Teaching Discrete Structures: A systematic review of the literature

Thomas Whelan
Dept of Computer Science
National University of Ireland
Maynooth, Co. Kildare, Ireland
thomas.j.whelan@nuim.ie

Susan Bergin
Dept of Computer Science
National University of Ireland
Maynooth, Co. Kildare, Ireland
susan.bergin@nuim.ie

James F. Power
Dept of Computer Science
National University of Ireland
Maynooth, Co. Kildare, Ireland
jpower@cs.nuim.ie

ABSTRACT

This survey paper reviews a large sample of publications on the teaching of discrete structures and discrete mathematics in computer science curricula. The approach is systematic, in that a structured search of electronic resources has been conducted, and the results are presented and quantitatively analysed. A number of broad themes in discrete structures education are identified relating to course content, teaching strategies and the means of evaluating the success of a course.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education—*Computer science education, Curriculum*; G.2.0 [**Mathematics of Computing**]: Discrete Mathematics—*General*

Keywords

Computing curriculum, discrete structures, discrete mathematics

1. INTRODUCTION AND MOTIVATION

The 2008 ACM Computer Science curriculum defines discrete structures as *foundational material* for computer science, that is required by many other areas in the syllabus [1]. This curriculum defines 43 core hours of topics, covering sets, logic, counting, graphs and probability. As a product of a long process of standardisation, adopted by the ACM and the IEEE Computer Society, it constitutes the standard definition of the topic.

However, in attempting to implement such a curriculum, many decisions remain for the educator. For example, frequently a choice must be made of which sub-topics to cover, bearing in mind the particular needs of a given programme, and the possibility that some topics may be covered elsewhere in the curriculum. Beyond this, the correct strategies for teaching and assessment must also be worked out. Finally, the ACM curriculum documents, while firmly based on years of experience, do not themselves provide a guide to the literature available in the field.

In this paper we address the last of these points, providing a literature survey, in order to provide a basis for addressing the others. In particular, we present the results of a *systematic* literature review of the teaching of discrete structures, based on the guidelines outlined by Kitchenham and Charters for performing literature reviews in software engineering [20]. Our evaluation contains cues from other sources of analysis, in particular the meta-analysis presented by Valentine [44].

The starting point of our review is to identify the main *research questions:*

Question 1: What approaches have been taken to designing the curriculum for discrete structures courses?

Question 2: What teaching methods have been employed in these courses?

Question 3: How has the success in teaching discrete structures courses been evaluated?

Rather than presenting the entire systematic literature review process, this paper provides an overview of the process and a summary of the results; fuller details can be found in the associated technical report [48].

We believe this review will be of interest on at least three levels: first, as as a review of the literature spanning many decades of teaching discrete structures, second at the meta-level, providing a meta-analysis of publications in the area, and third at the meta-meta level, as an example of applying the systematic review process applied to CS education.

Phases of paper selection and analysis

The paper selection strategy proceeded in three phases:

Phase 1: Use an automated search engine to identify papers that deal with topics relevant to our research. For simplicity we have restricted our search to *electronically-available* resources, but, given the topic, we do not believe this to be a major limitation of our review. This phase is described in Section 2.

Phase 2: Manually filter the results based on reading the abstracts, in order to identify the subset of these papers that are relevant to our research questions. This phase is described in Section 3.

Phase 3: Extract and synthesise the research by reading the remaining papers. Section 4 presents the results of this phase, discussing some common themes in the literature concerning course structuring, delivery and evaluation.

2. AUTOMATED SEARCH RESULTS

The **search space** for sources consisted of, in order, the *ACM Digital Library* (ACM DL), *ACM Guide to Computing Literature* (ACM Guide) and the *Google* search engine. All searches were carried out between July and September 2010.

It quickly became apparent that both "Discrete structures" and "Discrete mathematics" were good search terms for the kind of papers we wanted. Beginning with these search terms (referred to as "base terms" from now on) we also tried searches including the phrases "teaching", "education" or "learning" to enhance the search precision.

2.1 Search Results

When displaying results in the ACM DL there are two options. The first, "Sort by" was left at its default setting ("relevance"), while the second, "Form" was set to "condensed form", which contains 50 paper titles per page.

Since the search for each of the base terms returned about 20,000 papers, we needed to establish a cut-off point for manual filtering. In the remainder of this paper, we use a cut-off density of 4%: that is, we did not consider more pages of results after two or less papers in a page of fifty were judged relevant. In total, 65 papers were manually identified by title alone from the ACM DL.

