Leveraging AI's Breadth in CS 1

Zachary Dodds

Harvey Mudd College Computer Science Department
301 Platt Boulevard
Claremont, CA 91711
dodds@cs.hmc.edu

Abstract

Artificial Intelligence offers a compelling backdrop for student assignments and projects even very early in the computer science curriculum. We have leveraged so-called high-level AI, in the form of reasoning about language and game-playing to motivate students in CS 1. More recently we have added activities that might be described as "lower-level" AI: robotics programming via both simulated and real platforms and audio processing and classification. This work presents both student and faculty responses to this experiment and concludes that, with appropriate scaffolding, topics from AI's full breadth succeed equally as hooks into early CS.

Overview

Because students, as intelligent agents of their own, connect viscerally with the endeavor of Artificial Intelligence, AI has long been a source of motivating material for early CS curricula [1,2]. Game-playing assignments such as Connect Four are as comfortable and ubiquitous in CS 1 or CS 2 as the Towers of Hanoi or database-like applications [3,4]. Our introductory computer science course at Harvey Mudd College has for years included that assignment, as well as a number of others that draw inspiration from AI's "high-level" reasoning about natural language.

In 2006 our computer science department redesigned CS 1 in order to attract more students, and women in particular, as well as to better present the breadth of CS as more than simply programming skills. This redesign prompted introspection, too, about how we might better reflect the breadth of CS's central subfields, such as AI. As a result, we added two additional CS 1 assignments of two weeks each, both drawn from what is sometimes termed "low-level" AI. The first comprised a robotics project made accessible via a simulator [5] but also backed up by the opportunity to implement on the iRobot Roomba and Create [6,7]. The second project asked students to write a classifier for different groups of sounds, using raw audio data as input. This use of data-driven and agent-driving AI topics in early CS borrows wholly from a long tradition of educational robotics [8,9,10] as well as exciting and emerging media-based CS curricula [11].

Certainly the distinction between high- and low-level AI has disappeared among today's deep and mutually dependent interactions between artificial cogitation and the processing of sensory inputs. In fact, it may never have had much basis in true AI practice. Even so, the top-down and bottom-up paradigms are important metaphors for novice computer science students, and AI's versions of those metaphors are powerful because they offer an initial analysis of the capabilities all students have – and many have taken for granted for decades.

Building from this natural, if naïve, connection, we have measured students' attitudes toward our high-level and low-level AI assignments. Thus, with this work we hope to make three contributions to educators considering AI in their early computer science curricula:

- detailed descriptions of two CS 1 assignments from both high- and low-level AI (sections 2, 3)
- results from students' reflections on those assignments, across multiple course sections and years (section 4)
- reflections on their successes and drawbacks (section 5)

High-level AI in CS 1

The URL **`www.cs.hmc.edu/twiki/bin/view/CS5`** has detailed write-ups, supplementary resources, and lecture slides that accompany the assignments motivated in this section. All programming materials are in Python.

Connect Four: *reasoning via recursion*

As noted, Connect Four (C4) has long been a standard in early CS; our CS 1 is no exception. First, it provides a backdrop against which to tell the story of computer chess in the history of computing. We also use it to illustrate the findings of de Groot [12] demonstrating the extent to which humans play such games with an internal a look-up table (Figure 1, left).

These side stories help motivate interest in the game, but its pedagogical value derives from the CS 1 topics it reinforces. Students gain practice with two-dimensional

Submission to the 2008 spring symposium on *Using AI to Motivate Greater Participation in Computer Science*, Stanford, CA, 3/26-28/07.

arrays of data when implementing the board. They reinforce recursion via the n-ply search -- we do not look at pruning at all. Our implementation also practices object-oriented syntax, design, and concepts, as students implement a Board class to host the game and a Player class whose instances decide on the next move to make. Others use this assignment in order to motivate ideas in interface-building or graphics-based programming. We have provided students with a wrapper that removes the challenge from the graphics. Even so, it provides an opportunity to introduce the model-view-controller organizational principle: students create the data structures for Connect Four independent of output choice. Then, they may choose a graphical or text-based view into those structures (Figure 1, right). This assignment typically takes two weeks, and is accompanied by other problems as well.

