A flowchart-based intelligent tutoring system for improving problem-solving skills of novice programmers

D. Hooshyar,* R.B. Ahmad,* M. Yousefi,† F.D. Yusop‡ & S.-J. Horng§

*Department of Software Engineering, Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia
†Center of Systems and Machines Intelligence, College of Engineering, Universiti Tenaga Nasional, Kajang, Malaysia
‡Department of Curriculum and Instructional Technology, Faculty of Education, University of Malaya, Kuala Lumpur, Malaysia
§Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan

Abstract

Intelligent tutoring and personalization are considered as the two most important factors in the research of learning systems and environments. An effective tool that can be used to improve problem-solving ability is an Intelligent Tutoring System which is capable of mimicking a human tutor’s actions in implementing a one-to-one personalized and adaptive teaching. In this paper, a novel Flowchart-based Intelligent Tutoring System (FITS) is proposed benefiting from Bayesian networks for the process of decision making so as to aid students in problem-solving activities and learning computer programming. FITS not only takes full advantage of Bayesian networks, but also benefits from a multi-agent system using an automatic text-to-flowchart conversion approach for engaging novice programmers in flowchart development with the aim of improving their problem-solving skills. In the end, in order to investigate the efficacy of FITS in problem-solving ability acquisition, a quasi-experimental design was adopted by this research. According to the results, students in the FITS group experienced better improvement in their problem-solving abilities than those in the control group. Moreover, with regard to the improvement of a user’s problem-solving ability, FITS has shown to be considerably effective for students with different levels of prior knowledge, especially for those with a lower level of prior knowledge.

Keywords

computer programming, flowchart-based environments, intelligent tutoring system, novice programmer, problem-solving ability improvement.

Introduction

Nowadays, because of the fact that there is an increase demand in Web-based learning systems over traditional paper-based textbooks, Web-based learning systems have become more and more popular. However, Web courseware environments offer an even better flexibility and accessibility via the Internet, enabling students to control their own pace in their studies without any rigid classroom schedules and the presence of teachers (Brusilovsky, 1999). Interactive elements, such as video and audio as well as other forms of multimedia, can be incorporated into Web-based tutoring systems to convey a point, unlike printed textbooks. These environments also provide students with hyperlinks to directly transfer
them to other relevant sites. Nevertheless, the majority of the aforementioned systems suffer from a lack of adaptability or interactivity because they are static HTML Web pages (Brusilovsky, 1999). Thus, various needs of many students (i.e., adaptive and personalized guidance for users with different level of knowledge and background, additional material and extra explanations for some categories of users, removed irrelevant piece of content) cannot be addressed and covered by these types of systems, and thus they proved unsuccessful in satisfying the needs of the target users (Brusilovsky & Maybury, 2002). The next generation of Web-empowered systems which benefit from the practical applications of Artificial Intelligence is Web intelligent systems, which is the key direction for scientific research in this area (Web Intelligence Consortium, 2004). The development of a system with behaviour adaption to its task, interests and other features in order to satisfy its users is the only way of fixing the problem in traditional approaches, as stated by Brusilovsky and Maybury (2002). A new generation of learning systems offering ‘one-to-one’ individualized instruction by stimulating activities of human teachers, similar to one teacher and one student, is named Intelligent Tutoring System (ITS). However, research on the problem-solving ability of these ITSs is far from adequate and personalized, and adaptive problem-solving environments are even rarer in this area, similar to interactive problem-solving environment particularly in general construction method. Therefore, further investigation is required in this area. Basic computer programming skills acquisition cannot be obtained from others through passive learning and varies from theoretical knowledge learning. Taking this into consideration, developing such interactive problem-solving-based environments for learning computer programming should be considered in order to change traditional teaching methods by offering learners personalized feedbacks and worked examples. Thus, the main aim of this research is to propose a novel Flowchart-based Intelligent Tutoring System (FITS) for computer programming in order to support students in learning the imperative concepts of computer programming and in improving their problem-solving skills. A Bayesian network, a framework to manage uncertainty based on probability theory, is used in our proposed ITS for the process of decision making so as to aid students in both computer programming knowledge acquisition and problem-solving skills improvement. Similar to Johnson (2001) and Liu et al. (2008), students who use FITS are assisted in navigating the online learning materials. Nevertheless, unlike Johnson and Liu et al., learning goals, flowchart development with the aim of improving problem-solving ability and the appropriate reading sequences are recommended to students by FITS. Therefore, with the aim of improving problem-solving ability and visualizing the structure of a program, a multi-agent system applying an automatic text-to-flowchart approach is improvised in FITS to overcome the inconvenience of translating a problem specification to its corresponding program code solution. Our proposed FITS, which is able to visualize the solution development for a programming problem by converting the given problem statement to its relevant flowchart while engaging users in flowchart development, provides a novice with an accurate mental model of execution. It is noteworthy to mention that we discovered that there has yet to be any studies dealing with this specific issue (i.e., a flowchart-based Bayesian network ITS in computer programming). We believe that FITS not only helps novice programmers to overcome their difficulties in learning programming, but also can aid instructors in teaching programming. In addition, FITS has also addressed the navigational problem of users as they are now refocused on their study objectives by having an adaptable learning system. This paper is organized into seven sections. The second section explains previous works on ITS for computer programming. In the third section, the main architecture of FITS is illustrated and described along with its main components. In the fourth section, the evaluation method is presented, while the results of the experiments conducted are shown in the fifth section. Finally, a discussion and the conclusion are written in the sixth section.

