Operations Research: Broadening Computer Science In A Liberal Arts College

Barbara Anthony
Southwestern University
1001 East University Avenue
Georgetown, TX 78626
anthonyb@southwestern.edu

ABSTRACT

Operations research, while not traditionally taught at many small or liberal arts colleges, can be a significant asset to the offerings of a computer science department. Often seen as a discipline at the intersection of mathematics, computer science, business, and engineering, it has great interdisciplinary potential and practical appeal, allowing for recruitment of students who may not consider taking a CS0 or CS1 course. A special topics course in operations research was offered by the computer science department at Southwestern University as an upper-level elective, and it was also cross-listed as a business and mathematics elective. Not only did the course benefit computer science majors who appreciated the applications and different perspectives, but it provided a means for the department to serve a wider population, increased interdisciplinary education, and resulted in a filled-to-capacity upper-level course in computer science for the first time in recent memory. This course is now being considered as a permanent elective that will interest computer science majors and minors as well as draw in students from disciplines across campus. For departments with limited faculty resources for teaching non-major courses, offering an operations research course provides an alternative that simultaneously serves the department and the campus as a whole.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education

General Terms

Experimentation, Management

Keywords

computer science education, interdisciplinary, liberal arts, non-majors, operations research, outreach

1. INTRODUCTION

While larger universities may be facing a boom in computer science enrollments [19], smaller liberal arts colleges do not always experience the same trends [12]. Furthermore, at some liberal arts colleges, the question arises: does computer science have a place in a liberal arts education, and if so, what should its role be [17]? Liberal arts colleges aim to provide both breadth and depth, exposing students to a wide variety of disciplines and topics, as well as delving into their chosen major. Sometimes it is equally important to recruit students to take one or two classes in the discipline as it is to recruit majors or minors.

A variety of strategies have been tried to recruit students, but many of them have focused on getting students into the major or minor by having a broad CS0 or using CS1 as a recruiting tool [16]. While our CS0 course is marketed to all students, and our CS1 course requires no previous programming experience, we have had limited success with that approach at Southwestern University. All mathematics majors are required to take CS1, but few choose to take additional computer science courses. We do not have the faculty resources that some larger institutions have to teach additional courses that are geared towards non-majors, nor a large enough student body to have different versions of CS1 based on the audience.

Interdisciplinary courses and programs are strongly encouraged at many liberal arts institutions. Such ventures have been attempted and studied, including examples such as a "CS0++ course [combining] CS with statistics, ecology, and the philosophy and history of science" [10], and through "interdisciplinary application tracks [including] bioinformatics, computer information systems, geographic information technology, and human computer interaction" [18]. While courses of that type show promise as entry points for potential majors and minors, there is still room in the curriculum for students who are willing to consider a computer science course, but for various reasons do not want to take CS0 or CS1. Various universities have offered what are typically upper-level courses in computer science to students without the standard prerequisites, including "open AI" [14] and a database course designed for non-majors [11]. In part due to the large number of business majors at Southwestern University, we saw the potential for an upper-level elective in operations research, that could be cross-listed under business, computer science, and mathematics.

Operations research was a natural choice in our situation due to existing faculty member expertise, but there are other motivations for its selection which make it worth considering

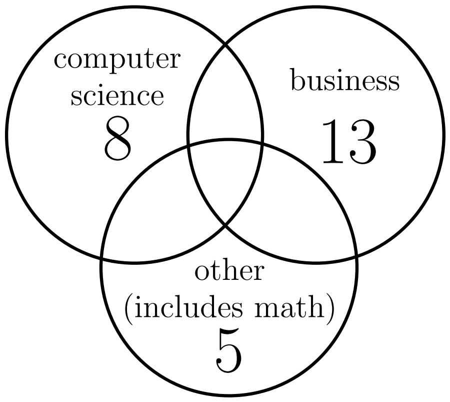

Figure 1: Class enrollment by major in Fall 2010; there were a few major/minor pairs, but no double majors.

by other colleges. At Southwestern University, since Computer Science and Mathematics are a combined department, with some faculty teaching in both disciplines, many students are aware that a number of the courses can be quite beneficial to students of either major. Some colleges where departments typically operate more independently may benefit from similar cross-listing. Recently, Macalester College introduced a course sequence in computational science which they have found beneficial in linking computer science not only to mathematics but also to the other sciences [15].