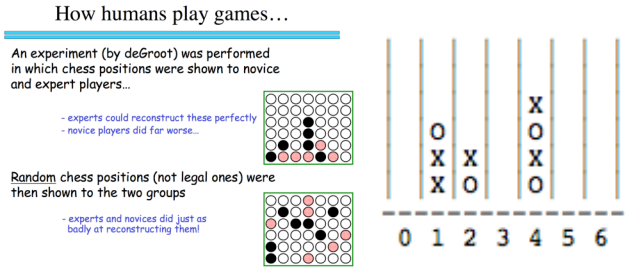


Figure 1. CS 1 slide showing de Groot's insights into human game representations (left). (Right) the ASCII representation students use to prototype and debug their C4 classes. A graphical representation is added only later.

Because the development of the data structures and search algorithm are paramount, work on a static board evaluator is left as an extra-credit exercise. A number of students do create a static evaluator, however, and we run a competition among those entries and announce with much fanfare a champion among each class section.

Toward Natural Language: *unnatural language*

One of the conceptual hurdles that challenges many new CS students is the concept of *text* as computational raw material. Numbers are natural for computation – after all, students have practiced those algorithms since elementary school. Yet many students find it quite difficult to consider text *without* the human-bestowed semantics it ordinarily carries. Because of this, we interlace a series of text-themed assignments throughout CS 1. Their purpose is to reinforce the extent to which computer programs *lack* the context for the text they manipulate. To the computer, that text is simply a set of character strings.

In the very first programming assignment of the term, students exercise their skills with console-based I/O and with printing by building a computer conversationalist. To emphasize the program's lack of understanding for its conversation, we encourage students to ask questions, but completely ignore the answers provided. Enough human conversations exhibit precisely this protocol that it is far from a stretch for students – and they enjoy the creativity (and attitude) they can bestow on their conversationalists so early in the term.

Three weeks later, we ask students to consider how string-manipulation can be an important part of a deeper "understanding" of text by implementing the Caesar cipher. Another staple of CS 1, the Caesar cipher simply rotates the characters of a plain-text message by $0 < n < 26$ places in the alphabet, wrapping from z to a as needed. The AI facet of the problem comes in classification: a **`decipher`** function that students write must generate the 26 possible ciphertexts and choose the one *most like English*. We encourage thinking broadly about what "*like English*" means, and students respond with one or more strategies:

- using letter frequencies to provide each possible decoding a first-order probability
- using dictionary look-up for common words
- seeking bigrams that do not appear in English
- identifying patterns common in English, e.g., ensuring each word contains a vowel or maximizing the number of vowels in the result

This open-endedness offers an opportunity to delve into the tradeoffs; it also makes the assignment a rewarding one for the students, as many feel for the first time like their code is making nontrivial decisions over which they have programmatic control. As a result, students regularly surprise us with their creativity: for example, it turns out that for short phrases minimizing the *scrabble score* of the possible ciphertexts more successfully distinguishes English than first-order, letter-by-letter frequency modeling.

After another four weeks, students further investigate models for the structure of text – in this case by creating a first-order Markov text generator based on units of words. This assignment offers practice in defining and using the dictionary data structure and loops, as students read in text files of their own choice and store a statistical summary of word co-occurrences. Running that summary in reverse can lead to wonderfully nonsensical and entertaining results. Together these three text-processing assignments expose students to the significant shift in difficulty when moving from straightforward character processing to the statistical analysis of natural language.

