you’re welcome to if you so desire), but you must provide some method for (a) accepting human moves, (b) reporting computer moves, (c) reporting the current board state, and (d) reporting the current score. If you choose to use an alpha-beta or other game-search method, you should also report on each turn the number of nodes that your program examined and the computer’s current estimate of whether it is winning or losing and by how much. If you’re not using alpha-beta, the program should report whatever criterion it is using to select moves.

Important: You *must* also provide a parameter, set at the beginning of the game, that fixes how much time the computer is allowed to take on each of its moves. The computer *must* not take more

time than that to make any of its moves (though it is free to, for example, plan while the human is making her or his move). If your program takes more than the allotted time to make a move, it is considered to have fouled and forfeits that game.

You may use whatever planning/strategizing method you desire in your program, but you must be capable of explaining what it is doing, how it is choosing to make moves, and why you developed it that way. Part of your grade will be based on a written report documenting your design decisions. (More on this below.)

Your program *must* stand on its own and it *must not* access the network during play, nor rely on somebody else's prebuilt Awari move database. Doing so both disqualifies your program from the tournament and will be considered academic dishonesty (i.e., cheating). You're free to implement any ideas that you find elsewhere, but you must implement them *yourself* and document them in your final report. If your program includes a learning component, you may *train* it by playing it against any other existing Awari programs you can find, though you are not allowed to simply *memorize* their move database.

4 The Tournament and Schedules

To determine whose program is smartest, we will hold a tournament. Your project grade will *not* be based on the outcome of the tournament, though the winner will receive a small number of bonus points. In addition, I reserve the right to assign bonus points to any program that I think is particularly elegant, clever, or creative.

The tournament will be held in two stages: an out of class round-robin tournament to determine the top 4 programs, and an in-class single-elimination championship to determine the final winner.

4.1 Round Robin Matches

In a round robin, each program will play one match against every other program. A match consists of two games: one in which your program plays North and one in which it plays South. Your program scores one point for each stone it captures in each game of the match. At the end of the round robin, the four programs with the most points (i.e., largest total number of stones captured across all games) will advance to the finals. The rules for the round robin are:

1. You are individually responsible for scheduling a time to meet your opponent. If one party makes every good effort to meet, but the other party refuses, doesn't appear at the agreed time/location, etc., the no-show party forfeits both games. The non-forfeiting player will receive 25 points per game (total of 50 for the match), and the forfeiting player receives 0.
2. If one program crashes, performs an illegal move, or takes more than its allotted time for a move during a competition game, that program forfeits that game. The non-forfeiting program receives 25 points for that game and the forfeiting program receives 0.
3. If either party requests it, you may have a third-party present to adjudicate disputes, ensure that both programs are playing by the rules, keep time, etc.

4. After the match, one party should report the results of the game on the class mailing list. Please report not just winner/loser, but also the number of stones captured by *each* program during the match, and the number of turns taken in each game (the number of turns doesn't figure into your score, but it will help me judge how much time to allow per move in games on the final day). Because the mail lists can be flaky here sometimes, please also CC: me and your opponent on such mails.
5. Each program must be set to a maximum turn length of 1 minute.
6. You may make modifications to your program (including changes in weights on evaluation functions or learning parameters) between *matches*, but not between *games* within a match. I.e., you can't change your program between games against a single opponent (even to fix a bug that caused your program to crash!). The program is free to modify *itself* (e.g., via learning) as play progresses, however.
7. All games must be completed and mailed to me/the class by no later than **noon, May 8**.
8. You may choose to play "practice" games against each other that don't count toward the round-robin scoring, but you *must* both decide on which pair of games will count toward the round robin *before* playing them.

As results come in, I will post current standings on the class web page at

http://www.cs.unm.edu/~terran/classes/cs427-s03/awari_robin.html

4.2 Final Elimination Round

On May 8, the last day of classes, you will turn in your program and program writeup at the beginning of class, and we will have an in-class final playoff between the top four programs from the round robin tournament. The winner will receive a small number of bonus points and the envy and adulation of her/his classmates. In addition, before each match, I will ask each player to give a short verbal description of the design of his/her program, including any novel elements, creative ideas, the design of the evaluation function, etc.

5 Deliverables and Grading

On May 8, you will hand in, by the beginning of class, a copy of your program code (possibly electronically) and a short (2-5 page) written description of your program design. You should document all significant design decisions, but especially describe the search strategy your program uses, its evaluation function, any learning component it employs, or any other elements that you think are creative or that you're particularly proud of. In addition, if you use any ideas from other sources (e.g., journal publications, web pages, etc.), please document them.

Your grade will be 30% on the written report (I *do* count off for grammar, spelling, or incoherence!) and 70% on producing a functioning, non-trivial¹ program that successfully plays through the tournament.

¹Your program must actually employ *some* significant form of strategy. A program that, e.g., only picks uniformly random moves or always makes the same move will receive 0 credit (unless you can produce a sound mathematical proof that that should lead to optimal play, and your program wins the tournament).