Int. J. Engng Ed. Vol. 11, Nos 4 and 5, pp. 284–292, 1995
Printed in Great Britain.

0949–149X/91 $3.00+0.00

A Laboratory to Improve Undergraduate Instruction in Artificial Intelligence

R. W. WEBSTER
Intelligent Machines Laboratory, Department of Computer Science, Millersville University, Millersville, PA 17551, USA

This paper describes a project to improve the quality of instruction of upper-level courses in computer science in artificial intelligence (AI). During this project undergraduate students have exclusive access to a laboratory of powerful, interconnected workstations and a programming environment that integrates AI techniques, graphics, windowing systems and object-oriented programming. With this equipment, students majoring in computer science, computer engineering, physics and engineering are able to develop advanced software projects utilizing state-of-the-art software tools. This AI laboratory has given students who wish to pursue careers in industry exposure to skills that are in high demand, which is crucial for today's science and engineering graduates. The AI laboratory has also provided better preparation for students who wish to go on to graduate school because of the opportunity to engage in advanced software projects with faculty. The most novel aspect of this project is that undergraduate students are exposed to sophisticated programming environments that integrate multiple software components such as graphics, windowing tools, an expert system shell, mouse-driven events and AI tools.

INTRODUCTION

Objectives of the project

IT IS THE belief of the author that artificial intelligence (AI) and symbolic computing will become integral components of many present day computer applications such as manufacturing automation systems, robotic systems, computer-aided design/computer-aided manufacturing (CAD/CAM) applications, and many real-world scientific, business and engineering applications. AI techniques will be used in the front-end (natural language interfaces to database systems) as well as the back-end (inference engines and rule-based expert systems) of many of the today's applications. This project attempts to give students who wish to pursue careers in industry exposure to these and other skills that are in high demand, which is crucial for today's science and engineering graduates. Thus, students are engaged in projects that provide them with experiences that will make them more competitive in the marketplace.

A great deal of research being performed in computer science in graduate programs centers around AI and its related fields of study. One of the primary objectives of this project is to boost student interest in pursuing more scholarly activities such as student publications, going on to graduate school and seeking employment in research-oriented institutions. The intention is to increase substantially the percentage of our majors who choose to go on to graduate school. In order to strengthen our position of preparing students for graduate work, it is essential to have a strong AI component to the curriculum and a state-of-the-art computer environment to perform experiments.

Deficiency addressed by the project

The main problem was that while the curriculum was strongly aligned with the national IEEE/ACM standards and the computer equipment was powerful enough, i.e. the institution has provided a basic level of support for the courses we offer, there were no modern graphical software tools. Our students are well prepared in the theory of computer science, the programming languages of the discipline and the foundations of computer science (operating systems, software engineering, computer architecture, mathematics). The students are exposed to editors and compilers (Pascal, C, C++, LISP, Prolog, Modula–2). The major deficiency in the program was that the students might never get to use a highly interactive, graphical, software environment for AI programming. A logical step at the time was to build a specialized laboratory and expose our students to an integrated software environment such as workstations equipped with windows, graphics, an expert system shell and AI compilers such as Common LISP with CLOS and add-on AI tools, and Quintus Prolog with a built-in X-window/object- oriented, graphics interface.

EQUIPMENT

At the outset we believed that a network of Sun SPARC workstations provided the most appropriate environment for this project. Although a case could be made for a configuration of DEC, HP, IBM, workstations or any other workstations, the overriding reason for the Sun system is its powerful hardware, the sophistication and flexibility of the software, and fact that Sun workstations appear to

be the most popular in the AI/computer science community. This means that our students would be most likely to use Sun workstations (or environments that are very similar) if they go on to graduate school or do sophisticated AI programming in industry. Also, the Sun systems appear to provide a computing platform to support AI technology and its integration into other conventional computer technologies. The workstations are networked together and connected to the university Internet backbone. The operating system is Unix, which provides access to a host of quality third-party software products. There are many third-party expert system shells and advanced AI add-on software components available for the Sun SPARC stations. Another important consideration was that Sun offered a high performance/cost ratio.

PROJECT BENEFITS

Most important benefit

In science and engineering the laboratory is the lifeblood of the discipline. It is usually in the laboratory where new ideas and new methods are tried out. All too often conventional computer laboratories are set up and managed so formally and are so structured that students fight for terminals in order simply to finish programming assignments. These laboratories are not designed to make the students experiment scientifically with the techniques and the technology.

In this project we intentionally set out to build a laboratory where the students felt they owned it and ran it, rather than the University Academic Computing department. The Intelligent Machines Laboratory (IML), owned by the Computer Science Department, is now the central focal point of the computer science program. It is a large laboratory of Sun workstations with access to publications and technical reports of faculty. There is also a video cassette recorder and monitor so that students can view technical video tapes when not programming. This laboratory is designed so that faculty and students can informally sit around and discuss various technical topics. Students can come and go as they please; there is no official 'sign in' procedure. The objective was to encourage students to experiment, to dig deeper into the knowledge of computer science, rather than just perform simple assignments. Our intent was that the students would be an integral part of the educational process because they are actively involved in the 'spirit of experimentation', the 'spirit of the laboratory experience'. The URL of the World Wide Web server for the IML lab is http://zansiii.millersv.edu.

Immediate Benefits

This AI workstation laboratory has improved undergraduate instruction in a number of ways. First, it has enabled students to develop all programs using windows and graphics; therefore, their programs can visually or graphically illustrate the techniques of AI. Some sample programming assignments have involved graphically illustrating on the monitor the various AI problem-solving techniques such as min-max, the A* algorithm, and-or graphs, hill climbing, and tree/graph searching algorithms such as breadth-first search, depth-first, best- first search, heuristic search, and uniform cost. Specific programming projects will be described such as an AI robotic mail delivery system, the classic eight- puzzle problem and heuristic path planning.

