Keeping the Focus on Heuristics:

Network Game Agent for Teaching AI

R. Andrew Lamonica*, Xudong W. Yu‡, Jerry Weinberg†
Department of Computer Science, School of Engineering
Southern Illinois University Edwardsville
*rlamoni@siue.edu, ‡xyu@siue.edu, †jweinbe@siue.edu

Abstract

Game playing programming assignments can provide useful hands-on learning experiences for teaching search tree programming techniques, space efficient data representation, and heuristic evaluation functions. However, a number of issues arise with game playing assignments that are not relevant to the focus of learning the AI material. Among these are interface development and agent communication. This paper describes the implementation of a network game playing system that allows students to center their attention on the relevant AI topics. Student evaluations and test scores over the last three years indicate that the use of this game playing system has enhanced the learning of AI concepts.

Introduction

Students' experience in Artificial Intelligence courses can be enhanced by the use of computer projects, both in and out of the classroom (Walker 1994). Game playing programming assignments in an AI course can provide an interesting and motivating approach for hands-on learning of search tree programming techniques and developing heuristics (Kumar 1999). Though, with non-trivial games, students can become overly focused on the less relevant aspects of the assignment such as a reasonable interface or finding ways to test their game playing agents against other students' agents.

This paper describes the design and implementation of a networked game playing system that allows students to center their attention on developing and testing heuristic evaluation functions and debugging their search tree code. The system allows students to easily pit their developing agents against one another over a network connection, to re-play games to see how their agents evaluated a move, and to make changes to their agent in the middle of a game to see how a change in the heuristic function will alter the play. Furthermore, the system provides a convenient API so students do not have to concern themselves with writing interface code. This makes it easier for them to debug their own code. The current work targets 2-player, board games such as Reversi, Connect Four, and Kuba.

Background

In our introductory AI course, we begin with a formal description of how to use searches as a problem solving technique by reviewing several brute-force search methods such as depth-first and breadth-first search (Russle and Norvig 1995). The students are then introduced to heuristic search methods such as A* and best-first search, followed by discussions of game playing as a search problem.

To help students better understand and appreciate the concepts of searching, heuristics, and search-tree programming, a set of 3 programming projects are assigned. These assignments cover the specific topics of blind search, heuristic search, and game tree search.

In the first project, students are asked to apply a blind search method to solve a relatively simple puzzle such as 8-puzzle, or a limited version of more a complex puzzle such as Fore & Aft or peg game (Klutz 1996). By limiting the board sizes of these games, the search space is made small enough that an efficient implementation of the breadth-first search algorithm will find an optimal solution for any initial state.

In the second project, students are asked to implement a heuristic search method to solve an extension of the 1st puzzle, such as 15-puzzle, Fore & Aft with board sizes ranging from 13x13 to 39x39 and peg game with up to 64 pegs. The search space for these puzzles is so large (e.g., O(15!) for 15-puzzle) that a good heuristic is necessary for successful completion of the assignment. Class competitions are also held to recognize the program that either has the best solution (e.g., fewest steps in 15-puzzle) or works on the largest board (e.g., Fore & Aft).

The following sections discuss the third and most important project in this sequence, the game playing project.

The Game Playing Project

The Game

Students are assigned to write a program that plays a game by selecting the best move given a board state. The most recent offering of the AI class used a game similar to Milton Bradley's Connect Four. In this game,

players alternate selecting a column of the board in which to drop one of their markers. A newly dropped marker rests on top of the highest marker already in the column. If a column has a marker in its top position, then it is considered full and may not be selected by either player. The goal is to position four markers in a row vertically, horizontally, or diagonally. The game ends when one player reaches the goal or when all the columns are full.

