Assessment Models and Qualitative and Symbolic Analysis Techniques for an Electrical Circuits eTutor

Adrian Muscat
Dept of Communications and Computer Engineering
University of Malta
Msida, Malta
Email: adrian.muscat@um.edu.mt

Jason Debono
Institute for Electronics
Malta College for Science and Technology
Corradino, Malta
Email: jason.debono@mcast.edu.mt

***Abstract*—This paper is about assessment models, domain expert models and user interfaces as components in an Intelligent Tutoring System that serves junior classes in electrical circuits. Two student models for the purpose of automated assessment are developed and tested. One of the models is a Markovian graph model, while the other is a histogram model. The effectiveness of these models in tracing the student's declarative as well as procedural knowledge is studied and compared to human assessment. The domain expert models are based on qualitative analysis and on symbolic quantitative techniques. These models are used to test declarative statements made by the student and also to generate a solution to the problem. The circuit analysis techniques are also studied from an educational point of view and are compared to numerical models on the basis of how much they help the student assimilate the knowledge. Two types of user interfaces are developed, one is text command line based, and the other comprises a graphical user interface. These three building blocks are used in the development of two independent systems, which are field tested with the engagement of polytechnic teachers and students at the higher national diploma level. The technical and pedagogical results obtained for the two modules are good and encouraging.**

***Keywords*-Electrical; Intelligent Tutoring System; Qualitative; Symbolic; Markov Model; Assessment;**

I. INTRODUCTION

Electrical circuit theory is one of the foundational courses studied in college, polytechnic and university degrees in the areas of electrical and electronics engineering. Later courses, such as electronic circuits and electrical machines, build on a good knowledge-base in circuit theory. It is therefore important that the student acquires a good handle in this theory. As with other foundational courses good mentoring from the very start is important in reaching this goal. As such Computer Aided Learning (CAL) or Intelligent Tutoring Systems (ITS) software can play a significant part in the progress of the student. In [1] the authors develop a prototype circuit simulator based on qualitative and symbolic reasoning, that emulates the process or sequence of steps that a person carrying out circuit analysis manually usually engages in. In this paper the system is augmented with the addition of an assessment module that is useful in giving feedback and following the progress of the student. These two models form the basis or kernel for an ITS or eTutor.

Personal human tutors are very effective in increasing the learning rate and studies show that personal tutoring helps students achieve significantly higher assessment scores [2] and [3]. A system of personal human tutors is however unsustainable and unrealizable due to the financial cost of the project as well as the lack of availability of human tutors. ITSs promise to deliver a personal mentor or a tutor to each student in class. The quest is to model the tutor using artificial intelligence techniques. Two early and substantially successful systems are PUMP [4] and SHERLOCK [5]. PUMP is a secondary school algebra tutor and SHERLOCK is a virtual practicing space for apprentices in electronics troubleshooting. More recent systems were designed for physics [6] and medical sciences [7], and the systems proposed and explored in [8] and [9] are probably the first ITS for electrical circuits. The system described in [8] is a production system and rules are defined to generate problems, solve problems and judge mistakes. The system generates and solves problems that consist of simple parallel and series combinations of impedances. Judging mistakes is carried out by analysing and coding several real-world student mistakes as mal-rules.

ITS were initially evaluated from an Artificial Intelligence point of view rather than from an educational impact focus. This approach is however changing and ITS is much more of an interdisciplinary research area today. Indeed today more emphasis is placed on evaluating the impact of ITS from an educational point of view. Nevertheless, the independent development of a number of components that contribute to the realisation of the ITS is a necessity.

This paper contributes a Markovian Assessment Model to assess solutions to problems in electrical circuits and a Nodal Analysis electrical circuits expert model for circuits of arbitrary topologies. Two systems that target electrical circuit classes are discussed. The first system accepts input circuits that are made up of an arbitrary number of resistors and one voltage source. This system processes serial and parallel connections of resistors to provide a machine generated full answer and allows the student to drive the process him/herself. In the latter case, the output log-file provides the input to a student assessment model that assesses the student for both declarative and procedural knowledge. The second system accepts input circuits that are made up of an arbitrary number of resistors, voltage sources and current sources. The software tool then tutors the user on how to select valid spanning trees and the corresponding fundamental cutsets for the input circuit. The symbolic Kirchhoff's Current Law (KCL) equation for each fundamental cutset is then generated by the program, which the user or student can compare to his/her workings. Both tools analyse the topology of the input circuit to accomplish their respective type of analysis. The first system is targeted to Malta Qualifications Framework (MQF) Level Four students while the second system is to be used by MQF Level Five students.

This section introduced and motivated the need for tutoring systems in electrical engineering. The rest of the paper is organised as follows: Section II gives background information and discusses related work in the literature. In particular Section II-C reviews circuit analysis techniques and section II-D reviews student models, both of which are important in this work. The framework and models developed are described in section III. Sections IV and V describe and discuss the results. Finally, section VI concludes the paper.

II. Background and Literature Review

This section provides (a) background to the current state-of-the-art ITS architecture and the current practices in schools teaching electrical theory, and (b) a literature review of domain expert models, which in this case are circuit analysis techniques, and students assessment models that have been proposed in various ITS research projects.

A. Intelligent Tutor Architecture

Fig. 1 depicts the general architecture for an ITS, summarized from [10] and [11]. The main components of such a system are a Domain or Expert model, a Student Model and a Tutoring or Pedagogical model. The problem Solving Environment or Human Computer Interface is another component that should not be underestimated. To these components we have added a human Tutor model, which is useful to tune the ITS system to the peculiarities of specific human tutors. This involves machine learning from data generated by the human tutor and may contribute to a more effective ITS.

The problem solving environment defines the way the human student interacts with the system. It defines for example whether the interaction is via a text editor, via a graphical editor and more recently via speech and vision. The interface selected has a profound effect on the pedagogical nature of the system. It can for example limit the types of inputs that a student is allowed to enter. This can be either thought of as a limitation, in other words less freedom of roaming space for the student or as a forced scaffolded learning pedagogy. In general, the ITS research community agrees that the problem solving environment should emulate as far as possible the real

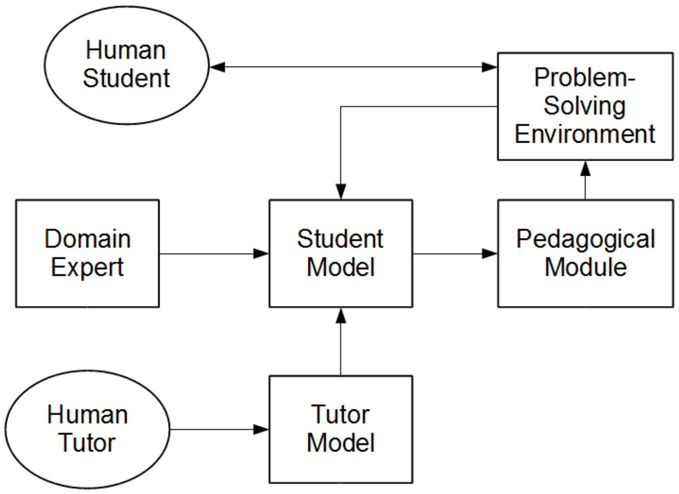


Fig. 1. Intelligent Tutoring System architecture, summarized from [10] and [11]

world environment and at the same time facilitate the learning process [10]. The latter requirement should be considered in the light that scaffolding should be completely removed at the tail end of the learning process [12].

The Domain Expert module provides an interpretation to the student's input. This module determines whether the atomic assertions of the students are correct in the specific domain area. Additionally the expert module should generate a full answer to problems given to the student, including an explanation in a natural humanistic language, symbols and diagrams included. This implies that the system must apply a causal human-like reasoning process when generating an answer. Finally, this module encompasses all the knowledge that a student is expected to learn and can therefore be termed as the *ideal student model*. In other words it is a benchmark that students strive to reach.

The student model is a record of the knowledge state of a student. In its most simple form it is a copy of the expert's model that is tagged with information of how well the student has demonstrated knowledge of each component in the expert model. Knowledge can be classified into classes; declarative and procedural [10]. Most often declarative knowledge implies learning rules and relationships. On the other hand procedural knowledge is not typically well defined and relates to the problem solving approach itself. For the case of declarative knowledge statistical models may suffice, while for procedural knowledge models that consider the sequence and order, in which declarative knowledge is applied are desired. In the electrical circuits ITS described in [9] and [8] the student model is limited to recording declarative knowledge and Mishra et al point out that the model should also capture the knowledge flow [13]. Finally, the student model is used to assess the progress of the student and its output is very useful for the pedagogical module that observes the student and controls the actions taken by the ITS.