This methodology allows the students to see the actual execution of the AI techniques and then experiment with them. One novel benefit is that programs are designed to utilize slider bars and interfacing (windows/mouse) such that the user can adjust the heuristics interactively without having to recompile. Thus, a student can experiment by changing the parameters and heuristics, and immediately see the results. One of the most difficult problems in AI is designing heuristics.

Secondly, the use of these tools gives students hands-on experience with software that is essential for preparation for graduate school, and is also in high demand in industry. There are a significant number of students who work in industry and take courses towards a degree in computer science. This AI laboratory has provided the skills that are being demanded by our student's employers and thus has also been beneficial to the co-operative education program on campus.

With this new AI laboratory we have seen an increase in undergraduate student involvement in independent study with faculty members, because we have provided a more sophisticated software environment. We have also seen that quality senior research projects have facilitated student interest in pursuing more scholarly activities such as student publications, going on to graduate school, and seeking employment in research-oriented institutions.

Most importantly, the students have begun to build a library of programs that are presently used as demonstration programs for students in future semesters. This has improved undergraduate instruction in AI because faculty members can call up from the library any number of graphical demonstration programs to illustrate a particular AI technique. Also, the students can later run the same demo program (and other programs), and can experiment with the concepts by interactively changing the parameters and heuristics, thus visualizing the results. The library of graphical AI demonstration programs is also used for tour groups and special programs at the university, and can be run over the World Wide Web.

On a broader scale this project is also part of the FLAIR (Flexible Learning with an Artificial Intelligence Repository) project at Temple University. Funded by the National Science Foundation (NSF), the FLAIR project is a consortium of Temple University, Drexel University, Villanova University and Millersville University to establish a

repository for AI teaching materials [1]. The repository, to be maintained and supported by Temple University, will consist of a variety of educational materials for teaching AI, such as demonstration software. The repository will be used by students for software projects and by faculty for classroom preparation and independent personal study. Millersville University is currently an active participant and a contributor of materials to the repository.

In addition, this AI workstation laboratory and software environment has proven to be *the* prototype for other courses or tracks in the computer science curriculum. The results of using the laboratory appear to be generalizable to other classes in computer science (e.g. how to structure laboratory manuals and what software assignments can facilitate student publications).

SAMPLE STUDENT ASSIGNMENTS

Heuristic path planning project

This demonstration module, written in the Common LISP (list processing) programing language and the Garnet GUI (graphical user interface) system, graphically illustrates the classic search methodologies used in AI and discusses the power and rationale of heuristic search [2]. The module reinforces the AI concept of state-space problem solving, i.e. representing the problem as a graph structure (the state-space) and performing an intelligent search of the states [3].

The students have been told in lecture that many practical problems can be formulated or represented in state-space form. This point is reinforced in the module using the problem of determining the best route to drive from city to city. Each city is represented as a node in the graph and the road to the next city is represented as an arc. The state-space is the graph. Each state or node is a city. The problem postulated to the students is: given a starting city, what is the best method to compute the route or path to the goal city?

The Search Module screen is shown in Fig. 1. If the user clicks on the Help button, a fully scrollable window pops up with many helpful hints and issues concerning AI search strategies. The References button pops up a scrollable window with many references from the AI literature on search strategies. Clicking on the Show Code button will pop up a scrollable window with the particular search