Integrating operations research with computer science is hardly unique. Carnegie Mellon University and Georgia Institute of Technology both have PhD programs in Algorithms, Combinatorics and Optimization which include the departments/schools/colleges of computer science, mathematics, and either business (operations research group) or industrial and systems engineering [2, 5]. Yet the offering of operations research is far from standard in undergraduate computer science curricula.

2. COURSE CONTENT AND STRUCTURE

2.1 Prerequisites

Teaching an upper-level course across disciplines requires careful planning, especially when there is no common core of mathematics or computer science courses that all students must take (as there might be at an engineering school). Mathematics majors at Southwestern University are required to take CS1, computer science majors have mathematics requirements including calculus and linear algebra, while business majors typically take statistics and calculus. Thus, to make this course truly accessible to students in all three majors, we could not require courses that otherwise might have been desirable prerequisites, such as linear algebra, algorithms, or even CS1. Actual enrollment numbers from the different majors are illustrated in Figure 1. Within each of the three majors, one or two students had a declared minor in one of the other listed majors.

A prerequisite of junior standing was used as a common way of ensuring a certain level of maturity of the student; case-by-case exceptions were made for a few students, primarily computer science majors. More important than the official prerequisites in guiding the appropriate self-selection of the course were email descriptions of the course, posted information on departmental bulletin boards, and informational presentations given in targeted courses, including the business capstone. Students were advised that they would need to be able to perform calculations (calculators or computers allowed), would need to use some software but would not need to know a programming language, and would be expected to consider problems from real-world perspectives and write for both general and technical audiences.

2.2 Resources

Numerous resources are available regarding operations research. Some of the materials that were most useful for this course came from INFORMS (Institute for Operations Research and the Management Sciences), and its Forum on Education [4]. The 2010 INFORMS Annual Meeting session on "The First Undergraduate OR Course" included a presentation by Martonosi of Harvey Mudd College. Her talk recognized that students may encounter operations research only once in their undergraduate career and they can be shown "both the power of O.R. to solve complex, real-world problems and its rich theoretical underpinnings" [13]. The operations research literature also talks about the value of 'real-world' projects, whether as one component of a course or as the focus of a capstone course in a program where the "OR-affiliated faculty includes members from the Departments of Mathematical Sciences, Management, Computer Science, and Economics and Geography" [8].

While there are some good, traditional choices for a textbook for a first course in operations research, they may assume the students have a particular background. Books aimed at business students may have a more spreadsheet-based design, focusing on the use of Excel without providing the same theoretical understanding. Other texts assume a programming or mathematical background that is often limited to either computer science or mathematics majors. For this course, *Schaum's Outline of Theory and Problems of Operations Research* [9] was chosen. It is intended more as a reference with a compendium of solved problems, highlighting the main ideas in a much briefer manner than a traditional text. Additionally, it allowed more freedom to select topics and order them as desired, without as much dependence on previous chapters. The relatively low cost as compared to most textbooks was also a factor in its favor. As is the norm for books from this series, all problems in the book are solved, so homework assignments pulled straight from the text do not work. In our case, many homework problems were devised independently, some were taken from other sources, and some required students to either arrive at the same answer as the text using software, or to provide a detailed written explanation of the answer and justification of the steps. Many students can benefit from practicing their technical written and explanatory skills, and when the answer is provided, it reduces complaints about points being deducted in spite of achieving the correct numerical solution.