The Pedagogical Module decides the problems and sequence that are presented to the student and, at which moment it offers support to the student. This model is usually considered to be domain independent. Typically there are six types of support that an ITS can provide to the students [14] and [10]; (1) Problem solving demonstration, (2) Scaffolding, (3) Monitored from a distance, (4) Goal seeking, (5) Free exploration and (6) No support is provided. The optimal choice of these six types of support services is the topic of a highly debated question in pedagogical research. In [13], Mishra et al cast this problem as an intelligent questioning system. Mishra et al argue that current electrical engineering ITS available are a "little more than a computerized version of home problems found in a typical textbook" and without an appropriate student model the pedagogical module does not function well or not at all. Another way to approach this problem of choice is to consider a domain independent help-seeking model, that aims at detecting inappropriate help requests and a gaming model that detects attempts at gaming the system. This approach is studied in depth by Roll et al in [15].

Notwithstanding the significant progress in the field of ITS, the products developed are still deemed not as effective as a human tutor in the situation of when he/she is leading discussions with students [13].

B. Electrical Theory Classes in Schools

Most college, polytechnic and university electrical circuit theory courses include theoretical as well as practical sessions. The practical sessions are important for two reasons; (a) students learn how to link theoretical models to the real-life circuits, and (b) students learn how to carry out the appropriate measurements using the right instrument. These practical skills are indispensable for professional engineers during the installation, testing and maintenance, of electrical and electronics systems. However instrumentation is generally expensive and its use is restricted to labs. In this respect numerical circuit simulators, such as SPICE, augmented with a graphical schematic capture front and back ends are very useful. With such elearning tools students connect virtual components together using virtual wires, choose and add virtual instruments to the circuit, and finally, carry out a computer analysis. The software outputs the variables chosen or measurements as displayed on the virtual instruments. Such measurements include numerical values, like for example electrical current on a virtual meter, and voltage waveforms on a virtual oscilloscope. This type of eLearning software, widely distributed among colleges and polytechnics helps students in the acquisition of practical skills including the selection of instrumentation. It also speeds up the process and reduces the cost since there is no need for building the circuit in real life. However it does not help the student in understanding how the circuit works or how to design the circuit. On the contrary, it encourages the student not to carry out a manual or mental analysis.

The second author has carried out a study based on a questionnaire regarding the effectiveness of using SPICE simulators as a pedagogical tool and using handouts, which explain step by step the symbolic calculations involved in electric circuit analysis. The questionnaire involved both open ended questions and Likert scale questions. The questionnaire was handed out to the students of the two first year classes of the National Diploma in Electrical and Electronics Engineering (MQF level 4) at Malta College of Arts, Science and Technology (MCAST), Malta and a total of thirty one filled in questionnaires were collected. The full report on this study is published in [16]. The report outlines two conclusions that are relevant to this paper; (a) in general although students find the SPICE simulator as motivating very few agree that it helps in understanding how circuits work, and (b) The larger proportion of students acknowledge that it would be much more useful if the simulator explains how the results are obtained. These results confirm what other researchers [9] and [17], who advocate the use of symbolic and qualitative techniques have stated in their papers, i.e., software that gives explanations, and not just results, is a better aid to students.

Apart from practical skills, electrical engineering students learn how to analyse and design electrical circuits. Traditionally students have been taught how to analyse electrical circuits using pen and paper through the application of the relevant theories, including Ohm's Law, Kirchhoff's Current Law and Kirchhoff's Voltage Law. As explained above SPICE simulators were not specifically designed to help students learn how circuits work, consequently SPICE simulators have some serious limitations when used as a pedagogical tool. The electrical theories taught to students are an essential part of the mental models that the students must develop. Using these theories students can write down symbolic (algebraic) equations that describe how the circuit being analysed behaves. In contrast numerical simulators calculate values iteratively, and this approach limits the understanding and insight that the simulator can impart to its user about how the circuit being analysed functions. In the last couple decades, symbolic simulators have been developed that build the symbolic equations that describe the circuit being analysed and display these equations explicitly. By examining the transfer function equations the student can then infer how the output changes when the parameter values are varied. Examples of recently developed symbolic simulators are SAPWIN [18] and SNAP [19]. On the other hand, these simulators do not explain how the transfer function is obtained.

Additionally, accomplished engineers apply mental models, during what-if analysis activities, to understand how a change in a parameter at a point of the circuit affects the other parameters of the circuit. A change in a parameter, like for example the input voltage, is thought of as influencing the parameters of its neighbouring components and nodes. In turn, these varying parameters affect their own neighbours and hence the changes propagate throughout the circuit. Furthermore, engineers only consider the direction of change, that is, an increase, a decrease or no change at all in the parameter's value. In other words, the quality of the change is considered and not the quantity of the change. This method of analysing a circuit was formally studied by Sussman and Stallman [20] and Johan De Kleer

[21]. In [9], Ahmed et.al. note that experts apply qualitative reasoning prior to calculation, whereas novices calculate first.

In summary, human tutors teach informal methods for what-if qualitative analysis and quasi-formal methods such as node and loop methods that yield quantitative answers. Likewise assessment is carried out as how well a student demonstrates the application of both qualitative analysis as well as quantitative analysis. The most creative teachers make use of real-world problems to motivate the students and include design questions to give a new meaning to circuit analyses. In [17], a Practical Relevance Module (PRM) and a Design Module (DM) are proposed for inclusion in an electrical circuit ITS.

C. Circuit Analysis Techniques

In this section, three circuit analysis paradigms, (the quasi-formal nodal and loop analysis, symbolic analysis and qualitative analysis), pertinent to ITS are discussed.

1) Ohm's Law, Nodal and Loop Analysis: The simplest way to analyse circuits is to identify parallel and series connections such that the original circuit is reduced in complexity to another equivalent circuit that is easier to analyse. However this method fails when one section of a circuit is mutually coupled to another section, in which case a global simultaneous solution is required. The electrical circuit domain model described in [8] and [9] is based on non-coupled serial and parallel combination of impedances and is there limited to such circuits. On the other hand a complex circuit can be described by using either the mesh or the nodal formulation. The mesh equations are based on Kirchhoff's Voltage Law (KVL), which states that the sum of voltage drops along any closed loop is zero, while the nodal equations are based on KCL, which states that the algebraic sum of currents leaving any node is zero. A more general definition of KCL is that in any fundamental cutset that separates the network into two parts, the sum of the currents in the cutset edges is zero. If the number of branches in the network is denoted by the letter b and number of ungrounded nodes is denoted by the letter n, then to solve a circuit; (a) the number of mesh equations required is equal to b - n, and (b) the number of nodal equations required is equal to n. In general nodal analysis yields less equations than mesh (loop) analysis and hence nodal analysis is usually easier to carry out [22].

The nodal and loop methods are widely manually applied in circuit analysis. Automation of these methods however requires formalising it in graph theory, for example as Signal Flow Graphs. The model implemented in this paper focuses on the use of graph theory to analyze the topology of electrical circuits, which is the study of inter-connected objects represented by 'edges' in a graph [22]. The points where the end-points of edges touch together are formally called 'vertices' or 'nodes'. A graph is extracted from the schematic diagram of a circuit by replacing the components with edges. For example the graph shown in fig. 2(b) is extracted from the circuit shown in fig. 2(a). A graph of an electrical circuit contains more than one spanning tree, and from each spanning tree a set of fundamental loops and fundamental cutsets can be extracted. A spanning tree of a graph is defined as any set of connecting branches that connects every node to every other node without forming any closed paths or loops [22]. Fig. 3(a) and fig. 3(b) show two different spanning trees for the graph shown in fig. 2(b). Once a spanning tree has been defined, the edges making part of the spanning tree are referred to as branches. The remaining branches are referred to as links or chords. A fundamental loop is a loop that contains one (and only one) link in its set of edges [22]. Fig. 4(a) shows the fundamental loop for link R1 when considering the Spanning Tree shown in fig. 4(b). To construct a loop that includes only the link R1 (shown as a thick black line) and no other links, the tree branches shown in red must be used. Therefore the fundamental loop of R1 is made up of the edges: R1, R3, and V1. A cut set is a minimal set of edges that when cut, divides the graph into two groups of nodes. A fundamental cutset is a cutset that contains one (and only one) tree branch in its set of edges [22]. Fig. 4(b) shows the fundamental cutset for tree branch V1 when considering the Spanning Tree shown in fig. 3(b). By cutting V1 node 1, shown in green becomes isolated

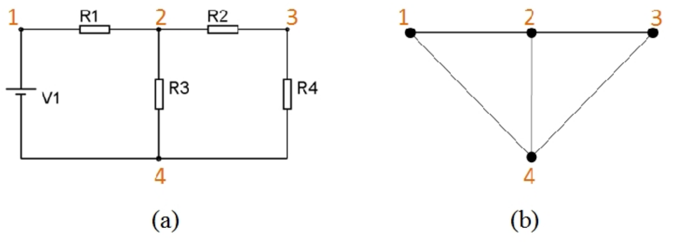


Fig. 2. (a) Example Electrical Circuit, and (b) Graph for Example Circuit.