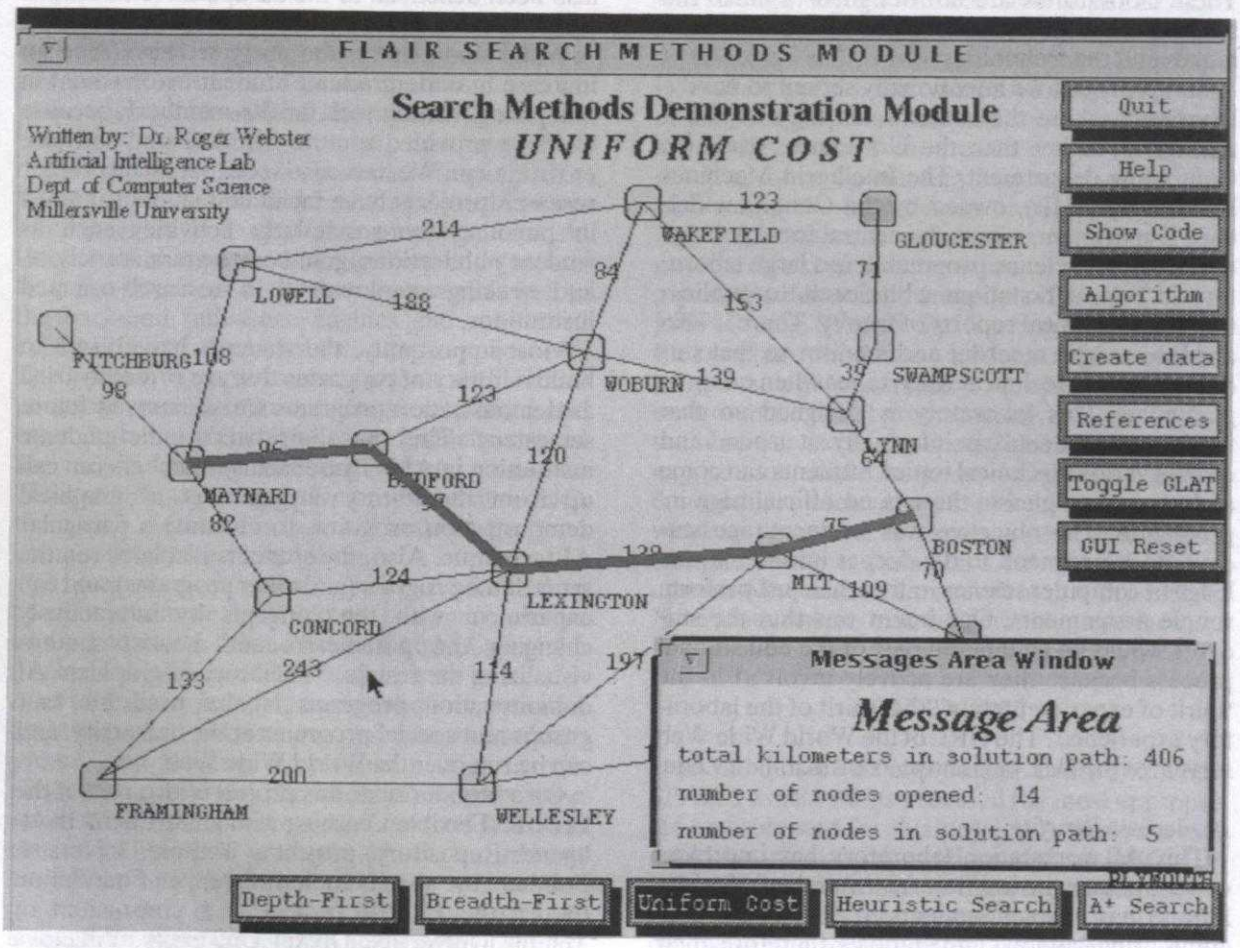

Fig. 1. Results of uniform-cost search using actual driving distances.

strategy code in it. Students can scroll through the code and look at it or print it off. The Algorithm button pops up the algorithm instead of the code.

The Toggle GLAT (General List Animation Tool) button turns on/off the list animation feature, which graphically shows the execution of the list manipulation being done in the particular search strategy chosen (described later in this paper). A window with all the cities pops up automatically in which the student can pick which city is the start node and which city is the goal node (not depicted in the figure). The Create Data button allows the user to create his/her own data set and save it or load in a created data set.

If students click on the Uniform-cost method button (starting at Maynard and going to Boston) they see the graphics and results shown in Fig. 1. The students are told in lecture that the uniform-cost method is always guaranteed to find the shortest cost path (optimal path) from start to goal node. However, by experimentation the students discover that with large state-spaces the uniform-cost algorithm (also with depth-first and breadth-first) can produce combinatorial explosion. With this particular example, using a heuristic search algorithm from Nilsson [4, pp. 55–60], the student sees that the solution path is only 16 km longer than the optimal solution, and the algorithm checked only five nodes instead of 14 (Fig. 2).

The students' experimentations reinforces that heuristic search tends to guide the search process toward selecting cities that are closer to the goal city. Thus, the computation can be significantly reduced. The FLAIR demonstration module produces the graphics and outputs the results for heuristic search shown in Fig. 2.

Experimenting with the module allows the students glean a number of important AI concepts relating to state-space search, i.e. that heuristic search:

- can greatly reduce the search effort (the number of nodes expanded) and computation time;
- is not guaranteed to find the optimum path to the goal as with the uniform- cost method;
- is identical to uniform-cost except that the g(N) evaluation function is an estimate of how close the node (N) is to the goal node (G);
- has the effect of pulling the search process towards selecting nodes that appear to move closer to the goal;
- is also called ordered search because the nodes are ordered by their respective evaluation functions.