Previous works

For assisting students towards a more efficient learning process, numerous computer-assisted instructional systems have been developed from exploring the area of learning environments (Sleeman & Brown, 1982; Urban-Lurain, 2004). Although these systems can aid users and thus are helpful, they are considered as weak systems from a more modern point of view, as they do not consider a user’s specific needs,
knowledge and limitations (Brusilovsky & Maybury, 2002; Yao & Yao, 2003). Computer Assisted Instructional (CAI) systems can provide neither a flexible nor a dynamic instructional plan for their users and therefore are unable to compete with instructors, considering the level of attention they could offer to the users (Bennett, 1999). As a result, and considering the abilities of Artificial Intelligence, researchers in this field have developed ITS to address the mentioned shortcomings (Burns & Capps, 1988). The aim of ITS, as a computer-based system, is to offer users personalized and flexible learning materials together with tailor-made instructions and feedbacks (Brusilovsky, 1999). Considering the exclusivity of each individual user, the logic behind developing an ITS can be justified. It has been proven that ITSs, which can be developed for traditional as well as distant learning courses, are significantly more effective in enhancing the performance of the users (Shute & Glaser, 1990). These systems can be distinguished from their CAI counterparts by their capability in a user’s learning while following up on a user’s knowledge during his/her studies (Shute & Psotka, 1996). Several efforts have been made to develop ITS for learning programming, namely Lisp-Tutor (Anderson & Reiser, 1985) which aims at teaching Lisp programming using rule-based approach, SQL-Tutor (Mitrovic, 2003) for SQL programming benefiting from constraint-based modeling approach, JITS (Sykes & Franek, 2003; Woolf, 2008) which aims at Java programming using decision tree approach, and ITS for learning PHP (Weragama, 2013). Although these systems have shown positive feedbacks, users are not provided with a navigational option to navigate through the online learning materials when they are given individualized recommendations from the system. Problem-solving ability improvement with the help of a stepwise guide, feedbacks, updating the student model and errors is the main aim of ITSs. Therefore, FITS was proposed and built to cover both problem-solving skills in the form of solution designing activities and learning material navigation. Applying Bayesian networks in ITS was first suggested by Villano (1992). Some researchers, such as Martin and VanLehn (1995), found that the assessment made by Villano cannot possibly reach the conclusion of what exactly users know and what they do not know. In our proposed system, three colour lights show whether a user knows the concept, does not know the concept but is not ready to learn, or does not know but is ready to learn. Martin and Van-Lehn (1995) proposed an ITS which only shows what users know, but FITS assesses both known and unknown concepts along with navigational options through unknown concepts. Many systems in this area have been proposed and developed for applying Bayesian networks in ITS systems (Butz, 2008; Collins, Greer, & Huang, 1996; Conati, Gertner, & Vanlehn, 2002; Gertner, Conati, & Vanlehn, 1998; Hartley & Tait, 1986; Henze & Nejdl, 2001; Park, Gyumin, & Meekyoung, 2009; Yang & Tzu-Hua, 2012; Zapata & Greer, 2001). Several applicable techniques have been reviewed by Jameson (1995) to be applied on ITS systems such as Dempster–Shafer theory of evidence, fuzzy logic and Bayesian networks. Bayesian networks were found to be the best among them all as proven by Pearl (1988), thus this algorithm was chosen to be used in FITS.