Criteria

Students are given a program stub (shown in Figure 1) that receives the current state of the board and returns a move. Students are assigned to add code implementing a heuristic search method (typically alpha-beta) to select that move. The search must be completed within a given time limit. Making an illegal move or exceeding the move time limit results in disqualification. The interface monitors for violations of these two rules and reports them with error dialogs when they occur. A rudimentary agent is provided that the students' agents must be able to beat in order to qualify for the tournament. The tournament is held during class and individual games are projected on to a screen. Tournament winners are rewarded with bonus points on the assignment. In addition to submitting code, students are required to submit documentation describing what their heuristics do and why they think their heuristics will be effective.

```
main()
{
    Link4Serverl4Interface(nameServerAddress);
    l4Interface.setName("Smith");
    while(true)
    {
        l4Interface.waitForConnection();
        gameState =
            l4Interface.waitForMyTurn();
        if(gameState == BOARD_READY)
        {
            BoardState gameBoard =
                l4Interface.getCurrentBoard()
            moveToMake =
                findBestMove(gameBoard);
            l4Interface.sendMove(moveToMake)
        }
    }
}
```

Figure 1: Game Stub

Provided Materials

Students are provided with a Student Package containing the software and materials necessary to complete the assignment. The 2003 version of this package contained several parts. These parts included the Game Interface shown in Figure 2, a sample project called the Game Stub summarized in Figure 1, a compiled agent called NotRealBright, and some instructions. The interface is the same software that will be used in the tournament and is described in greater detail below. The Game Stub is a ready-to-compile Microsoft Visual Studio Project that performs all the steps necessary to connect with the interface through the included Link4Server library. The only difference between the Game Stub and the program that students are expected to hand in is that the stub's findBestMove function prompts its user to type a move rather than performing a heuristic search to find one. NotRealBright is the agent that the students' agents must beat to qualify for the tournament. NotRealBright considers only the next turn when deciding what move to make.

The System

Three pieces of software comprise the system the Game Interface, the Link4Server Library, and the Name Server. The Game Interface and the Link4Server Library are Included in the Student Package. The Game Interface provides a graphical method for students to play games against the agents they have written and to watch games played between two agents. The Link4Server Library provides an API for communicating between the Game Interface and a student agent. The Name Server program runs on a server and allows students using the Game Interface to find any agents that are waiting to play.

The Interface

The Game Interface (shown in Figure 2) is a client program that connects to one or two Link4Servers. Each Link4Servers provides a game-playing agent. After a student selects the agent or agents to participate in a game, he/she presses the start button and Player One's turn begins. If Player One is an agent, then it is sent a copy of the board so that it can select a move to make. Otherwise, the student makes a move by clicking on the game board. When the move is made it is displayed on the board and the next player's turn begins. When a player wins or there are no more legal moves remaining, the game stops and the result is displayed on the interface.

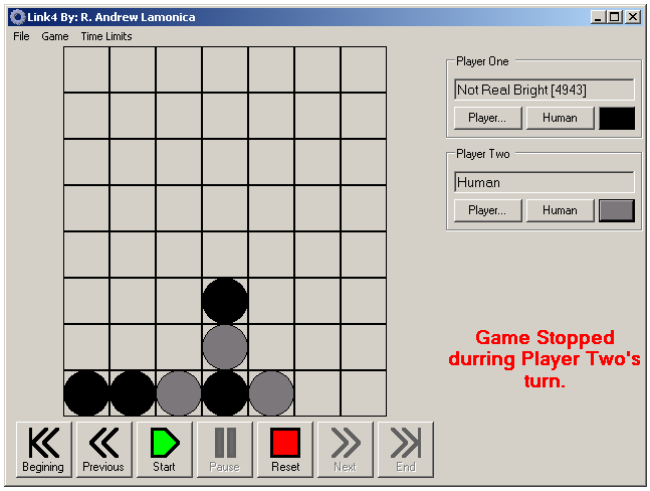

Figure 2: Game Interface

Name Server

The Name Server provides a list of waiting agents to Game Interface instances. Figure 3 shows what this list looks like in the Link4 Game Interface. The Link4Server library registers its agent's name with a Name Server when it starts and removes the name when it closes. When a user presses the "Player…" button to select a player in the Game Interface the interface asks the server to list the currently registered names and addresses. Students can manually type the address and port of an agent in the textbox at the bottom of the dialog, but it is easier to just point and click.