A variety of software is available for solving linear and integer programs, both on a small scale and for commercial applications. Since many college students are familiar with Microsoft Office products, using the Excel Solver Add-In by Frontline Systems [1] that is included with the Windows version of Excel (a Mac version is available for download) provides a low barrier to entry. The GNU Linear Programming Kit (GLPK) [3] is a powerful package, capable of accepting a variety of input formats, but is often more intimidating to students without a computer science background. Google Docs Spreadsheet also provides a Solver [6], but with far fewer features (in particular, no sensitivity analysis). Note that while the options all provide subsets of the same numerical information, the terminology used differs; to mitigate this potential source of confusion for students, instructors can use this opportunity to talk about how names develop, simultaneous discoveries, and how applications can affect naming conventions. For some courses and students, it may be appropriate to have students write their own solver implementing the simplex method, among other algorithms.

2.3 Integrating the Students and the Material

As a special topics elective that does not serve as a prerequisite for future courses, there was some degree of freedom in the material covered in this operations research course. Keeping in mind the audience, it was designed so that all students would benefit, regardless of their background in algorithms, real-world case studies, and proofs. Certainly linear and integer programming would be included in an operations research course. The term *programming* alone leads to interesting discussions about the histories of computer science and operations research, as many student (regardless of their major) associate the word programming with programming languages. The course also provides numerous opportunities to reiterate that computer science is far more than 'just' programming. Solving linear programs by the simplex method is one of the earlier algorithms introduced in the course, though the graphical technique for lower-dimension problems is also introduced. Students also encounter duality early in the semester, and gain an understanding of the economic interpretation of the dual and that the dual is a powerful concept elsewhere in computer science. The dual aids in the study of sensitivity analysis, which also leads to questions about the accuracy of data and assumptions.

Throughout the study of various types of linear programs, the necessity of exact terminology was evident. Naturally, binary numbers were familiar to computer science and mathematics majors, yet their utility in ensuring certain constraints (such as exactly one of these five situations must happen) was novel to many. Having to think about variables and numbers as being binary, integer, or general, as well as the importance of non-negativity constraints, gave new significance to those terms. Likewise, the word *or* can be ambiguous or context-dependent in English as to whether or not it is an exclusive or, yet has a precise meaning in the realm of computer science. The importance of word choice was evident when students had to translate between word problems and systems of equations.

Sometimes computer scientists think of trees as so fundamental to their work that they forget that what they think of as a tree is not an innate concept for much of the world. Branch and bound techniques for integer programming provide a memorable example of how a tree can be used in storing relevant information and making decisions in a relatively efficient way. Yet, at the same time, these problems quickly illustrate the potential complexity of integer programming, making the exponential growth abundantly clear.

Algorithms were familiar to all students to various extents, though perhaps not by that terminology or with much thought about runtime or resource usage. Throughout the semester, students saw a variety of algorithms, with varying degrees of difficulties in implementation, explanation, and motivation. For instance, understanding the simplex algorithm in two dimensions leads to reasonable intuition about why it succeeds for problems with many more variables. It provides the rationale behind the steps in the manipulation of the linear equations, but does not generally make the solution trivial, whether the student is calculating it by hand or writing a program to complete the steps. Degenerate cases and ambiguous procedures (such as tie-breaking) show up and illustrate the need for handling all cases. Students who had previously taken an algorithms course were reminded of the variety of algorithms available for network analysis problems, and how a graph representation of a problem can be useful. All students saw that the edges on a graph and associated numerical values can have a variety of meanings, among them measures of capacity or speed, indicators of quantity shipped or used, or indicating precedence for project planning.

