A New Start: Innovative Introductory AI-Centered Courses at Cornell

Eric Breck*, David Easley†, K-Y Daisy Fan*, Jon Kleinberg*,
Lillian Lee*, Jennifer Wofford‡, and **Ramin Zabih***

Departments of Computer Science* and Economics† and Faculty of Computing and Information Science‡
Cornell University
Ithaca, NY 14853

Abstract

We describe an array of novel introductory-level courses based on exciting topics in modern artificial intelligence. All present a great deal of often research-level technical content in a rigorous manner while keeping the material accessible to lower-level students. On the other hand, they differ in subject matter and style, since a "one-size-fits-all" approach cannot be expected to be effective given the wide variety of student interests and backgrounds. Thus, the courses cover topics ranging from computer vision to natural language processing to game theory and emphasize perspectives from hands-on implementation with robots to mathematical foundations to societal implications. The courses range in format from lab-style to seminar discussion-style to large lectures with out-of-class blogging activities, and some have been held during summer sessions expressly to attract high-school students.

The evidence shows that these courses are succeeding in drawing a broad audience to learn about ideas in computing. For example, several exceeded their initial enrollment estimates or limits; one drew over 200 students from 25 different majors in its first running; one reports over 30% female enrollment. Course materials are available on the Web and two textbooks based on some of these classes are in progress.

Pub. info: AAAI Spring Symposium on Using AI to Motivate Greater Participation in Computer Science, 2008.

Introduction

One of the current paradoxes in computer science is the increasing divergence between the research vitality of the field and its declining undergraduate enrollments. This decline is taking place at all levels, but it is particularly alarming at the introductory levels: in the US, the number of entering freshman expressing an interest in computing has fallen by 70% from Fall 2000 to Fall 2005 (Vegso 2007, quoting an HERI/UCLA survey). Thus, while there may also be problems in keeping upper-class students in the major once they enter it, or in maintaining the interest of those college students who originally intended to pursue CS-related studies, the far more significant battle is getting entering freshmen to consider the field in the first place.

At a superficial level, this drop in student interest may seem surprising in light of the pervasively digital culture in which these students are growing up. More than in any earlier period, students entering college are highly conversant with on-line environments and interfaces; indeed, many of these interfaces are compelling AI applications, including Web search engines, email spam filters, the grammar checker in Microsoft Word, automated telephone answering systems, and many others. Yet most students do not perceive the intellectual content that goes into the creation of these applications, and instead too often form a view of computer science that is shaped by negative stereotypes of the field. A successful introduction to computer science, then, needs to break through these preconceptions by exposing the deep ideas of computing and relating them to the concrete experiences of students and to the computing applications that they use every day.

An overview of the new Cornell courses. At Cornell, we have been actively pursuing such a plan to energize our introductory curriculum through the design of novel courses, many of them based on exciting topics in modern artificial intelligence. All of these new courses are rooted in a long tradition at Cornell of moving research material as quickly as possible into early undergraduate syllabi, and all of the courses emphasize ideas such as algorithmic thinking over simple skill acquisition. All have instructors who are actively engaged in research on the fields presented. And finally, they all also present a great deal of technical content in a rigorous manner while keeping the material accessible to lower-level students.

But although these courses hold the aforementioned fundamental tenets in common, by design the courses differ quite widely in subject matter and style — it is clear that a "one-size-fits-all" approach cannot work in the current setting. Students arrive with backgrounds ranging from fundamentally non-technical to years of prior computer programming experience, and they also come to college with interests that range all over the map. As such, the courses emphasize perspectives ranging from hands-on implementation with robots to mathematical foundations to societal implications; they range in format from lab-style to seminar discussion-style to large lectures with out-of-class blogging activities; and two have been held during summer sessions expressly to attract high-school students.

