C. Frasson G. Gauthier G. I. McCalla (Eds.)

Intelligent Tutoring Systems

Second International Conference, ITS '92
Montréal, Canada, June 10-12, 1992
Proceedings

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

A Framework for Intelligent Knowledge Sequencing and Task Sequencing

Peter L. Brusilovsky

International Centre for Scientific and Technical Information,
Kuusinen str. 21b, Moscow 125252, Russia

Abstract. Most effective human tutors possess the skill of adaptive sequencing of knowledge and tasks. This skill is also the key function of many important tutoring systems and learning environments. A number of workers in the field of intelligent tutoring systems have tried to build a framework for intelligent knowledge and task sequencing. In this paper we briefly discuss previous work on building a framework and strategies for knowledge and task sequencing. Then we suggest several additional components we have designed to complete the framework and describe a pragmatic strategy for multiple-kind, multiple-concept task sequencing based upon the framework.

1. Introduction

One of the most important duties of a human tutor is to extend the student's knowledge of a given subject. Extending the student's knowledge step by step, the human tutor decides what to teach next and how to teach. The teacher provides the student with a sequence of pieces of new knowledge and suggests a sequence of learning tasks to master the knowledge. In Soviet psychology, a learning task is any teaching operation designed to organize the student's learning activity. Learning tasks are the only way for the student to acquire new pieces of knowledge. An effective human tutors skillfully adapts the sequence of knowledge and tasks according to the student's needs, abilities, and knowledge. Effective tutor can generate an individual task sequence for any student.

Both knowledge sequencing and task sequencing are the key functions of many tutoring systems. Task sequencing is also a function of many learning environments [13]. A number of researchers in the field of intelligent tutoring systems have tried to build a framework for intelligent knowledge and task sequencing. In the first part of the paper we briefly discuss the problem itself, previous work on knowledge and task sequencing, and the components of the framework suggested in several papers and projects. We refer to the following systems, which demonstrate several components of the framework and several strategies for knowledge and task sequencing: SCHOLAR [5], GCAI [11], BIP [1], BIP-II [20], WUSOR [7], QUADBASE [10], ReGIS [9], Assembler Tutor (AT) [17], Intelligent Tutor for Basic Algebra (ITBA) [13], SCENT-3 [3,14], and TOBIE [18,19].

In the second part of the paper we describe several additional components we have designed to complete the framework and give a pragmatic strategy for multiple-kind multiple-concept task sequencing based upon the framework. The complete framework and the strategy was designed and tested for the Intelligent Tutor, Environment and Manual for Introductory Programming (ITEM/IP). Details of ITEM/IP development are given elsewhere [4]. Here we consider features of ITEM/IP related to the topic of this paper.

2. The curriculum and the student model for knowledge sequencing

The tutoring system's choice of information to convey is limited to content from the knowledge representation. How these pieces of knowledge are sequenced in instruction is determined by the developmental level and current comprehension of the student, by the teaching method, and by the evolutionary structure of information on the given subject, represented in the system. [6, p.336]. The evolutionary structure forms a *syllabus* of

knowledge [6] or *curriculum* [12] from which the tutor or coach can select. The student's developmental level and current comprehension are represented by the student model. The curriculum and the student model form the basis of any framework for intelligent task sequencing. A variety of teaching strategies can be built upon this base to provide knowledge sequencing according to a variety of teaching methods.

2.1 The simple case

The simplest curriculum (rather a syllabus) [8] is just a set of unrelated knowledge elements (knowledge items). Each knowledge element represents a subset of the expert's knowledge. Depending on the subject, the amount of knowledge represented by the element can be varied from an atomic skill or concept to a complex curriculum topic. The simplest student model is an overlay model [8]. The overlay student model represents student knowledge as a subset of expert knowledge. To build the overlay model we should augment each knowledge element with a tag (yes/no), which reflects whether the modelled student has mastered this element of knowledge or not.

Combining the simplest curriculum with an overlay student model, we obtain the simplest framework for knowledge sequencing. Such a framework is too weak to support more intelligent knowledge sequencing than random floating from one element of knowledge to another. This simplest framework was used in SCHOLAR for both random knowledge sequencing and random question sequencing.

2.2 The advanced case

To form a real basis for intelligent knowledge sequencing we should represent in the curriculum both the domain knowledge elements (KE) and the relationships between them. The set of these relationships forms a pre-specified curriculum structure.

The advanced curriculum can be represented as a network of knowledge elements (nodes) connected with relationships (links). Several kinds of domain knowledge elements of different levels of generality can be used to form the curriculum. Several kinds of links can be used to represent pedagogically important relationships between knowledge elements. The most important relationship is the *prerequisite* relationship, the only one used in several projects to form the curriculum structure (GCAI, QUADBASE, ReGIS, AT, TOBIE). The curriculum based on the prerequisite relationship can be used by a teaching strategy to produce any admissible sequence of represented knowledge.

To build "more intelligent" teaching strategies we can represent more information about curriculum structure by adding more kinds of relationships. The systems BIP-II and SCENT-3 use the links "part of" ("component"), "is a" ("kind"), and others to represent important relationships between KEs.