Table 1: Major Topics Covered

Mathematical Modeling
Graphical Solutions of Linear Programs
Simplex Method
Duality and Sensitivity Analysis
Integer Programming
Transportation Algorithm
Assignment Problem
Traveling Salesman Problem
Network Analysis
Maximum Flow and Minimum Cut
Project Planning

The Traveling Salesman Problem (TSP), a familiar topic in algorithms and complexity courses in computer science, can be introduced from a different perspective in operations research. First, cover the assignment problem which has a simple solution procedure (the Hungarian Method) and obvious applications. From there, proceed to the TSP, and show that a solution to the assignment problem may (but does not have to) satisfy the TSP constraints. Then, apply branch and bound strategies to either require or preclude one of the assignments. For students who have already learned that TSP is NP-hard, it is often revealing to see that two straightforward procedures can be combined to solve the TSP, yet the growth of the branch and bound tree presents an obvious challenge, illustrating the hardness of the problem in a more concrete manner. A listing of the major topics and the order in which they were covered is provided in Table 1.

Mathematics and computer science majors both have substantial experience with proofs, but often find them rather unrelated to the applications. However, in this course they were provided with more examples where the proofs were necessary to truly understand the algorithms and solve the problems. For instance, with the simplex method, even with the correct intuition that optimal solutions occur only at the corners, or on boundaries/faces between optimal corners, a proof is still necessary. While the proof is not that difficult, it is also not trivial, and provides students with an example where it is easy to see how the proof develops. As both mathematics and computer science majors at Southwestern University take linear algebra, they were provided with more examples of how linear algebra has both theoretical and applied or practical uses.

Generally, each group of majors brought a somewhat different set of strengths to this course. Mathematics majors were the group most familiar with using geometry to solve problems, despite its relevance in computer science. Yet they were often the least likely to be comfortable with the applications, and applying what had been proved to real-world problems. Business majors were comfortable with case studies with real problems, including picking out which information was pertinent, but often struggled with translating that information into equations when needed. To varying degrees, all groups of students struggled with letting the algorithm do its work to find an optimal solution rather than trying to either seed it with a perceived good value or adding constraints that were not truly part of the problem but 'seemed logical'. To that end, it was useful when the instructor could provide specific, realistic counterexamples to these inaccurate assumptions, enabling students to develop that skill as the semester progressed.

As mentioned, different software options were available for students to use, primarily the glpsol solver in the GLPK, and Excel's Solver. Both were used for in-class demonstrations. Interestingly, while Excel was familiar to the majority of the business majors, there were computer science majors who had not previously used it. Not surprisingly, most students quickly mastered the necessary tasks in Excel. Several computer science majors observed that portions of Excel were intuitive because it prompts the user that one of the inputs to the sumproduct function should be an array. The computer science majors are quite familiar with the concept of an array, and were easily able to explain it to some of their classmates. The particularities of the various file formats required by glpsol reinforced (especially for the non-computer science majors) that a computer can only understand syntax which it has been programmed to accept, and reminded the computer science majors of how they should think about their design choices and what a parser has to be able to handle.

2.4 Group Project: Case Study

All students were required to complete a semester-long group project, a case study, modified from one used at Harvey Mudd College. Due to the composition of the class, each group of approximately four students was required to have at least one computer science major/minor, and at least two business major/minors. With that requirement, students were allowed to self-select their groups. The mixed nature of the groups was intended to provide different perspectives and background knowledge, and to ensure that at least one student would be comfortable using glpsol if the size of the problems warranted, as the built-in Microsoft Excel Solver limits users to 200 changeable cells or decision variables.

To ensure adequate progress throughout the semester, there were several intermediate deadlines. Selection of groups was completed within the first weeks of the semester. A multi-page proposal about the problem, the work completed, and the outline of future progress was due one-third of the way through the semester. At this time, students were provided substantial feedback about their decisions thus far, given advice about factors they should consider that they had not, and guided where necessary to ensure that the scope of the project was reasonable given the constraints of the semester. Halfway through the semester, each student in each group had to meet with another student to discuss the status of their project, with the listener writing up a brief status report and providing additional feedback. In this way, each group collectively got feedback not only from the instructor but from (typically) four of their peers, and had to choose how to reconcile the potentially conflicting recommendations. A final portfolio was due at the end of the semester, which had to include both a brief executive summary (aimed at a non-technical manager) as well as a full technical report, complete with recommendations, acknowledged limitations, and sensitivity analysis. Students recognized throughout the semester that some assumptions had to be made, and some data was likely to be inexact, gaining a better appreciation of why sensitivity analysis is important. Many also experienced the potentially undesirable effects of adding constraints because it seems like they will be necessary in the final solution, only to discover that linear programming does best when given the freedom to find optimal solutions given only the true constraints.