Low-level AI in CS 1

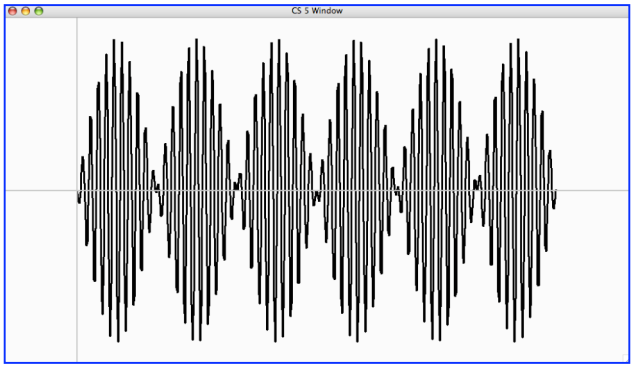
Audio processing: "*pass*" *or* "*fail*" ?

To convey the depth of abstraction that modern-day computational interfaces provide and to build atop the remarkable resources now available for media-based CS 1

[11], we introduced a two-week audio-processing assignment in the fall of 2006 and 2007.

To support students' work with audio data, we built tools for plotting the raw pulse-code modulated samples of sounds and for reading in, playing, and writing out sounds in .wav format. In the first week students practice their skills in manipulating one-dimensional lists by reversing sounds, changing their volume, splicing and reordering segments, and generating pure tones and chords from scratch.

In the second week students write a function to correlate two sound waves as illustrated below. This exercise motivates an introduction to the different ways in which humans interpret sound frequency: Figure 2 shows a 440 hz sine wave, subsampled only for plotting purposes, multiplied pointwise with a 3 hz wave. When played, the 440 hz A demonstrates a perceptible warble in volume



rather than an accompanying low-frequency tone.

Figure 2. The csplot.py module provides functionality for visualizing in 1d and 2d. Here, a 440 hz sine wave – aliased in the image – undergoes 3hz amplitude modulation, demonstrating humans' different interpretations of a sound's frequencies.

Summing the resulting samples from these pointwise multiplications yields the discrete Fourier transform, i.e., the algorithm by which a machine can "hear" frequency or pitch. Students then use their home-grown DFT in order to build a chord classifier that can distinguish between major and minor triads. The change of medium – from visual and textual to auditory in this assignment – sparks enthusiasm from some of the students who otherwise simply mark time from assignment to assignment without personally engaging in the material. Finally, students use their own voices as input, extending their chord classifier into a speech classifier that knows whether a male or female voice is speaking and whether the phrase spoken is "pass" or "fail." This progression from straightforward signal processing to the more nuanced and creative design of a speech classifier conveys the spirit of the speech recognition field in a manner accessible – and fun – for introductory students.

Embodied intelligence: *robot navigation*

The second sensory-based, AI-inspired assignment is a two-week project in which students program a simulated robot. Depicted in figure 3 (top left), the simulator and visualizer are versions of the Pyrobot system [5] tailored to our environment. The robot provides odometry, bump data, and a single range-finder on a panning mount. The task is navigation: the students implement a state machine which will navigate the robot to a human-specified goal (the green circle) in an environment with unknown obstacles. Typical approaches include opportunistic random wandering and/or wall-following. Both are strategies that still play an important role in the business and practice of robotics today, e.g., in iRobot's Roomba vacuums.

Because our curriculum postpones event-driven programming until CS 2, the students write their control programs within a traditional sense-plan-act loop that polls inputs to decide the next time step's motor velocities. This paradigm stretches their conception of control flow: the merging of discrete action selection with the continual choice of motor velocities leads naturally to a finite-state-machine architecture. This dovetails with the computational-models portion of the course both in time and spirit.

Although the 200+ students who take our CS 1 each fall do not all have access to real robot hardware, the interface and task is designed to transition smoothly to iRobot's Create platform. Students who wish run their code on the Create – invariably they are surprised at the large impact of noise and wheel slippage on their carefully, perhaps too-carefully, designed algorithms. To encourage this experience, students' performance on the real-world trials can not hurt their project grade. It does, however, build appreciation for the difficulties inherent in computational interactions with the physical world.