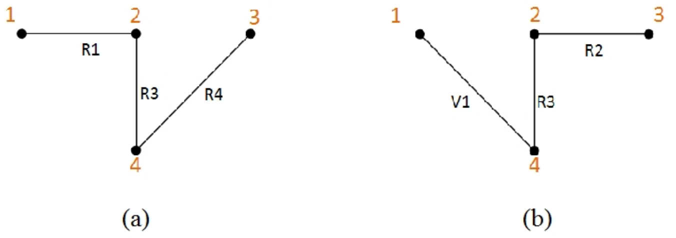


Fig. 3. (a) Spanning Tree I, and (b) Spanning Tree II.

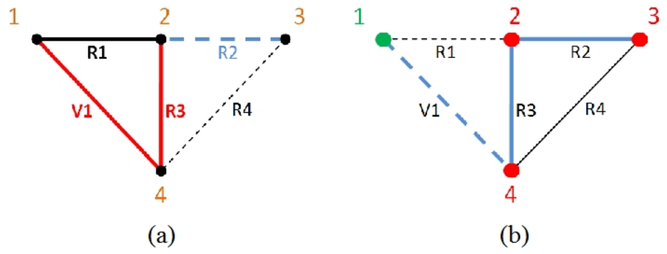


Fig. 4. (a) Fundamental Loop for R1, (b) Fundamental Cutset for V1.

from the group of remaining nodes, that is nodes 2, 3 and 4, which are shown in red. Together with branch V1, the link R1 has to be cut to keep the two groups of nodes separated, hence the complete fundamental cutset is: V1, and R1.

2) Symbolic Simulators: Symbolic simulators are based on formal circuit theory and are able to generate the transfer function of circuits input to them. The transfer function is a commonly used symbolic expression that describes how a circuit behaves. Using the transfer function the output signal that the circuit generates for a given input signal can be calculated. The advantage of using a symbolic transfer function is that the circuit is analysed symbolically only once to obtain the transfer function and then as many numerical answers as needed can be obtained from the transfer function by substituting the symbols with the numerical values being considered. Considerable research has been carried out on the symbolic analysis of electrical circuits in the late 1960's and a number of software Symbolic Simulators were developed in the 1980s [23]. For example De Kleer developed a symbolic simulator called SYN together with Sussman in 1979 [24]. De Kleer states that SYN has several limitations that are were overcome in EQUAL, the Qualitative Analysis Simulator that he developed [21]. These limitations include the lack of the ability to use approximations that drastically simplify the algebra without sacrificing accuracy. Some of these problems have been addressed in modern symbolic simulators [25]. Good examples of modern symbolic simulators that are equipped with a Graphical User Interface (GUI), including a schematic capture front end, are SAPWIN [18] and SNAP [19].

3) Qualitative Electrical Circuits Analysis: De Kleer [21] divides qualitative analysis of electrical circuits into two independent types of analysis, which are; (a) causal analysis, and (b) teleological analysis. The way that the components are connected together in a circuit gives a specific structure to the circuit. The schematic diagram of a circuit describes this structure. Each component in the circuit causes some effects on the other components that are connected to it, and these in turn affect the components that are connected to them, and so on. The aim of causal analysis is to combine the behaviour of the individual components to explain the behaviour of the overall composite system. That said, a composite system is built so that it serves a purpose. The purpose of a circuit is also referred to as the function of the circuit. Teleological analysis describes how by knowing the behaviour of a circuit one can deduce its function. Causal analysis relates structure to behaviour and teleological analysis relates behaviour to function. These two types of analysis were also investigated by Marc Fossprez in 1988 [6]. Marc Fossprez states that it is relatively easy to deduce how a circuit behaves once its function is known, but it is much harder to deduce how a circuit behaves if only the circuit's structure (its schematic diagram) is given. His work focuses on this latter task and he gives definitions about the different structures that circuits can possibly have and mathematical proofs that employ topology and graph theory.

The "Propagation of Constraints" technique, developed by Sussman and Stallman [20], is the first attempt at formalizing qualitative analysis. This method is inspired by the what-if qualitative and informal procedure applied by experts in electrical engineering and described in section II-B. The algorithm developed by Sussman and Stallman calculated the numerical values of voltages and currents and therefore goes beyond qualitative analysis. In 1984 Johan De Kleer implemented the Qualitative Analysis of electrical circuits in a program he called EQUAL [21]. This program is able to explain how a circuit works using qualitative arguments and even categorize the circuit as being a power-supply, logic-gate, amplifier or multivibrator. Furthermore, the propagation of Constraints has proved to be a powerful algorithm in circuit analysis. Fossprez recommends its use when searching for a pair of compatible current and voltage (i, v) orientations, while analysing circuits qualitatively [26]. In 2006, Rehman et al developed a type of software authoring tool for an 'Intelligent Book' [27], in which this algorithm is used to find the currents, voltages and component values inside different circuits. The values generated by the Propagation of Constraint algorithm are used to verify the values input by the students that make use of the 'Intelligent Book'.

D. Student Models For Assessment

Alongside the domain expert model, the student model is indispensable in an ITS since this stores information about the student's knowledge state, progress and learning behaviour. The two main characteristics of an ITS system is adaptivity to the student and assessment. In the case of being adaptive it is necessary to build a model for each student that is updated as the student progresses through the learning process. In the case of assessment it may not be necessary to store a model for each student. Instead models that the define certain levels of attainment, such as distinction, merit and pass, to which the student's profile is compared will suffice. In an assessment system the goal is to determine what the student knows or the knowledge state. Knowledge is often described as being either declarative, procedural, or a mixture of both. Ideally the assessment system models all three types. Additionally other variables that relate to the student's human attributes and aptitudes, such as cognitive, conative, meta-cognitive, motivational and affective can be added to the student model to deliver systems that consider hidden skills and states of mind [28] and [29]. Finally, the terms knowledge tracing and model tracing are often used to distinguish between assessment and adaptivity. Knowledge tracing refers to what the student knows, whereas model tracing refers to the steps taken by a student when solving a problem. Model tracing implies the sequence of selecting the right or wrong items versus time, that lead to a solution or an impasse. It is our view that model tracing can be split into two types, those that are directly related to the domain and therefore can be linked to the knowledge model and those that related to generic items or skills, such as self-regulation. In summary, student models are compared and contrasted on the basis of how well they can model knowledge that is either declarative, procedural, or a mixture of both.

The pre-cursor of ITSs are Computer Adaptive Testing (CAT) systems that attempt to adjust test questions to suite the ability of the examinee. The principle behind CAT is to build and update a student model that reflects the knowledge state versus time. The choice of the next item or problem to solve is therefore based on the model output. Item Response Theory (IRT) was developed for this purpose [30],[31] and was the prevalent method of choice until the Bayesian modeling approach was extensively studied and adapted to modeling the student's knowledge state [32], [33], [34] and [35]. More recently Hidden Markov Models (HMMs) have been proposed to study the metacognitive behaviour of students during the learning process [36] and [37].

