Hard theorems made easier: visualisation and interactivity

Michael Reynolds and Ljiljana Brankovic
School of Electrical Engineering and Computer Science, The University of Newcastle
Corresponding Author's Email: Ljiljana Brankovic@newcastle.edu.au

Structured abstract

BACKGROUND
This paper deals with the challenges associated with studying difficult and complex mathematical problems arising in the area of discrete mathematics, which is of great relevance not only to mathematics students but also computer science and software engineering students. Traditional mathematical notation and presentation, while necessarily precise and concise, is sometimes a barrier to understanding, especially when dealing with difficult concepts and proofs. There is a need to facilitate the understanding of deep or hard mathematics. It is proposed that a computer-based graphical tool could assist in this process.

PURPOSE
The goal is to improve understanding of difficult and complex mathematics with dynamic diagrams that illustrate the concepts and are capable of responding to user interaction.

DESIGN/METHOD
We are developing software that will attempt to make difficult and complex theorems and concepts in discrete mathematics more accessible to students, professional mathematicians and computer scientists alike. We have focused initially on two famous problems, namely the Four Colour Theorem and the Graceful Tree Conjecture, and we are developing specialised software for this purpose. The software will allow for both web and PC based technology in order to maximise accessibility and is intended to be (1) extensible, thus allowing other fundamental concepts and theorems to be added, using the overall architecture and underlying building blocks; (2) multi-level, catering for various levels of sophistication of the learners; (3) interactive, allowing the user to examine, investigate and experiment for themselves with many of the concepts involved. We ran an instruction session for student volunteers enrolled in BCompSc and BEng at the University of Newcastle, who subsequently filled in a questionnaire and participated in a focus group to evaluate the software interface, diagram clarity and interactivity level, as well as their interest in, motivation for and understanding of the problem.

RESULTS
We intend that our software will inspire and motivate students, as well as assist in the understanding of challenging mathematical theorems and conjectures.

CONCLUSIONS
Interactive teaching software has great potential, not only to assist students and professionals in understanding difficult mathematical concepts, but also to possibly inspire and motivate students to pursue mathematics, science or engineering at university.

KEYWORDS
Teaching software, intrinsic learning, interactive software

Introduction

This paper deals with challenges associated with studying and teaching discrete mathematics and theoretical computer science. We aim to supplement traditional teaching material such as textbooks and exercise collections with specialised software that would offer a graphical, interactive, and multi-level approach to teaching and learning.

The rationale for this study is two-fold. On one hand, there is a need to promote mathematical problem solving skills among high school and first year university students. On the other hand, there is a need to facilitate the understanding of deep or difficult mathematics by senior undergraduate and postgraduate students. This project deals with a computer based graphical tool that could greatly assist in achieving both of these aims.

The organisation of the paper is as follows. In the next section we discuss the current state of affairs in mathematical education at various levels, starting from primary school, via high school, to University students. We describe a teaching software tool that has potential not only to assist in understanding difficult theorems but also to motivate and invoke interest in maths education. We then evaluate the software through a questionnaire and a focus group. We conclude the paper with discussion of the results.

Mathematical education in trouble

Mathematics is an enabling science that provides foundations for many disciplines, including science, engineering, economics, finance, business and health. Moreover, it is “the only science subject whose study consistently enhances performance across all fields of science”, and it is also “a critical skill that every Australian citizen should be able to develop in order to improve their lives and the lives of those around them” (Rubinstein, 2009). There is a general consensus that mathematical capability in any country is of great importance for its future prosperity (Mathematics And Statistics, 2006, cf. Brown, 2009, Rubinstein, 2009).

Unfortunately, it seems that the state of affairs in Australian maths education has been steadily declining over that last decade and perhaps even longer. A recent Review of Education in Mathematics, Data Science and Quantitative Disciplines for The Group of Eight pointed out that “state of the mathematical sciences and related quantitative disciplines in Australia has deteriorated to a dangerous level, and continues to deteriorate” (Brown, 2009). The problem is evident at all levels of maths education: (1) there is a lack of mathematical education and apparent maths phobia among primary school teachers, which naturally spreads to primary school students (Brown, 2009); (2) many high school maths teachers are not suitably qualified - in 2007, 40% of secondary maths teachers did not have a 3 year degree in maths (Rubinstein, 2009), and the number of year 12 students who choose intermediate and advanced maths is steadily declining - from 41.3% in 1995 to 34.3% in 2004 (Mathematics And Statistics, 2006), with a further decline in recent years (Brown, 2009); (3) the number of Australian students enrolling in maths majors is steadily declining in the recent years, down 15% from 2001 to 2007 (Brown, 2009) and many university programs that once included a significant number of maths subjects no longer do so (Mathematics And Statistics, 2006); maths departments in Australia are shrinking, while the demand for maths sciences graduates is growing at annual rate of 3.5% (Brown, 2009, Rubinstein, 2009).

