On Enhancing Application-Ability Training in Discrete Mathematics

Tun Li, Wanwei Liu, Liqian Chen, Xiaoguang Mao
College of Computer Science and Technology
National University of Defense Technology
Changsha, Hunan, P.R.China, 410073
{tunli, wwliu, lqchen, xgmao}@nudt.edu.cn

***Abstract*—In this work-in-progress innovative practice paper, we argue the application-ability training in Discrete Mathematics (DM) should be enhanced for students majoring in computing science. Our motivation is based on the analysis of the differences in learning outcomes between DM and other branches of Math courses, the special role of DM in computer science (CS) courses and the gaps between DM and other CS courses. Motivated by the above analysis, we rethink of CS undergraduate education program as a DM-centric program. Furthermore, we make an experimental implementation of the DM-centric program and enhance application-ability training by designing and adopting many large-scale projects from various related topics, such as database, satisfiability, deductive proof and so on. Each project is decomposed into several sub-projects, which are integrated with a project-based learning environment. The large-scale projects derived from related CS courses enable students to get in touch with various computing topics related to DM applications at an early stage in the learning process. The learning environment with the ability of automatic assessment enables students to complete the projects in a step-by-step manner. The preliminary feedback from 219 students after taking the redesigned DM course shows the promising effects on students' following learning.**

***Index Terms*—Computer Engineering, Discrete Mathematics, Application-Ability, Online, Training**

I. Introduction and Related Work

The field of Discrete Mathematics (DM) provides an important analytical toolset for computing disciplines. Students in CS, software engineering and related programs need a foundation in topics like logic, relations, graph theory, and abstract algebra. However, there are several variations for teaching DM when regarding to teaching other branches of mathematics.

First of all, DM is the mathematical foundations of CS, and problem solving through the medium of the computers is the essence of computing disciplines. Students majoring in CS like to be engaged with computers, and any course in CS curriculum should take advantage of this intrinsic motivation. Therefore, in addition to solving problem by paper and pen, students in DM course should be trained to solve (practical) problems by computers.

Therefore, we argue that the students majoring in computational science should strengthen the training of the application-ability of discrete mathematics. Especially, the training of computer-based application-ability.

Secondly, the principal ideas and methods of DM are taught relatively early in students' college years, while the applications of DM subjects are scattered among many courses studied during the subsequent semesters, which makes an application-ability training gap between DM and other courses. On one hand, when students study DM topics such as set, relationship and mathematical logic, they may ask: "Why do we have to learn all this?". On the other hand, when they study courses such as operating system, computer network, and computer organization, they will forget or ignore the mathematical concepts and notations that are utilized in technical treatments in these courses.

To sum up, the biggest challenge of teaching DM course is how to fill the application-ability training gap between DM course and other CS courses. Individually, teachers around the world have developed various methods to deal with the challenge. The dominate solutions are to integrate DM topics and programming practices, so as to directly integrate DM and the follow-up courses (i.e. Algorithms, Data Structures, etc.). Among the practices, from the perspective of application-ability training, DM topics are used to improving programming skills [1] , and engaging problems are solved in a DM course as well as a programming course to reveal the intrinsic differences between the two courses [2]. Among the solutions, teachers also tried to use programming exercises and motivating applications to reinforce DM topics [3]–[5].

The other research activities focus on CS courses that have no obviously connection with DM course. Teachers try to use the problems in software development to stimulate mathematical analysis in modern software development [6], or use the project derived from DM to stimulate software development activities [7], so as to link data mining with software engineering courses. Teachers have also tried to develop a set of resources for students to practice DM topics in order to realize application-ability training. For example, using Alloy language and analysis tool to teach DM topics in an exploratory and programming-intensive way [8], proposing a DM enhancement project which consists of various learning exercises and a set of programming and interactive resources that are directly related to the material encountered by CS student during their studies [9], [10]. In [11], teachers were ambitious to develop a systematic solution to the challenge.

However, most of the current practices focus on linking DM course with one course in each research, and the developed resources are offline, which are not so rich and extensible in terms of filling the gap. Especially, currently there are no systematic solutions for the big challenge of teaching DM course. Therefore, the major concern of our innovative practice is attempt **to propose and develop a systematic solution to enhance the students' application-ability training**.

II. Methodology

A. Background

As shown in Table I, in our CS curriculum, the courses are distributed along the first 6 semesters. The discrete mathematics course is scheduled for the 3rd semester. The core courses of CS, namely, data structure (DS), operating system (OS), software engineering (SE), database system (DBS), computer architecture (CA), compilation principle (PC), algorithm analysis (AA), Digital Circuit Design (DCD) etc., are arranged in the same semester or in the following semesters.