The ACM Guide contains over 1.2 million citations from over 3,000 publishers, enabling a larger search space than the ACM DL alone. However, many of the results were not relevant to the topic or the research questions, or did not have a readily available on-line copy. Only two new papers were found in the ACM Guide (references [24, 50]).

The final search strategy used the search engine *Google*, keeping in mind that many results returned from such a search engine will not be relevant and/or academically published. In order to counteract this, the base terms were only ever queried along with either "teaching", "education" or "learning". All queries were done as individual words and also as complete phrases, and the results were examined manually up to the 5th page. Interestingly, having removed unrefereed or non-relevant sources such as lectures and coursework, no new papers were identified by this search.

Following these searches a total of 67 unique papers were identified by title alone as meriting further analysis.

2.2 Analysis of search terms

Table 1 shows the results of searching the ACM DL using the base terms and their extensions. As can be seen, the vast majority of relevant papers were retrieved using the two base terms, with the extra terms yielding just six new papers in total, only two of which were unique.

Figure 1 contains a bar chart showing the distribution of relevant papers retrieved, broken down by search term over five year periods. In this bar chart, when a paper was found using more than one search term its score was divided between these terms. From this figure it can be seen that whereas "discrete structures" was the dominant term until 1985, since then "discrete mathematics" has become more common, with an increasing number of these being additionally qualified by one of the secondary search terms.

3. FURTHER MANUAL FILTERING

Following their selection by title, the 67 papers were then manually filtered by reading their abstracts and determining whether they contained information relevant to the research questions. The main reason for rejection of a paper in this phase was the *nature* of the paper (e.g. book review, poster and discussion sessions, documentation) rather than the content. This phase resulted in the exclusion of a further 20 papers, leaving 47 papers for full consideration.

Table 1: Numbers of ACM DL papers judged relevant based on examining the results using various search terms.

	Number of Papers		
Search Term	**Inspected**	**Accepted**	**Added**
Discrete Structures	100	18	
+ Education	150	11	+1
+ Learning	50	2	+0
+ Teaching	150	14	+2
Discrete Mathematics	200	55	
+ Education	250	24	+1
+ Learning	200	21	+1
+ Teaching	150	30	+1

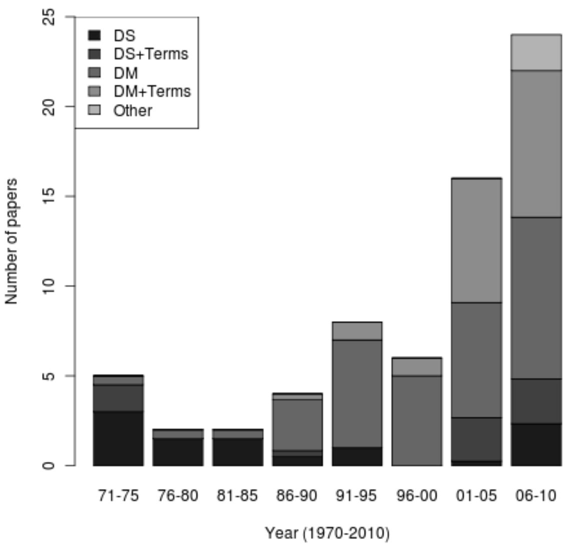

Figure 1: Distribution of relevant papers from the ACM DL by search term. Each bar represents the number of relevant papers in a given five-year period; subdivided by the search term used.

Precision and Recall

In order to measure the exactness and completeness of our search for sources we can calculate the *precision* and *recall* associated with each search term. *Precision* is defined as the number of relevant documents retrieved by a search divided by the total number of documents retrieved by that search, while *recall* is defined as the number of relevant documents retrieved by a search divided by the total number of existing relevant documents (which should have been retrieved).

