

# CASE STUDIES IN TEACHING MATHEMATICAL MODELLING\*

C. S. TONG†

**Abstract.** We discuss our experience in using group projects on real world problems for teaching Mathematical Modelling. Three sample problems, which were well tackled by students, are described and we summarize what we've learned from these experience in the form of five rules for teaching modelling.

**Key words.** mathematical modelling, finite state machine, graph theory

**AMS subject classifications.** 93A99, 05C45, 05C90

**1. Introduction.** At Hong Kong Baptist University, modelling is deemed the key element of our curriculum in Mathematical Science major and many subjects include project work specifically to enhance student's ability in modelling. Indeed, the only compulsory core subject in the final year is the subject on Mathematical and Statistical Modelling. I have been involved in team teaching this subject for the past seven years and I consider it the easiest subject to teach, but the hardest subject to teach *well*.

What's easy to teach are the skills and techniques of standard mathematical models, which are well covered in standard texts such as [1-2], and this was the approach we adopted early on. In order to expose students to a wide range of techniques, we covered four topics in a semester and required students to carry out group projects for each topic. The workload was intense, and students became well drilled in tackling textbook problems. However, whenever students were asked to apply the techniques they have learnt to real world and they invariably froze! As Powell *et al.* succinctly put it, students "*are talented in manipulating symbols and following textbook recipes but terrified of applying mathematical knowledge in unknown territory*" [3]. This brings us to the hard part, namely, teaching the art of mathematical modelling. Indeed, students themselves are aware of the problem as reflected in teaching evaluation feedback.

Thus in the past five years, we adopted a different approach. The subject is now divided into two parts. The first part involved the previous skills-oriented approach, albeit reduced to just 2 topics. The second part is problem solving and group projects. During class, students are given small scale problems to work on and the emphasis lies in encouraging discussion and brainstorming. The problems chosen do not require very deep mathematical machinery but usually involve subtle tricks in all branches of mathematics. They serve to arouse student interests and warm up their mathematical instincts. Examples include the famous Lloyd's 15-puzzle, explaining the effect of gaining a day in the 80-days Around the World story, non-transitive games and well know probability "paradoxes".

Students are also given a small list of interesting real world problems to think about. By about the third week of the semester, they are to form groups of 3 to 4 and each group to propose a real world problem to work on, which can be taken from the list but are strongly encouraged to come up with their own.

---

\*Partially supported by a Teaching Enhancement Grant from the Hong Kong Baptist University

†Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong (cstong@hkbu.edu.hk)

In the next section, I shall describe three sample problems in which students displayed flair and ingenuity in modelling and use them to illustrate five rules which we have come to value as sound principles for teaching the art of mathematical modelling. These rules are summarized in section 3. We hasten to add that every class is different and unique and there cannot be any golden rules that work for every class, except perhaps rule 5 !

## 2. Case Studies.

### Case 1: Cross Chess

#### The Problem

Cross Chess is a simple but surprisingly difficult game played on a  $5 \times 5$  board. Each grid in the board is marked with either a nought or a cross (in the electronic version of the game, each grid is either lit up or remain unlit). A move in the game consists of pressing a grid position which would toggle the status of the current grid as well as its immediate neighbours: above, below, left or right. For grids in the middle, there will be 4 neighbours, whereas grids on the sides will have 3 neighbours and the corner grids have only 2 neighbours. Thus each move would toggle 3 to 5 grids, depending on the position of the move. The game starts with a random pattern of noughts and crosses and the goal is to find a minimum number of moves to toggle all the crosses to noughts.

EXAMPLE 1. A sample game is shown below: each move is indicated by shading.