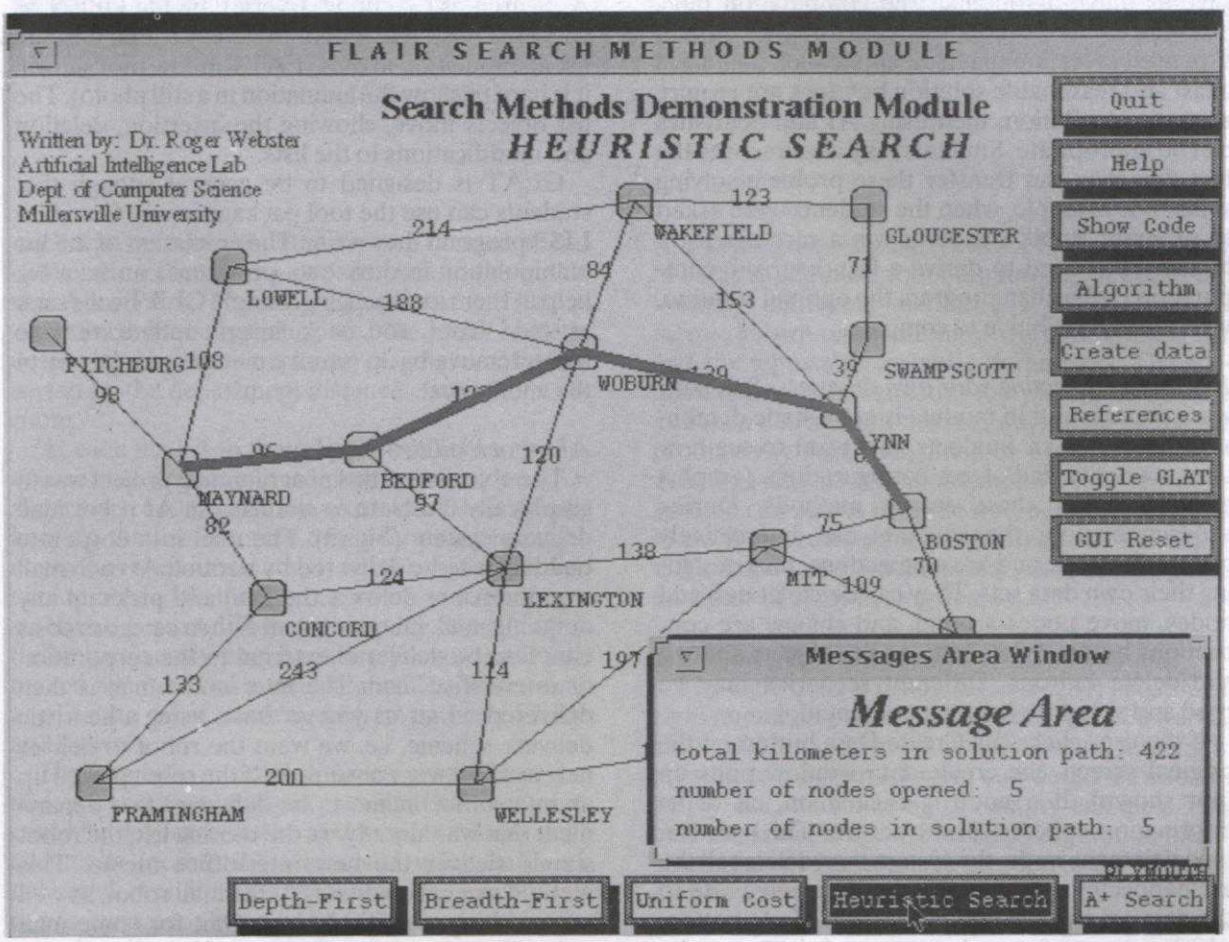

Fig. 2. Results of heuristic search.

The particular heuristic used in this path planning example is the Euclidean distance between the node (N) and the goal node (G). Given the latitudes and longitudes of the cities, we can derive the distance from any node to the goal node in Euclidean space or as the crow flies:

g(N):= Euclidean-distance (node[N].latitude, node[N].longitude, Goal.latitude, Goal.longitude)

where: Euclidean-distance = sqrt ((node[N].latitude – Goal.latitude)^2 + (node[N].longitude – Goal.longitude)^2)

This is reasonable since one would expect that if a city is closer than all the rest of the choices in Euclidean distance, then probabilistically, that city is along the shortest path to the goal (but not always!). The students observe the fundamental techniques used in AI programming and problem solving through these graphically illustrated examples and, more importantly, they experiment with them in the laboratory. The results of this path-planning demonstration software are described in additional detail in [5]. A demonstration of this software module can be run over the Internet using a World Wide Web browser such as Netscape or Mosaic at URL http://zansiii.millersv.edu/cs451.html (click on either LISP or Prolog demos).

It becomes evident from the experiments that using a heuristic to compute the routing path can save as much as one-half the computation time. Therefore, the students reaffirm the notion espoused in lecture that if an application mandates a fast and reasonable solution but does not require an optimal solution, then using AI and heuristics can be appropriate. Students may also uncover the fact that they can transfer these problem-solving skills. For example, when the students were asked to program Rubik's cube to win a race against a human, they tried to derive a fast, heuristic computation rather than program the optimal solution, which was prohibitive to compute.

Flexibility—Creating your own data set This software demonstration module is not a static demonstration, however. Students may want to see how their own city and street configurations (graphs) perform under these search methods. During experimentation, the students can interactively change the node and arc connections, thus designing their own data sets. They can delete nodes, add nodes, move nodes around, and change arc connections by simply clicking on the objects and the add/delete buttons. Different scenarios may be tried and saved for later experimentation.

If the user clicks the Create Data button on the original screen, the create-data window pops up (not shown). If a node is clicked on, all of its information appears on the screen. The user can then delete the node, the system then deletes all the arc connections. The user can add a new node to the data set by clicking on the Add Node button, which prompts for node name and then puts the node on the screen. The user can move the node around (or any node) by simply selecting and dragging the node to any location on the screen. All the arcs automatically move with the node because they are constraints built in to the node objects via the Garnet GUI interface for Common LISP.

Arcs can be created by simply selecting a node, clicking the Create Arc button and then selecting the node to connect to. Arcs can be deleted the same way. The user can decide to start afresh by deleting all of the data set and creating an entirely new data set. Any modification to the data set can be saved and later recalled. Numerous data sets can be stored. The authors are currently working on a Data Object, such that data sets can be shared among users of the repository and ported to other demonstration modules as simple objects.

How does the LISP code work? Students may also be interested in how the code works. How do these various search strategies work in LISP ? They can click on the Show Code button and the code for any of the search methods will pop up in a scrollable window so that they can browse through it. In addition, they can turn on GLAT, to show how the list manipulation is done in the code.

GLAT is provided in the Search Methods demonstration module to help show students how lists are manipulated while a particular AI search technique, such as breadth-first, heuristic search or A* search, is executing. Every time the OPEN or CLOSED list is updated, the GLAT animates the list manipulation in the GLAT window (not shown: it is hard to show the animation in a still photo). The list objects move, showing the insertion, deletion and modifications to the lists.

GLAT is designed to be general enough that students can use the tool package in any Common LISP program they write. The animation of the list manipulation in their own programs can be a big help to them in debugging. Simple GLAT calls such as: [add front, add back, insert, update, remove front, remove back, remove member, etc.], control the animation.