Calculating *precision* requires determining the total number of documents retrieved by a search, but there are a number of possible choices here. For example, the search of the ACM DL for the term "discrete mathematics" yielded 19,013 papers, but only 200 were actually examined by title, using the 4% density cut-off mentioned above. Thus, dividing by

Table 2: Precision and recall values for each search term, (total relevant = 47 abstract-filtered papers)

Search Term	Papers Retrieved	Papers Relevant	Percentage Precision	Recall
DS	23,855	13	0.05	27.66
DS+E	2,401	10	0.42	21.28
DS+L	6,128	1	0.02	2.13
DS+T	1,407	12	0.85	25.53
DM	19,013	38	0.20	80.85
DM+E	1,695	18	1.06	38.30
DM+L	3,374	17	0.50	36.17
DM+T	1,094	25	2.29	53.19

19,013 gives a slightly lower result than is fully accurate, but dividing by 200 inhibits comparison with other terms whose density may have reduced at a different rate. Bearing this caveat in mind, Table 2 uses the full total figure to facilitate comparison between search terms.

Table 2 gives the precision and recall values as percentages, assuming that the relevant papers are the 47 remaining after filtering based on abstract. As can be seen from this tables, the terms "discrete mathematics teaching" and "discrete mathematics education" provide the most precise search terms. Consistent with the results shown above in Figure 1, the term "discrete mathematics" provides the greatest recall value, yielding 81% of the final 47 relevant papers.

Source of publications

Finally, Table 3 shows a breakdown of the 67 title filtered papers according to the search term used and the source of the publication. Just under three-quarters of the relevant papers came from four sources, with the top eight sources accounting for nearly 90% of the papers. These results must be qualified by noting that most came from the ACM DL, thus causing a natural clustering, but are nonetheless broadly in line with expectations.

4. ANALYSIS AND SYNTHESIS

Having acquired the 47 source papers, the final step is to read, analyse and synthesise the results. While we cannot provide universal recommendations for any syllabus in the subject, we hope to at least provide a framework within which they can be worked out.

In this section we discuss the research in teaching discrete structures under five broad *themes*:

- **Whether** discrete structures should be taught as a single course, or have its content distributed over multiple courses.
- **When** students should study discrete structures
- What should the material be linked with: we have identified two main overlapping areas: **with programming** and **with data structures**.
- **How** should the course be taught; in particular, what teaching methods have been identified.

Table 4 presents a summary of the papers by each of these themes, and Figure 2 shows the distribution of the themes over time.

A further theme, the *evaluation of course success*, is discussed at the end of this section.

Whether to teach DS

This somewhat existential question is not so much related to whether the topics in discrete structures should be taught, as to whether they should be taught in a single

Table 3: Place of publication for the 67 title-filtered papers, broken down by search term used.

Place	DS	DM	Other	Total	Cum
SIGCSE	7.5	14.5	2	24	36%
JCSC	1	11	0	12	54%
SIGCSE Bulletin	3	5	0	8	66%
ITiCSE	0	4	0	4	72%
JERIC	0	4	0	4	78%
SIGACT News	0	3	0	3	82%
CCSC	0	2	0	2	85%
CSC	0.5	1.5	0	2	88%
Others	1	5	2	8	100%
Total	*13*	*50*	*4*	*67*	

Table 4: Themes covered in the 47 abstract-filtered papers (some papers cover multiple themes).

Category	Sources	Total
Whether to teach DS	[3, 4, 6, 8, 12, 32, 43]	7
When students should learn DS	[2, 3, 4, 6, 8, 13, 18, 22, 25, 26, 27, 29, 35, 36, 37, 46, 47]	17
With programming?	[5, 7, 10, 14, 15, 16, 23, 28, 30, 31, 33, 39, 41, 45, 49, 50]	16
With data structures?	[4, 5, 6, 11, 22, 28, 31, 34, 39, 40, 45, 49]	12
How to teach DS	[2, 5, 7, 9, 10, 14, 16, 21, 24, 32, 33, 37, 41, 42, 45, 49, 50, 51]	18

course, or moved into the courses where they are applied. Examples include moving logic into software verification or formal methods or moving all study of trees and graphs into data structures. Similar questions arise when considering the mathematical techniques relevant to computer science, such as calculus or statistics, and appear across a large time span, from Berztiss, Tremblay and Manohar in the 1970's to Fleury and Neff in the 1990's and 2000's [6, 43, 12, 32].

It is highlighted quite frequently that one of the main ideas of teaching students discrete structures is to prepare them for more theoretical content they meet in courses later on [25, 27], which brings up the question that if computer science majors are finding the content too difficult or aimless so early in their studies, why not push it back to "where it belongs"? With Tremblay and Manohar for example, counting techniques, permutations and probability are omitted from their discrete structures course (although part of the curriculum) as they are covered in other courses [43].