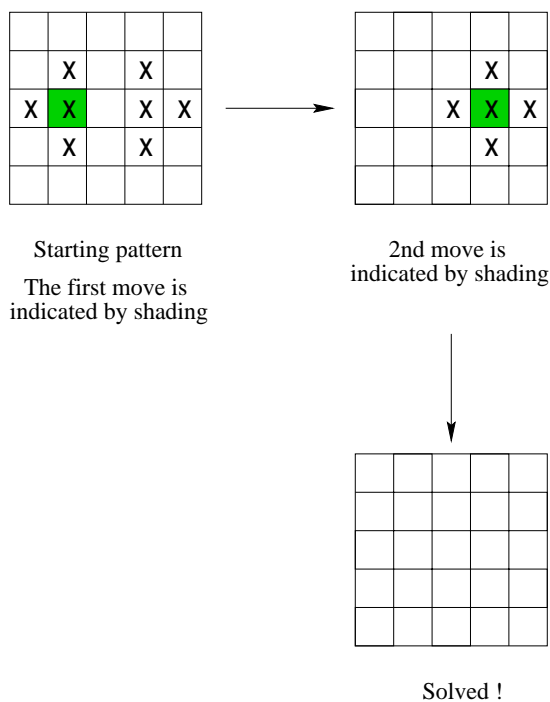


FIG. 2.1.

#### The solution

Since each grid position can take only one of two states, we shall work in the





If these two necessary conditions are satisfied, then solutions are given by the first 23 rows:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_{23} \end{bmatrix} = x_{24} \times \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + x_{25} \times \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} b_1 + b_3 + b_5 + b_8 + b_{10} + b_{11} + b_{12} + b_{20} + b_{21} + b_{22} + b_{24} + b_{25} \\ b_1 + b_2 + b_4 + b_5 + b_7 + b_{13} + b_{14} + b_{15} + b_{19} \\ b_1 + b_3 + b_4 + b_5 + b_6 + b_8 + b_9 + b_{13} + b_{14} + b_{16} + b_{17} + b_{18} + b_{19} + b_{20} + b_{22} \\ b_3 + b_4 + b_5 + b_7 + b_{11} + b_{12} + b_{14} + b_{15} + b_{17} + b_{23} + b_{24} + b_{25} \\ b_2 + b_3 + b_5 + b_6 + b_{11} + b_{13} + b_{15} + b_{18} + b_{20} + b_{21} + b_{22} \\ b_1 + b_6 + b_{13} + b_{16} + b_{18} + b_{19} + b_{21} + b_{23} + b_{25} \\ b_2 + b_4 + b_6 + b_7 + b_9 + b_{10} + b_{14} + b_{16} + b_{17} + b_{18} + b_{22} \\ b_1 + b_3 + b_6 + b_8 + b_9 + b_{15} + b_{16} + b_{18} + b_{20} + b_{21} + b_{23} \\ b_3 + b_7 + b_8 + b_9 + b_{11} + b_{14} + b_{15} + b_{16} + b_{19} + b_{22} + b_{23} \\ b_3 + b_5 + b_7 + b_8 + b_{10} + b_{11} + b_{13} + b_{15} + b_{16} + b_{17} + b_{21} + b_{25} \\ b_1 + b_2 + b_4 + b_9 + b_{10} + b_{11} + b_{12} + b_{13} + b_{14} + b_{16} + b_{17} + b_{18} + b_{19} + b_{21} + b_{24} + b_{25} \\ b_1 + b_2 + b_4 + b_5 + b_{11} + b_{12} + b_{14} + b_{15} + b_{17} + b_{23} + b_{24} + b_{25} \\ b_2 + b_3 + b_5 + b_6 + b_{10} + b_{13} + b_{14} + b_{18} + b_{21} + b_{22} \\ b_1 + b_2 + b_3 + b_7 + b_9 + b_{13} + b_{14} + b_{15} + b_{19} \\ b_1 + b_2 + b_5 + b_8 + b_9 + b_{10} + b_{11} + b_{14} + b_{15} + b_{16} + b_{17} + b_{19} + b_{21} \\ b_3 + b_7 + b_8 + b_9 + b_{11} + b_{15} + b_{16} + b_{18} + b_{20} + b_{21} + b_{23} \\ b_3 + b_4 + b_7 + b_{10} + b_{11} + b_{12} + b_{15} + b_{17} + b_{18} + b_{19} + b_{23} \\ b_1 + b_6 + b_7 + b_{11} + b_{13} + b_{17} + b_{18} + b_{19} + b_{23} + b_{25} \\ b_2 + b_3 + b_6 + b_9 + b_{11} + b_{14} + b_{15} + b_{17} + b_{18} + b_{19} + b_{23} \\ b_1 + b_3 + b_5 + b_6 + b_8 + b_{10} + b_{16} + b_{18} + b_{20} + b_{21} + b_{23} \\ b_4 + b_5 + b_8 + b_{12} + b_{13} + b_{15} + b_{16} + b_{18} + b_{20} + b_{21} + b_{22} \\ b_3 + b_4 + b_5 + b_7 + b_9 + b_{11} + b_{12} + b_{13} + b_{21} + b_{22} + b_{23} \\ b_1 + b_3 + b_4 + b_5 + b_6 + b_9 + b_{12} + b_{17} + b_{18} + b_{19} + b_{21} + b_{22} + b_{23} + b_{25} \end{bmatrix}$$