Growing the AI beyond CS 1

One of our goals with the iRobot Create assignment in CS 1 is to hook the interest of students who might not otherwise have continued their studies of computer science. Our department is particularly concerned about the low number of women who major in CS. Our experiences in fall 2006 sparked the interest of three first-year women, who opted to stay on campus in the summer of 2007 in order to extend the computational capabilities of the Create.

They decided to focus their summer efforts toward an entry into the 2007 Tapia robotics competition. The competition's task echoed that of our CS 1 project: finding distinctively colored markers in a partially known environment. The student began the summer with an unadorned Create, which does provide actuation, bump-sensing, and odometry through an interface identical to

the simulator's. From there they evolved the hardware and software of the platform until it included two sonar rangers on panning motors, an iSight firewire camera, and two USB-based controller boards. A Mac laptop onboard the robot provides the processing power. The robot ultimately earned a well-deserved nickname, *Insomnia* (top left). Figure 3 (bottom) depicts a frame in one of the marker-finding test runs, along with the final state machine for the students' entry.

Because the Create can be run wirelessly, too, it offers applications inviting to a wide range of students. Other platforms share this scalability – in fact, the Myro robot from the Institute for Personal Robots in Education has a form factor that for many students is more inviting than the utilitarian Create. The crucial feature in both, however, is that students can push beyond CS 1-level tasks to the point of direct engagement with the broader CS community both inside and outside their home institution.

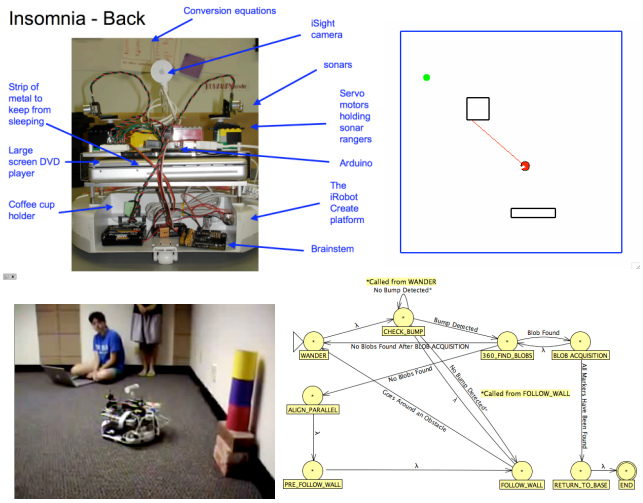


Figure 3. The simulator (top right) offers an identical interface to an iRobot Create, an expandable and low-cost platform shown with several added components (top left). Students can leverage the active competition community as early as they like: three first-years developed this Create for the Tapia 2007 robot competition, in which the Create uses a state machine (lower right) to seek out distinctive markers (lower left).

The specifics of the Create and the Tapia competition notwithstanding, this summer experience illustrates a best-case result from an AI-based "outreach" in CS 1:

- Three first-year women, none of whom had planned on taking more than the minimum requirements in CS, discovered a computational interest in robotics.
- CS 1's scaffolded projects fostered a confidence and familiarity with computation that those students chose to pursue.
- Although students' choice of major does not occur until later in the sophomore year, it seems very likely that one or more of these women will choose CS – largely because their engagement in the field has grown so deep through this project.

Of course, this best-case scenario in which an AI-themed CS 1 assignment becomes a foundation for is not typical. The following section examines more critically the student feedback we have received on our use of AI in CS 1. However, such results ***do not have to be typical to make a big impact*** on a computer science program. For example, our department has languished in recent years, with only 3-5 women majoring in CS at any given time. If this kind of CS 1 experience sparks interest in only one additional woman, it bolsters our small community of women computer scientists significantly.