The use of AI as a basis for introductory courses has turned out to have many advantages. First, it imparts a cer-

tain concreteness and immediacy to the material that can be hard to achieve with traditional introductory programming assignments — for example, there is considerable appeal in getting a robot to actually move around and respond to its environment. Even early assignments can be crafted so that students are able to think about what is involved in creating agents that can behave "intelligently," and to see into the complexity behind applications that they have been familiar with for years. At an intellectual level, the long-term goal of making an entity that can act independently is very exciting (though, of course, also potentially scary — but many students find debating this point to be part of the thrill).

The interdisciplinarity of AI — always a part of the field, and only increasing in recent years — is another source of benefits in these courses. The ability to draw naturally on connections to psychology, economics, linguistics, philosophy, and other fields broadens the range of students to whom such courses can appeal. Moreover, many mathematically advanced topics that feature prominently in AI can actually be described fairly intuitively. For example, cutting-edge material such as linear discrimination or random-graph models can be introduced via geometric concepts that students have learned in high school or via "naive" probability (as described later).

The courses that Cornell has developed to incorporate AI ideas at the freshman level can be briefly summarized, in order of increasing hands-on contact with computers, as follows; a more complete description of each course appears in the sections below. (Note that the three non-programming courses do not exempt students from any programming requirements: if students choose to enter a computing discipline after taking them, they must achieve programming proficiency through the same sets of classes that all the other students do.) *Computation, Information, and Intelligence* is a non-programming, mathematically-oriented course that exposes students to algorithmic thinking as it arises in machine learning, search engine technology, and natural language processing. *Networks* is a course situated at the intersection of of computer science, economics, information science, and sociology; it explores how the social, technological, and natural worlds are connected, how agents interact in these systems, and how the study of networks sheds light on these phenomena. *Computation and Culture in a Digital Age* is an applications-oriented course for summer session high school students that pulls together AI, programming, Web development and social issues. *Introduction to Computation with Robotics* is an honors version of our standard Matlab introduction that uses camera-controlled robots — Sony Aibos (robot dogs) and the iRobot Roomba, both appealing platforms — to teach fundamental computer science concepts, with emphasis on computer vision techniques.

This curriculum evolution is intended to attract students with a wide variety of interests and experiences and to make more students aware of the opportunities that computer science offers. While many of the above courses are too new for us to evaluate formally, early evidence indicates that they will achieve their desired outcomes. For example, several exceeded their initial enrollment estimates or limits; one drew over 200 students from 25 different majors; one reports over 30% female enrollment; and all synthesize material that was recently viewed as research-level, much of it never having appeared before at the freshman level anywhere.

Pedagogical Framework

Ten years after the publication of the National Science Education Standards (National Research Council 1996), student-centered learning approaches have spread from their beginnings in K-12 classrooms to undergraduate courses across science, technology, engineering, and mathematics (STEM) disciplines (Apedoe & Reeves 2006; McIntosh 2000; Powers & Hartley 1999). This shift from old guard skill-transmission to more progressive "inquiry-based" and "constructivist" learning incorporates three primary pedagogic tenets: a philosophically inquisitive approach to curriculum where ideas and knowledge are discovered or created by students through teacher facilitation; *in situ*, often project-based problem-solving; and a focus on cross-context transferable skills like critical thinking and research design (Oliver 2007; Phillips 1995; Brown, Collins, & Duguid 1989).

Programming skills are important components of some of our new courses, and remain the central focus of our traditional "CS100-style" classes. However, the new suite of introductory courses is meant to broaden participation in computer science, focusing on fundamental CS concepts (in particular, algorithms and models) instead of, or in addition to, programming skills. By their very nature, concepts are amenable to this new style of pedagogy. Brown et al (1989, p. 32) discuss "situated cognition," a pedagogy that, like inquiry-based and constructivist educational approaches, privileges knowledge-in-use. In their discussion, a concept "will continually evolve with each new occasion of use, because new situations, negotiations, and activities inevitably recast it in a new, more densely textured form. So a concept, like the meaning of a word, is always under construction" (Brown, Collins, & Duguid 1989, p. 32). A concept-centered curriculum is therefore more naturally suited to inquiry-based and constructivist learning than skills-based approaches. Such a focus also allows us to present cutting-edge material without sacrificing rigor yet without mandating technical prerequisites. Students individually and collaboratively reconstruct a knowledge base from a common conceptual starting point, and instructors can assess students' understandings of the material from effective, situated application, rather than the often misleading skill-and-drill.