**FITS: a Flowchart-based ITS**

An effective way to acquire basic computer programming skills is to involve users in the solution designing activities from the early stages of computer programming. This concept consists of two aspects: one is ‘learning by doing’ and the other is ‘being guided stepwise by the system’. In this paper, we have proposed FITS which is a flowchart-based Bayesian ITS. It has three advantages over other developed ITSs for learning computer programming, namely: it emphasizes on the process of problem-solving activities in the form of solution designing activities by extending the traditional model of ITS; it develops an interactive problem-solving environment which actively engages users in flowchart development of imperative and practical programming problems; and it uses a novel flowchart-based ITS using Bayesian networks to aid the learning process of students. In addition, a multi-agent system benefiting from an automatic text-to-flowchart conversion approach is incorporated into the FITS to engage the students in stepwise problem-solving activities. While the majority of existing ITSs support static HTML Web pages of learning materials, FITS offers users the ability to navigate through the online learning materials, generates appropriate reading sequences and suggests learning goals.
importantly, since Markup languages are applied in the development of FITS, it offers students different options for flowchart development with various levels of guidance based on their level of knowledge using Bayesian networks and a multi-agent system which benefits from an automatic text-to-flowchart conversion approach. This feature considerably advances FITS in comparison to other related works in problem-solving skill improvement of students (i.e., the generated guidance regarding flowchart development of ‘Factorial’ might be different for each user based on their level of knowledge), taking full advantage of Bayesian networks with the aid of Markup languages. In this section, FITS is presented for C++ programming language; FITS’s entry page with its navigational menu is illustrated in Figure 1. The key components of FITS are outlined in this section and the way they interact with each other is described here as well. In addition, how to apply Bayesian networks in FITS for modeling and inference in order to design the pedagogical options for each single user is demonstrated. The main components of the proposed system are shown in Figure 2.

Overall, FITS benefits from several Artificial Intelligence techniques to assist users in computer programming, such as Natural Language Processing (NLP), Bayesian networks, knowledge base and knowledge expansion, and a multi-agent system. The multi-agent system operates in the background of FITS, located at the knowledge base, comprising of the NLP, knowledge expansion and knowledge base systems, and a Web Crawler for searching online for the relevant websites in order to improve the knowledge base without any human interference. This multi-agent system has benefited from an interactive approach of text-to-flowchart conversion in an automatic manner; this is used to involve the users in flowchart development and problem-solving activities (Hooshyar, Ahmad, Nasir, & Mun, 2014; Hooshyar, Ahmad, Shamshirband, Yousefi, & Horng, 2015).

Bayesian networks in FITS

How Bayesian networks are applied in FITS as an inference engine in order to assist a student’s learning based on his/her level of knowledge is discussed in this subsection.

Problem domain modeling

Two tasks, which are modeling the structure of the problem domain as well as tracking users’ knowledge concerning each specific concept in the problem
domain, are dealt with in assisting the navigation of a personalized Web-based learning environment, while a single task is involved in problem-solving ability improvement. In regard to the problem-solving ability improvement, the generated sub-flowcharts for each learner must be different in accordance to their level of knowledge. All three objectives are met with the help of a Bayesian network. The scope of the problem is restricted in our proposed system so as to simplify the task of developing FITS. The limitations applied are: FITS was developed to guide novice programmers in the C++ programming language; only the imperative and elementary concepts are taught, namely variables, assignments and control structures. It is noteworthy to mention that other concepts which are more advance, such as inheritance and pointers, are not covered. As a result, a set of imperative and fundamental learning concepts of C++ programming language (Programming I) in University of Malaya was identified. A graph is designed for those aforementioned concepts where each node represents a concept. Learning dependencies are shown among knowledge concepts using a directed edge from one concept to another, indicating that the former concept (node) is a prerequisite for understanding the latter. Therefore, by referring to the textbook used in the ‘Programming I’ course, the directed acyclic graph (DAG) is manually built. This graph encodes the appropriate learning sequences of the entire aforementioned concepts. We used the results of previous ‘Programming I’ final exams in order to obtain a conditional probability distribution (CPD) for the DAG. Generally, these questions comprise true\false options. In the situation that the question is answered wrongly, the concept is labelled as unknown, and if the question is answered correctly, the concept is labelled as known. Afterwards, the Bayesian network determines the probability that each concept can be known \( P(v_i = \text{known}) \). In addition, \( P(v_i = \text{known}, P_j = \text{known}) \) can also be computed; the Bayesian also calculates the desired CPD using the equation shown below.

\[
P(v_i = \text{known} | P_j = \text{known})
\]

\[
P(v_i = \text{known}, P_j = \text{known})
\]