Broader Student Feedback

Part of the success of these four AI-themed assignments: a game-player, text processing, audio classification, and robot programming stems from the excitement the instructors feel for artificial intelligence topics. Even so, a solid majority of CS 1 student projects and assignments neither exhibit nor build upon AI themes. To assess the differences between these classes of assignments, we contrasted student opinions of the AI-themed problems' ***worthwhileness*** and ***difficulty*** against their opinions on other course assignments as measured on a 7-point Likert scale (1 = least; 7 = most). Figures 4 and 5 depict the results: all of the AI-based work in CS 1 appears in the upper 50% of the list of our assignments, with most near the overall top. Those figures appear on the back page.

Ultimately, it was even more than the motivation to explore facets of intelligence that engaged students in the AI-themed assignments. As Figures 4 and 5 suggest, it is also the challenge of the AI assignments that draws participation. From there, it is the open-endedness of AI-based pursuits that prompts further investigations and invites much deeper thinking than many other CS 1 assignments provide.

Perspective

The student-survey numbers of the previous section indicate only the most statistically defensible advantages that we have found using AI in CS 1. Less quantifiable, but more motivating for us instructors, are the handful of students who become passionate about one or more of the assignments in CS 1 and pursue them far beyond their original scope. The *Insomnia* team is only one example.

Many students get excited about Connect Four when their program defeats them for the first time. One student continued working on his static evaluator throughout the following semester and summer, returning triumphantly to demonstrate his resulting (very formidable!) C4 player.

We continue to use his software as an "unbeatable" example when teaching the class [13]. A quote from his page underscores the assignment's scalability:

> I've worked on this project for about a year and a half off and on through my freshman year and summer. The data structure has been rewritten about 4 times and I most recently reworked the entire applet structure. It now uses a better layout and has a more professional code style, which makes it easy to change. When I stopped working on it I had finally solved the horizon problem (with the exception of one case, try to find it) and surprisingly I used the solution to increase the algorithm's efficiency in the end game play.
>
> When I first wrote it, the game tree was only recursed 3ply, badly, in about a minute. Now it projects the game tree to 10ply and beyond in less than five seconds for a standard board and in about a minute for larger boards (This is dependent on the speed of your machine as well). This is accomplished with alpha-beta pruning among other pruning techniques to reduce the search space, as well as an extremely efficient non-trivial evaluation method that was implemented based on some heuristic graph theory.

The audio programming engages our many musicians in a more personal way than the other assignments – and the results have included a Midi-like interface that one enthusiastic student implemented atop the raw sound samples. The unnatural-language assignments are often where students feel their greatest sense of programming accomplishment: when their decipher function turns gibberish into English or when their models generate a surprisingly natural phrase.

Ultimately, it is this individual enthusiasm – and the choices that it can engender – that we strive to create as educators. As the student feedback suggests, AI-themed assignments – both at a high-level and low-level of abstraction from raw sensory input – provide a powerful combination of features. They are

- fun, in their contrast with our human abilities
- CS 1-accessible, with available support materials
- scalable, so that students see an path for further investigations extending all the way to open problems

Encouraged both by the numbers from our surveys of student opinion and by personal observations, we look forward to developing additional AI-based assignments in coming semesters. Computer vision is a natural domain we have yet to explore at the low-level, though others have reported success with AI-based pixel processing [14]. At the high-level, we hope to create scaffolding to wrap an accessible example from automated theorem proving. Regardless of the details, however, we look forward to working with other educators to broaden the audience and effectiveness of AI-themed assignments in the early CS curriculum.

Acknowledgments

The authors gratefully acknowledge support from National Science Foundation DUE CCLI #0411176 and funds provided by Harvey Mudd College.