AI robot mail delivery project

The objective of this programming project was to graphically illustrate or simulate an AI robot mail delivery system (Fig. 3). The mail in a corporate building is to be delivered by a robot. At each mail stop the robot delivers the mail and picks up any outgoing mail. Outgoing mail is then categorized as either: to be delivered external to the corporation or inter-office mail. The inter-office mail is then delivered on an 'as you go' basis using a heuristic delivery scheme, i.e. we want the robot to deliver new mail if it was convenient. If the robot picked up an inter-office memo to be delivered to a department that was already on the routing list, the robot should deliver the new inter-office memo. This method can get confusing to the mail robot, as well as cause a prohibitively long wait for some mail stops. Therefore, a heuristic was used.

The program begins by placing the robot at the

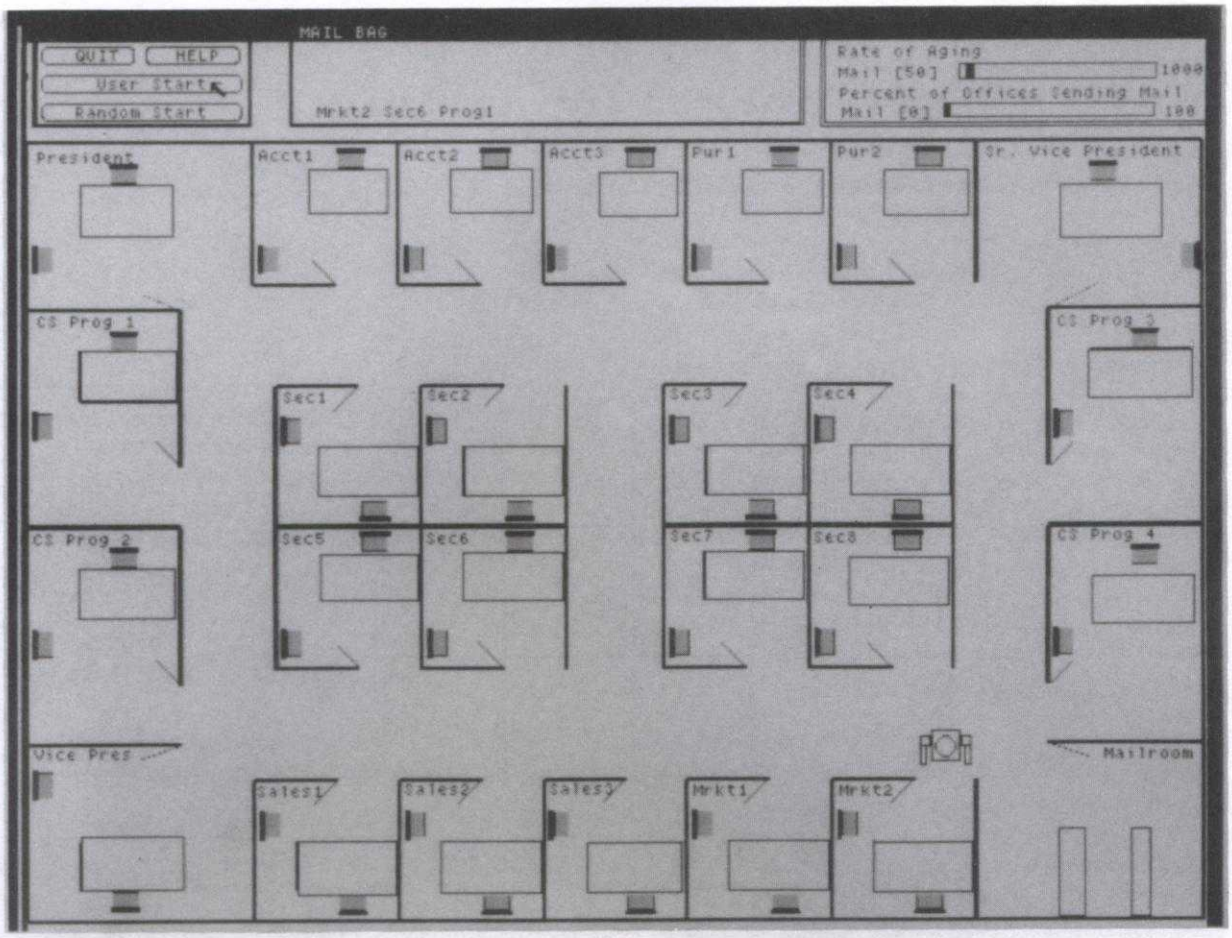

Fig. 3. An AI robot mail delivery simulation.

mailroom with a random amount of mail to be delivered to the various departments in the building: Accounting, Marketing, Engineering, Quality Assurance, Manufacturing, etc. The initial route is determined by distance, i.e. an original routing list sorted by the department distances from the mailroom.

At each mail stop the robot would collect interoffice mail (randomly adjustable by a window slider bar), and update the routing list. Just simply keeping the list in sorted order by distance might cause the robot continuously to pick up and deliver mail to only a few local departments and it may get stuck in this mode. Therefore, ageing was added to the heuristic evaluation function. The new mail 'just picked up' received a time stamp. The older mail aged as the robot continued on its path of pick up and delivery.

The heuristic evaluation function weighted the distance and age to assign a priority. Thus, new mail could be delivered to the next stop, but older mail eventually increased its heuristic value and was delivered, even if that mail stop was far away (so as not to keep people waiting). Students could interactively adjust the heuristic weights and 'see' the results.