We next take a closer look at two undergraduate student cohorts that are particularly relevant to our study: computer science students and education students. Typically, a first-year computer science student completes two or three courses in mathematics, including discrete mathematics. This is prescribed as the minimum to ensure students can grasp the subsequent computer science courses. Students who are mathematically inclined have an opportunity to undertake additional mathematical subjects in the second and third year as electives, while others can focus on other areas of computer science. In the second year, computer science students who have presumably been equipped with mathematical reasoning ability in their first year, normally enrol in the core theoretical computer science courses, including Theory of Computation and Algorithms. These courses are designed to build a deep fundamental understanding of the power and limitations of computing, such as distinguishing between problems that can and cannot be solved by means of computers, recognising those problems that can be solved efficiently and becoming skilled at techniques for solving them, and appreciating the strategies for dealing with those problems for which there are no known efficient solutions. Students who have mastered these core areas have a sound basis on which they can build by acquiring more practical skills and becoming competent computer scientists and software engineers (Figure 1).

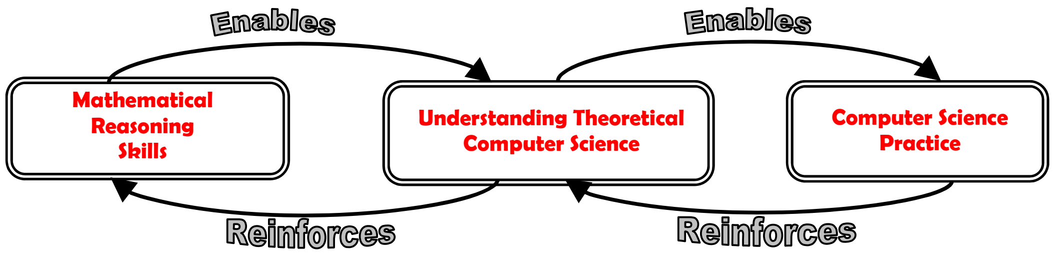

Enables
Enables
Mathematical Reasoning Skills
Understanding Theoretical Computer Science
Computer Science Practice
Reinforces
Reinforces

Figure 1: Mathematical reasoning skills enable understanding of theoretical computer science, which in turn enables computer science practice.

While there are certainly some practical skills in programming and software engineering that can be attained even in the absence of mathematical foundations, such foundations are necessary for those students who aspire to more satisfying and better paid IT positions involving problem solving rather than just routine, laborious tasks (Figure 2). Indeed, "industry and government agencies stress that modern graduates need software and programming skills, as well as teamwork experience, but not at the expense of deep mathematical and statistical understanding" (Mathematics And Statistics, 2006).

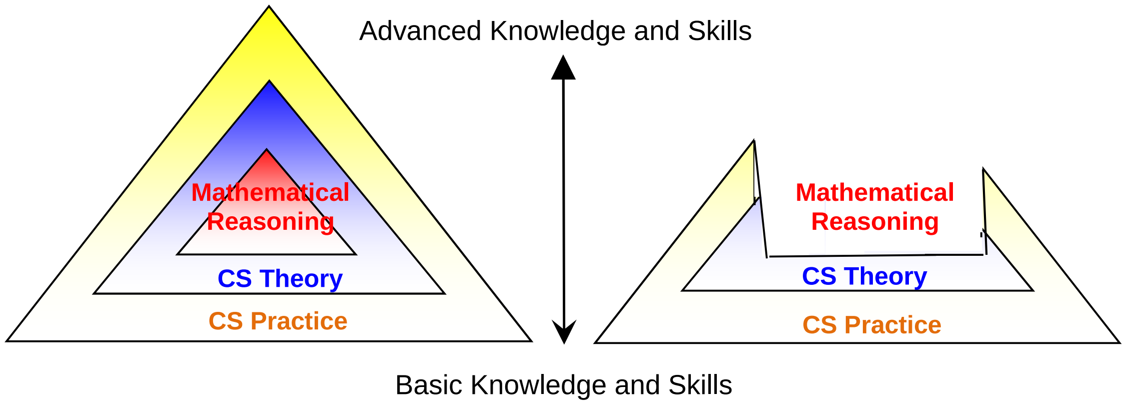


Figure 2: An illustration of attainable competence levels for students with high and rudimentary mathematical reasoning skills