References

[1] Fox, S. "Introductory AI for Both Computer Science and Neuroscience Students" – Proceedings, *FLAIRS* 2007.
[2] Ladd, Brian C. "Artificial intelligence in CS 1" *Journal of Computing Sciences in Colleges* , 20(4) pp. 101-103, 2005.
[3] www.cs.rit.edu/~cs1s/project/index.html
[4] cs.nyu.edu/courses/spring00/V22.0102-002/index.html
[5] Blank, D. S., Kumar, D., Meeden, L., and Yanco, H. "Pyro: A Python-based Versatile Programming Environment for Teaching Robotics." *ACM Journal on Educational Resources in Computing*, 3(4), pp. 1-15, December 2003.
[6] Tribelhorn, B. and Dodds, Z. "Evaluating the Roomba: A low-cost, ubiquitous platform for robotics research and education." *Proc. ICRA,* pp. 1393-1399, March 2007.
[7] Matarić, M, Koenig, N. and Feil-Seifer, D. "Materials for Enabling Hands-On Robotics and STEM Education", *AAAI Spring Symposium on Robots and Robot Venues: Resources for AI Education*, Palo Alto, CA, Mar 2007.
[8] Kumar, D. and Meeden, L. "A robot laboratory for teaching artificial intelligence" *In Proc, SIGCSE*, pp. 341-344, 1998.
[9] Fagin, B, Merkle, L. D., and Eggers, T. "Teaching computer science with robotics using Ada/Mindstorms 2.0" *ACM SIGAda Ada Letters* XXI(4), pp. 73-78, Dec. 2001.
[10] wiki.roboteducation.org/
[11] Forte, A. and Guzdial, M. Motivation and Nonmajors in Computer Science: Identifying Discrete Audiences for Introductory Courses. *IEEE Transactions on Education*, 48 (2). 248-253.
[12] deGroot, A. *Thought and Choice in Chess*, 2nd Ed. Mouton De Gruyter, publishers. 1978.
[13] http://www.stanford.edu/~ccecka/research/C4.html
[14] Wicentowski, R. and Newhall, T. "Using image processing projects to teach CS 1 courses." *In Proc. SIGCSE*, pp. 287-291, St. Louis, MO, 2005.

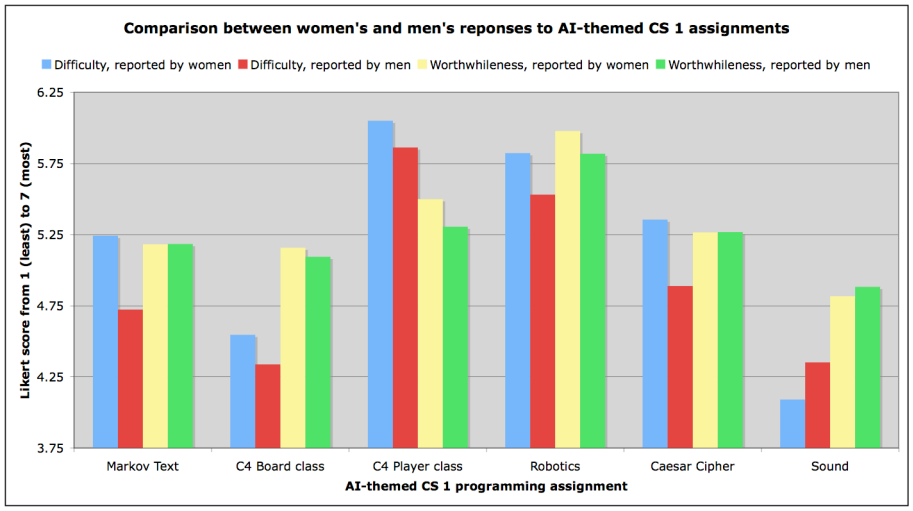


Figure 4. A comparison between female and male students' reported worthwhileness and difficulty for the AI-themed assignments in 2006's CS 1. Women found the AI assignments significantly more difficult than men, though no more so than for all of the assignments in the class. The data point to the success of all of the AI-themed assignments, along with a need to better scaffold our C4 player assignment!