A more novel approach is taken specifically by Neff [32], where parts of the discrete structures course are outsourced and the subject is then taught with an approach of rather than "Here's the tools, you'll need them sometime", it says "Here's a computer science problem we need to solve, what tools do we need to solve it?".

When students should learn DS

It is acknowledged that discrete structures is just a basis for the more specialised computer science topics which students may not encounter for quite some time [6]. Thus a conflict arises between teaching discrete structures early in the syllabus or late, potentially allowing for greater depth, but causing difficulties for subjects that depend on it [25].

Marion draws us back to our previous point about dismantling the subject, but instead segments the course rather than completely dissolving it [27]. This can allow an easier introductory course for discrete structures and a subsequently more in-depth course later on when students are more experienced. But the appreciation of the subject is always a problem early on, with a difficulty noted in teaching students material that they believe they may never use [13]. If discrete structures is to serve as a basis for a wide variety of more in-depth modules later on, it needs to justify its existence by demonstrating its necessity and application.

DS with Programming

A number of papers describe curricula that involve coding up a discrete structures related problem. One approach uses real world examples in order to aid students in the applicability of what they are being taught and motivate them to excel at the course work [39]. Martin proposes allowing students to implement an algorithm of their choice which they learnt in their discrete structures course, enforcing their understanding of the material [28]. Cigas and Hsin, introduce the use of *specification* for problems; one of the primary applications of the logic learnt in discrete structures [10].

There is also the opportunity to exploit the link between discrete structures and declarative programming languages. For example, Berry describes a more advanced application of coding involving the use of Scheme [5]. As a language which contains elements of functional programming there is already a clear link between this approach and the *functions* aspect of discrete structures. Similarly, Hein outlines the use of FP, ML and Prolog to teach students discrete structures concepts [15]. Clearly, an issue here is how this interacts with the teaching of programming and language paradigms in the syllabus as a whole.

McMaster *et al.* draw the conclusion that teaching discrete structures with programming makes the subject easier for students - or at least for those students who already know how to program [31]. Implementing a theory learnt in discrete structures as a program requires a full understanding of it, rather than the regurgitation of a rote learnt mathematical proof. However, it is important not to turn discrete structures into another programming course, since omitting the theoretical material could lose much of the intended flavour of the subject.

DS with Data Structures

Another noticeable feature of many of the papers reviewed is an apparent crossover between topics covered in discrete structures and data structures (CS2). The ACM curriculum for discrete structures contains a number of intersections with data structures such as graphs and trees and such a crossover (as well as some others) is further explored by Decker and Ventura [11].

Even though the context in which these topics are to be studied is different between the two subjects, this again raises questions regarding the coherence of a discrete structures course in a syllabus. A number of the papers note that the concept of coding up actual algorithms in discrete structures is very similar to the way in which data structures are studied [39, 28, 5, 31]. Teaching an inductive structure such as a list or tree, and then asking students to implement them, particularly in a functional language [34], is similar to implementing these as data structures, albeit often with different motivation and analysis mechanisms.

A different angle on this issue arises from studies of factors influencing student success at discrete structures. There is significant coupling between data structures and discrete structures when evaluating student performance [6, 40, 34, 11], giving strong support to closely examining this link in a syllabus.

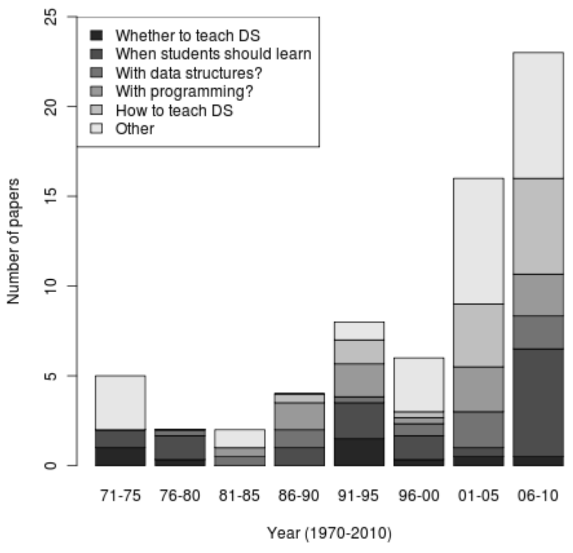

Figure 2: Distribution of topics over time. Each bar represents the number of relevant papers over a five-year period, subdivided by the topic covered.