The program was a great deal of fun to write and to use; the graphical robot ran furiously around the building delivering mail to the various departments. Shown on the monitor were the routing list and the adjustable heuristic parameters. The students learned that instead of just picking up all mail and then delivering it the next day, some increased productivity could be gained by using a 'heuristic' delivery scheme. A demonstration of this software module can be run over the Internet using a World Wide Web browser such as Netscape or Mosaic at URL http://zansiii.millersv.edu/cs451.html (click on either LISP or Prolog demos).

Eight-puzzle program with tree representation

The eight-puzzle problem (see Fig. 4) consists of eight numbered, movable tiles set in a 3×3 frame. One cell of the frame is always empty, making it possible to move an adjacent numbered tile into the empty cell, thus moving the empty cell also. The problem is to change the initial configuration into a given goal configuration. A solution to the problem is an appropriate sequence of moves [6].

The demonstration module begins with the screen in Fig. 4. Students start the demo by either clicking on the Shuffle button which randomly

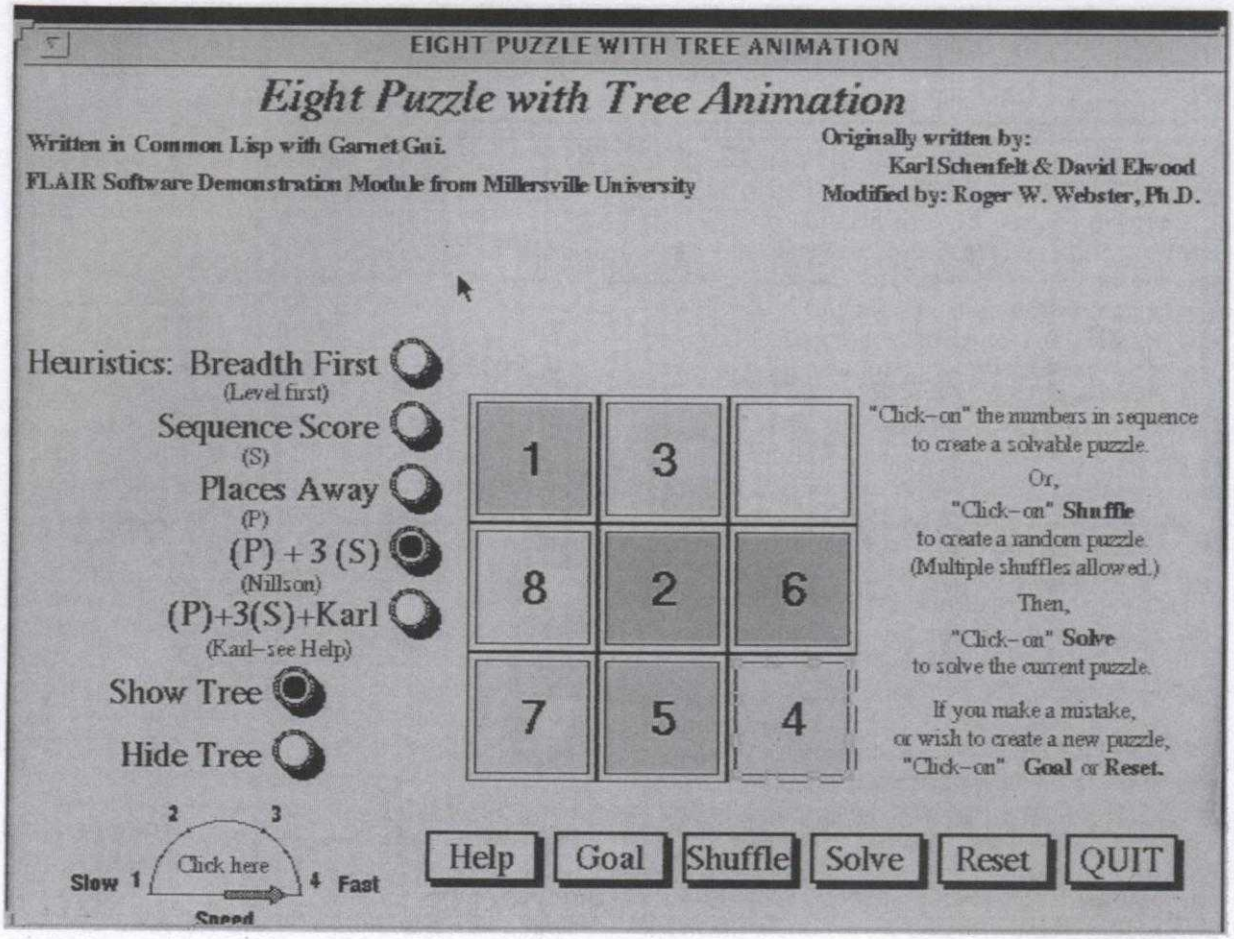

Fig. 4. Eight-puzzle user window.

shuffles the eight puzzle tiles, or the student can manually build the initial state to any desired tile configuration by simply moving the tile objects into place. Once the initial state is set, the student can pick any of five heuristics to be used to solve the puzzle (search the state-space). The five heuristics include: breadth-first, sequence score only, number of places away from the goal position, Nilsson's classic $P(n) + 3S$, or $P(n) + 3S$ + Karl (a student-added heuristic). A facility exists within the module which allows a student to add his/her own component to the heuristic and experiment with it. Nilsson's classic $P(n) + 3S$ computation is the summation of the number of places away from the goal position for each tile plus 3 times the sequence score, which is obtained by checking around the non-central squares, scoring a 2 for every tile not followed by its correct successor and scoring a 0 for every other tile; a piece in the center scores a 1 [4, pp. 64–69].