TABLE I
Scheduling of Courses in Our Curriculum

Semester	Courses
S1	• Introduction to Computer Science
S2	• Introduction to Programming
S3	• Discrete Mathematics • Data Structure (DS) • Digital Circuit Design (DCD)
S4	• Algorithm Analysis (AA) • Networking (NW) • Database System (DBS)
S5	• Operating System (OS) • Software Engineering (SW)
S6	• Principle of Compiling (PC) • Computer Architecture (CA) • Artificial Intelligent (AI)

The topics taught in discrete mathematics are as follows:

- Sets, relations and functions
- Propositional logic and predicate logic
- Proof techniques
- Natural deduction system
- Graphs and trees
- Abstract algebra

Before taking the DM course, students have already learned Python [12] and C++ programming languages [13] in the first and second semesters, respectively. In addition, they have also received intensive training on computational thinking and programming [14], [15]. The topics covered in these prerequisite courses are:

- Foundations of programming using Python
- Applications of computing techniques:
  - Numerical computation
  - Modeling and simulation
  - Data storage and processing

The disadvantage of our curriculum arrangement is that it may lead to an application-ability training gap between DM and other courses. On the other hand, however, prior to the DM course, the prerequisite courses related to computational thinking provide an alternative solution to this challenge. The prerequisite courses make it possible to develop a DM-centric program and therefore online application-ability training resources.

B. Overview

Here we attempt to present a systematic solution to application-ability training in DM course. According to the situation we introduced in Section II-A, the basic idea is to develop a DM-centric program, and implement the application-ability training enhancement by designing and adopting a number of large-scale projects from various courses on related topics, such as database, satisfiability and deductive proof.

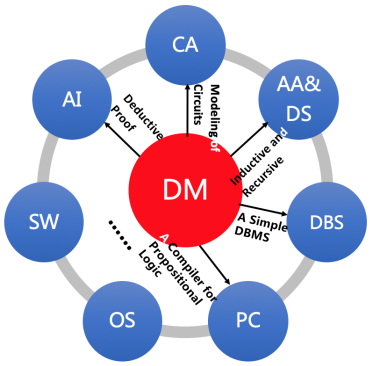

Fig. 1. DM-centric Program and Online Application-Ability Training Projects

Figure 1 shows the overview of the proposed method, where we take all the CS courses topics as the applications of DM subjects. First, the application projects that could reflect the distinct character of the selected course are developed incorporating with teachers of the selected course. Then, a proper online platform is selected to deploy all the projects. The selected platform should support automatic grading, and train students step by step according to the their abilities.

C. Implementation and Deployment

We have preliminarily implemented and deployed the DM-centric program and online application-ability training resources on Educoder.net platform (in Chinese)[1]. The reasons for selecting Educoder.net platform are as follows:

[1]https://www.educoder.net/paths/453

- The Educoder platform provides an automatic grading system. Students could practice the training projects at any time of the day. At the same time, students could get feedback on training evaluation immediately after completing a project.
- The Educoder platform provides a mechanism for arranging projects in a step-by-step way. A project could be divided into several stages, each of which is based on the results of the previous stage. Therefore, for every project, students could be trained from easy to difficult, and gradually gain confidence in the process of completing the project.
- The Educoder platform collects detailed data of students' learning activity, and outputs various statistical results, thus providing sufficient information for teachers to track the progress of students' training process.

TABLE II
THE APPLICATION-ABILITY TRAINING PROJECTS

Chapter	Projects	Related Courses
Sets	Application of set datatype	AA&DS
	Natural number system	AA&DS
Relations	Relations modeling and operation	AA&DS
	A simple database management system	DBS
Function	Recursive algorithm	AA&DS
Logic	Truth table generation	PC, AI, DCD
	Deductive proof system	AI
Boolean Algebra	Digital circuits modeling	CA, DCD

At present, we have developed and deployed projects for 5 subjects in the DM course, which includes 8 projects and 94 stages. The ability of programming using Python is required for students to complete all the projects. Table II gives an overview of all the projects along with the brief descriptions and the related courses.