How to teach DS

Finally, in response to the uncertainties of educating students in discrete structures a number of teaching methods are described in the literature.

There is much focus on reducing the unfamiliarity of the subject. For example Zeng and Jiang builds on the foundation of staying in a student's comfort zone when introducing a new idea [51]. This appears to ease students into the formality of certain areas of discrete structures. Berry, Cigas and Hsin use visual tools to try to reduce the complexity for students [10, 5]. Exploitation of a student's familiarity with programming is also an interesting way to tackle teaching problems since students may be a lot more comfortable in a software engineering setting than in a pure mathematical one in which discrete structures is sometimes presented. To deal with unfamiliar notation, a simple reinforcement of exactly what a symbol means can uncloud a problem for a student, allowing them to focus on understanding the theory rather than the theory's representation [21].

The collaborative approach discussed by Buchele takes a more traditional pen and paper approach but the increased student engagement can certainly be seen as beneficial [9]. Collaborative homework also enables mingling of those who understand certain concepts and those who don't, hopefully making the educational experience "viral". Similarly, Neff suggests that a problem-directed approach can go some way to tackling the appreciation and motivation problems with discrete structures [32].

It is interesting that while the previous themes explored the relationship of discrete structures with other subjects, many of the papers on approaches to teaching seek to exploit this relationship positively.

Table 5: Ways of evaluating course success.

Category	Sources	Total
Grade Based Statistics	[11, 19, 30, 34, 35, 40, 50]	7
Survey Based Statistics	[9, 26, 28, 37, 38, 42, 45]	7
Anecdotal Feedback	[6, 7, 8, 14, 16, 17, 21, 24, 31, 41, 43]	11

4.1 Evaluation of Course Success

Given that a very large number of papers discuss some kind of discrete structures course it is important to analyse the ways in which these courses are evaluated. We found that a total of 25 papers discussed some form of evaluation, though this varied between papers. Overall we can identify three broad categories of information regarding the evaluation of a teaching approach: based on student grades, based on a quantitative analysis of student feedback, and based on anecdotal information from students. The papers falling into each of these three categories are shown in Table 5.

Notably, it appears that statistics in some cases can be quite sparse when it comes to discrete structures. With Martin for example, a set of 20 students may not tell us as much as a set of 100 students would when it comes to certain questions [28]. Going by statistics alone, in taking radical approaches such as scattering discrete structures course content to later on modules we effectively move into unknown territory. For this reason, it is necessary to collect a significant amount of statistics for a wide variety of approaches taken to teaching the subject. Having said that, there are some more sizable data sets which analyse the final grades for students [11, 34, 40].

As well as these grade-based statistics there are also some survey-based course evaluations which provide more anecdotal type feedback on courses [28, 9]. This anecdotal type evaluation is a lot stronger in some papers than in others, sometimes with just a sentence or two given for student thoughts on a course [6, 21, 31, 43]. There appears to be quite some variety in course evaluation techniques and as a result it can be difficult to fully evaluate, much less compare, course outcome analyses.

5. CONCLUDING REMARKS

This paper has presented a review of research on discrete structures teaching. Since the initial submission of this paper our search parameters were extended to include the ERIC collection, however this did not result in any significant changes to this review; details can be found in the associated technical report [48].

We hope this work will be of use as a reference point for those preparing to teach discrete structures, and as an example of a systematic literature review in CS education.

6. ACKNOWLEDGEMENTS

This work was partially funded by the National Academy for Integration of Research, Teaching and Learning (NAIRTL).

7. REFERENCES

[1] ACM/IEEE. Computer science curriculum 2008: An interim revision of CS 2001. Report from the interim review task force, ACM and the IEEE CS, Dec. 2008.

[2] V. L. Almstrum, P. B. Henderson, V. Harvey, C. Heeren, W. Marion, C. Riedesel, L.-K. Soh, and A. E. Tew. Concept inventories in computer science for the topic discrete mathematics. In *Working group reports on ITiCSE on innovation and technology in CS education*, pages 132–145, Bologna, Italy, 2006.

[3] D. Baldwin, B. Marion, and H. Walker. Status report on the SIGCSE committee on the implementation of a discrete mathematics course. In *35th SIGCSE Tech. Symp. on CS Education*, pages 98–99, Norfolk, VA, 2004.

[4] D. Baldwin, C. H. Smith, P. B. Hendersen, and V. Vadisigi. CS1 and CS2 (panel session): foundations of computer science and discrete mathematics. In *31st SIGCSE Tech. Symp. on CS Education*, pages 397–398, Austin, TX, 2000.