Clicking on the Solve button then solves the puzzle and graphically depicts the search-space tree with the solution (Fig. 5). The demonstration code also detects unsolvable puzzles as may happen when the student picks his/her own initial state. Although it is not possible to test all unsolvable puzzles, certain tests can easily be implemented. One test for an unsolvable state is that if any two tiles are swapped from their goal sequence. A scrollable help window is available to explain further the AI concepts of the eight-puzzle problem, state-space search, heuristics, etc. The tree of the state-space and the solution are graphically displayed as the program is searching the state- space. Also printed in the window is the depth of the solution, the number of nodes in the solution path, the number of nodes expanded and the number of nodes generated. The students can also bring up a scrollable window to show the eight-puzzle LISP code.

Last semester's projects included writing an expert system and AI controller to play the game Stratego™, a battlefield strategy game. This was a major challenge, since playing the game is very intricate and the strategy is not trivial. It also has given students experience with writing a sophisticated expert system for strategic operations, which is popular in many military applications. A demonstration of this software module can be run over the Internet using a World Wide Web browser such as Netscape or Mosaic at URL http://zansiii.millersv.edu/cs451.html (click on either LISP or Prolog demos).

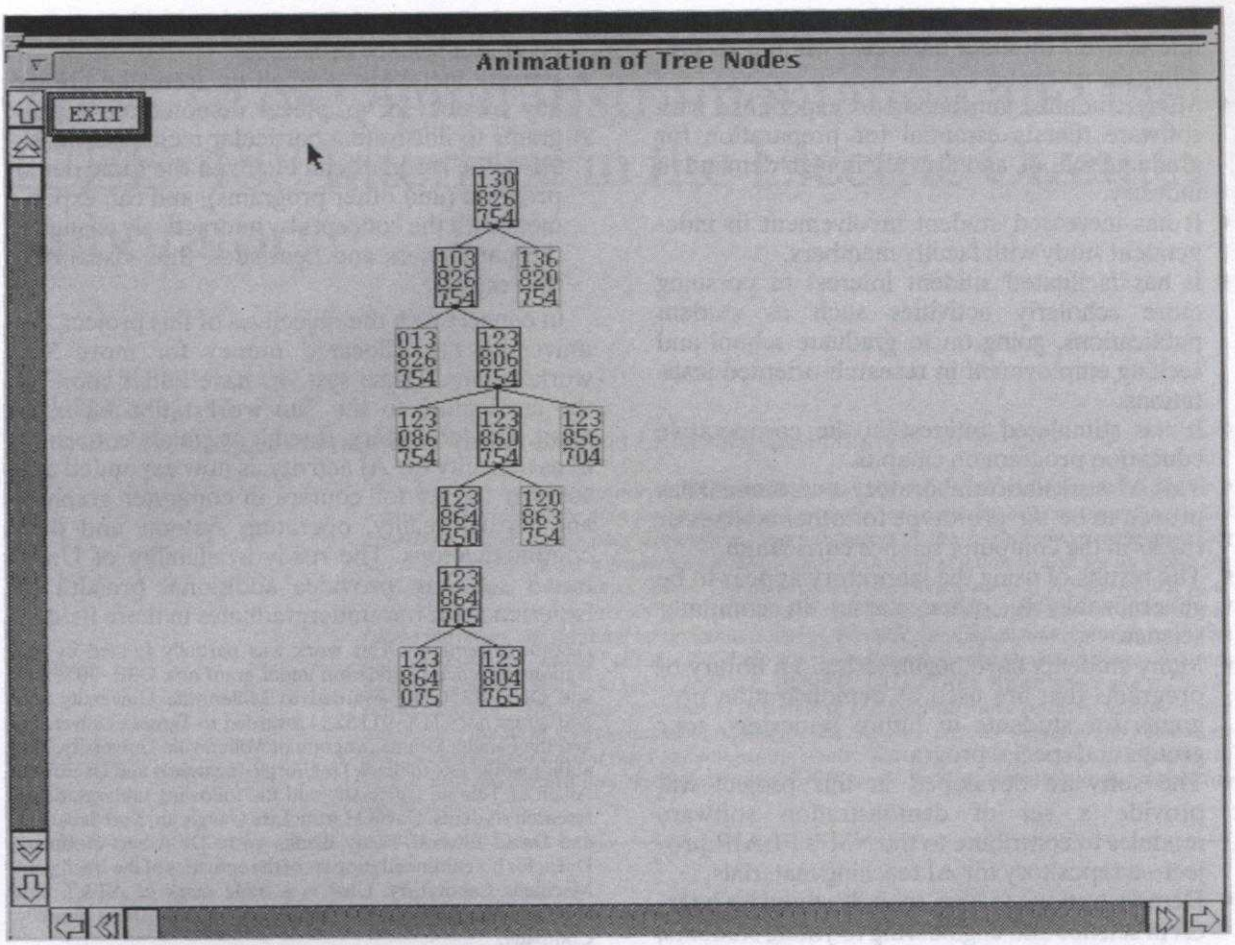

Fig. 5. Eight-puzzle graphic solution window showing tree of search space.

STIMULATING INTEREST IN ENGINEERING IN YOUNGER STUDENTS

Demonstrations of the equipment and the projects are performed regularly in the laboratory. These presentations to area schools and the community bring recognition to the university and to the project, but, most importantly, stimulate interest in science and engineering among young students.

The faculty and undergraduate students of the AI laboratory have given presentations and demonstrations to many diverse community groups: high school students, middle school age children, first graders, a Rural Partnership Program for tenth and eleventh graders, and to the public in various university programs. There is also a three-week special research project in robotics and AI with gifted high school students as part of the university Summer Science Training Program (SSTP). These young students spend three weeks, seven hours per day, five days per week with a university professor engaged in a small research project. This is a tremendous opportunity for high school students to prepare for college.