\[
p(P_i = \text{known})
\]
Personalized learning

In order to offer personalized environments in a system, users’ interaction with the system should be observed to provide a profile for the individual user in order to update the Bayesian network. Two different approaches are used in FITS to obtain the profile from users: (i) in the case where a user knows about a specific concept, he/she responds to the query generated by the system; (ii) in the case where a user does not know about the concept, a quiz will be generated by the system for that specific concept with the aim of defining whether the user has understood the concept or not. As soon as the user replies to the queries generated by the system, the Bayesian Network algorithm gets updated and as a result of that, the relevant lecture notes or reading materials will be given to the user. Afterwards, the user is expected to provide the system with feedbacks in order to update the algorithm. The user can simply choose one of the options given by the system which are: (1) I understand; (2) I do not understand; and (3) I am not sure. These options are illustrated in the top right hand corner of Figure 3. With regard to option (1), the algorithm will be updated instantly and the knowledge level of the user will be reflected by this. In option (2), a proper quiz will be extracted from the system database and will be presented to the user. For instance, a user is unsure concerning ‘if statement’, a sample quiz regarding ‘if statement’ will be generated. After the quiz is completed by the user, the system extracts the stored answers from its database and compares them with the user’s answers. Then an instant feedback will be shown to the user regarding the correctness of his/her answers. In the case where the answers are all correct, the Bayesian network is updated and the navigational menu, refer to subsection ‘Navigational Support’, is presented again while that specific concept is now marked as known. In the case where most of the answers are wrong, the user will be recommended to review the learning materials regarding the concept again along with displaying the correct answers. The Bayesian network is also updated accordingly. Therefore, the Bayesian Network algorithm gets updated frequently and based on the students’ knowledge, the system recommends the relevant lecture notes or sub-flowcharts and asks the students to complete them by providing the necessary guidance, workspace and editor, system chat, instant feedback and visualization notations. For example, the sub-flowchart generated for a Factorial Exercise by the system might be different.

**Figure 3** Learning Materials on ‘Assignment’ Concept and Asking Whether the Student has Understood the Concept or Not
from user to user and some users will automatically be given more hints to complete the sub-flowcharts while some of them will be given less based on the knowledge level of each user.

**Knowledge base**

The knowledge base is divided into two different sections. The first section consists of the lecture notes in Web page form, a repository of sample quizzes and questions along with their key solutions, while the second section includes a flowchart-based multi-agent system, where the main architecture of the multi-agent system for flowchart development is shown in Figure 4. The lecture notes are presented when a user learns a new concept, while quizzes and questions are presented to determine whether the user comprehends a specific concept or not. These learning materials containing lecture notes and quizzes are stored by their specific concept in programming which refers to the relevant nodes in the Bayesian network. This separation strategy not only enables instructors to add, edit and compose the learning materials independently, but also a student’s level of knowledge can be estimated by the inference mechanism of the Bayesian Network algorithm applied in the proposed system. As each single node in the Bayesian network refers to a specific concept, the result of quizzes and the feedbacks taken from the lecture note are used to update the belief of the system. Moreover, in order to make any changes in the learning materials, the Bayesian network needs only to be re-indexed and not reformulated wholly. The system
can also recommend some learning sequences based on a user’s level of knowledge which is a result of this separation mechanism.

**Flowchart development**

Before users start interacting with this section, they are required to enter their target programming problem into the system. Afterwards, the multi-agent system recommends users with two different options for flowchart development based on their level of knowledge, which are either the toolbar or guidance option. In the toolbar option, users will be given an editor, a flowchart template, a workspace to complete the sub-flowchart, a sub-flowchart of the entered programming exercise (some parts of the complete flowchart) and a system chat. A screen shot of the toolbar option is shown in Figure 5. After dragging and dropping and completing the flowchart, the multi-agent system provides them with brief feedbacks next to each of the shapes in the flowchart, as brief feedbacks are a classical way of making users think after a failure/error.

In the guidance option, users are provided with a template of a flowchart having some shapes or sub-flowcharts placed in the right positions; unlike the first option, users are required to complete the flowchart using the editor. A screen shot of the guidance option is shown in Figure 6. Users are not allowed to drag and drop a shape in the wrong position as the system could immediately identify any errors. Unlike the toolbar option, if the users dropped a shape in the right position at the flowchart template, the appropriate context will be extracted from the database and will be placed in the flowchart shape. Instant feedback for each drag and drop is provided by the system chat. Once the users are done dragging and dropping shapes, the system offers them a full flowchart of the programming exercise extracted from the Internet to allow a comparison to be made with the flowchart developed by the users.

During the flowchart completion process using either the toolbar or guidance option, if any error occurs, it will be detected and stored by an error detection component. Afterwards, it will be passed to a Web Crawler in order to find any relevant, additional information and definitions for improving the database automatically without any human intervention for the next users. It should also be mentioned that the worst case scenario can also be predicted by the multi-agent system. In case no keyword is found in the database, the system will refer each of the main word and keyword to their corresponding shapes and will develop a sub-flowchart in addition to providing the users with a workspace for them to complete the
sub-flowchart, an online system chat to guide users step-by-step and a flowchart template. The workspace for the worst case scenario is shown in Figure 7.