Note that  $x_{24}$  and  $x_{25}$  can take arbitrary values and hence there are at most 4 distinct solutions corresponding to the 4 possible combinations of values of  $(x_{24}, x_{25})$ . The shortest solution (the one with the smallest number of moves) is the solution with the smallest one-norm, which can be easily checked with a computer.

A Java implementation of the game, and the solution as give above in case it is needed, can be accessed on the website at (<http://www.math.hkbu.edu.hk/~cstong/sci3510/xchess.html>).

**Case 2:** Breaking a digital lock

The problem

In Hong Kong, many buildings and apartments have front gates with digital locks which can be opened using a 4-digit password. For security purposes, the password is changed regularly and it is quite common for people to forget the password! If every 4-digit sequence is tried at random, the person would need to type in 40000 keys to guarantee opening the lock.

However, typically, a digital lock accepts a continuous input of keys until a sequence of 4 digits matches the required password whereupon the lock will be released. Thus it should be possible to type in a shorter sequence of digits that contains all the possible passwords as 4-digit substrings. The problem is to find (one of) the shortest such sequence. Clearly, the shortest string must be at least 10003 digits long in order to contain at least 10000 4-digit substrings.

Solution A

The problem is modelled by a finite state machine. There are 10000 states, each represented by a distinct 4-digit substring. Two states are connected by a directed edge labelled  $n$  if after adjoining the digit  $n$  to the 4-digit substring of the first state, the last 4 digits now forms the 4-digit substring represented by the second state. An example is illustrated graphically below:

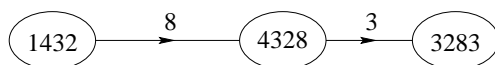


FIG. 2.3.

The entire finite state machine is a digraph with 10000 vertices, each with 10 in-edges and 10 out-edges.

The solution to the digital lock problem now becomes a problem in graph theory: find a Hamiltonian circuit which visits each and every vertex once and once only, in the digraph constructed (as described above).

Given a Hamiltonian circuit, the solution to the digital lock problem can be obtained by arbitrarily starting at a vertex, say, (0000), which is equivalent to typing in the zero key 4 times. Then follow the circuit and typing in the number associated with the edges in order.

By the time we reach the last vertex in the circuit, we would have typed in  $4+9999=10003$  keys which is the theoretical minimum.

Note that there is no need to key in the last edge label as we need not return to the starting vertex.

To illustrate this solution, consider a 3-digit lock where each digit is 0 or 1. The associated directed graph has 8 vertices (for the 8 possible passwords) and each vertex has 2 in-edges and 2 out-edges.

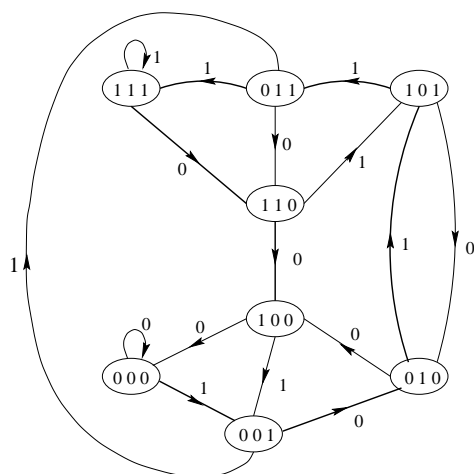


FIG. 2.4.