Bayesian Networks and Markovian Models are graph models that consist of nodes and directed edges. Bayesian networks are acyclic, while Markovian models allow cycles. Markovian models therefore have the property of representing knowledge in a more compact form at the expense of granularity and inference. Bayesian networks can be more complex and computationally intensive, however they have been shown to perform well in knowledge tracing and model tracing [28] whereas Hidden Markov Models have been limited to model tracing [36]. The graphical networks are characterized by two types of variables or nodes; target variables and evidence variables. Target variables are for example the knowledge state (both procedural and declarative) and cognitive skills. Evidence variables are attributes that can be measured such as answers to questions, selection of items, assertions and sequence of events including timing information. Granularity has an effect on computational time required and may have an effect on the accuracy of the judgment made on a student. Therefore, models have to be compared on this attribute. Depending on the target variables the ITS designer has to decide on what to model with a network. For example in [6], separate Bayesian networks are built for each student and for each problem presented to a student. The total number of networks that a system handles can therefore be high. Finally, in most ITS Bayesian network implementations, such as in [6] and [33] the networks are knowledge engineered and very few are constructed from empirical data. One such case is described in [35].

The work reported in [36] is motivated by an emphasis on preparation for future learning. Therefore, the goal is not to assess the knowledge state but to study strategies and behaviours that students engage in during the learning process. The proposed solution then consisted of deriving a HMM from the measured and tagged students' activity sequences. The model identifies sequence patterns to a student behaviour type or hidden state. In other words one model is required for each behavioural trait.

In this paper a Markovian Model or finite-state machine and a Histogram-based model are developed and studied in terms of how well they perform in knowledge and model tracing.

III. Implementation of Models and Framework

Two separate systems were developed. The first tool is text based and input is provided via a command-line interface. The user interface for the second tool provides a Graphical User Interface for part of the input and output, which makes it more adequate to be used as an eTutor. For both tools circuits are input via a text file. The student assessment model is integrated with the first tool only and does not provide feedback during the problem solving process. The second tool uses the expert model directly to provide feedback after each event.

A. Qualitative Analysis based Expert Model

This module is intended for entry level students and deals directly with the analysis of simple circuits made up a voltage source and a set of resistors. The software tool requires the user to input the circuit to be analysed as a text file. As such the circuit has to be specified in a matrix, in which the rows represent nodes and columns represent components. The first row of this matrix is reserved for the components' values, that is the voltage of the battery and the resistance of each resistor. In the cells of the other rows a '1' means that terminal one of the corresponding component is connected to the node corresponding to the cell's row, '2' means that terminal two of the corresponding component is connected to the node corresponding to the cell's row and '0' means that the corresponding component is not connected to the particular node considered.

The first software tool accepts input circuits that are made up of an arbitrary number of resistors and only one battery. This program then analyses the connections of all the resistors and identifies resistors that are connected in parallel. Each group of resistors connected in parallel is replaced by one equivalent resistor. The program then identifies serially connected resistors and replaces each group by one equivalent resistor. This process is depicted in fig. 5. At each step the program outputs a matrix in text format, which specifies the connections in the resultant simplified circuit. If the resultant circuit contains other groups of resistors that are connected in parallel or in series further reductions are done. This process is repeated until no further reductions are possible. The process is traced backwards and the required voltages and currents across and through the various resistors are calculated. The program is used in this mode when a demonstration or a full solution to the problem is required. In this mode, the student can then manually compare the eTutor's solution to his/her own and discover where s/he erred.

The program can be used in an interactive mode, where the student solves the problem on the eTutor console, or at least records his/her steps and calculations on the eTutor. The learner enters statements via a command-line interface, for example *Parallel(1,R1,R2,5)*, which means that in circuit 1, R1 is in parallel with R2 and the equivalent resistance is 5 ohms. The Domain Model checks that the statement is correct or incorrect. The assertion is recorded in a log-file and tagged as correct or not correct, following output from the expert model. Changes in the circuit are entered via a labeled text

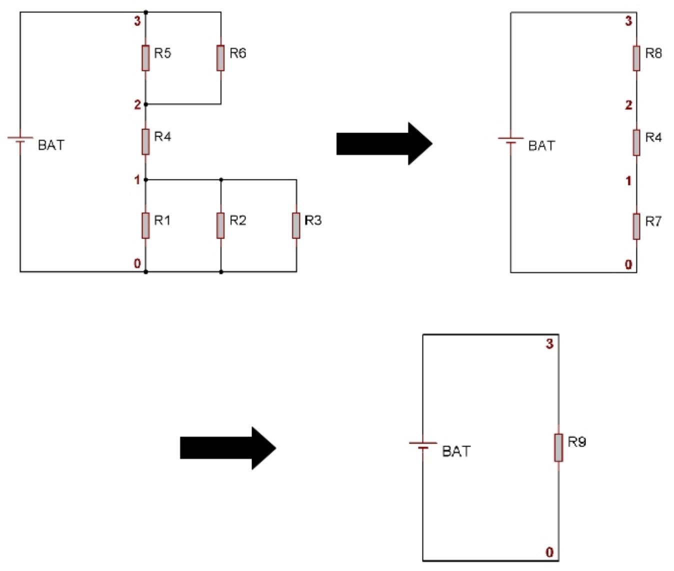


Fig. 5. Example of the parallel and series resistors reduction processes carried out by the first program.

file, whose format was described above, and the circuit is linked to the previous assertion that motivated the change in the circuit. If the same assertion is repeated, i.e. the learner back-tracks and repeats statements, the previous copies are deleted. Although deleting repeated assertions results in a loss of behavioural traces, it relaxes the task of the assessment module, which is required to output a summative value for the assessment. It also reduces the possibility of a student gaming the system. Furthermore the number of commands and inputs possible are grouped into six events. The granularity is thus reduced with some possible loss of precision. The latter step is a pre-processing step to relax the requirements from the data mining technique. It also has a positive effect of minimizing overfitting. The six events are; (a) The correct assertion of a series or parallel combination and the calculation of the equivalent resistance, (b) the incorrect assertion of the latter, (c) The correct modification of a circuit, (d) the incorrect modification of a circuit, (e) the correct assertion of a voltage or a current, and (f) the incorrect assertion of the latter. These six events labelled as (SP, $\neg SP$, C, $\neg C$, VI, $\neg VI$) are used by the models described in the next section, III-B. In summary, for the purpose of developing the assessment models described in section III-B, the sequence of events are saved to a log-file. The relationship in between events is not however preserved, although these are implicity coded in the sequence. On the other hand all the activity, including the commands and circuit scripts are saved for the purpose of manually assessing and marking the solutions. The test group for this software tool included 27 students and 3 members of the academic staff. In general the students and the academic staff think that the tool is very useful and should be expanded to include further expert knowledge. The text based user interface is a bit daunting, especially when modifying the circuit. The teachers think that the restrictions imposed by the program during the circuit analysis process can give some false expectations to students. The assessment module is described in the next section.

B. The Assessment Models

The assessment module is required to assess the performance of a student after attempting a solution to a given problem. Two different types of models are considered. One model is a Markovian Model that outputs a probability distribution over four levels of knowledge attainment and the other model is a histogram-based model that outputs a measure of difference between the ideal student model and the given case input. The six variables used in these models are the events described in the previous section, III-A and the sequence of events pertinent to a given student is obtained from the log-file described in the previous section, III-A. The following sections describe the implementation of these models.

1) The Markovian Model: Fig. 6 depicts the architecture for the model. The model consists of eight states or nodes,

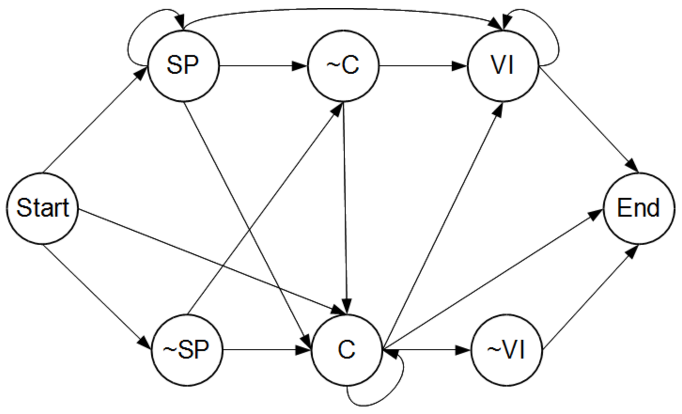


Fig. 6. The Markovian Model. Not all possible transitions are shown.

the six events described in section III-A and two new states (START and END). The value of the directed edges represent the probability of the student moving to a new state. A fully connected graph will have 64 edges in total, although the START and END states are typically only visited once. Four such models are required to assess a student in a discrete probability distribution over four levels of mastery. In this paper these four levels are labelled as 1, 2, 3, and 4, 1 being mastery of electrical circuit theory and 4 being a failure in learning electrical circuit theory. 3 and 4 are intermediate levels. The four models are all based on the same architecture in fig. 6 but the transition probabilities are different. Each case therefore models typical sequences that a student from a particular attainment level typically follows. In other words the model classifies patterns of sequential events. On the other hand atomic knowledge is assessed by the domain expert knowledge. Therefore intuitively the expert model and the Markovian Model will together achieve both knowledge and model tracing.