Computation, Information, and Intelligence

Computation, Information, and Intelligence is a non-programming yet rigorous and mathematically-intensive freshman-level introduction to computer and information science through the lens of AI, with major units built around modern machine learning, information retrieval, and natural language processing.

These topics are quite natural to use to attract students entering college to CS, since many of these students use systems that incorporate machine learning, information re-

trieval, and natural language processing technologies on a well-nigh daily basis. Examples are Web search engines such as Google and the grammar checker in Microsoft Word. (And, students certainly use natural language quite frequently.) Moreover, the goals of machine learning seem to embody precisely what young students (and others) often think are completely unachievable, and so are perhaps most apt to be curious about. For example, they already know that Deep Blue defeated Kasparov, but they are not particularly impressed, because they think that "machines only do what you tell them." They are thus quite struck to learn about systems that learned championship-level backgammon skills through self-play and cars that surpassed their human trainers at driving in reverse.

The decision to neither require nor teach programming has a number of advantages. First, it has the effect of deprivileging students who have had prior programming experience, thus serving to level the playing field and make the course more welcoming to students without such background. Second, it conveys the message that CS is not just about programming; rather, the focus is on models, algorithms, and critical thinking. Third, it makes coordinating with other course selections easier for students: the teaching of programming skills is done well in other courses, so there is no need to repeat such training in this class. Thus, coursework for the class currently consists of challenging "pencil-and-paper" problems. Finally, it has been suggested that an emphasis on hacking (i.e., the more "gory" aspects of programming) can be unappealing to certain populations of students (De Palma 2001). The fact that a relatively high number of women enroll in this class, as mentioned below, may be related to this suggestion.

The main question, of course, is how to present modern material in a way that is accessible to college freshmen, but that is still technically rigorous and promotes critical thinking — since, in the end, CS is in large part a technical field, and there is no point in misleading students about this fact. The transmission of critical-thinking skills is arguably one of the highest priorities in the educational mission of a university. What follows is a brief sketch of how various units of the course accomplish this goal.

- **Search and game-playing.** We begin with the most "traditional" unit. We find that freshmen have very few difficulties understanding the ideas of trees as data structures, depth- and breadth-first search, and elementary forms of alpha-beta pruning.
- **Perceptron-based learning.** Students are introduced to the idea of function computation by neurons, and hence to representation of instances by feature vectors, and from there to the notion of linear classifiers and perceptrons. We then present a version of the perceptron learning algorithm and go through a complete proof of convergence under certain assumptions. Crucial for this unit to work is that while background in linear algebra cannot be assumed, luckily, elementary trigonometry is still fairly fresh in ex-high-schoolers' minds, and so notions like orthogonal projection and inner product can be framed in terms of familiar geometric concepts like the cosine.

  In order to sharpen the students' understanding, the centerpiece homework problem of this unit involves asking the students to suitably alter the proof (or show how the proof breaks) if one of the initial assumptions is altered. This question illustrates one of the underlying themes behind the design of this course: one comes to truly understand a concept or algorithm if one can predict the effects of changes in the underlying assumptions, conditions, or steps. One of our deepest goals is to get students to independently ask themselves such questions later on in their academic (and "real") lives.
- **Information retrieval within the vector-space model.** Students are already quite familiar with search engines and their uses, if not with their internals, so little motivation is needed for this unit. Having already dealt heavily with vector-based operations in the previous unit, cosine-based retrieval comes naturally to the students. A typical homework problem here is to ask what is the effect of using a somewhat different form of a previously-introduced term-weighting function.
- **Link-based information retrieval.** Given students' prior experience with the World Wide Web, this unit tends to be one of the most popular. Their familiarity with the Web also means they find graph-theoretic notions (e.g., in-degree, connected components) intuitive. We study PageRank and hubs-and-authorities in technical detail, as well as random-graph models for describing the evolution of the Web graph. Interestingly, the iterative nature of the two link-based document-scoring algorithms echoes the iterative nature of the updates in the perceptron learning algorithm presented earlier.