A Hamiltonian path is shown in bold edges, starting from (000), which involves typing in the zero key 3 times, and then key in the edge labels as we traverse the Hamiltonian path. The solution obtained is 0001011100 and it is easily verified that all the passwords are contained in this sequence.

Solution B

The above solution is unsatisfactory as finding a Hamiltonian path in a general graph is NP-complete, which essentially means that the problem is computationally intractable for large graphs. A separate group of students found a much neater solution using a more subtle finite state machine.

In this machine, there are now 1000 states, each represented by a distinct 3-digit substring. The connection between states follow a similar rule as before, so that this finite state machine can be represented by a directed graph with 1000 vertices and 10,000 edges. A small portion of this graph is shown below

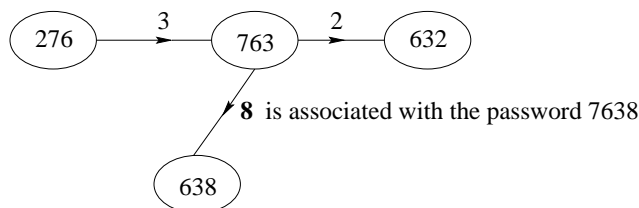


FIG. 2.5.

Note that each edge is uniquely associated with a distinct 4-digit substring by adjoining the edge label to the 3-digit substring of the vertex from which the edge emerges (see diagram above).

Thus in this directed graph, every edge represents a password and hence the solution to the digital lock problem is equivalent to finding an Eulerian circuit for the graph, which visits each and every edge once and once only.

As an illustration, consider a 3-digit lock where each digit is 0 or 1. There are 8 possible passwords and the associated directed graph is given below:

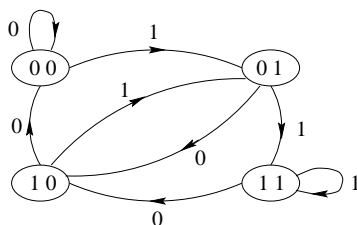


FIG. 2.6.

An Eulerian circuit starting at the (00) vertex is:

$$(00) 0 (00) 1 (01) 1 (11) 1 (11) 0 (10) 1 (01) 0 (10) 0 (00),$$

where the vertices are the 2-digit substrings in parenthesis and the single digits are the edges. The solution for the digital lock problem involves keying in the zero key twice (to start at vertex (00) and then key in the edge labels along the circuit, i.e. 0001110100. It is easy to verify that all possible 3-digit passwords are contained in this sequence.

For the 4-digit lock case where each digit can take values from 0 to 9, the solution would require typing in 3 digits (for the starting vertex)+10000(for each of the edges)=10003 keys, which is the theoretical minimum.

Since for each vertex, there are exactly 10 in-edges and 10 out-edges, a simple generalization of a result by Euler proves that an Eulerian circuit exists. Moreover, there exists an efficient, linear time algorithm (Fleury's algorithm [4]) to find it.

**Case 3:** Fair scores for Multiple Choice Questions.

The Problem

Multiple choice questions are commonly used as a convenient assessment tool. However, it has been criticized for not being able to award marks for partial knowledge. For example, if a student correctly narrows down the answer to one of two choices but made a wrong guess, he or she would receive exactly the same mark as another student who is totally ignorant and merely made a random guess that is also wrong. Of course, when the number of questions is very large, it can be argued that the first student should still expect to score higher than average.