2) Histogram Model: The histogram-based model computes the difference between the ideal student model histogram and the case to be assessed. In the test case available there was only one ideal student model. This was due to both the nature of the problem and the fact that the human-interface module inherently restricted the freedom of the student. Two different histograms are considered. One histogram is a frequency count of the six events (SP, $\neg SP$, C, $\neg C$, VI, $\neg VI$) that a student engages in. This will be called the state vector histogram. The difference between a given case and the ideal student model is computed as,

$$\sum_{i} \sqrt{(S_i^{model} - S_i^{test})^2} \tag{1}$$

where $\mathbf{S}^{model}$ is the state vector histogram for the ideal student model and $\mathbf{S}^{test}$ is the state vector histogram for the given test case.

The second histogram is a frequency count of the 64 state transitions possible. This is termed the transition matrix histogram. The difference between a test case and the ideal student model is computed as,

$$\sum_{i,j} \sqrt{(T_{ij}^{model} - T_{ij}^{test})^2} \tag{2}$$

where $\mathbf{T}^{model}$ is the transition matrix histogram for the ideal student model and $\mathbf{T}^{test}$ is the transition matrix histogram for the given test case.

Since the data set includes cases that have been marked by the highest and lowest marks possible and the output for the ideal answer is zero then the results are scaled to reflect the zero to ten marks range and it will then be possible to compare the results from the histogram model to the human generated assessment. In doing this we are assuming a linear mapping.

C. Nodal Analysis Based Expert and Tutor model

The aim of this system is to eTutor students that are learning how to identify a fundamental tree and the corresponding fundamental cutsets in a given circuit and how to generate the KCL current equations for each fundamental cutset. This topic is covered in a unit called 'Further Electrical Principles' that higher national diploma (MQF level 5) students follow in the second year of their course at the MCAST.

The format of the input text file for the second program is more compact and easier to write since in it a text line is dedicated to each component and the numbers of the two nodes, to which the component is connected are stated in the corresponding line. This does away with the '0s' that were used for the first program. The other information included in each line of this text file is the X and Y coordinates of where the component is to be drawn in the GUI, the name of the component and its value.

Once a circuit is specified correctly in the input text file it can be loaded in the program. Fig 7. shows an example of a loaded circuit. The user is asked to chose a spanning tree, by clicking on the components in the circuit. Once the user selects a group of components that s/he think makes up a valid spanning tree, s/he must press the 'Check Spanning Tree' button so that the program verifies if the selected group of components makes up a valid spanning tree. If it does not the program informs the user and gives relevant feedback to the user of why the selection does not make up a valid spanning tree. The program informs the user whether s/he selected the right amount of components and whether s/he captured all the nodes in the circuit with the group of components selected. The program also informs the user if there are loops present in the selection made.

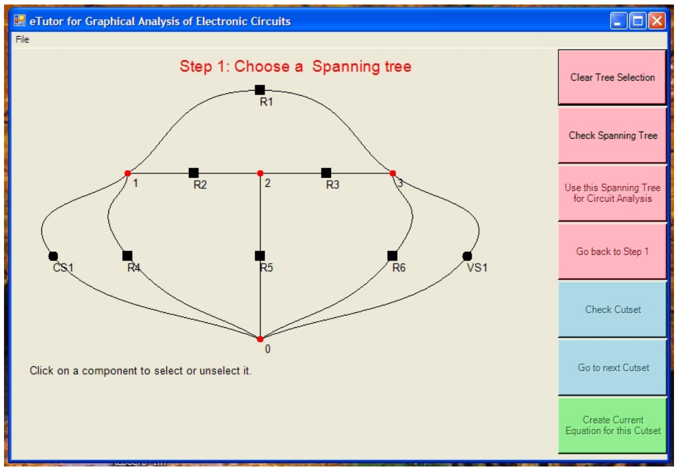
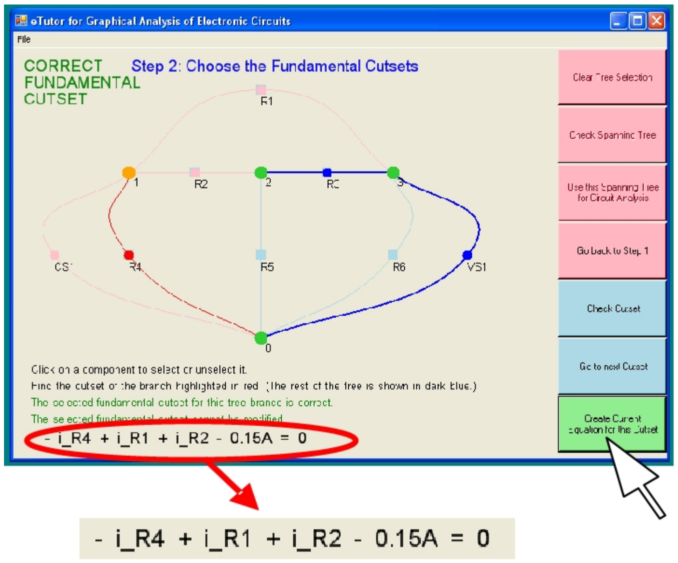

eTutor for Graphical Analysis of Electronic Circuits
File
Step 1: Choose a Spanning tree
R1
1
R2
2
R3
3
CS1
R4
R5
R6
VS1
0
Click on a component to select or unselect it.
Clear Tree Selection
Check Spanning Tree
Use this Spanning Tree for Circuit Analysis
Go back to Step 1
Check Cutset
Go to next Cutset
Create Current Equation for this Cutset

Fig. 7. Example of a loaded circuit in the program's GUI.

On the other hand, if the selection makes up a valid tree the program informs the user and allows the user to select this spanning tree to continue with the circuit analysis. To do this the user has to press the 'Use this Spanning Tree for Circuit Analysis' button. Once this button is pressed the program goes into Step 2, in which the user has to select the correct fundamental cutset for each of the tree branches inside the selected spanning tree. The tree branch, for which the user has to select the links that make up the fundamental cutset, is highlighted in red, as shown in fig. 8.

The fundamental cutset must separate one of the group of nodes from the remaining group of nodes. To help the user the program highlights all the nodes in one of these groups in orange and the nodes in the other group in green. After that the user selects the components that s/he thinks make up the fundamental cutset, s/he must press the 'Check Fundamental Cutset' button. Once this button is pressed the program checks if the selected components make up a valid fundamental cutset. If this is not the case the program gives relevant feedback to the user. The program states whether one or more components that should be included in the selection are not selected and it also states if one or more components that should not be included in the selection are in fact selected. In the case when the selected components make up a valid fundamental cutset, then the user is informed accordingly and is allowed to press the button labelled "Create Current Equation for the Cutset". When this button is pressed the KCL equation for the


eTutor for Graphical Analysis of Electronic Circuits
File
CORRECT FUNDAMENTAL CUTSET
Step 2: Choose the Fundamental Cutsets
R1
1
R2
2
R3
3
CS1
R4
R5
R6
VS1
0
Clear Tree Selection
Check Spanning Tree
Use this Spanning Tree for Circuit Analysis
Go back to Step 1
Check Cutset
Go to next Cutset
Create Current Equation for this Cutset

Click on a component to select or unselect it.
Find the cutset of the branch highlighted in red. (The rest of the tree is shown in dark blue.)
The selected fundamental cutset for this tree branch is correct.
The selected fundamental cutset cannot be modified

- i_R4 + i_R1 + i_R2 - 0.15A = 0

- i_R4 + i_R1 + i_R2 - 0.15A = 0

Fig. 8. Example of a fundamental cutset KCL equation generated when the correct fundamental cutset is selected and the appropriate button is pressed.

fundamental cutset is generated by the program and displayed at the bottom of the screen as shown in fig. 8. The user can then press the ’Go to next Cutset’ button to find the fundamental Cutset of the next branch in the spanning tree. This process has to be repeated until the fundamental cutsets of all the branches in the spanning tree are found. At this point it is desirable that the program tutors the user on how to find the fundamental loop for each link present in the graph, but this feature has not been implemented yet. The author plans to have this feature functional in the future so that it can be used by the higher national diploma students.