  We make heavy use of probability in this section, both for the aforementioned random-graph models and for the random-surfer motivation of the PageRank equation, but are able to rely on an intuitionistic understanding of it. (We do not cover any eigenvector-based interpretations.) Our analysis of random-graph models does require some elementary calculus skills, but homework questions explicitly state the solutions to any integration problems the students are asked to face.

  A typical homework problem involves presenting the students with an example of a graph on which PageRank or hubs-and-authorities has unexpected behavior (e.g., the "dangling link" issue with the "usual" PageRank equation that leads to probability "draining out" of the system), having them verify through hand-simulation that a problem exists, and proposing how to modify the relevant equations or algorithms accordingly.
- **Natural language processing.** Relying on students' instincts about and experience with language, we introduce context-free grammars as a basic model of syntax, thus going beyond the "bag of words" representation from the previous unit. We then introduce Earley's algorithm as an example of an efficient dynamic-programming algorithm for parsing that uses some interesting, linguistically-motivated reasoning constraints (and is thus more complex than the CKY algorithm one usually meets in standard courses on the theory of comput-

ing). Not surprisingly, we then move on to the question of language learning, using statistical machine translation as the prime example. Here, the use of an auxiliary hidden variable (specifically, word-to-word alignments in the IBM expectation-maximization-based models) corresponds nicely with the idea of using the auxiliary information of "hubness" in the link-based information retrieval section.

We believe that "Computation, Information, and Intelligence" effectively conveys the notion that computer science is not just about programming, and the non-programming nature of the course appeals to a broad range of students. In the 2007 spring semester, 34% of the 47 students were women, and enrollees included a psychology major and a French/English double major. Interestingly, during an internship visit to the offices of the New York Times, this double major challenged an editor's assertion that on the Internet, all information is equal with a disquisition on PageRank. This anecdote underscores the importance of outreach introductory courses: even those who do not end up pursuing technical disciplines can be empowered by understanding more about computer and information science.

Lecture guides are available at `http://www.cs.cornell.edu/courses/cs172`, and a textbook based on the course is currently in progress.

Social and Economic Interaction on Networks

The study of networks focuses on how the social, technological, and natural worlds are connected and how the structure of these connections affects each of these worlds. The topic thrives on insights that reach across very different domains. For example, models for the spread of epidemic diseases have been used to analyze the movement of fads and rumors through word-of-mouth communication; principles behind cascading failures in power grids have been applied to cascading extinctions that unravel food webs as well as economic crises that spread through the worldwide financial system; and the notion that certain individuals occupy powerful roles in social networks has helped form the basis for link analysis ranking methods in Web search engines.

In this way, the subject has a natural inter-disciplinary appeal, and in particular connects to the interests of many entering college students, who — regardless of their intended majors — inhabit on-line networks such as Facebook and tend to see themselves as part of an increasingly "connected" world. Motivated by these developments, we designed an introductory undergraduate course entitled *Networks*, which covers social and economic interactions in network settings. Cross-listed in computer science, economics, information science, and sociology, it seeks to cover the recent developments in this subject, and more generally to use the topic as a way of conveying both fundamental ideas from computing and information as well as fundamental mathematical models in the social sciences.

This blending of areas is consistent with a growing interest by computer scientists — from both artificial intelligence and other subfields —- in ideas from economics and sociology, including game theory, agent interaction, and social networks. Within this context, it becomes possible to convey these current trends in computer science to students who might never have otherwise intended to take any courses related to computing during their time in college.