It is noteworthy to mention that based on the profile taken and the knowledge assessment by the Bayesian network for each concept, FITS can manipulate the guidance and sub-flowcharts so as to provide users with an intelligent, personalized and dynamic environment for problem-solving improvement. This feature, taking full advantage of Bayesian networks with the aid of

**Figure 6** Workspace Provided by the Guidance Option Along with a Full Flowchart from the Internet

**Figure 7** The Workspace for the Worst Case Scenario

© 2015 John Wiley & Sons Ltd
Markup languages, can be considered as the main novelty of this study, as so far no such system has been developed that benefits from such an interesting feature to make problem-solving activities easy, interesting and an appealing subject to students. Based on the estimation of the probability of knowing the knowledge concept, FITS recommends three different flowchart development options. If the probability is between 0% and 50% or 50% and 75% or 75% and 100%, FITS recommends the worst case scenario, the guidance option or the toolbar option, respectively. For instance, the guidance given to user A for a programming problem (i.e., Fibonacci Series) is different from user B; in Figure 8, a sub-flowchart and guidance given to user A are displayed, while in Figure 9, a sub-flowchart for the same programming problem is shown indicating less guidance for user B based on his/her profile.

**User interface module**

Using the interface component, users are able to interact with FITS. This component is divided into two subsections: input and output. The former one is from users to FITS and updates the Bayesian network according to the evidence obtained from users, as described in section ‘Bayesian network in FITS’; while the latter is from FITS to users which is shown in a Web browser. The recommendations or guidance provided by FITS are either learning techniques or adaptive guidance, as will be discussed in the next section, or an animated flowchart development multi-agent, as discussed in the flowchart development section. All of the aforementioned learning materials are displayed for users using the output module of FITS.

**Adaptive guidance**

FITS can provide users with designed options for supporting a specific user individually by using a Bayesian network and estimating the knowledge level of each user. Three various sorts of adaptive guidance are offered to users by FITS, navigational menu, prerequisite recommendations and flowchart development for problem-solving skills, as was discussed in section ‘Flowchart Development’.

**Navigational support**

In order to navigate through the concepts which are shown by its proper light, where each one indicates the knowledge level of the users in that specific concept, such as a red light indicates the concept is unknown for that specific user, a navigational menu is used. Each
concept is marked with a colour (yellow representing ‘already know the concept’, red representing ‘the concept is unknown and not ready to learn’, and green representing ‘unknown concept but ready to learn’) showing the knowledge level of each user in that particular concept. It is noteworthy to mention that the Bayesian network computes these lights in an automatic manner from the initial profile, quizzes and feedbacks taken from users, as shown in Figures 1 and 3.

Prerequisite recommendations
In the case that users choose their own level of understanding after they have been exposed to the learning materials and guidance, FITS also offers users learning materials for prerequisite concepts, such as a link to lecture notes regarding the parent set of specific concept in the Bayesian network. For instance, offering the ‘assignment’ and ‘IncreDecrementOperator’ concepts for the ‘for statement’ concept, since ‘Assignment’ and ‘IncreDecrementOperator’ concepts are parent nodes to the ‘for statement’ concept in the graph, as shown in Figure 10. This method is used to avoid repeating a problem concept and to enable users to visit the prerequisite concepts in order to make sure that they have understood a specific concept. By doing so, a user who believed that a prerequisite concept has been understood when it has yet to be can confirm that the prerequisite concepts have actually been understood and they do not influence his/her learning process.

Methods
Participants
One tutor and 44 first year undergraduate students from University of Malaya took part in this research. The tutor taught ‘Programming I’ and has enough experience using FITS and traditional Web-based environments. All of the students were computer literate, although their abilities differ. The students were randomly divided into two different groups: an experimental group and a control group. While the teaching methods varied for these two groups, the learning materials, the practising time and the tutor remained the same for both groups. The control group had access to the same materials, organized in the form of folders, as the experimental group. However, the control group did not practise their skills with FITS while the experimental group did.
Instruments

Learning materials
The learning materials presented to students consist of some theoretical knowledge in basic computer programming, algorithm and solution design, and flowchart development. These learning materials were taught by the tutor during lecture and practical periods. Additional learning materials were also presented for FITS in the experimental group, while for the control group they were presented in the form of folders.

Prior knowledge assessment and summative assessment
In order to test a student's level of computer programming ability, a prior knowledge assessment was utilized. The test comprised 40 multiple-choice questions, each of which the students can leave blank if they do not know the answer to the question, and true-false to allow students to fully demonstrate what they know, and the total score was 100. One lecturer vetted all items' content validity, and the reliability was assessed by internal consistency [Kuder–Richardson 20 (KR-20)] through a pilot study with 44 undergraduate students with no prior knowledge of computer programming before the experiment. A KR-20 reliability coefficient of 0.87 was obtained. In addition, there was no significant difference between the mean scores on the pre-test for the groups. In accordance to the e-learning materials, a summative assessment was designed. These four items, either in FITS or folder form, were not considered in the summative assessment. It is also worth noting that the summative assessment’s pre-test is the same as the prior knowledge assessment. Finally, in order to show the learning outcome, post-test scores were obtained. In order to organize and administer the exams, an intelligent assessment system named iTAS was designed based on FITS to allow students to sit for the summative assessment and the prior knowledge assessment. For the time being, the specific management process will not be elaborated.