These demonstrations to area schools not only stimulate interest in science and engineering to young students, but appear to generate interest in our own computer science program. These young students are very impressed with our undergraduate research projects, the equipment, the faculty involvement, the laboratory, and therefore usually enquire about admission to the program. In addition, workshops are being planned in the laboratory for the engineering and scientific community.

CONCLUSION

This investigator has been extremely pleased with the use of the laboratory so far. Students are very anxious to use the systems and spend many long hours programming, experimenting and exploring the techniques of AI.

In summary, the benefits of this AI workstation laboratory are as follows:

- It has enabled many students to develop programs using windows and graphics; therefore, their programs can visually or graphically illustrate the techniques of AI.
- Because software programs graphically illustrate the techniques of AI, many students can

experiment with the heuristics and parameters interactively by slider bars, etc., without recompiling the software.
- Many students gain hands-on experience with software that is essential for preparation for graduate school, and also are in high demand in industry.
- It has increased student involvement in independent study with faculty members.
- It has facilitated student interest in pursuing more scholarly activities such as student publications, going on to graduate school and seeking employment in research-oriented institutions.
- It has stimulated interest in the co-operative education program on campus.
- This AI workstation laboratory environment has proven to be *the* prototype for other courses or tracks in the computer science curriculum.
- The results of using the laboratory appear to be generalizable to other classes in computer science.
- Many students have begun to build a library of programs that are used as demonstration programs for students in future semesters, tour groups and special programs.
- The software developed in this project will provide a set of demonstration software modules to contribute to the NSF's FLAIR project—a repository for AI teaching materials.
- Demonstrations to area schools stimulate interest in science and engineering to young students, and also appears to generate interest in our own computer science program.
- Faculty members can call up from the library any number of graphical demonstration programs to illustrate a particular technique in AI. Also, the students can later run the same demo program (and other programs), and can experiment with the concepts by interactively changing the parameters and heuristics, thus visualizing the results.

In concert with the objectives of this project, the university has allocated money for more Sun workstations. These systems have added more of the curriculum to the Sun workstation environment. The laboratory, though originally conceived of as a facility for AI activity, is now expanded as a support facility for courses in computer graphics and virtual reality, operating systems and data communications. The ready availability of Unix-based software provides additional breadth of experience for our undergraduates in these fields.

Acknowledgements—This work was partially funded by the National Science Foundation under grant nos. USE–9050371 and DUE–9350841 awarded to Millersville University and NSF grant no. CDA–9115254 awarded to Temple University, and the Faculty Grants program of Millersville University. The author would like to thank Dr Giorgio Ingargiola and Dr Robert Alken of Temple University and the following undergraduate research students: David Martin, Lisa Gaughran, Karl Schenfelt and David Elwood. Many thanks go to Dr Albert Hoffman, Dean, for his continued support of the activities of the Intelligent Machines Laboratory. Unix is a trade mark of AT&T Bell Laboratories, Stratego is a trademark of the Milton Bradley Company.

REFERENCES

1. R. Aiken and G. Ingargiola, FLAIR—flexible learning with an artificial intelligence repository, National Science Foundation grant September 1991–February 1995, NSF Grant no. CDA–9115254, Temple University, Philadelphia, PA.
2. A. B. Meyers *et al.*, Garnet Reference Manual Version 2.0, Technical Report CMU-CS–90–117–R2, Carnegie Mellon University (May 1992). The Garnet GUI system is a powerful object-oriented software package for Common LISP written by Carnegie Mellon University (available free of charge from ftp a.gp.cs.cmu.edu).
3. P. Winston, *Artificial Intelligence*, Addison-Wesley, Reading, MA, pp. 87–106 (1983).
4. N. Nilsson, *Problem Solving Methods in Artificial Intelligence*, McGraw- Hill, New York, pp. 55–60 (1971).
5. R. Webster, Useful artificial intelligence tools—a review of heuristic search methods, *IEEE Potentials J.*, pp. 51–54 (October 1991).
6. S. Tanimoto, *The Elements of Artificial Intelligence*, Computer Science Press, Rockville, MD, pp. 169–178 (1987).

Dr **Roger Webster** earned a Ph.D. degree (1988) in computer science from the School of Engineering at Temple University in Philadelphia, Pennsylvania. From 1979 to 1982 he worked at the Hewlett-Packard Corporation's Medical Systems Division in Waltham, Massachusetts as a software engineer. Since 1983 he has been a Professor of Computer Science and Director of the Intelligent Machines laboratory at Millersville University in Millersville, Pennsylvania (see World Wide Web server at http://zansiii.millersv.edu). Dr Webster has been awarded five grants in six years in the area of robot vision, artificial intelligence and real-time systems. The most recent, a National Science Foundation Grant equipping his laboratory with Sun SPARCstations, a mobile robot and a virtual reality system. He has published articles in robot vision and AI, and currently serves as a referee for the *IEEE Computer* journal, *Artificial Intelligence* journal, and the *IEEE Transactions on Systems, Man, and Cybernetics* journal. Dr Webster holds memberships in the IEEE Computer Society, the Association for Computing Machinery (ACM), the American Association for Artificial Intelligence (AAAI), the Canadian Artificial Intelligence Society (CAIS), and the American Society for Engineering Education (ASEE). His research interests are robot vision, real-time systems engineering, artificial intelligence and virtual reality, especially real-time mobile robot navigation, and virtual world modelling.