In the system developed, the nodes are implemented in a list data structure. The branches or components at a given node are defined in another list. The algorithms then operate on these lists. From graph theory it is known that a valid spanning tree must be made up of n-1 edges, where n is the number of nodes. Hence the first check made to verify the input tentative spanning tree is to count the number of selected components and check if it equal to n-1. If this is not the case it means that the selected components do not make up a valid spanning tree.

The next step to carry out is to check that the selected components capture all the nodes inside the circuit (graph). The algorithm just has to go through all the selected components and mark the two nodes, to which each component is connected as captured. After that the algorithm has to go through the nodes and check that none of them is non-captured. If one or more nodes are non-captured then the selected components do not make up a valid spanning tree. There exist cases, in which the two checks explained above are satisfied but the selected components still do not make up a valid spanning tree. In this case the selected branches will not be continuously connected and at least one loop will be present in the selection. To check for such cases the spanning tree algorithm starts off with one of the selected tree branches. It checks, to which nodes this branch is connected and proceeds to discover the other branches that one of these nodes is connected to. If there are more than one branch connected to this node the algorithm starts considering the first branch and it takes note, of which branch this is so that once it finishes checking it and returns to the last node considered, it continues looking for the correct branch. This process is repeated for each node. When at least one branch is found connected to a node the algorithm jumps to the other node, to which this branch is connected and hence travels further away from the first node that it considered at the start. Naturally the larger the selected tentative spanning tree is, the more searching the algorithm has to do. But in the case of invalid spanning tree selections there are two possible ways, in which the algorithm completes. One way is that the algorithm steps forward (not backwards) into a node that it already checked, and hence a loop is discovered. The other way, in which the algorithm can complete in the case of an invalid spanning tree selection, is that it finds out that it exhausted all the branches and nodes that are connected to the first branch considered, but it did not find all the nodes present inside the graph. In this case it means that the algorithm has found one continuous length of connected branches, which is not connected to the remaining branches of the selected tentative spanning trees. Since spanning trees should not contain any discontinuities in their branches’ connection, this means that the selected components do not make up a valid spanning tree.

Another algorithm used in the graphical analysis program is the one that highlights in different colours the two groups of nodes that are to be separated by a fundamental cutset. The searching that this algorithm does is very similar to that done by the algorithm that verifies spanning trees. However in this case, the fundamental cutsets algorithm does not check for loops because it is used after that a valid spanning tree is already selected, so it is already guaranteed that no loops are present. The important feature that this algorithm possesses, similarly to the previous algorithm, is that it always remembers, which branch it checked last when jumping from one node to another, so that when it returns back to the node from where it jumped, it continues checking from the correct branch.

IV. Results

This section describes results pertaining to (a) the Markovian Assessment Model, (b) the Histogram Assessment Model, and (c) the Problem Solving Environment.

A. *The Markovian Assessment Model*

In this paper, the Markovian models are tuned using empirical data. For this purpose twenty-seven students are given a problem to solve and their solution is recorded as described in section III-A. A human tutor then assesses the twenty-seven solutions and marks them over a 21 point scale from 0 to 10. The data set is clustered on these marks as; marks equal or above 7.5 corresponding to level 1, marks in between 5.0

and 7.4 as level 2, 2.5 to 4.9 as level 3 and marks less than 2.5 as level 4. The twenty-seven cases where approximately uniformly distributed across the four levels. Finally these groups are used to find the transition probabilities for each of the four models. Assessment consists of first computing the log likelihood probability distribution for a given new student and the case is classified by choosing the maximum log likelihood. The Markovian Model was first tested using samples from the data set. Fig. 9 shows the classification results for all twenty-seven cases, out of which two were incorrectly classified. Fig. 10 depicts the probability distributions for four correct cases and the two incorrectly classified cases. To further test the model, the twenty-seven test data set was split into two data sets, one being the tuning set and the other being the test set. The ratio of the tuning set size to the test set size was varied. When the training set size is 27 the correct classification rate is 93%, for a training set size of 23 the correct classification rate is 89%, for a training set size of 19 the correct classification rate is 85%, for a training set size of 15 the correct classification rate of 85% and for a training set size of 8 the correct classification rate is 70%.

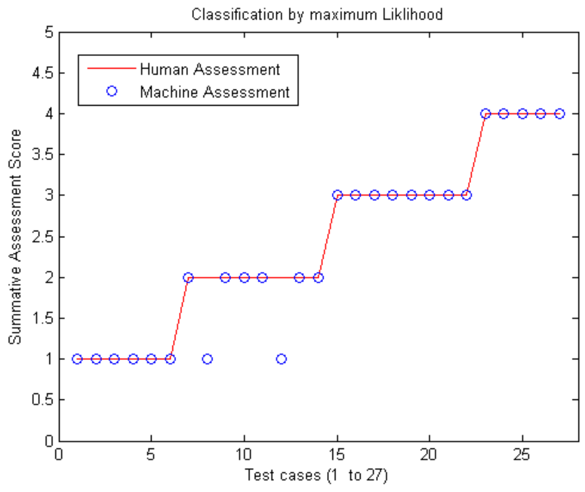


Fig. 9. Classification of results from the Markovian model by maximum likelihood.

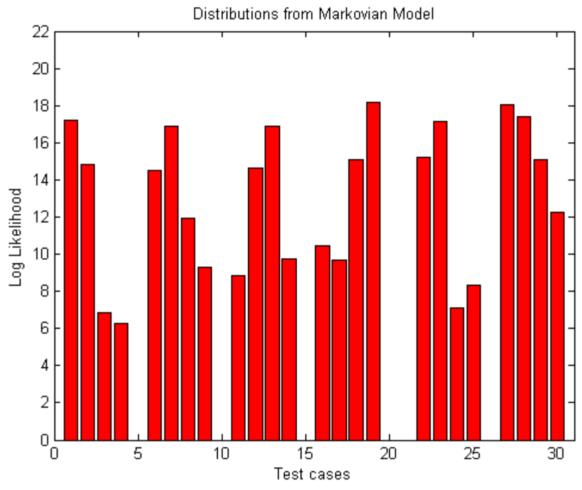


Fig. 10. Probability distributions obtained from Markovian model. The first four groups are correctly classified cases for each assessment score level. The last two groups are the two out of twenty-seven incorrectly classified cases.

is accurate only for cases close to the ideal student model, fig. 12.

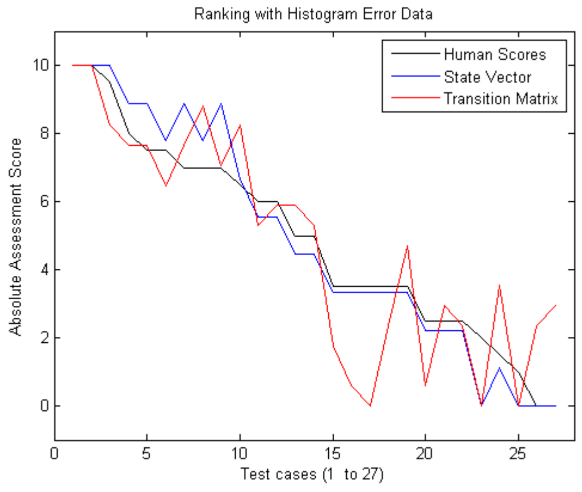


Fig. 11. Absolute assessment scores obtained from the state vector histogram and transition matrix histogram.