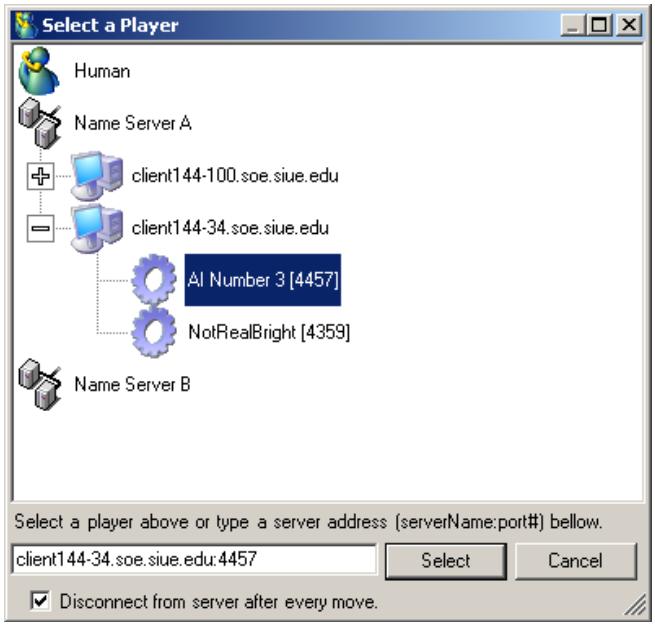

Figure 3: Player Selection Dialog

Link4Server Library

The Link4Server Library provides the connection functionality to an agent written by a student. This library is included in a stub project that is provided to students. The stub project calls the methods necessary to allow for Game Interface instances to make connections to this agent and ask it for a move. Students are only concerned with the findBestMove function in Figure 1, however they are allowed to make other changes if desired.

System Features

The system was designed to help students test and improve their Intelligent Agents (IA's) with ease and to handle problems unique to this type of programming assignment.

Robustness. One of the first design decisions made about this system was to keep the students code separate from the graphical user interface code. This solves a number of problems likely to come up when students are developing their agents and adds some convenience at the same time. To accomplish this separation the students are provided with a simplistic single threaded library that performs all communication with the multi-threaded event-driven Game Interface through sockets. With the separation in place, students can now debug, stop, or crash their agent software without serious consequences to the Game Interface. This decreases development time because students do not need to restart the Game Interface when their agent needs a correction and they do not need to worry about the complicated internals of the Game Interface when debugging.

Clearly, sockets communication provides the needed separation between Game Interface and agent but it has another obvious attraction as well. Using sockets to link Game Interface instances and student agents allows these agents to be running on different machines than the Game Interface. This fact adds two useful features. First, if two students want to test their agents against each other's they can do so without sending each other code or having to trust each other's binaries. All that is necessary is that both students have an Internet connection. The other feature is that when conducting the tournament the instructor does not need to worry about student agents interfering with one another because the two competing agents can be run on two different machines.

The separation of student agents onto different computers created the need for the Name Server, which allows students to easily connect their Game Interface instances to waiting agents. However, this addition could have caused crashed student agents to become a problem because a crashed agent would fail to un-register itself from the Name Server. To address this

concern, it was decided that Name Server entries should be set to expire after a few minutes if they are not explicitly renewed or deleted by the registering agent. This causes entries from stopped or failed agents to disappear from the list quickly and all the Link4Server library needs to do is periodically renew its entry while it waits for a connection. This Link4Server feature was added to the library in a manner transparent to the students.

Convenience. In order to maximize the time students spend working on the heuristic search part of their agents and minimize the time they spend learning project specific information, we added several convenience features to our system. We have already discussed how the Name Server makes connecting to agents quick and easy, but it is not the only graphical user interface feature added for convenience.

The Game Interface has a video-player like interface that gives a student the ability to browse through the game just played and see how his/her agent performed (see Figure 2). This can help find errors or weaknesses with the heuristic that would not be visible without close examination of the game as a whole. When the student's agent is stateless (as discussed in the next section) the student can use this interface for more than just reviewing a completed game. The video-player functionality can be used to make incremental improvements to heuristics easier. If the student finds a position in the game where his/her agent made a mistake with its next move the student can make appropriate corrections and then replay the game from just prior to the mistake. This joined with the ability to pause a playing game gives a student the option to undo any move made for any reason, including because the agent just won by catching an opponent's blunder.