Nevertheless, the concept of Partial Credit Multiple Choice Bubble Format has been proposed and used for short quizzes at the Department of Education in Hong Kong Baptist University since 1992. In this format, each question has 3 choices  $A$ ,  $B$  and  $C$  but the student may choose one of 13 options as indicated in the bubble chart below

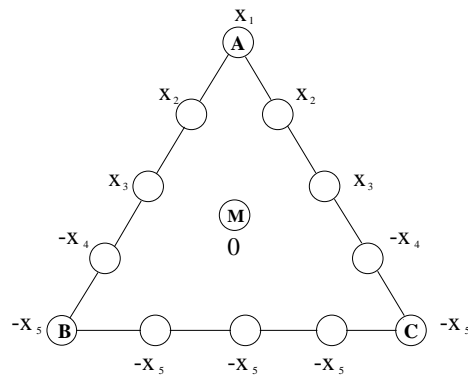


FIG. 2.7.

Without loss of generality, let  $A$  be the correct answer, then marks associated with each of the choices are given next to the boxes: all the  $X_i \geq 0$ .

Note that all the boxes on the bottom line receive the same negative marks since all of them rejected  $A$  as the answer and thus are equally wrong. The middle box  $M$  means the student has no idea which answer is more likely and is awarded zero mark for zero knowledge. The question is: how do we determine the values of  $X_1, X_2, \dots, X_5$  to ensure fair scoring?

Solution

Since the middle box is given zero mark, the student effectively only has 12 choices. Assuming the student randomizes, then the expectation of the mark  $X$  would be

$$E(X) = \frac{1}{12}(X_1 + 2X_2 + 2X_3 - 2X_4 - 5X_5)$$

Assumption 1: the scores should be chosen so that random choices lead to

zero marks. That is,

$$E(X) = 0 \quad \Rightarrow \quad X_1 + 2X_2 + 2X_3 - 2X_4 - 5X_5 = 0$$

In practice, there will be variations from the mean so that students who simply randomize may still score positive or negative depending on their luck. Thus a fair scoring scheme should minimize the elements of chance. This leads to Assumption 2: the variance of  $X$  should be minimized. Assumption 3: there should be higher marks for boxes closer to the correct answer (i.e.  $A$ ). Thus  $X_1 > X_2 > X_3$  and  $X_4 < X_5$ . The 3 assumptions together constitute a constrained quadratic programming problem:

$$\begin{aligned} \min E(X^2) &= \frac{1}{12}(X_1^2 + 2X_2^2 + 2X_3^2 + 2X_4^2 + 5X_5^2) \\ \text{subject to} &\begin{cases} X_1 + 2X_2 + 2X_3 - 2X_4 - 5X_5 = 0 \\ X_1 > X_2 \\ X_2 > X_3 \\ X_5 > X_4 \\ X_1, X_2, X_3, X_4, X_5 \geq 0 \end{cases} \end{aligned}$$

The problem is solved using a software package called LINDO [5] and after arbitrarily scaling to set  $X_5 = 10$ , the solution is:  $X_1 = 22$ ,  $X_2 = 12$ ,  $X_3 = 2$ ,  $X_4 = 0$ , and  $X_5 = 10$ . For a quiz of 20 questions using this Partial Credit Multiple Choice format, the highest mark possible is 440, the lowest mark possible is -200, and a monkey taking the test is expected to score 0, with standard deviation of  $9.3/\sqrt{20} \sim 2.1$  for the average score per question.

**3. Discussions.** Here we summarize our experience in the form of 5 rules.

**Rule 1:** The hook

Hook students to the problems! Once the students are motivated, they will find ways to solve the problem. The best way to do so is of course to encourage students to propose their own problems. However this can be rather difficult and in my experience, it is best to provide a small pool of interesting “real” problems — i.e. problems that naturally arise from everyday situations — for the students to choose. Once students have seen such examples, they are more confident to propose similar problems. This is well illustrated by the students who proposed the problem in case 1. Although they found the problem difficult, they never gave up and continued to search for solution, which included reading journal papers, books and searching the internet for help. When they finally solved the problem and presented it in class, the sheer joy, excitement, and satisfaction was plain to see and very infectious.

**Rule 2:** Big brother is watching you

Let me first recount an anecdote of Manuel Blum. When Blum was a little boy, he was working on a difficult math problem when his big brother watched over his shoulder and said “there’s a trick you know”. Sure enough, Blum soon found the trick and solved the problem.