Students were provided with some topic suggestions, but were free to select their own, which all groups did. Some students chose projects that were relevant to the campus or the community, such as equipping the redesigned athletic training room, or how the local food pantry should distribute items to satisfy nutritional and taste requirements while serving the needs of clients. Other groups thought about problems related to their jobs and hobbies, including stocking a bar and constructing the ideal basketball team with a salary cap. Students were sometimes creative and usually successful in finding information relevant to their problem, and the campus offices and community organizations seemed willing to provide what they needed. For instance, a group whose project involved the location of materials in the library was able to get data about the number of times each item in the collection had been checked out in various recent time intervals. While students were not required to implement their solution (due to logistical constraints), in some courses that may be an option, and it may be possible to work directly with campus or community organizations, though that might be more appropriate for a capstone course. Students potentially can continue or expand their projects as independent studies in subsequent semesters; as many of the students graduated at the end of the semester, none took advantage of that opportunity this time.

3. COURSE HISTORY AND LOGISTICS

The idea of offering an operations research course at Southwestern University was first conceived when the author used operations research techniques to solve a recurring scheduling problem on campus [7]. Each fall, approximately 30 first-year seminars are taught, and each seminar has an orientation to the library, as well as some other common modules. Typically this scheduling had been done by hand by administrative staff, on an ad hoc basis. Since it is fairly straightforward to devise the appropriate integer program to find an optimal solution to scheduling these modules, incorporating the desires of the faculty and the librarians as well as space limitations as constraints, we volunteered to do so, using a Java program to efficiently write out all the constraints in an appropriate format that could be handled by the solvers in the GNU Linear Programming Kit. The results were well-received, and provided a means of sharing some applications of computer science beyond the games, popular software, or web-based applications that many typically consider.

Some assignments were designed to target the students' different majors. For instance, students could either solve several examples of the transportation problem by hand using the transportation simplex algorithm, or they could work from prototype code to implement the missing details. In either case, students would become familiar with the steps of the algorithm; as would be expected, only computer science majors chose the code, but some business majors did discuss some of the ideas of the code and what was involved with their peers outside of the classroom. While part of the motivation may have been determining perceived fairness of the options, it reinforced the idea that computers are great tools to automate repetitive processes, yet can only do what the programmers specify. Due to the nature of cross-listing at Southwestern University, students can designate the course as business, computer science, or math, and make changes retroactively, so we could not require students to complete particular assignments or enroll based on their major, as is often done in various universities (for instance, with a course that is both undergraduate and graduate level). While perhaps a desirable option, careful assignment design and targeting can help most students select the variant that is most appropriate for them. Likewise, such differentiated assignments can help ensure that the course is rigorous enough for each major yet mindful of the different backgrounds and expectations of the students in each major.

3.1 Students: Recruiting and Enrollment

The first key to success in this course was having buy-in from all of the departments. As mentioned, when computer science and mathematics are a combined department, communication between those faculty is easy. In our case, the business program readily recognized the value of this course, did not have anyone in their department who was excited about teaching it themselves, and had an abundance of students so there were no concerns about this course affecting the size of their elective courses. In fact, they invited us to use their internal channels (bulletin boards, departmental listservs, etc.) to share information with potential students.

The second factor involves the students. While students will often take a cross-listed course taught by a faculty member in their discipline, a business major can be understandably reticent about taking a course taught by a computer science faculty member. In our case, it helped immensely that we had a few business majors who had taken one or more computer science classes, who became the champions of the operations research course to their fellow business majors. Additionally, students generally like to have their friends in classes, so this course provided a way for computer science majors to encourage others to take the course in a way they typically can not. Finally, since students were provided evidence that operations research techniques had been used successfully to solve a campus problem (scheduling first-year seminar modules) the course was more appealing.