Understanding the core concepts in computer science requires mathematical thought. It is more important to the students' success that they have developed mathematical reasoning skills than knowledge of particular mathematical facts, as the latter can be, and typically are, revised within a theoretical computer science course. However, current teaching practices often produce second and third year students who lack the necessary mathematical reasoning skills, which can have a detrimental effect on students' progress in the computer science program. If students struggle with understanding the fundamental concepts, such as what constitutes a mathematical proof, then their ability to grasp the fundamental concepts of computer science is significantly impaired; this, in turn, makes them less motivated and interested in theoretical computer science subjects. As a result, many students leave such theoretical subjects until the very end of their degrees, and thus miss the opportunity to reinforce and deepen their understanding of fundamental computer science concepts while learning the practical areas of computer science (see Figure 1). Importantly, this situation inhibits students from developing problem solving and critical thinking skills that enable them to design novel practical approaches and critically evaluate available practical methodologies in computer science and software engineering (Figure 2).

The education students' cohorts of particular interest for our project are both future primary teachers and secondary mathematics teachers. Both professions have profound influence on students, their attitude towards mathematics and their subject choices in final years of high school. Unfortunately, primary education students are typically not required to undertake mathematics in high school as an entry requirement and it appears that many future educators do not choose mathematics for the HSC in order to maximise their UAI (Brown, 2009). Moreover, it appears that at most Australian Universities primary education students are not required to complete any purely mathematical course taught by mathematicians (Brown, 2009). One of the exceptions to this is The University of Newcastle where there is a course in Elementary Mathematics taught by the Mathematics department and tailored specifically towards the needs of primary education students, in order to "encourage students to think mathematically and to increase students' confidence in their mathematical ability".

The situation is even more serious in the secondary education sector. Statistics show that only up to one third of year 12 students elect intermediate or advanced mathematics (Brown, 2009). A big contributing factor to such a situation is insufficient qualification of mathematics teachers or even lack of mathematics teachers altogether, as schools report that the highest rate of vacancies is in mathematics (Rubinstein, 2009). This effectively means that in many schools students are unable to undertake advanced or even intermediate mathematics, as there are no available teachers. However, the situation is different in private schools, which creates unequal opportunities for Australian students (Rubinstein, 2009). Given such a generally unfavourable general state of mathematical education, there are initiatives to improve both the motivation and education of students and teachers, including recommendations made to the Group of Eight Universities (Brown, 2009). In this paper, we are proposing that interactive software that can illustrate mathematical concepts and motivate students could contribute to improving this situation.

From textbooks to software

There are already several software packages for graph theory (Peeters et al., 2009, Fajtlowicz et al., 2005, Carbonneaux et al., 1996) but many of them have been designed predominantly for research. For example, Graffiti has so far generated hundreds of conjectures, some of which have been worked on by the very top world graph theorists (Fajtlowicz et al., 2005). Similarly, it was reported in 2005 that there were at least 92 research papers in which the authors acknowledge using the software system GRAPH, the ancestor of newGRAPH (Cvetkovic et al., 2005). Conjecture generator was incorporated in some other packages, including GRAPH, newGRAPH, Graffiti, GraPHedron and AutoGraphiX but they all have focused more on research than teaching. One exception is GrInvIn, which builds upon Graffiti and Graffiti.pc, but has been designed with teaching in mind (Peeters et al, 2009). Some systems are highly specialised, for example, plantri and fullgen by Brinkmann and McKay and surftri by Sulanke are programs for generating certain classes of graphs (plantri and fullgen), while nauty by McKay finds isomorphisms and automorphisms of graphs (nauty). There are several software packages designed also for teaching discrete mathematics and graph theory, including Combinatorica and Graph Magics. LINK (Berry, 2009) is one software package that was designed with both research and teaching in mind. The project was initiated at DIMACS and included the capacity for the development and analysis of conjectures in graph theory.