Research design
A quasi-experimental design was adopted by this research where the students were divided into two groups. As aforementioned regarding the prior knowledge assessment, the skills of both groups were not considerably different ($F_{1,43} = 1.111, p > 0.05$). During the 4-week period, some theoretical knowledge in C++ programming language, algorithm and solution design, and flowchart development was presented to the
students. These learning materials were taught by the tutor during the lecture and practical periods. The students were required to take part in lecture and computer lab sessions for 80 and 100 min/week, respectively. The tutor was first introduced to the main aim of this research, the teaching techniques and methods, and the research design. Afterwards, the prior knowledge assessment was performed on all students. The summative assessment’s post-test was taken by all students after a 4-week period of experimental control.

Data collection and analysis

The summative assessment’s post-test scores and the prior knowledge assessment’s scores are two groups of qualitative data which were collected in order to show the outcome of this research. In total, four students in both the experimental and control groups did not perform the summative assessment due to reasons such as time conflict and machine failure. The extreme value was eliminated so as to shun its effect on subsequent analysis results. Thus, in both the summative assessments’ post-test and prior knowledge assessment, 40 students’ scores were collected. Using spss 19.0, the data collected were analysed in three different ways. Firstly, the students were grouped into three categories based on their scores in the prior knowledge assessment, namely high, middle and low prior knowledge level groups. Secondly, the ‘summative assessment’s post-test scores’ which was considered as a dependent variable, along with the ‘different prior knowledge level’ considered as a fixed factor, were taken as input parameters by one-way ANOVA in order to test the relationships between these two factors for the two different types of teaching methods. In addition, the ‘summative assessment’s post-test score’ considered as a dependent variable and the ‘different types of teaching methods’ considered as a fixed factor were taken by the one-way ANOVA in order to test the relationships between these two factors for all of the three prior knowledge level groups. Thirdly, in order to compare the problem-solving ability improvement of students having different prior knowledge levels in the FITS group and in the control group, a least significant difference (LSD) post hoc test was performed. Lastly, in order to compare the problem-solving ability improvement of students with low and middle prior knowledge levels in the two different teaching methods, a LSD post hoc test was performed. Additionally, the problem-solving ability improvement of students having high and middle prior knowledge levels was also performed across two different teaching methods.

Results

The effect of ‘different types of prior knowledge and teaching methods’ on student’s problem-solving ability improvement

As aforementioned, based on the level of the prior knowledge scores, all students were categorized into three groups (refer to Table 1). It should be noted that owing to the fact that some of the students’ scores were the same, the number of students is not the same in the three groups. We also tested the homogeneity of the variance assumption ($F_{5,36} = 1.403$, $p > 0.05$) using two-way ANOVA and it was shown that this assumption was not violated. As shown in Table 2, the ‘different prior knowledge level’ factor ($F_{2,39} = 11.573$, $p < 0.05$) had a significant effect on the problem-solving ability improvement. In addition, the ‘different types of teaching methods’ factor ($F_{1,39} = 17.050$, $p < 0.05$) also had a significant effect on the problem-solving ability improvement. Furthermore, the ‘different types of prior knowledge and teaching methods’ interaction effect was also significant ($F_{2,39} = 4.213$, $p < 0.05$). Table 1 shows the statistical description of the three groups.

<table>
<thead>
<tr>
<th>Group</th>
<th>Prior knowledge assessment (mean scores)</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group with prior knowledge of high</td>
<td>33.62</td>
<td>0.657</td>
</tr>
<tr>
<td>including 15 students</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group with prior knowledge of middle</td>
<td>31.54</td>
<td>0.673</td>
</tr>
<tr>
<td>including 13 students</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group with prior knowledge of low</td>
<td>28.06</td>
<td>0.694</td>
</tr>
<tr>
<td>including 12 students</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1. The Statistical Description of the Three Groups

© 2015 John Wiley & Sons Ltd
p < 0.01) and the ‘different types of teaching methods’ factor ($F_{1,39} = 27.946, p < 0.01$) had a great impact on the ‘summative assessment’s post-test scores’. In accordance to the students’ problem-solving ability improvement, the results of the LSD post hoc test as shown in Table 2 illustrate that the students’ problem-solving ability improvement in the FITS group was considerably better than in the control group ($p < 0.01$).