B. The Histogram Assessment Model

Fig. 11 shows how the two histogram models compare with the human generated assessment. A non-monotonically decreasing graph indicates deviations from the human assessor. The state vector model is characterized by a closer match than the transition matrix histogram model. This is expected since the former tests for the correct event selection and its frequency of selection, whereas the latter tests for the transitions. These results omit the states that describe an incorrectly executed event. This makes a fair comparison since we know that the human assessor did not negatively mark the solutions. When these events are included in the histogram model the deviations and oscillations increase and the model

The real number outputs from the histogram models are used to classify the students into one of the four levels of attainment. Fig. 13 shows that the classification results are not very good. The state vector model classified eight instances in the wrong class, while the transition matrix model classified twelve instances in the wrong class. Fig. 14 shows the classification results for the model that included negative states. In the case of the state vector model ten instances are not correctly, whereas for the transition matrix model eleven instances are in error. In summary, the histogram models can only be used to classify students in two states, mastery or non-mastery. On the other hand, the Markovian Model grades the

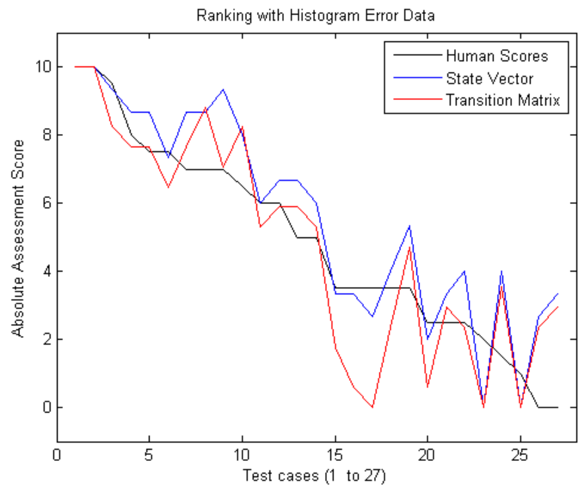


Fig. 12. Absolute assessment scores obtained from the state vector histogram and transition matrix histogram. These results include the negative states.

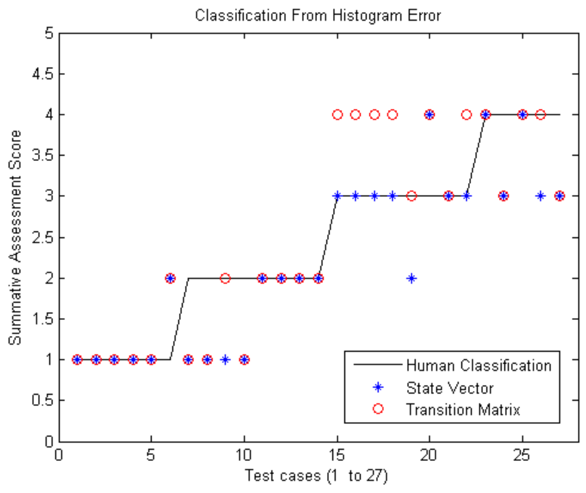


Fig. 14. Linear classification of test data using the state vector histogram and transition matrix histogram. These results include the negative states.

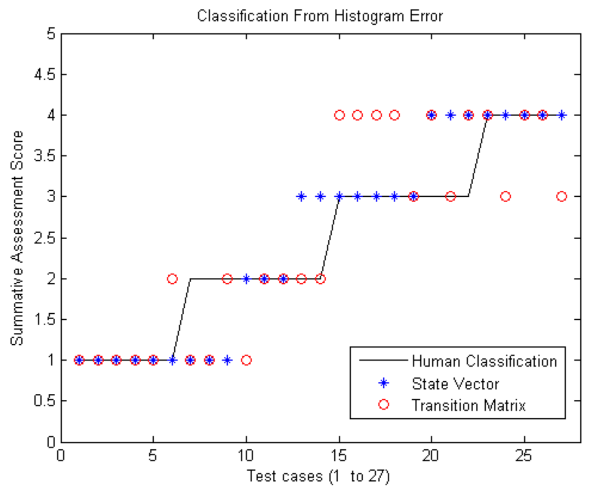


Fig. 13. Linear classification of test data using the state vector histogram and transition matrix histogram.

student on a scale of four points.

C. Problem Solving Environment

The Nodal Analysis Based Tutor System is aimed at second year higher national diploma (MQF level 5) students. Its aim is to tutor these students on how to find correct spanning trees and fundamental cutsets in graphs of electrical circuits. This program was tested and verified to function correctly. It interacts with its users through a GUI. It lets the user input the circuit of interest and select a valid spanning tree. When a valid spanning tree is selected the program lets the user work out all the valid fundamental cutsets corresponding to this spanning tree and then generates the corresponding KCL equations. Whenever the user does an incorrect choice during the selection process, the tool explains why the choice is incorrect, and hence acts like a Tutor. The tool was first tested by fifteen students that undertook courses that included the topic under consideration in the previous academic year and all these students stated that this tool would have been of great help to them. The program was then tested with a class of 18 novel HND students. These students were able to choose their own personal set of branches that make up valid spanning trees and fundamental cutsets in class. This reduced the amount of time that the students needed to learn and understand these two concepts, as well as the success rate among students.

V. Discussion of Results

The field studies demonstrated that students are keen to experience software tools that help them learn how to analyze electrical circuits in the same way a human tutor would do. Two systems that give explanations of the analysis carried out on circuits were developed. The expert models in both systems are based on symbolic and qualitative analysis, and feedback is provided on declarative and procedural knowledge. These systems therefore simulate the full-time availability of a tutor and immediate assessment results given to students increased their motivation to discover and learn. These characteristics can have a significant impact on attainment levels for a large number of students. Besides being used by one of the authors, this program was demonstrated to two lecturers that teach circuit theory and both are of the opinion that this program will help them deliver the concerned topic more efficiently, leading to higher success rates among students.

The text-base system, whose expert model is based on Ohm's Law, is targeted towards first year national diploma (MQF level 4) students. The expert model identifies resistors that are connected in parallel and in series and replaces them by equivalent resistors. Alternatively the process of analysis is carried out by the student and the system responds with immediate feedback on every assertion. Students reported, that the expert explanation provided by the system is very useful. However, the text-based interactive environment is not straightforward and the students had to adapt to it. The main problem with this environment relates to interpretation errors specifically when the student needs to translate text into a circuit diagram, either mentally or on a side paper note-pad. Augmenting the system with a graphical view of the circuit in question would therefore mitigate this problem. Other than the latter shortcoming, both the students and teachers liked the tool and think that it is a useful tutor.

The user interface for the Nodal Analysis based expert model is based on a graphical layout. So it is not surprising that students and teachers found it easier to use. The students and teachers noted that sometimes the system either gives too much feedback or the feedback is too wide and not selective. This is understandable since no personalized student model was integrated in this system. It also shows, how important it is for an ITS to keep the student motivated and interested. Nonetheless field tests showed a reduction in the amount of time that the students needed to learn and understand the concepts in electrical circuit theory classes, as well as an increase in the success rate among students, when subjectively compared to previous groups that did not use the system.

An important aspect of the contribution in this paper is the Markovian assessment model that traced both declarative and procedural knowledge components in the student solution. The main drawback of the Markovian model is the fact that a training set is required for every problem set by the human tutor. It may be possible to generate the ideal student model from the solution generated by the domain model. Perturbations from the ideal model can then generate inferior answers to the problem and the models are tuned or fitted with simulated data. It may also be possible to use past answers and sample human generated assessments to generate synthetic answers and assessments. This solves the problem of requiring the human tutor to correct a sample class to tune the model with.

The student assessment models reported in this paper were deployed to provide feedback any time the student engages in a learning activity. This means that these systems can be used to assess students more often, providing valuable data to help teachers allocate resources more effectively and also helps in tackling challenges in mixed-ability classes. Additionally the Markovian Model is trained using human assessment data and this means that the model can simulate specific characteristics of teachers. This leads to a personalised teacher machine assistant, which learns how to assess from the human teacher.

From this experience, we see three areas that are worth improving. It would be ideal to have machine tunable Markovian assessment models. The two domain expert models should be merged into one, yielding a model that can cover most of the electrical circuit theory dealing with linear components. A student model for each and every student, possibly a probabilistic one, should be included in order for the system to provide personalized and more appropriate feedback.

VI. Conclusions

A Markovian Assessment Model and a Nodal Analysis electrical circuits expert model for circuits of arbitrary topologies have been developed and tested using lab and field tests. These two contributions are a significant improvement over the respective models described in [8] and [9].

The Markovian Model traces declarative and procedural knowledge in solutions to problems in electrical circuits. A simpler histogram model that traces only declarative knowledge is developed and compared to the Markovian Model. The Histogram Model is useful to test whether the student has mastered the topic in question and is similar to models installed in current electrical circuits ITSs [13]. The Markovian model can be possibly improved, in terms of providing finer granularity in assessment, by feeding a regressor with the four element vector that is ouput from the Markov Model.