This project aims at utilizing modern software technology for promoting mathematics and especially mathematical problem solving skills among high school and junior undergraduate students, as well as making difficult and complex theorems and concepts in discrete mathematics accessible to senior students, professional mathematicians and computer scientists alike. To that end we are developing specialised software suitable for teaching and learning the famous Four Colour Theorem (4CT), many of the graph theory concepts necessary for understanding this theorem, as well as the infamous Graceful Tree Conjecture (GTC) and related Alpha labellings of graphs. The 4CT, a famous and historically extremely fruitful map colouring problem, was selected because of the ease of actually stating the problem. It takes only a few minutes even for school students to understand the challenge, but the existing proof is remarkably complicated and deep. This, as well as the importance and outstanding role it has played in graph theory as a whole (see section “Importance” for more detail), and the controversy around the use of computers in the proof, make it a very suitable theorem to consider. The GTC and associated Alpha labellings are also very easy to state and understand and like the 4CT have connections with many problems in a variety of areas of mathematics, including map colouring, Latin squares and graph decomposition.

The software we are developing is designed to be extensible, thus allowing other fundamental concepts and theorems to be added, and multi-level, catering for various sophistication levels of the learners. At the high school and junior undergraduate levels this software can invoke interest in discrete mathematics and teach basic concepts and critical mathematical thinking in a problem based learning environment, where the 4CT and the GTC/Alpha labelling can be used as the motivating problems. At the senior undergraduate and postgraduate levels, the software could be used to facilitate understanding and checking of the long proof of the FCT, and as a tool for progressing towards a solution for the GTC.

Software

The 4CT and Alpha software is based on the Object Oriented (OO) approach to programming where each graph theory construct will be encoded as a class, so as to enable a simple, clean and reusable design and assist in potential future software extensions. It employs the Generics, and Component based programming models (available in the C# language used, as well as in some other modern languages); similarly to the OO approach, this powerful model will facilitate simpler design and reusability of efficient algorithms necessary for accomplishing the tasks. Finally, it allows for both web and PC based technology so as to maximise accessibility.

The software will eventually illustrate the following: (1) the elementary graph theory concepts necessary for understanding of the Four Colour Theorem; (2) the Four Colour Theorem itself; (3) other key results in graph theory including the famous and fundamental Theorems of Kuratowski and Jordan; (4) the computer algorithms involved in the proof of the Four Colour Theorem; (5) Alpha labellings of trees with bounded number of vertices of degree 1, and (6) the algorithms for growing such trees from seeds.

The main features of the software will include the following: (1) levels of sophistication that provide a thread of instruction at various levels of sophistication, with the ability at any stage to change the level; (2) interactivity, allowing the user to examine, investigate and experiment for themselves with many of the concepts involved; (3) extensibility, potentially allowing other adopters to structure other courses and educational material in a similar way, using the overall open architecture and underlying building blocks.

The main distinguishing feature from much other mathematical software is that our project does not aim at replacing textbooks and human tutors, but rather takes a step further and aims at presenting difficult theorems in a way not achievable by traditional means and human explanation. For example, in our 4CT example, the software will graphically represent each of the 633 unavoidable configurations and also generate on demand all of the large number of possible colourings for each of these configurations, giving the learner not only the ability to verify arbitrarily selected parts of the theorem but also an opportunity to gain a deeper insight into any single building block of this complex proof. That is far beyond what traditional textbook and/or human instructor could possibly achieve. Our software has potential to become a tool for problem based learning at the high school and junior undergraduate level. The Four Colour Theorem in its early years, received many false proofs, some of which survived for more than ten years. Due to their simplicity many of these attempted "proofs" and their flaws, can be used to develop critical mathematical thinking in junior students. Additionally, the developed software will improved accessibility, and understanding of the 4CT and other related complex areas of graph theory and computer science, to a wide range of professionals and students.

Importance

The Four Colour Theorem (4CT) undoubtedly had a serious impact on modern graph theory, a critical and dynamic area in the intersection of maths, computer science and communications. Indeed, it is commonly claimed that most of modern graph theory essentially originated with attempts to solve the 4CT (Ore, 1967). To further illustrate the importance of the 4CT for graph theory in general and more specifically for graph theory education, we note that there are actually undergraduate graph theory courses essentially based on 4CT itself where a student learns those elementary graph theoretic concepts that are necessary for understanding the theorem (Wilson, 2002). We hope that software we are developing can take this approach further so that the 4CT can be used as a motivating example in problem based learning for high school and junior undergraduate students.