Hong Kong students work hard, but they can be very dependent on the teachers for standard, authoritative answers. Such an attitude is disastrous in a course like mathematical modelling. Thus my rule number 2 is based on the general principle “never give students answer!”. When students come to

me for advice, I would tell them, in Blum's style, "there is a trick, you know". In my experience, once students are assured that the problem can be solved, then they are quite likely to find a solution. Again, case 1 is illustrative here. After several weeks on the problem, the students were despairing and were beginning to think the problem is impossible — after all, all the other math games such as Rubik's cube have solutions attached to the game, but no such solution is attached in cross chess! I told them, in total confidence, that "there is a trick, find it!" and they did!

**Rule 3:** Question is the answer

There are times when rule 2 cannot apply. For instance, when students are stuck and are beginning to lose interest (remember rule 1!), or when they are going in the wrong direction. Still, the principle of never giving them answers must be upheld. It is very tempting to break this principle but you must resist the urge at all times! But if you are pinned to a corner and you have to give them some tips then do so by asking more questions.

For example, in case 2, two groups of students presented their draft solutions in a mid-project report session. Instead of telling the first group that their solution is no good for large problems, I simply ask both groups to estimate the computation complexities of their algorithms. Very soon, the first group realized that their algorithm is inferior to that of the second group. When they tried their algorithm to the full problem of a 4-digit digital lock with 10 keys, they learned a very painful lesson about NP-complete problems. Through class discussion, the whole class then learned about complexity of algorithms and how intractable NP-complete problems are. In my experience, only lessons learned the hard way last!

**Rule 4:** Judge not, lest ye be judged

The fourth rule is: never criticize student's underlying assumptions in their models. Indefinite questions such as "how do you justify your methods" or "why do you make such assumptions" usually do not work well especially when student do not even realize they have implicitly made an assumption. Students tend to build models based on their mathematical intuition and cannot articulate such intuition well, especially since it is not very well developed yet (this is of course the *raison d'être* for a course in mathematical modelling!)

I would suggest asking definitive questions that help focus students on their possibly implicit assumptions and let the students challenge them, and in the process strengthen their mathematical intuition. To illustrate, consider case 3. In their solution, the marks for box  $A$ ,  $B$ ,  $C$  add up to  $22 - 10 - 10 = 2 > 0$ . Hence if a student does not choose the answer boxes uniformly randomly (as assumed) then the marking scheme would favour such a student who is so confident that he/she always choose from the corner boxes!

Now if their assumption 1 is challenged in class, it would quite likely put the group into a highly defensive position in which they could only see the rationale for their assumptions (all of them, not just the one being challenged!) and couldn't widen their horizon to explore other alternatives. Thus instead of asking the students to defend their assumption 1, a better strategy is to ask them what would happen if a student always choose from one of the three corner boxes. In this way, students can learn to analyse their own solutions and come to appreciate just how deep the concept of "fairness" can be.

Another useful question is: “out of a 20 questions quiz, what is the minimum mark that a student must score for the teacher to be confident that the student is knowledgeable, and does the result make sense?”. Such a question would lead the students to analyse their proposed scoring scheme using hypothesis testing, and to compare the results with the standard multiple choice format.

**Rule 5:** The rule to end all rules

There are exceptions to every rule. When all else fail, just be with the students and follow your instincts!

#### REFERENCES

- [1] R. HABERMAN, *Mathematical Models*, Prentice-Hall, 1977.
- [2] M. S. KLAMKIN (ed), *Mathematical Modelling: Classroom Notes in Applied Mathematics*, SIAM, 1995
- [3] J. A. POWELL, J. S. CANGELOST, AND A. M. HARRIS, *Games to Teach Mathematical Modelling*, SIAM REV, Vol.40, No1, pp.87-95, 1998.
- [4] J. A. BONDY AND U. S. R. MURTY, *Graph Theory with Applications*, Macmillan, New York, 1976
- [5] *User's Manual for Linear, Integer and Quadratic Programming with LINDO*, Release 5.0, Linus Schrage, Scientific Press.