In order to allow students the opportunity to catch a mistake made by their agent when playing against another agent, we added a minimum move time to the interface, in addition to the maximum move time used during the tournament. If a minimum move time is set and the currently running agent returns its move choice before that minimum time has elapsed, then the interface waits for the minimum time to pass before starting the next agent's turn. This has the effect of slowing down the game to a more easily observable speed. This is a handy feature to have during the tournament because students usually optimized their agents to run too fast rather than too slow, to avoid time limit violations.

Flexibility. Because the goal of this project is for students to learn to write heuristics, one of the design decisions made early was to try and keep students' agents stateless. For our purposes, a stateless agent is one that can be sent any board arrangement at any time and it will respond with what it thinks is the best move, given that arrangement. With stateless agents playing, the student has the option to interrupt a game and change a move, ask the agent to try the same move over again, or even have a different agent make the next move. These changes can be accomplished by pressing the pause button on the Game Interface then changing players by clicking on one of the two "Player…" buttons (Figure 2). We cannot guarantee that students will follow the stateless model; however, if they do testing becomes easier. To help students keep to the stateless model we designed the underlying communications architecture to accommodate it.

The underlying communications architecture for this system supports stateless agent execution by retransmitting the entire board at the beginning of each move instead of just the last move made. Additionally, the board arrangement is sent in a player-neutral format. This means that in Link4 pieces are labeled MY_PIECE or OPPONENTS_PIECE not player-one or player-two which would require the agent to know which player it is. A stateless agent playing when "Disconnect after every move" option has been selected in the Game Interface can even play against itself with just one instance running.

Evaluating Results

Our experience from the last three years indicates that puzzle solving, game playing projects, and competitions can have positive effects on student learning (see Table 1). Specifically, students' experience is significantly enhanced, along two dimensions:

- Better motivation and higher interest level - Overall, these competitions succeeded in increasing students' interest in the topics and motivating the students to learn the concepts and technique and apply them in a challenging environment. The competition clearly inspired the students. Many of them spent days and even weeks creating and adjusting their programs, working until the last minute. The actual contests attracted interested faculty members, students not enrolled in Artificial Intelligence classes, and even people from the community.
- Better understanding of the related material – Analysis of our examination records indicates better student understanding of basic concepts, especially those related to the projects such as blind and heuristic search, and game playing as search. As shown in the table below, class averages on project-related materials are generally higher than the overall class averages. The averages are even higher among students who successfully completed at least two of the three projects.

Term	Class Average (all material)	Class Average (project-related material)	Avg. of Students who completed project (project-related material)
Spring '03	75%	86%	91%
Spring '02	72%	75%	82%
Spring '01	67%	77%	85%

Table 1: Exam Result Summary

Future Work

Now that this project has shown it is successful in motivating students to learn heuristic search techniques, we have decided to focus on refining the tools used in this project.

The game interface and name server were originally written to be game independent. However, due to time constraints, only the Name Server has been used for more than one game. For next semester's class, we will focus on making the Link4 module of the game interface interchangeable with modules for other popular board games.

In the previous semester, several students indicated that they tested their heuristics against real players in online game rooms. It has been suggested that we could create a link between the Link4Server Library and a game web site. It is even possible that this link could be used instead of the game interface for running the competition. Students would then be able to test their agents against real people in addition to each other's agents.

Material Availability

The game playing system is available for educational use. The current system can be downloaded at the following website: http://www.cs.siue.edu/gameplaying/

References

Kumar, D. 1999. Pedagogical Dimensions of Game Playing. *ACM Intelligence Magazine*, 10(1), 9-10.

Peg Solitaire. 1996. Palo Alto: Klutz Press

Russell, S. J., and Norvig, P. 1995. *Artificial Intelligence: A Modern Approach*. New Jersey: Prentice Hall.

Walker, E. L. 1994. The Use of Computers for Teaching Artificial Intelligence at Rensselaer. *Working Notes of the AAAI Symposium*: *Improving Instruction of Introductory Artificial Intelligence*: 47-50.