Although the formulation of the 4CT dates back to the nineteenth century, the first "generally" accepted proof of this famous and challenging theorem came in 1976 (Appel et al., 1977). However, the enormous complexity of the proof, and the heavy reliance on opaque computer algorithms meant that there was general dissatisfaction with the achievement, since it appeared virtually impossible to independently verify the proof, and in fact nobody had fully done so (Robertson et al., 1997). A much simplified and clearer proof appeared in 1997 (Robertson et al., 1997), which resulted from an abandoned attempt at verification of the previous proof (Appel et al., 1977) but this is still far removed in complexity and approachability from the level where many interested professionals and students might follow. More specifically, the original paper reporting the new proof of the theorem is 43 pages long, and yet is not self-sufficient but rather relies on 2 other auxiliary papers (Robertson et al., 1997 - Reducibility, Robertson et al., 1997 - Discharging), containing an abundance of technical details. To fully understand the proof, one must examine a pair of long but tersely expressed computer programs in the C language which requires a multidisciplinary expertise spanning both maths and computer science.

Our approach is to make this proof understandable to a far wider audience, starting from researchers, PhD and masters candidates, to honours and undergraduate students, and even high school students. Importantly, the appeal of this famous problem can be used to promote problem based learning in maths at the high school and junior undergraduate levels. The developed software will be extensible and other problems could be subsequently added.

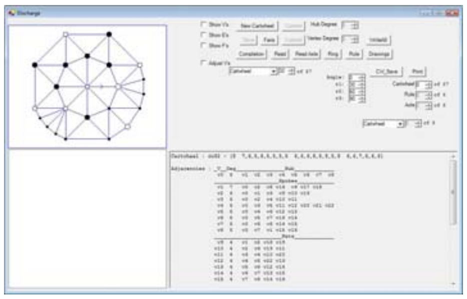


Figure 3: A Cartwheel

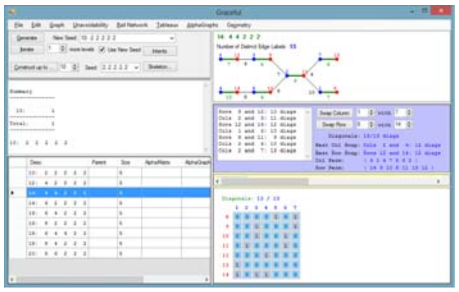


Figure 4: An Alpha labelling of a tree, and Alpha matrix representation of the labelling

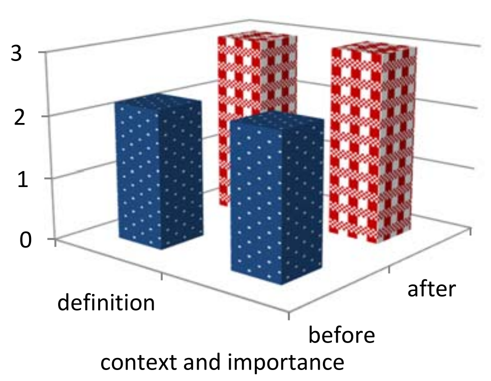
Figure 3 illustrates some of this initial work and shows a "cartwheel", a configuration instrumental in the Discharging half of the Proof of the Four Colour Theorem (Robertson et al., 1997 - Discharging). One of the major achievements of the work of "RSST" was in confirming Heesch's conjecture that only up to the “second neighbourhood" of an overcharged vertex needed to be considered. This was a major improvement on Appel-Haken’s proof (Appel et al., 1977), which had needed to consider configurations possibly wrapping around on themselves. The concept of a Cartwheel embodies this distinction, and so forms one of the objects illustrated by the software.

Our second problem, The Graceful Tree Conjecture (GTC) is still unsolved despite the effort of a large number of researchers and doctoral students around the globe. Just like the 4CT, the GTC is also easy to state and small individual instances can easily be solved by primary and high school students. This problem is also connected to many other famous problems in maths, including the Oberwolfach problem and the enumeration of near-transversals in Latin squares. Figure 4 shows some of the initial work in illustrating Alpha Labellings connected to the GTC. The ability to try labellings for oneself, and to do a computer search looking for the best outcome from a range of possibilities is made available to the user, as well as the associated Alpha matrix, which provides another way of visualising the same problem.

