Model AI Assignments

Todd Neller
Gettysburg College
tneller@gettysburg.edu

John DeNero and **Dan Klein**
University of California, Berkeley
{denero, klein}@cs.berkeley.edu

Sven Koenig and **William Yeoh** and **Xiaoming Zheng** and **Kenny Daniel** and **Alex Nash**
University of Southern California
{skoenig, wyeoh, xiaominz, kfdaniel, anash}@usc.edu

Zachary Dodds
Harvey Mudd College
dodds@cs.hmc.edu

Giuseppe Carenini and **David Poole**
University of British Columbia
{carenini, poole}@cs.ubc.ca

Chris Brooks
University of San Francisco
cbrooks@cs.usfca.edu

Abstract

The Model AI Assignments session seeks to gather and disseminate the best assignment designs of the Artificial Intelligence (AI) Education community. Recognizing that assignments form the core of student learning experience, we here present abstracts of eight AI assignments that are easily adoptable, playfully engaging, and flexible for a variety of instructor needs. Assignment specifications and supporting resources may be found at http://modelai.gettysburg.edu.

The Pac-Man Projects Software Package for Introductory Artificial Intelligence - John DeNero, Dan Klein

The Pac-Man projects apply a variety of artificial intelligence (AI) techniques in the setting of the classic video game Pac-Man. Despite their video game theme, the projects are targeted very broadly. They cover a range of foundational AI concepts, including informed state-space search, probabilistic inference, and reinforcement learning. These concepts underlie real-world application areas such as natural language processing, computer vision, and robotics. The Pac-Man projects promote effective learning through several design principles. Graphical interfaces allow students to visualize the results of the techniques they implement. Harness code contains commented examples and clear directions, but does not force students to wade through undue amounts of scaffolding. Finally, the domain of Pac-Man itself provides a challenging problem environment that demands creative solutions; real-world AI problems are challenging, and Pac-Man is too. In our course at UC Berkeley (CS 188), these projects have substantially boosted enrollment, teaching reviews, and student engagement. The projects have been field-tested, refined, and debugged over multiple semesters. We are now excited to release them to other universities for instructional use. This software package includes four primary projects on search, multi-agent search, reinforcement learning, and probabilistic tracking. It also includes an open-ended final contest and an initial Python tutorial.

A Project on Fast Trajectory Replanning for Computer Games for "Introduction to Artificial Intelligence" Classes - Sven Koenig, William Yeoh

This standalone path-planning project for an undergraduate or graduate artificial intelligence class relates to video game technologies and is part of our effort to use video games as a motivator in projects without the students having to use game engines. Heuristic search and, in particular, A* are among the most important search techniques and thus are good candidates for a project in artificial intelligence. In this project, the students need to code A* and then extend it to Adaptive A*, a fast trajectory replanning algorithm, to move game characters in initially unknown gridworlds to a given target. Adaptive A* is an incremental version of A* that often searches faster than A* since it updates the heuristics between searches to find solutions to series of similar search tasks potentially faster than is possible by solving each search task from scratch. This project requires students to develop a deep understanding of A* and heuristics to answer questions that are not yet covered in textbooks. The project is versatile since it allows for theoretical and implementation questions. We list a variety of possible project choices, including easy and difficult questions. The project text and additional support material (such as a maze generator and pointers to the literature) can be found at http://idm-lab.org/gameai.

Getting Set with OpenCV - Zachary Dodds

"Getting Set with OpenCV" is a two-week assignment used as a hands-on introduction to computer vision in an undergraduate AI course. Set is a game of visual percep-

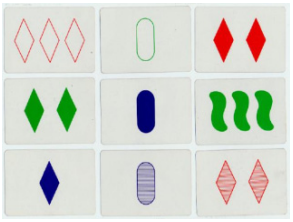

Figure 1: Set Image.

3	4	1	3	1
3	3	3	G	2
3	1	2	2	3
4	2	3	3	3
4	1	4	3	2

Figure 2: Example Rook Jumping Maze. Starting at the circled cell, each *jump number* indicates the exact number of cells one may move in a straight line horizontally or vertically. The object is to find a path to the goal marked "G".

tion and reasoning: players look through a group of face-up cards that contain a variety of shapes. Each of those shapes has four attributes: color, texture, form, and number. The game's goal is to find a trio of cards among which these attributes have all identical or all distinct values. As an example, Figure 1 shows nine of the 81 Set cards. Within this image the "central" card, the "east" card, and the "northeast" card form a set.

The goals of this assignment are twofold. First, it stands alone as an introduction to image-processing suitable in a junior- or senior- level AI course. As we use it, however, it also motivates the use of the OpenCV library, the largest and most ubiquitous software foundation for real-time vision. For all of its strengths, OpenCV continues to be a challenge to newcomers. This assignment offers a smooth and task-directed path through which undergraduates can get "set" with OpenCV before applying it in more advanced investigations.

Rook Jumping Maze Generation - Todd Neller

A Rook Jumping Maze (Figure 2[1]) consists of a grid of *jump numbers*, a start cell, and a goal cell. Each jump number indicates the exact number of cells one may move in a straight line horizontally or vertically. The object is to find a path from the start cell to the goal cell.