The project design criteria are:

- For basic DM structures such as sets and relations, we try to design projects to cultivate students' application-ability, that is, how to model these structures, and then solve problems directly related to these structures. Therefore, we developed
  - some projects that directly use predefined Python data type Set to solve problems, such as construction of power-set and Cartesian products of given sets, and palindromes counting in "A Midsummer Night's Dream".
  - some projects that implement not only the theory of constructing natural number system with set theory, but also, addition, multiplication and power operation on natural numbers.
  - some projects that model relation structure and implement most operations on relation.
- For comprehensive DM topics, we try to design comprehensive projects to cultivate students' application-ability of applying theory to specific topic of the selected courses. The purpose of these projects is to answer the questions such as "Why do we have to learn all this?". As listed in TableII, each project is connected with one or more subsequent courses. Therefore, we developed:
  - a project that implement a simple database management system which is based on the n-ary relation theory and the operations (selection, projection and join) defined in the theory.
  - some projects that design and implement recursive algorithms using mathematical induction.
  - some projects that implement a propositional logic compiler and display truth table for a given formula based on the compiler .
  - some projects which implement a predicate logic compiler and a deductive proof checker for a given deductive prove system.
  - some projects that model and simulate various digital logic gates and logic units commonly appeared in central processing units.

Therefore, the projects could be classified into 2 types. The first category is the modeling of basic DM structures, which focus on training of application-ability on how to solve problems by computers. For example, the "Basic of Relations" project, which consists of 24 stages, is designed to train students how to model Relation structure and implement various operations on Relation structures. We first define a Python class framework to model Relations, and then students are required to implement a Relation operation one at a stage. The operations include relation composing, relation closure computation, operations on equivalence relation and partial order, and so on.

```
class Proposition:
    def __init__(self, name):
        # Please delete pass to complete the difinition of __init__ method
        pass

    def __eq__(self, other):
        if not isinstance(other, Proposition):
            return False
        return self.name == other.name

    def __str__(self):
        return self.name

    def __hash__(self):
        return hash(str(self))
```

Fig. 2. The Proposition class framework

The second type is to practice the application of DM theories in various courses, focusing on cultivating the application-

ability on how to apply DM theories to CS subjects. For example, in the "Truth table generation" project we first define an implementation framework as the starting point, and ask students to write the omitted code fragments to complete the assigned task. Then, according to the essential tasks of compiler design and truth table generation, the whole project is divided into 8 stages :

1) Stage 1: Students are assigned to complete the definition of the semantic objects of propositional formula, such as Propositions and operators (e.g. Not, And, Or, Implicit, Equivalence) classes in Python. Figure 2 shows the Proposition class framework. The task is to complete the "$\_\_init\_\_$" method, which will initialize all the necessary data to present the semantic of a proposition object.
2) Stage 2: According to the classes defined in the first stage, students are required to complete the task that extract all the propositions in a given formula, so as to prepare for the generation of the truth table.
3) Stage 3: Students should complete the task of assigning "True" or "False" values to each proposition extracted in the previous stage.
4) Stage 4: Students should write programs according to an assignment in Stage 3 to evaluate the value of the given formula.
5) Stage 5: Students should connect the syntax parsing results of the propositional logic complier with the semantic objects defined in the former 4 stages. Here, we use Pyparsing [16] to construct the compiler.
6) Stage 6: The task is to convert a given formula in human-readable format to the compiler's internal representation.
7) Stage 7: Based on the return results of stage 4, students should generate all the values for the given formula according to the whole proposition assignments.
8) Stage 8: Students are required to draw the truth table according to the results of stage 7.

Finally, after completing all the tasks of the project, students are able to develop a program, which takes a propositional logic formula as input and visually outputs a corresponding truth table. For example, for the formula $(P \wedge R) \wedge \sim Q$, the output is shown in Figure 3.

```
+---+---+---+---------------+
| P | Q | R | (P /\ R) /\ ~Q |
+---+---+---+---------------+
| F | F | F |       F       |
| F | F | T |       F       |
| F | T | F |       F       |
| F | T | T |       F       |
| T | F | F |       F       |
| T | F | T |       T       |
| T | T | F |       F       |
| T | T | T |       F       |
+---+---+---+---------------+
```

Fig. 3. The corresponding truth table

III. Practices and Preliminary Results

Over the past two years, we have been practicing this method to enhance students' application-ability training. The discrete mathematics course is an 80-hour course, which is taught in 2-hour sessions three times a week over 13 weeks. Among the three 2-hour sessions, 2 sessions are used for lecturing and 1 session is used for application-ability training according to the lecturing contents.

Every year more than 300 students attend the DM course, thus receiving the training of application-ability. Based on the students' learning trace data from Educoder.net, we find that the time spent on the first type of projects (as defined in Section II-C) is averagely 1.5 hours, while for the comprehensive projects, the average time is 2–2.5 hours. 90% of the students could complete a project stage after 1–8 trials, while for some tough stages, 10% students have to take 78–94 trials (i.e. the deductive proof system project). The results show that the projects are moderately difficult and suitable for the cultivation of students' application-ability.