Interest in the first offering of the course, Spring 2007, exceeded our expectations; it came from many parts of campus, with an enrollment of over 200 students representing 25 different majors and intended majors. The second offering of the course in Spring 2008 attracted an even larger number of students from a still broader diversity of majors.

In addition to other parts of the coursework, students actively contributed to a class blog, where they wrote about connections between course topics and material they had found in current news or elsewhere on-line. A number of blog posts attracted positive responses from the outside world, including from founders of technology companies and press liaisons at financial institutions. The class blog is available at `http://expertvoices.nsdl.org/cornell-info204`, with a digest version including comments by the course staff at `http://expertvoices.nsdl.org/cornell-info204-digest/`.

To give a more specific sense for the focus of the course, we now describe a brief outline of the content, organized around five main topics. The first two topics served as background in context for the main analytical techniques, and the latter three topics illustrated extended applications of network thinking.

- **Graph theory and social networks.** The course begins with a discussion of graph theory as the area of mathematics that studies networks. It develops this through examples from social network analysis, including Granovetter's famous "strength of weak ties" hypothesis, and connects this to recent large-scale empirical studies of on-line social networks.
- **Game theory.** Since most network studies require us to consider not only the structure of a network but also the behavior of the agents that inhabit it, a second important set of techniques comes from game theory. This too is introduced in the context of examples, including the design of auctions and the Braess paradox for network traffic.
- **Markets and strategic interaction on networks.** The interactions among participants in a market can naturally be viewed as a phenomenon taking place in a network, and in fact network models provide valuable insights into how an individual's position in the network structure can translate into economic outcomes. Thus, this topic provides a very natural illustration of how graph theory and game theory can come together in the development of mathematical models. Our discussion in this part of the course also built on a large body of sociological work using human-subject experiments to study negotiation and power in networked settings.
- **Information networks and the World Wide Web.** The Internet and the Web of course are central to the argument that computing and information is becoming increasingly networked. Building on the earlier course topics, we describe why it is useful to model the Web as a network, discussing how search engines make use of link information

for ranking, how they use ideas related to power and centrality in social networks, and how they have implemented network-based matching markets for selling advertising.

- **Cascading behavior in networks.** Finally, networks are powerful conduits for the flow of information, opinions, beliefs, innovations, and technologies. We discuss how models of agent interaction can give us ways of reasoning about processes that cascade through networks. In particular, we describe "herding" models in which agents make decisions based on Bayesian analysis, and diffusion models in which agents are embedded in networked coordination games. Here too, we connect the models to recent empirical studies, illustrating for example how the rich-get-richer dynamics inherent in cascades lead to heavy-tailed distributions for on-line popularity.

Handouts, problem sets, readings and references are available at `http://www.infosci.cornell.edu/courses/info204/2007sp/`, and a textbook based on the course is in progress.

Computation and Culture in a Digital Age

Offered for the first time in 2007 through Cornell's Summer Explorations Program for high school students, *Computation and Culture in a Digital Age* is an introduction to computing and information science from both the technological and social perspectives. The goal of the course is to show the students, most of whom are about to apply to college, the wide range of exciting opportunities in computing and information science. We intend to follow up with the students, tracking their college applications to see how many enroll in computer science or related programs.

AI makes up one quarter of the course contents — the other topics are programming, user-centered Web development, and the legal and social issues of computing. Technologies and issues that are relevant to the students' everyday life are highlighted. In the AI portion of the course, the central theme is for the students to understand that intelligence isn't a monolithic concept realized only in fictional androids, but instead is a collection of behaviors that can be individually understood and implemented. To this end, the course discusses three research areas of AI: natural language processing, information retrieval, and machine learning. In each area, we focus on one application, addressing both the implementation (at a high level) and the user's perspectives.