The circuit expert model based on the formal theory of nodal analysis and on qualitative assertions is suitable for analyzing a circuit of arbitrary topology and for providing a detailed account of the analysis process. The nodal analysis based model is an improvement over the simpler symbolic and qualitative circuit model based on Ohm's law implemented in [8] and [9]. From an electrical circuit theory ITS system point of view it would be ideal to combine the features of both models into one. Other ITS or CAL systems, described in [27] and [17], make use of ”the propagation of constraints” algorithm to calculate circuit parameter values. However this method does not yield an explanation as one would expect from a human tutor. So, alternatively further research may develop the ”propagation of constraints” model to be better suitable for an ITS.

Finally, field tests confirmed two important points. During learning, domain models based on qualitative and symbolic analysis are more effective than simulators based on numerical analysis, which may be better suited for expert use in industry. Students prefer a problem solving environment that comprises both text and graphics-based input/output systems.

References

[1] J. Debono and A. Muscat, “An electrical circuits e-tutor based on symbolic and qualitative analysis,” in *The Fifth International Conference on Advanced Engineering Computing and Applications in Sciences, ADVCOMP2011*, November 2011.

[2] B. S. Bloom, “The 2 sigma problem: The search for methods of group instruction as effective as one-to- one tutoring.,” *Educational Researcher*, vol. 13, pp. 4–16, 1984.

[3] P. A. Cohen, J. A. Kulik, and C. C. Kulik, “Educational outcomes of tutoring: A metaanalysis of findings.,” *American Educational Research Journal*, vol. 19, pp. 237–248, 1982.

[4] J. R. Anderson, A. T. Corbett, K. R. Koedinger, and R. Pelletier, “Cognitive tutors: Lessons learned,” *The Journal of the Learning Sciences*, vol. 4, pp. 167–207, 1995.

[5] A. Lesgold, G. Eggan, S. Katz, and G. Rao, “Possibilities for assessment using computer-based apprenticeship environments,” in *Cognitive Approaches to Automated Instruction* (J. Regian and V. Shute, eds.), (Hilisdale, NJ), Lawrence Eribaum Associates, 1992.

[6] C. Conati, A. S. Gertner, and K. VanLehn, “Using bayesian networks to manage uncertainty in student modeling,” *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 371–417, 2002.

[7] S. Suebnukarn and P. Haddawy, “Modeling individual and collaborative problem-solving in medical problem-based learning,” *User Modeling and User-Adapted Interaction*, vol. 16, no. 3-4, pp. 211–248, 2002.

[8] A. Yoshikawa, M. Shintani, and Y. Ohba, “Intelligent tutoring system for electric circuit exercising,” *Education, IEEE Transactions on*, vol. 35, no. 3, pp. 222–225, 1992.

[9] M. Ahmed and M. Bayoumi, “An artificial intelligent instructor for electrical circuits,” in *Circuits and Systems, 1994., Proceedings of the 37th Midwest Symposium on*, vol. 2, pp. 1362 –1365 vol.2, aug 1994.

[10] A. T. Corbett, K. R. Koedinger, and J. R. Anderson, “Intelligent tutoring systems,” in *Handbook of Human-Computer Interaction* (M. G. Helander, T. K. Landauer, and P. Prabhu, eds.), vol. 37 of *2*, (Amsterdam, The Netherlands), Elsevier Science, 1997.

[11] H. S. Hwana, “Intelligent tutoring systems: An overiew,” *Artificial Intelligence Review*, vol. 4, pp. 251–277, 1990.

[12] R. R. V. D. Stuyf, “Scaffolding as a teaching strategy,” *Adolescent Learning and Development*, November 17 2002.

[13] M. Mishra, V. Mishra, and H. Sharma, “Intellectual ability planning for intelligent tutoring system in computer science engineering education,” in *Emerging Trends and Applications in Computer Science (NCETACS), 2012 3rd National Conference on*, pp. 26–30, IEEE, 2012.

[14] D. M. Towne and A. Munro, “Supporting di verse instructional strategies in a simulationoriented training environment,” in *Cognitive Approaches to Automated Instruction* (J. Regian and V. Shute, eds.), (Hilisdale, NJ), Lawrence Eribaum Associates, 1992.

[15] I. Roll, R. Baker, V. Aleven, B. McLaren, and K. Koedinger, “Modeling students’ metacognitive errors in two intelligent tutoring systems,” *user modeling 2005*, pp. 151–151, 2005.

[16] J. Debono, “Effectiveness of using circuit analysis software in vocational electronics engineering courses,” tech. rep., Malta College of Arts, Science and Technology (MCAST), Corradino, Malta, September 2010.

[17] R. Amarin, K. Sundaram, A. Weeks, and I. Batarseh, “Importance of practical relevance and design modules in electrical circuits education,” in *Global Engineering Education Conference (EDUCON), 2011 IEEE*, pp. 792–796, IEEE, 2011.

[18] A. Luchetta, S. Manetti, and A. Reatti, “Sapwin - a symbolic simulator as a support in electrical engineering education,” *IEEE Transactions on Education*, vol. 44, p. 9, May 2001.

[19] D. Biolek, “Snap - program with symbolic core for educational purposes,” *Proceedings of 4th World Multi-Conference on: Circuits, Systems, Communications and Computers*, pp. 1711–1714, July 2000.

[20] G. Sussman and R. Stallman, “Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis,” *Artificial Intelligence*, vol. 9, pp. 135–196, October 1977.

[21] J. de Kleer, “How circuits work,” *Artificial Intelligence - Special volume on qualitative reasoning about physical systems*, vol. 24, pp. 205–280, December 1984.

[22] J. W. Nilsson and S. A. Riedel, *Electric Circuits*. Addison Wesley, 5th ed., 1996.

[23] G. Gielen, P. Wambacq, and W. Sansen, “Symbolic analysis methods and applications for analog circuits: A tutorial overview,” *Proceedings of the IEEE*, vol. 82, pp. 287–304, 1994.

[24] J. de Kleer and G. Sussman, “Propagation of constraints applied to circuit synthesis,” *International Journal of Circuit Theory and Applications*, vol. 8, pp. 127–144, 1980.

[25] H. Floberg, *Symbolic Analysis in Analog Integrated Circuit Design*. Kluwer Academic Publishers, 1997.

[26] M. Fossprez, *Qualitative Analysis of Non-linear, Non-reciprocal Circuits*. John Wiley and Sons, 1992.

[27] K. Rehman, W. Billingsley, and P. Robinson, “Writing questions for an intelligent book using external ai,” *Proceedings of the Sixth IEEE International Conference on Advanced Learning Technologies*, pp. 1089 – 1091, 2006.

[28] E. Milln, T. Loboda, and J. L. P. de-la Cruz, “Bayesian networks for student model engineering,” *Computers and Education*, vol. 55, no. 4, pp. 1663 – 1683, 2010.

[29] R. S. Baker, “Mining data for student models,” in *Advances in Intelligent Tutoring Systems* (R. Nkmabou, R. Mizoguchi, and J. Bourdeau, eds.), (Secaucus, NJ), pp. 323–338, Springer, 2010.

[30] A. Birnbaum, “Some latent trait models and their use in infering an examinee’s ability,” in *Statistical theories of mental test scores* (F. Lord and M.Novick, eds.), pp. 397–472, Addison-Wesley, 1968.

[31] W. J. van der Linden, *Hanbook of modernitem response theory*. Springer-Verlag, 1997.

[32] M. C. Desmarais, A. Maluf, and J. Liu, “User-expertise modeling with empirically derived probabilistic implication networks,” *User Modelling and user-Adapted Interaction*, vol. 5, no. 3-4, pp. 283–315, 1995.

[33] J. Martin and K. Vanlehn, “Student assessment using bayesian nets,” *Int. J. Human-Computer Studies*, vol. 42, pp. 575–591, 1995.

[34] E. Millan, M. Trella, J.-L. P. de-la Cruz, and R.Conejo, “Using bayesian networks in computerized adaptive tests,” in *Computers and Education in the 21st Century* (M. Ortega and J. Bravo, eds.), pp. 217–228, Kluwer, 2000.

[35] J. Vomlel, “Bayesian networks in educational testing,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 12, no. Supplement-1, pp. 83–100, 2004.

[36] H. Jeong and G. Biswas, “Mining student behavior models in learning-by-teaching environments,” *Educational Data Mining*, 2008.

[37] H. Jeong, G. Biswas, J. Johnson, and L. Howard, “Analysis of productive learning behaviors in a structured inquiry cycle using hidden markov models,” *Educational Data Mining*, 2010.