Based on the findings, it can be seen that FITS was the most effective especially at problem-solving ability improvement. In addition, based on the results of the LSD post hoc test, higher learning effectiveness was shown for students with high and middle prior knowledge levels than students having a low prior knowledge level ($p < 0.01$). How learners associate new knowledge with what they already know can be affected by prior knowledge, as stated by Spyridakis and Lsakson (1991). According to Table 2, there is a considerable effect on the interaction between the ‘different types of teaching methods’ factor and the ‘different prior knowledge level’ factor ($F_{2,16} = 6.297, p < 0.01$). Thus, we used one-way ANOVA to analyse this further as demonstrated below.

### Problem-solving ability improvement of students with different prior knowledge levels in different types of teaching methods

The homogeneity of the variance assumption in the FITS group ($F_{2,22} = 3.013, p > 0.05$) and the control group ($F_{2,14} = 0.350, p > 0.05$) was tested using one-way ANOVA and the results showed that the assumption was not violated. Table 3 shows the results of one-way ANOVA. As shown in Table 3, unlike the control group, there is no considerable impact from the ‘different prior knowledge level’ factor on the ‘the summative assessment’s post-test scores’ ($F_{2,22} = 0.320, p > 0.05$) in the FITS group. This means that the prior knowledge of students does have considerable effect on their problem-solving ability improvement. The considerable impact of the ‘different prior knowledge level’ on the ‘the summative assessment’s post-test scores’ ($F_{2,14} = 13.940, p < 0.01$) in the control group is also shown in Table 3. According to the results of the LSD post hoc test, the middle and high prior knowledge level students have considerably better problem-solving ability improvement than those with low prior knowledge.

### Table 2. Two-Way ANOVA on 40 Students for ‘Different Prior Knowledge’ and ‘Different Teaching Methods’

<table>
<thead>
<tr>
<th>Foundation</th>
<th>SS</th>
<th>df</th>
<th>MS</th>
<th>F value</th>
<th>Post hoc</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A) = Different types of teaching methods</td>
<td>524.209</td>
<td>1</td>
<td>524.209</td>
<td>27.946</td>
<td>FITS group &gt; control group</td>
</tr>
<tr>
<td>(B) = Different levels of prior knowledge</td>
<td>403.628</td>
<td>2</td>
<td>202.319</td>
<td>11.573</td>
<td>High prior knowledge &gt; low prior knowledge</td>
</tr>
<tr>
<td>A × B</td>
<td>248.612</td>
<td>2</td>
<td>126.220</td>
<td>6.302</td>
<td>Middle prior knowledge &gt; low prior knowledge</td>
</tr>
<tr>
<td>Error</td>
<td>2183.106</td>
<td>36</td>
<td>18.009</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected total</td>
<td>3162.193</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 3. One-Way ANOVA on Two Groups Having Different Prior Knowledge Levels

<table>
<thead>
<tr>
<th>Group</th>
<th>Variable</th>
<th>Level of prior knowledge</th>
<th>M</th>
<th>F value</th>
<th>Post hoc</th>
</tr>
</thead>
<tbody>
<tr>
<td>FITS group (22)</td>
<td>Different level of prior knowledge</td>
<td>High</td>
<td>33.502(0.891)</td>
<td>0.298</td>
<td>High prior knowledge level &gt; low prior knowledge level</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Middle</td>
<td>33.112(0.887)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low</td>
<td>32.563(0.816)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Control group (14)</td>
<td>Different level of prior knowledge</td>
<td>High</td>
<td>32.298(0.891)</td>
<td>13.940</td>
<td>High prior knowledge level &gt; low prior knowledge level</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Middle</td>
<td>30.638(0.990)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low</td>
<td>24.300(1.038)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
knowledge level in the control group. However, no considerable difference between the problem-solving ability improvement of students with middle prior knowledge level and students with high prior knowledge level can be seen ($p > 0.05$). In addition, in order to discover the problem-solving ability improvement of students with different prior knowledge levels across two different types of teaching methods, one-way ANOVA was performed. The homogeneity of the variance assumption of the high, middle and low prior knowledge level groups which is ($F_{1,14} = 1.531$, $p > 0.05$), ($F_{1,12} = 0.756$, $p > 0.05$) and ($F_{1,10} = 0.228$, $p > 0.05$), respectively, was tested before one-way ANOVA and no violation was found in the homogeneity assumption. Table 4 shows the results of the one-way ANOVA and indicates that ‘different types of teaching methods’ had a considerable impact on the ‘summative assessment’s post-test scores’ for the middle prior knowledge level group ($F_{1,12} = 5.220$, $p < 0.05$) and for the low prior knowledge level group ($F_{1,10} = 32.767$, $p < 0.01$). In the FITS group, the high prior knowledge level group performed better, but no significant differences in the statistical points were seen. Students can achieve a deeper conceptual understanding when they are provided with detailed and sophisticated feedbacks (Winne, 1989). From Tables 3 and 4, it can be seen that FITS allows students to experience better problem-solving ability improvement for any prior knowledge levels. More importantly, by using FITS, a better total learning experience can be achieved by students with low prior knowledge level and the improvement of their problem-solving abilities can be clearly seen compared with students with high prior knowledge level. As more guidance and feedbacks are needed for students with a low prior knowledge level, FITS provides them with more guidance and feedbacks in a stepwise manner. An equivalent problem-solving ability improvement in learners having different prior knowledge levels was also seen in the experimental group even though the same design was not used in the group. As a result, low prior knowledge level students have a considerably lower performance in the control group. The results, in an intuitive way, are presented in Figure 11.