Courses at Southwestern University are typically capped at 25 students, but computer science classes are often smaller. It is not unusual to have upper-level elective courses have single digit enrollment. In part because of the promotion of the special topics course, all the available spaces filled during preregistration, and we allowed a few students to enroll above the cap, knowing that there are typically some registration changes during the first week of the semester. The semester ended with 26 students completing the course, and only one student dropping after the first week of classes. Computer science majors and minors made up a substantial percentage of the course, but were not the majority. Enrolled students listed majors in accounting, business, computer science, economics, mathematics, and sociology. Having a full class is good for increasing awareness of the discipline to students, faculty, and advisors on campus, and is also viewed favorably by the administration. It is unlikely that Southwestern University course sizes will change by an order of magnitude, but the course has the potential to scale for larger class sizes, especially since much of the group work is done independently of the instructor.

3.2 Feedback and Evaluation

The first time teaching a course is rarely perfect, and this course proved no exception. Despite being aware of the mathematics cross-listing, there was the business major who expressed irritation at having to do arithmetic in the simplex method. There were the computer science majors who wanted to deal only with the abstracted versions of the problems and did not like having to consider constraints such as the impossibility of placing half of a person in one location and half in another. Some students also thought that homework should be evaluated only based on the procedure, not if they also got the correct answer, and were reminded that they would like the engineers building bridges to get the numbers correct. Yet, typical of many courses, for each student who disliked a particular topic, project, or aspect of the course, there was another who rated it favorably, whether it was the number/style of assignments, the software available to use, the group project, or the textbook. Some of the students who were searching for jobs after graduation noted that the group project and other more realistic problems solved during the semester were useful talking points during their interviews. Recent graduates have since noted the applicability of this material in their jobs.

The final grade distribution was fairly typical for an upper-level elective, and students of all majors were equally likely to succeed. In fact, the percentage of students receiving an A or A- in the course was identical for both computer science and business majors, at slightly more than one-third of each group, and similar for math majors (the smallest group). The proportions of students receiving the lowest grades were also comparable. The suggestion that seems the most useful for future offerings is to make more direct connections with specific topics covered in the required business curriculum, as was done for the computer science and math courses. More handouts were provided in this course than typical, in part because of the varied backgrounds of the students, and this minor change seemed to help some students feel more comfortable. Likewise, because some of the students had the perceived advantage of having had exams in other courses written by the same instructor, an optional review session was held before the midterm exam. While the direct benefit of these features is hard to measure, and may be negligible, it seems worthwhile to do in this situation, and will be retained in future years. Repeated offerings could provide additional data and a larger sample size for more extensive evaluation.

3.3 Future Plans

While this course was initially offered as a one-time special topics course, its success and a current campus-wide interdisciplinary focus make it a prime candidate to become a regularly-offered elective. Over the next year, when the department reviews its elective offerings, it seems likely that Operations Research will be added to the offerings, and a typically low-enrollment elective will revert to independent study status. A permanent cross-listing of the course as both Computer Science and Mathematics is almost certain. Discussions will take place with the Business department to see about a continued cross-listing that will allow Operations Research to count as one of the free electives in the major, as it did as a special topics course. It may be possible to co-teach the course with a member of the Business faculty, allowing for more of a tie-in with other Business courses or some collaborative projects. Finally, options similar to cross-listing but that allow for differentiated assignments (mandatory as opposed to self-selected) based on students' majors may be worth exploring.