Evaluation

Although our 4CT and Alpha Labelling software is still under development, it is already sufficiently developed to be considered as a proof of concept if we can show that it can be used successfully to improve the teaching, learning and motivation of undergraduate students. To that end we recruited 5 student volunteers, two of them BCompSc students who had done more than 3 maths courses, including graph theory, and the remaining 3 BEng students who had done 3 or less maths courses, none of them graph theory.

We presented two twenty-five minute instruction sessions, one for the 4CT and the other for the Alpha software, and we asked them to fill in a questionnaire for each software and to subsequently participate in focus groups. The questionnaires identified 4 important concepts from the area of Alpha labelling and the same number from the 4CT. These concepts were initially introduced on whiteboard, and subsequently revisited with the aid of the software. Students were asked to evaluate their understanding of the concept definition, as well as the concept context and importance, both before and after their experience with the software. The results are summarised in Figure 5, one the scale 1 to 5, where ‘strongly agree’ corresponds to 5, ‘agree’ to 4, ‘neutral’ to 3, ‘disagree’ to 2 and ‘strongly disagree’ to 1. The student responses to the remaining questions are summarised in Table 1.



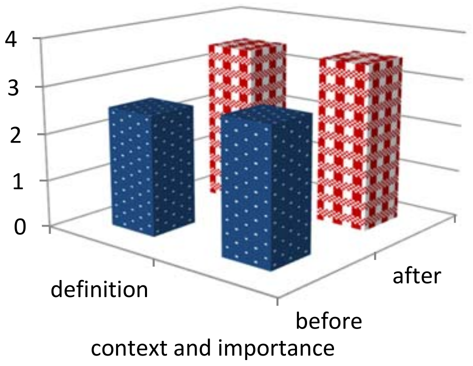


Figure 5: Understanding the concept definition, and the concept context and purpose, averaged over all concepts and all respondents. Left: Alpha Software. Right: 4CT Software

During the focus group session on the 4CT, a participant expressed appreciation for certain features of the software (“I think he visuals are really good, the ability to be able to change

Table 1: Questionnaire Responses – averaged over all respondents on the scale 1 to 5

Question	4CT	Alpha
My overall understanding of the concept has been enhanced by the software.	3.2	3.8
I now have a reasonable understanding of the structure of the 4CT proof/computer search for alpha labelling.	3.2	3.4
The software preforms a useful service.	3.6	3.8
Software diagrams and illustrations are very helpful.	4.2	4.6
The software interface is intuitive.	3	3.4
The software has a sufficient level of interactivity.	3.6	3.6
This session with the software inspired my interest in the 4CT/graceful labelling.	3.4	3.4

from vertex to edge colourings"), while another participant noted that although they did not feel they understood the problem very well, the software got them interested in the in it ("I came into it with no understanding, I don't really feel I understand it, but what it did do for me was create an interest"). Another participant felt that the interface was not easy to get around if you are not an expert and it was suggested to have a basic and an advanced version with different interface, where the former could offer simple things such as invite a user to colour a given graph (map) themselves, while the letter can allow more knowledgeable users to "play bells and whistles". It was also noted that while the 4CT is difficult to prove, it is easy to state and thus should be easily accessible even to high school students providing that there is a basic version for non-experts as suggested above. One of the participants was colour-blind so a lot of the discussion revolved around alternative visual representations, and it was suggested that different shapes in place of colours would provide more clarity than numbers.

There was quite a lot of enthusiasm about the Alpha software among the participants who had graph theory background: "I thought that it was really really good", "I've never heard of that problem before but just from different colours and being able to see the perfect matching as occurring and all that it just made it almost click immediately as to what you are trying to do", "The visualisation is really clever", "Even if you don't understand why at the start you can at least understand how to check if it is a feasible solution". The participants who have not done graph theory did not quite grasp all the concepts so we provided additional explanations during the focus group and subsequently there was a general consensus that the Alpha software "seems to be a lot more helpful that trying to look it up in a textbook". It was suggested to allow users to attempt to find their own labelling.