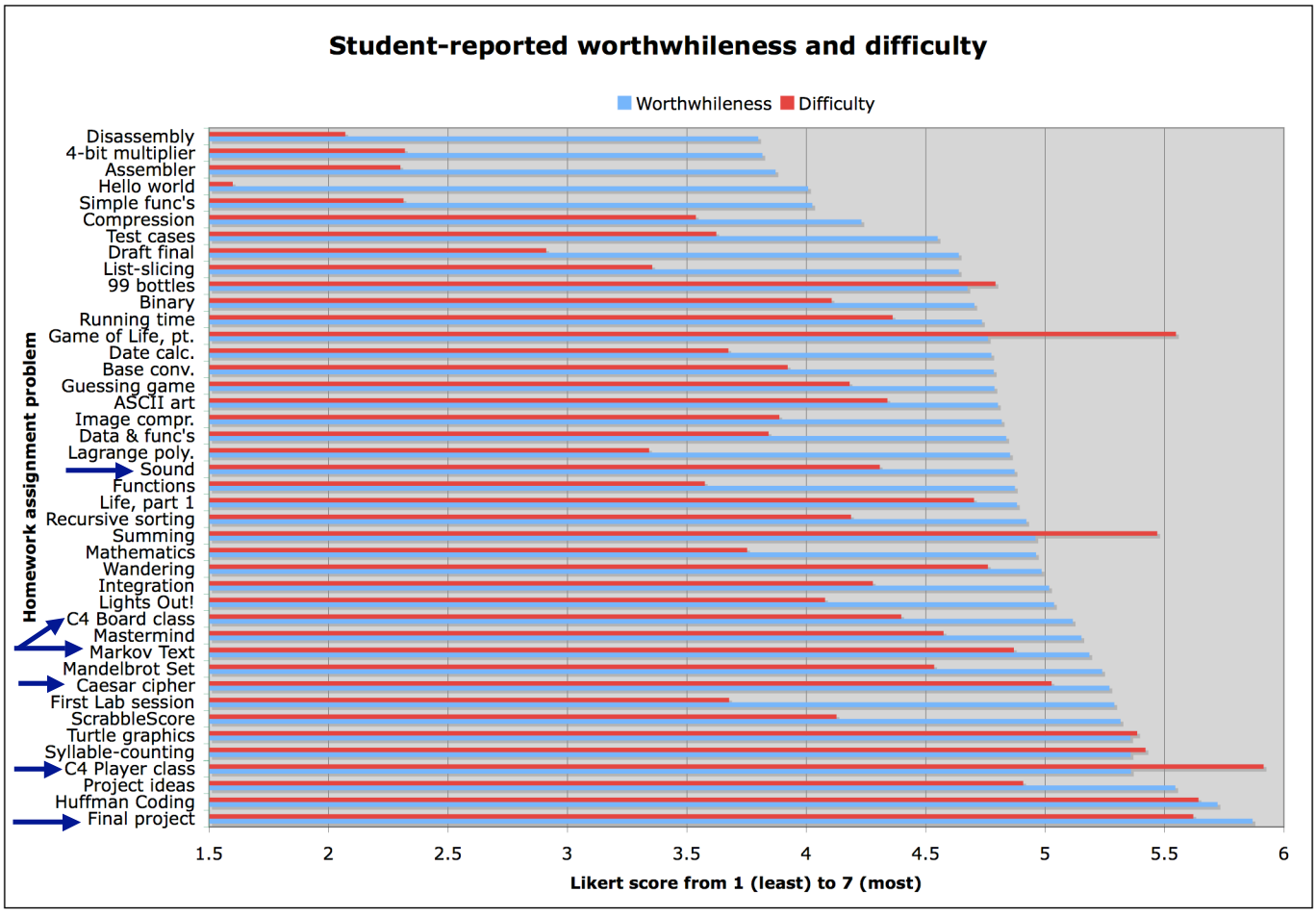


Figure 5. A direct comparison among students' reported worthwhileness and difficulty for the assignments in 2006's CS 1. The AI assignments (with arrows) are clustered near the positive side of the chart, indicating that they are motivating despite (or, perhaps, because of) their difficulty. The correlation coefficient between difficulty and worthwhileness is 0.83: that is, students appreciate challenges, as long as they feel they can make progress. AI offers a wealth of such challenging – but accessible – problems for early CS.