We conduct a survey on the effects of our application-ability training approach one year later for students who have finished DM course study and have taken several consequent CS courses. There are 219 students finished the survey. The survey questions and corresponding results are listed as follows:

1) Do the training help you understand the applications of DM topics: 50.7% of the students selected "Very helpful", 45.2% selected "Helpful", 4.1% selected "Little help", while no one selected "Almost no help".
2) Is the application-ability training helpful for your subsequent CS course study: 34.2% of the students selected "Very helpful", 59.4% selected "helpful", 5.9% selected "Little help", while 0.5% selected "Almost no help".
3) What do you think the difficulty of the current training: 18.3% of the students selected "Very difficult", 74% selected "Moderate", 7.8% selected "Easy". This result is consistent with the statistic of the average time spend on completing training in previous discussion.
4) Which subsequent CS course do you think the training gives you greatly help on studying: The top 5 selected courses are AA&DS (24.1%), DCD (13.5%), CA and NW (9.7%), AI (9.5%) and PC (9.4%).

Although the results are promising, there are some lessons learned, from which we can improve our future practices in the design of projects. Especially, we find that emphasizing the cultivation of application-ability will weaken the study of theoretical contents. We need to make a very good compromise between these two aspects.

IV. Conclusion

In this work-in-progress innovative practice paper, we report our experiences of enhancing the training of application-ability in DM course. Survey results show that our approach is effective in improving students' ability of computing related curriculum. The results also provide impetus for continuously improving our method. In the near future, we will continue designing more comprehensive projects using Python and on Educoder platform.

REFERENCES

[1] R. L. Page, "Software is discrete mathematics," *SIGPLAN Notices*, vol. 38, no. 9, pp. 79–86, September 2003.

[2] R. Y. Flatland and J. R. Matthews, "Using modes of inquiry and engaging problems to link computer science and mathematics," in *Proceedings of the 40th ACM Technical Symposium on Computer Science Education (SIGCSE'09)*, Chattanooga, TN, USA, pp. 387–391, 2009.

[3] K. McMaster, N. Anderson and B. Rague, "Discrete math with programming: Better together," in *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education (SIGCSE'07)*, Covington, Kentucky, USA, pp. 100–104, 2007.

[4] R. Arnold, M. Langheinrich and W. Hartmann, "Infotraffic: teaching important concepts of computer science and math through real-world examples," in *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education (SIGCSE'07)*, Covington, Kentucky, USA, pp. 105–109, 2007.

[5] T. VanDrunen, "Functional programming as a discrete mathematics topic," *ACM Inroads*, vol. 8, no. 2, pp. 51–58, 2017.

[6] J. P. Cohoon and J. C. Knight, "Connecting Discrete Mathematics and Software Engineering," in *Proceedings of the 36th IEEE Annual Conference on Frontiers in Education (FIE'06)*, San Diego, CA, USA, pp. 13-18, 2006.

[7] T. Li, W. Liu, J. Chen, X. Mao and X. Mao, "Towards connecting discrete mathematics and software engineering," Tsinghua Science and Technology vol. 25, no.3 pp. 325–335, March 2019.

[8] L. C. Ureel, C. Wallace, "Discrete mathematics for computing students: A programming oriented approach with Alloy," in 2016 IEEE Frontiers in Education Conference (FIE'16), Eire, PA, USA, pp. 1–5, 2016.

[9] J. W. McGuffee, "The discrete mathematics enhancement project," Journal of Computing Sciences in Colleges, vol. 17, no. 5, pp. 162–166, 2002.

[10] A. Remshagen, "Making discrete mathematics relevant," In Proceedings of the 48th Annual Southeast Regional Conference (ACM SE '10), ACM, New York, NY, USA, Article 37, 1–6, 2010.

[11] G. David, E. Michael, E. Ali and H. James. "Discrete mathematics/structures: how do we deal with the late appreciation problem?" Journal of Computing Sciences in Colleges. vol. 24, pp. 110–112, 2009.

[12] Python, www.python.org, 2021.

[13] C++ Standard, https://isocpp.org/std/the-standard, 2021.

[14] T. Li, T. Wang, "A Unified Approach to Teach Computational Thinking for First Year NonCS Majors in an Introductory Course." IERI Procedia 2.1, pp.498–503, 2012.

[15] T. Li, W. Liu, X. Mao and H. Zhou, "Introduction to programming: science or art?" in 2013 ACM Innovation and Technology in Computer Science Education conference (ACM ITiCSE'13), Canterbury, United Kingdom, pp. 324, 2013.

[16] P. McGuire, Getting Started with Pyparsing (1st ed.). Sebastopol, CA: O'Reilly, 2007.