[5] J. Berry. Improving discrete mathematics and algorithms curricula with LINK. In *2nd Conference on Integrating Technology into CS Education*, pages 14–20, Uppsala, Sweden, June 1997.

[6] A. T. Berztiss. The why and how of discrete structures. In *6th SIGCSE Tech. Symp. on CS Education*, pages 22–25, July 1976.

[7] S. Bridges. Graphics assignments in discrete mathematics. In *24th SIGCSE Tech. Symp. on CS Education*, pages 83–86, Indianapolis, IN, 1993.

[8] D. T. Brown. Discrete mathematics II. *SIGCSE Bulletin*, 25(4):13–17, 1993.

[9] S. F. Buchele. Increased student participation in a discrete mathematics course. *J. of Comp. Sci. in Colleges*, 20(4):68–76, Apr. 2005.

[10] J. Cigas and W. J. Hsin. Teaching proofs and algorithms in discrete mathematics with online visual logic puzzles. *J. Educ. Resour. Comput.*, 5(2), June 2005.

[11] A. Decker and P. Ventura. We claim this class for computer science: a non-mathematician's discrete structures course. In *35th SIGCSE Tech. Symp. on CS Education*, pages 442–446, Mar. 2004.

[12] A. E. Fleury. Evaluating discrete mathematics exercises. In *24th SIGCSE Tech. Symp. on CS Education*, pages 73–77, Apr. 1993.

[13] D. Gries, M. Eckmann, A. Erkan, and J. Heliotis. Discrete mathematics/structures: How do we deal with the late appreciation problem? *J. of Comp. Sci. in Colleges*, 24(6):110–112, June 2009.

[14] V. J. Harvey and S. H. Rodger. Editorial for the special issue on software support for teaching discrete mathematics. *J. Educ. Resour. Comput.*, 5(2):1, 2005.

[15] J. L. Hein. A declarative laboratory approach for discrete structures, logic, and computability. *SIGCSE Bulletin*, 25(3):19–25, Sept. 1993.

[16] P. B. Henderson. Discrete mathematics as a precursor to programming. In *21st SIGCSE Tech. Symp. on CS Education*, pages 17–21, Washington D.C., 1990.

[17] P. B. Henderson. Reflections on teaching discrete math for the first time. *SIGCSE Bulletin*, 39(2):24–24, 2007.

[18] P. B. Henderson, R. DeCoste, and K. L. Huggins. Preparing to teach discrete math for the first time. In *Working group reports on ITiCSE on Innovation and technology in CS education*, pages 20–21, Bologna, Italy, 2006.

[19] L. Jamba-Joyner and W. F. Klostermeyer. Predictors

for success in a discrete math course. *SIGCSE Bulletin*, 35(2):66–69, 2003.

[20] B. Kitchenham and S. Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

[21] J. Krone. Meeting the challenges of discrete mathematics for computer science: the power of symbols. *J. of Comp. Sci. in Colleges*, 23(1):31–37, Oct. 2007.

[22] M. D. LeBlanc and R. Leibowitz. Discrete partnership: a case for a full year of discrete math. In *37th SIGCSE Tech. Symp. on CS Education*, pages 313–317, Houston, TX, 2006.

[23] G. M. Levin. ISETL: a language for teaching discrete mathematics (abstract). In *ACM Annual Conference on Cooperation*, page 455, Washington, D.C., 1990.

[24] Y. Li. A new approach to teaching logic in discrete mathematics. In *9th Intl. Conf. for Young Computer Scientists*, pages 2432–2437, Nov. 2008.

[25] B. Marion. Discrete mathematics: support of and preparation for the study of computer science. In *Seventh annual CCSC Midwestern Conference on Small Colleges*, pages 190–199, Dec. 2000.

[26] B. Marion. Final oral report of the SIGCSE committee on the implementation of a discrete mathematics course. In *37th SIGCSE Tech. Symp. on CS Education*, pages 268–269, Houston, TX, 2006.

[27] W. Marion. Discrete mathematics for computer science majors - where are we? How do we proceed? In *20th SIGCSE Tech. Symp. on CS Education*, pages 273–277, Feb. 1989.

[28] K. E. Martin. The role of discrete structures and operations research in a computer science curriculum. *SIGCSE Bulletin*, 16(4):4–6, Dec. 1984.