In this suite of introductory AI assignments, the student begins by representing a randomly generated maze and performing breadth-first search to compute the minimum solution path length, one measure of maze quality. There are many stochastic local search exercise options for iterative improvement of the maze according to the evaluation function. Creative opportunities also exist for more sophisticated evaluation function computation. Reinforcement learning may also be applied to the tuning of stochastic local search. Thus, this suite of exercises may be used as a common thread tying together various topics of an introductory AI course.

[1]Shortest 13-move solution for Figure 2: down, right, left, up, down, left, right, up, left, left, right, down, up

Assignment on CSPs for First Undergraduate AI Course - Giuseppe Carenini, David Poole, CPSC322 Team[2]

This assignment covers most of the basic principles and techniques involved in solving constraint satisfaction problems (CSPs). The complexity of the questions increases smoothly. The student is first required to represent and solve a simple problem with the help of AIspace, an interactive tool for teaching and learning AI (`http://www.aispace.org/`). After that, the student is asked to implement several CSP techniques (Arc Consistency, Search by Domain Splitting, Stochastic Local Search) and apply them to solve more realistic problems ranging from the Sudoku game to configuration and scheduling problems. The target audience is students taking the first undergraduate AI course (focusing on Representation and Reasoning, not Learning). This is a very challenging assignment with a strong programming component (in Java). Students should be encouraged to work in pairs on the programming questions. The assignment has never been offered exactly in this form, but from past experience with rather similar formats it should take students between 15 and 20 hours. The assignment can be simplified in several ways. For instance, the code for arc consistency could be given to the students. The same is true for the code for the SLS algorithms. This would make for an assignment more focused on representation than reasoning, and more focused on the interpretation of results. Another simplification could be to remove the Sudoku question.

A Project on Gesture Recognition with Neural Networks for "Introduction to Artificial Intelligence" Classes - Xiaoming Zheng, Sven Koenig

This standalone neural network project for an undergraduate or graduate artificial intelligence class relates to video-game technologies and is part of our effort to use video games as a motivator in projects without the students having to use game engines. Neural networks are among the most important machine learning techniques and thus are good candidates for a project in artificial intelligence. In this project, the students need to understand and extend an existing implementation of the back-propagation algorithm and use it to recognize static hand gestures in images. This project requires students to develop a deep understanding of neural

[2]Parts of this assignment were designed by Kevin Leyton-Brown. The code was implemented by Mark Crowley, Erik Peter Zawadzki and David R.M. Thompson. AIspace is a project co-led by David Poole and Alan Mackworth.

networks and the back-propagation algorithm. It extends a project from Tom Mitchell's "Machine Learning" book and builds on ideas, text and code from that project (courtesy of Tom Mitchell). The project is versatile since it allows for theoretical and implementation questions. We list a variety of possible project choices, including easy and difficult questions. The project text and additional support material (such as code and image data sets) can be found at `http://idm-lab.org/gameai`.

An Introduction to Genetic Algorithms - Christopher Brooks

Genetic algorithms (GAs) are a popular topic for students in introductory AI classes, and a nice way to introduce a variety of search and constraint problems. Two common challenges in teaching students about GAs are the wide variety of design choices for mutation, crossover, selection and elitism, and the challenge of applying a GA to a non-trivial problem that is still small enough to be used in a class.

This assignment addresses these challenges by providing students with a mixin-based template for working with a GA, allowing them to easily change selection, crossover, mutation or elitism operators without editing or recompiling code. It also provides three different problem domains: the bitstring problem, the traveling salesman problem, and the nurse scheduling problem, and encourages students to explore the effect of different design and parameter choices on each problem in an attempt to better understand the GAs algorithmic behavior and the underlying diffculties of the fitness landscape.

It also provides students with exposure to two new programming paradigms: the use of multiple inheritance to implement a mixin, thereby combining functionality from several different objects at runtime, and the use of higher-order functions, which allow the user to specify a fitness function separate from the GA implementation. Both of these techniques allow the student to quickly experiment with a variety of configurations without any editing of source code.

A Project on Any-Angle Path Planning for Computer Games for "Introduction to Artificial Intelligence" Classes - Sven Koenig, Kenny Daniel, Alex Nash

This standalone path-planning project for an undergraduate or graduate artificial intelligence class relates to video game technologies and is part of our effort to use video games as a motivator in projects without the students having to use game engines. Heuristic search and, in particular, A* are among the most important search techniques and thus are good candidates for a project in artificial intelligence. In this project, the students need to code A* and then extend it to Theta*, an any-angle path-planning algorithm, to plan paths for game characters in known gridworlds. Theta*, like A*, propagates information along the edges of the gridworld (to achieve a small runtime) but, unlike A*, does not constrain the paths to be formed by grid edges (to find short "any-angle" paths). This project requires students to develop a deep understanding of A* and heuristics to answer questions that are not yet covered in textbooks. For example, Theta* and A* have many different properties despite their similarities. Exploring the differences between Theta* and A* thus helps students to improve their understanding of A*. The project is versatile since it allows for theoretical and implementation questions. We list a variety of possible project choices, including easy and difficult questions. The project text and additional support material (such as pointers to the literature) can be found at `http://idm-lab.org/gameai`.