- **Natural language processing.** We discuss the problems posed by ambiguity at all levels of language. The students explore some of these issues by solving puzzles from the North American Computational Linguistics Olympiad. We discuss practical applications of natural language processing, including question answering and summarization, and then delve into the problem of machine translation. In a lab exercise, students evaluate a number of publicly available translation systems.
- **Information retrieval.** As students are already familiar with Google and other Web search engines, we emphasize the connection between Web search and other information retrieval technologies, such as desktop search and library indexing. We introduce some of the basic ideas of information retrieval, such as indexing, and have students compare and evaluate different search engines.
- **Machine Learning.** The idea that machines can learn is a popular one in science fiction — the students can readily name movies that feature some kind of (fictional) machine learning. We discuss the state of the art in machine learning and introduce the concepts of training and test sets, the basics of several types of machine learning models, and we explore how to evaluate them. Continuing with the theme of human language technologies established so far, we introduce the problem of spam classification, culminating in the construction of a Naive Bayes spam classifier.

In addition to topics in AI, other activities in the course include programming to manipulate digital media, conducting Web usability tests, and debating issues such as illegal music sharing and the use of blocking and monitoring software in high schools. We also present a mini seminar series to showcase other research activities and facilities in computing and information science. By exploring the digital technologies that students use everyday and studying the related issues, we show students that the field of computing and information science is broad, exciting, and has important humanistic and societal dimensions.

Slides and problem sets for this course are available at `http://www.infosci.cornell.edu/courses/info153/2007su/`.

Introduction to Computation with Robotics

Introduction to Computation with Robotics is an honors-level introduction to computer science based on camera-controlled robots. The course primarily emphasizes the issues involving sensing, rather than reasoning or control, for several reasons. First, many exciting computational issues naturally arise in the analysis of real-world data. For example, it is necessary to robustly handle noise and to perform some kind of model fitting. Such issues arise quite naturally in the context of machine perception. Second, Matlab provides excellent tools for the analysis of noisy data sets, in an interpreted environment that is easily accessible to freshmen. Finally, sensing is a vital component of any robotic system; sophisticated sensing capabilities, combined with relatively simple reasoning and control, can lead to impressively complex robotic behavior.

The topics in the course are focussed on creating sensing capabilities for robots that would be of broad use (in fact, the topics are to some degree motivated by the common sensing requirements of student-run robot projects, such as Robocup). In the first portion of the course, the goal is to determine the position and orientation of a red light stick (like those used to guide an airplane into its gate). This leads to numerous important algorithms such as sorting, median finding, depth-first search, connected components, and convex hull. Even advanced topics, such as robust statistics, graph algorithms and problem reductions, naturally appear in simplified forms.

In a later project, the students use least-squares fitting to analyze odometry data from the robots in order to build a robot speedometer and accelerometer. This introduces some of the math behind least squares, and the students implement a simple form of gradient descent. In another project, the students design simple clustering algorithms to enable the robot to distinguish between primarily red objects (e.g. Coke cans) and primarily blue ones (e.g. Pepsi cans).

The robots are a combination of the Sony Aibo (robot dogs) and iRobot Create (which is essentially a Roomba without the vacuum cleaner attachment). At the end of the course, the students complete a final project of their own choosing, drawing on the robotics capabilities that they build up over the semester. Some of these projects have proven to be quite impressive, with teams of robots cooperating to accomplish a task.

Slides and problem sets for this course are available at `http://cs100r.cs.cornell.edu`.

Conclusion and Future Directions

We have described four new courses at Cornell that, while differing widely in certain important respects, all focus on important topics in modern AI and use these topics to interest students in the ideas in CS as a whole. Course materials we have developed are available on the Web, and we are in the process of writing two textbooks based on some of these classes.