[29] W. D. Maurer. An alternative to group theory in the discrete structures course. *J. of Comp. Sci. in Colleges*, 21(5):110–115, 2006.

[30] J. W. McGuffee. The discrete mathematics enhancement project. *J. of Comp. Sci. in Colleges*, 17(5):162–166, 2002.

[31] K. McMaster, N. Anderson, and B. Rague. Discrete math with programming: better together. In *38th SIGCSE Tech. Symp. on CS Education*, pages 100–104, Mar. 2007.

[32] N. Neff. Problem-directed discrete structures course. In *41st SIGCSE Tech. Symp. on CS Education*, pages 148–151, Mar. 2010.

[33] D. Nohl. Using an automated reasoning program as a CS application in discrete mathematics. *J. of Comp. Sci. in Colleges*, 22(4):7–13, 2007.

[34] R. L. Page. Software is discrete mathematics. In *8th Intl. Conf. on Functional Programming*, pages 79–86, Aug. 2003.

[35] B. T. Pioro. Introductory computer programming: gender, major, discrete mathematics, and calculus. *J. of Comp. Sci. in Colleges*, 21(5):123–129, 2006.

[36] R. E. Prather. Another look at the discrete structures course. In *6th SIGCSE Tech. Symp. on CS Education*, pages 247–252, Feb. 1976.

[37] D. A. Schoenefeld and R. L. Wainwright. Integration of discrete mathematics topics into the secondary mathematics curriculum using Mathematica: a summer institute for high school teachers. In *24th SIGCSE Tech. Symp. on CS Education*, pages 78–82, Indianapolis, IN, 1993.

[38] A. Settle and C. Settle. Graduate student satisfaction with an online discrete mathematics course. *J. of Comp. Sci. in Colleges*, 21(1):79–87, 2005.

[39] B. Setzer. A lab course for discrete mathematics. In *47th Annual Southeast Regional Conf.*, Mar. 2009.

[40] J. R. Sidbury. A statistical analysis of the effect of discrete mathematics on the performance of computer science majors in beginning computing classes. In *17th SIGCSE Tech. Symp. on CS Education*, pages 134–137, Feb. 1986.

[41] D. E. Stevenson, M. R. Wick, and S. J. Ratering. Steganography and cartography: interesting assignments that reinforce machine representation, bit manipulation, and discrete structures concepts. In *36th SIGCSE Tech. Symp. on CS Education*, pages 277–281, St. Louis, MO, 2005.

[42] K. Sutner. CDM: Teaching discrete mathematics to computer science majors. *J. Educ. Resour. Comput.*, 5(2):4, 2005.

[43] J. P. Tremblay and R. Manohar. A first course in discrete structures with applications to computer science. In *4th SIGCSE Tech. Symp. on CS Education*, pages 155–160, Feb. 1974.

[44] D. W. Valentine. CS educational research: a meta-analysis of SIGCSE technical symposium proceedings. In *35th SIGCSE Tech. Symp. on CS Education*, pages 255–259, Norfolk, VA, 2004.

[45] R. L. Wainwright. Introducing functional programming in discrete mathematics. In *23rd SIGCSE Tech. Symp. on CS Education*, pages 147–152, Kansas City, MO, 1992.

[46] J. S. Warford. An experience teaching formal methods in discrete mathematics. *SIGCSE Bulletin*, 27(3):60–64, 1995.

[47] J. Waxman. Reflections on B3, discrete structures. *SIGCSE Bulletin*, 7(2):51–54, 1975.

[48] T. Whelan, S. Bergin, and J. F. Power. Discrete structures teaching: A systematic literature review. Technical report, Department of Computer Science, National University of Ireland Maynooth, NUIM-CS-TR-2010-01, Sept. 2010.

[49] M. R. Wick and P. J. Wagner. Using market basket analysis to integrate and motivate topics in discrete structures. In *37th SIGCSE Tech. Symp. on CS Education*, pages 323–327, Houston, TX, 2006.

[50] W. Xiuguo. Discrete mathematics teaching reformation: Adding experiments. In *Intl. Workshop on Education Technology and Computer Science*, volume 2, pages 566–569, 2009.

[51] H. Zeng and K. Jiang. Teaching mathematical proofs to CS major students in the class of discrete mathematics. *J. of Comp. Sci. in Colleges*, 25(5):326–332, May 2010.