Discussion and conclusion

There was a clear distinction in opinions between the two participants who studied graph theory, and the three participants who did not. While the level on instruction we provided was clearly ample for the first group, it might have not be sufficient for the second group.

Both 4CT and Alpha software were well received and appreciated by the participants of the focus group. It appears that Alpha software generated more enthusiasm which could be at least partly due to the fact that it was somewhat easier to fully grasp its concepts than the 4CT, at least for the participants with graph theory background. The suggestions made in the focus group to enable novice users to construct their own colouring/labelling is consistent with our own vision where instruction will be provided at various levels of sophistication. Since both the 4CT and Alpha software are highly specialised to support learning of difficult maths, it is not surprising that a novice user without any knowledge of graph theory needs some preliminary instruction and/or a basic version of the software/interface, where the user would be invited to colour/label a given graph. In that way the software would teach novice users "without them realising they are learning", as cleverly put by one of the participants.

Interactive teaching software, such as the 4CT and Alpha software, has great potential not only to assist students and professionals in understanding difficult mathematical concepts, but also to possibly inspire and motivate students to pursue mathematics, science or engineering at university. The crucial requirements for this appear to be an intuitive interface and engaging activities that could be pursued without any previous knowledge required.

References

Appel, K. and Haken, W. (1977). Every Planar Map is Four Colorable, Part I. Discharging, and Part II. Reducibility, *Illinois J. Math*. 21 (1977), 429-490 and 491-567.

Berry, J. (2009). Improving discrete mathematics and algorithms cirricula with LINK. ACM's 1997 SIGCSE/SIGCUE Conference on Integrating Technology into Computer Science Education , 2009.

Brown, G. (2009). *Review of Education in Mathematics, Data Science and Quantitative Disciplines*. Report to the Group of Eight Universities, December 2009.

Carbonneaux, Y., Laborde, J-M. and Madani, R. M. (1996). CABRI-Graph: A tool for research and teaching in graph theory. *Graph Drawing, Lecture Notes in Computer Science*, 1027/1996.

Cvetkovic, D. and Simic, S. (2005). Graph theoretical results obtained by the support of the expert system "Graph" - An extended survey. In *Graphs and Discovery, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 69.

Fajtlowicz, S., Fowler, P.W., Hansen, P., Janowitz, M. F. and Roberts, F. S. Eds. (2005) Graphs and Discovery. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 69.

LINK, http://dimacs.rutgers.edu/~berryj/LINK.html, accessed October 2013.

Mathematics And Statistics: Critical Skills For Australia's Future. The National Strategic Review Of Mathematical Sciences Research In Australia, December 2006. Retrieved May 2010 from www.review.ms.unimelb.edu.au.

nauty, http://cs.anu.edu.au/people/bdm/nauty/ *, accessed on 2 October 2013.*

Peeters, A., Coolsaet, K., Brinkmann, G. and Van Cleemput, N. and Fack, V. (2009). GrInvIn in a nutshell. *J Math Chem*, 45:471–477.

plantri and fullgen, http://cs.anu.edu.au/people/bdm/plantri/, accessed on 2 October 2013.

Rubinstein, H. (2009). *A National Strategy for Mathematical Sciences in Australia*, National Committee for the Mathematical Sciences and Australian Council of Heads of Mathematical Sciences.

Robertson, N., Sanders, D. P., Seymour, P. D. and Thomas, R. (1997). The four colour theorem, *J. Combin. Theory Ser. B.* 70, 2-44.

Robertson, N., Sanders, D. P., Seymour, P. D. and Thomas, R. (1997). Reducibility in the Four Colour Theorem. Unpublished paper. Retrieved September 2013 from http://people.math.gatech.edu/~thomas/PAP/reduce.pdf

Robertson, N., Sanders, D. P., Seymour, P. D. and Thomas, R. (1997). Discharging Cartwheels. Unpublished paper. Retrieved September 2013 from http://people.math.gatech.edu/~thomas/PAP/discharge.pdf

Ore, O. (1967). *Four-colour Problem*. Academic Press.

Wilson, R. A. (2002). *Graphs, Colourings and the Four-colour Theorem*. Oxford University Press.

Acknowledgements

The authors thank Catherine Greenhill for reviewing a part of this paper and to Leanne Rylands for bringing the work of Brown's and Rubinstein's reports to our attention.

Copyright statement