It should be mentioned that the four courses described in this paper are part of a broader initiative at Cornell in reenvisioning the entire introductory computing curriculum. Other efforts in progress include the following: an honors version of our standard Java introduction featuring projects that are much less structured and, as a result, much more research-like; a course entitled *Computing in the Arts* that looks at randomness and stochastic processes in the context of poetry, visual art, music, and sculpture, with some Java programming; self-paced ("auto-tutorial") on-line introductions to programming; and a course entitled *Visual Imaging in the Electronic Age* that integrates ideas from architecture, art, and computer science

We believe that through these innovations, we will be able to bring more students into CS and related majors, and, just as importantly, that we will be able to bring key CS ideas to more students regardless of their eventual choice of academic and career paths.

Acknowledgments

We owe a large debt of gratitude to Charles Van Loan, former Chair of the CS Department, who inspired and encouraged us to create these courses and who provided very generous support of many kinds during the process of development; the new suite of introductory courses is due in large part to his vision, creativity, and commitment to excellence in undergraduate education. We are also grateful to all the teaching assistants and course consultants who have helped us with creating and maintaining these courses: Steve Baker, Jared Cantwell, Tze Jian Chear, Chris Danis, Ray Doyle, Rafael Frongillo, Nick Gallo, Jon Guarino, Abraham Heifets, Amanda Holland-Minkley, Marek Janicki, Tian Liang, Homan Lee, Yuzhe Liu, Selina Lok, Ezra Kannof, Devin Kennedy, Blazej Kot, Elliot Kulakow, Shannon McGrath, Brian Mick, Anton Morozov, Milo Polte, Ben Pu, Neeta Rattan, Brian Rogan, Gurmeet Singh, Sara Tansey, Mark Yatskar, Adam Yeh, Chong-Suk Yoon, and Yisong Yue. We also thank Stephen Chong and the reviewers for helpful comments on earlier drafts of this paper.

This paper is based upon work supported in part by the National Science Foundation under grant nos. IIS-0329064, CCF-0325453, and CNS-0403340, and BCS-0537606, a Cornell University Provost's Award for Distinguished Scholarship, a Yahoo! Research Alliance gift, a grant from the G.E. Fund for furthering educational outreach to underrepresented groups, an Alfred P. Sloan Research Fellowship, Intel, Microsoft, the GRASP laboratory at the University of Pennsylvania, Cornell's Faculty Innovation in Teaching Program, a Google Research grant, the Institute for Social Sciences at Cornell, and the John D. and Catherine T. MacArthur Foundation. Any opinions, findings, and conclusions or recommendations expressed are those of the author and do not necessarily reflect the views or official policies, either expressed or implied, of any sponsoring institutions, the U.S. government, or any other entity.

References

Apedoe, X. S., and Reeves, T. C. 2006. Inquiry-based learning and digital libraries in undergraduate science education. *Journal of Science Education and Technology* 15(5):321–330.

Brown, J. S.; Collins, A.; and Duguid, P. 1989. Situated cognition and the culture of learning. *Educational Researcher* 18(1):32–42.

De Palma, P. 2001. Why women avoid computer science. *Communications of the ACM* 44(6):27–29.

McIntosh, W. J. 2000. Beyond 2000 - the changing face of undergraduate science education. *Journal of College Science Teaching* 29(6):379–380.

National Research Council. 1996. *National science education standards*. Washington, D.C.: National Academy Press.

Oliver, R. 2007. Exploring an inquiry-based learning approach with first-year students in a large undergraduate class. *Innovations in Education and Teaching International* 44(1):3–15.

Phillips, D. C. 1995. The good, the bad, and the ugly: The many faces of constructivism. *Educational Researcher* 42(7):5–12.

Powers, M. L., and Hartley, N. K., eds. 1999. *Promoting Excellence in Teacher Preparation: Undergraduate Reforms in Mathematics and Science*. Fort Collins, CO: Rocky Mountain Teacher Education Collaborative.

Vegso, J. 2007. Continued drop in CS bachelor's degree production and enrollments as the number of new majors stabilizes. *Computing Research News* 15(2):4.