4. CONCLUSIONS

While CS0, CS1, and dedicated non-majors courses will continue to provide the majority of access and recruiting into computer science, there are other viable options for some schools to consider. In particular, liberal arts colleges with relatively low enrollment in computer science as compared to some other disciplines, particularly business, may find offering an operations research course to be beneficial to both majors and non-majors, and the campus as a whole. Especially at small colleges, it seems that favorable word of mouth about courses within a discipline can have a substantial impact on which electives and distribution courses students choose to take, and interdisciplinary courses with limited prerequisites that students can take with their friends are often popular. Note that this course is not expected to have a noticeable and direct impact on the number of majors and minors, nor was it intended to; it has a prerequisite of junior standing and is targeted at students who have declared particular majors (business, computer science, and/or mathematics). Rather, the goal is to serve the majors and minors, while providing another opportunity for students to explore computer science. Preliminary feedback at Southwestern University suggests that the course strengthened the curriculum for majors and non-majors alike, as students saw connections between computer science and other disciplines.

5. ACKNOWLEDGEMENTS

Thanks to Susan Martonosi from Harvey Mudd College for sharing the materials from her Introduction to Operations Research course, in particular the idea and details for the group project case study.

6. REFERENCES

[1] FrontlineSolvers: Developers of the Excel solver. `http://www.solver.com/`.

[2] Georgia Tech's Ph.D. program in algorithms, combinatorics, and optimization. Retrieved August 12, 2011 from `http://www.aco.gatech.edu/`.

[3] GLPK (GNU linear programming kit). `http://www.gnu.org/software/glpk/`.

[4] INFORMS forum on education. Retrieved August 12, 2011 from `http://www.informs.org/Community/INFORM-ED`.

[5] Program in algorithms, combinatorics and optimization at Carnegie Mellon University. Retrieved August 12, 2011 from `http://aco.math.cmu.edu/index.html`.

[6] Using solve : Tools. Retrieved August 12, 2011 from `http://docs.google.com/support/bin/answer.py?hl=en&answer=139704&topic=15158`.

[7] B. M. Anthony. Scheduling of first-year seminar modules. Unpublished technical report.

[8] A. P. Armacost and J. K. Lowe. Operations research capstone course: A project-based process of discovery and application. *INFORMS Trans. Ed.*, 3(2):1–25, 2003.

[9] R. Bronson and G. Naadimuthu. *Schaum's Outline of Theory and Problems of Operations Research*. McGraw-Hill, 1997.

[10] J. B. Cushing, R. Weiss, and Y. Moritani. CS0++ broadening computer science at the entry level: interdisciplinary science and computer science. *J. Computing Sciences in Colleges*, 23:51–57, December 2007.

[11] D. Goelman. Databases, non-majors and collaborative learning: a ternary relationships. In *Proceedings of the 13th annual conference on Innovation and Technology in Computer Science Education*, ITiCSE '08, pages 27–31, New York, NY, USA, 2008. ACM.

[12] M. Goldweber. TauRUs: a "Taulbee survey" for the rest of us. *ACM Inroads*, 2:38–42, May 2011.

[13] S. Martonosi. The first (and often only) O.R. course: Something for everyone. INFORMS Annual Meeting, Austin, TX, November 2010.

[14] M. Merzbacher. Open artificial intelligence - one course for all. *SIGCSE Bull.*, 33:110–113, February 2001.

[15] G. M. Schneider. Computational science as an interdisciplinary bridge. In *Proceedings of the Thirtieth SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '99, pages 141–145, 1999.

[16] T. Urness and E. Manley. Building a thriving CS program at a small liberal arts college. *J. Computing Sciences in Colleges*, 26:268–274, May 2011.

[17] H. M. Walker and C. Kelemen. Computer science and the liberal arts: A philosophical examination. *Trans. Comput. Educ.*, 10:1–10, March 2010.

[18] M. Zhang, E. Lundak, C.-C. Lin, T. Gegg-Harrison, and J. Francioni. Interdisciplinary application tracks in an undergraduate computer science curriculum. In *Proceedings of the Thirty-Eighth SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '07, pages 425–429, 2007.

[19] S. Zweben. Undergraduate CS degree production rises; doctoral production steady: 2009-2010 Taulbee survey. *Computing Research News*, 23(3), May 2011.