### Discussion and conclusions

In the research on learning systems and environments, intelligent tutoring and personalization are considered as the two most important factors. An effective tool for problem-solving ability improvement is an ITS which is capable of mimicking a human tutor’s actions in implementing one-to-one personalized and adaptive teaching. A novel FITS for computer programming was proposed with the aim of improving the problem-solving ability of novice programmers. The developed system benefits from Bayesian networks in order to handle uncertainty management and decision making in a student’s learning process. A novel architecture for designing an ITS for C++ programming language aimed at novice programmers, using a flowchart-based multi-agent system.

<table>
<thead>
<tr>
<th>Group</th>
<th>Variable</th>
<th>Level</th>
<th>M (standard error)</th>
<th>F value</th>
<th>Post hoc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group with prior knowledge of high including 15 students</td>
<td>Different types of teaching methods</td>
<td>FITS group</td>
<td>33.502(0.891)</td>
<td>1.104</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Control group</td>
<td>32.298(0.891)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group with prior knowledge of middle including 13 students</td>
<td>Different types of teaching methods</td>
<td>FITS group</td>
<td>33.112(0.887)</td>
<td>5.220</td>
<td>FITS group &gt; control group</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Control group</td>
<td>30.638(0.990)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group with prior knowledge of low including 12 students</td>
<td>Different types of teaching methods</td>
<td>FITS group</td>
<td>32.563(0.816)</td>
<td>32.767</td>
<td>FITS group &gt; control group</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Control group</td>
<td>24.300(1.038)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Figure 11](image_url)
along with Bayesian technology, was discussed in this paper. In addition, the application of a Bayesian network to be taken full advantage of in FITS for modeling and inference purposes was elaborated. More importantly, a multi-agent system benefiting from an automatic text-to-flowchart conversion approach was applied in FITS to support problem-solving skill improvement. The assessment of student’s level of knowledge by the use of Bayesian networks was first suggested by Villano in 1992. It has been argued that such systems using Bayesian networks are unable to identify the knowledge components which are necessary to be taught in addition to what students do not exactly know. By using FITS, each concept in programming language is shown by an appropriate traffic light which enables students to choose the appropriate instructional content to study and the learning goals are suggested when a specific concept is not understood or is unknown. Additionally, there is a possibility of learning a specific concept without having to go through all of the previous concepts by presenting the minimum prerequisite knowledge required to understand the desired concept. To the best of our knowledge, FITS is the first flowchart-based Bayesian ITS which aims at improving the problem-solving ability of novice programmers through solution designing activities. Using FITS, not only can a student’s knowledge state be assessed, but the student is given guidance on how to navigate through the unknown concepts. The majority of existing Web-based learning environments use only static HTML Web pages which makes them nothing more than a regular textbook. Therefore, two main weaknesses have been discovered in these forms of instructional materials, namely the lack of interactivity and adaptability shortage. Hypermedia-structured learning materials such as lecture notes, quizzes, feedback and examples are provided by FITS. Finally, by offering various options in flowchart development along with step-by-step guidance, FITS enables students to improve their problem-solving skills through solution designing activities. It is noteworthy to mention that based on the profile taken from users and the knowledge assessment by the Bayesian networks for each concept, FITS can manipulate its guidance, change the sub-flowcharts’ components for each user, so as to provide the users with an intelligent, personalized and dynamic environment for problem-solving improvement. This feature can be considered as the main novelty of this study, as currently there is no such system that has been developed benefiting from such an interesting feature to make problem-solving activities easy, interesting and an appealing subject to students. FITS has been implemented and was recently used in a 4-week duration by 44 students in University of Malaya for an initial computer programming course named ‘Programming I’. FITS has proved to be effective in a student’s problem-solving improvement during the experimental study. Based on the results of the experiments, students in the FITS group experienced better problem-solving ability improvement than those in the control group. Moreover, with regard to problem-solving ability improvement, FITS had shown great effectiveness even for students with a lower prior knowledge level as compared with others. We believe, to support tutors teaching computer programming, FITS can be useful in institutions especially for the target audience suffering from a lack of problem-solving skills as well as offering a personalized instruction that they need.

Acknowledgement

This research has been financially funded by the University of Malaya with project number RG327-15AFR in Malaysia.

References