The genetic graph [7] is the most advanced approach to representation of the curriculum. Its nodes represent elements of knowledge (originally procedural skills) of varying level of expertise and its links include the relationships of analogy, specialization, generalization, prerequisite, and deviation. Thus the genetic graph serves to represent static relationships as well as evolutionary relationships. All these relationships can be used by the tutoring system for advanced intelligent knowledge sequencing.

Another component required to build intelligent knowledge sequencing is an advanced student model. Most systems use for this purpose an advanced overlay model which can reflect more than two discrete states (yes, no) of student's knowledge on every KE: from three states (SCENT-3) to seven states (ReGIS) or more. Some systems use wide integer (QUADBASE) or real (GCAI) intervals to measure student knowledge of the KE. To represent discrete states of knowledge the interval is divided into subintervals by thresholds.

2.3 Curriculum based knowledge sequencing and task sequencing

Several systems successfully use the curriculum and the student models described above for knowledge sequencing. The curriculum is learnt element by element along the links. Prerequisite links restrict the choice of the next knowledge element (all the prerequisite KEs should be learnt beforehand). Thus the set of prerequisite links defines implicitly all "legal" orders of KE learning. The links and the student model are used by a teaching strategy to generate the best KE sequence for the given student.

Knowledge sequencing provides a good basis for simple task sequencing. A *simple* task is *single-concept* task, that is, the teaching operation aimed to work with one KE only. Having selected the current knowledge element, the system selects the best task or chooses at random from a limited set of simple tasks related to the current knowledge element. The system can select several tasks of different kinds for the student to master the current KE. The student model is used to determine when to leave the current KE for the next one. The most advanced plan-based approach for simple task sequencing was developed in SCENT-3.

3. Task spectra for complex task sequencing

A *complex* task is a *multiple-concept* task - a teaching operation related to several curriculum KEs. Presentation tasks (information, demonstration) present or explain the related KEs. Problem tasks test the student's knowledge of the related KEs (To complete a task a student should know all the related KEs). The list of curriculum knowledge elements related to the task is the *task spectrum*. The task spectrum links explicitly the complex task with the curriculum and provides the tutoring system with a key to the sensible choice of tasks.

The curriculum, the overlay student model, and the set of tasks augmented with spectra form the framework for complex task sequencing. The problems of multiple-concept task sequencing are not well-studied in the field of ITS. Three systems can build an adaptive sequence of multiple-concept tasks for a student: namely, BIP, BIP-II and ITBA. These systems use the framework described above together with different strategies for task sequencing. BIP uses unrelated curriculum with unrelated knowledge elements and a simple strategy for task sequencing. More advanced task sequencing requires more knowledge about curriculum to be represented either by links between KEs (BIP-II) or by links between sets of KEs (ITBA).

We propose a two-step teaching strategy for task sequencing. The first step is compiling the list of all relevant (ready to be selected) tasks according to the current student model. The task is relevant if it contains some "goal" concepts in the spectrum and has all the spectrum KEs learnt or ready to be learnt. The second step is selecting the best task from the list of relevant ones for a student at a given moment. The following simple approaches have been used. BIP presents the task with the greatest number of unlearned skills. ITBA presents the task with the fewest non-target skills. The BIP-II strategy is as follows: if the student is doing well, then select the relevant task that has the fewest learnt skills; if the student is progressing slowly, then find the relevant task with the fewest unseen skills.

4. Task complexities for the choice of the best task

Experience with BIP, BIP-II and ITBA systems demonstrates that task spectra related to curriculum provide a good basis for compiling a lists of relevant tasks at each moment, but spectra contain insufficient information from which to select the best task from the list of relevant ones. Additional information about the student and the tasks should be taken into account to select the best task.

To guide the choice of the best task in the ITEM/IP system [4] we used the notion of task complexity. >From analysing the protocols of students solving programming problems, we conjecture that each task has a measurable complexity. Furthermore, each student at each moment prefers a particular value of complexity, an optimal value. The task is less preferable to the student if its complexity is not optimal. The same idea was suggested by another Soviet group in the domain of mathematical differentiation [15]. We have noticed also that the optimal complexity is not fixed for the given student, but increases according to the development of his or her experience in problem solving.

We have also suggested that the complexity of problem has two independent components: conceptual complexity and structural complexity. The conceptual complexity is just a number of not quite well learnt KEs in the task spectrum. Thus the conceptual complexity of the task depends on the state of student's knowledge reflected in the student model and should be computed directly in the process of learning. Structural complexity is a constant part of complexity. It reflects the number of steps required to solve the problem. The structural complexity is added to task frame beforehand by measuring the model solution of the task with special procedure.

Thus in ITEM/IP we use a precompiled set of tasks augmented with both spectra and the structural complexities of tasks. We also use an advanced overlay model: a set of counters corresponded to domain KEs and two additional counters for each student-- one for the current optimal structural complexity and the other for the current optimal conceptual complexity. Task spectra are used to compile a list of relevant tasks. The difference between task complexities and optimal complexities for the given student are used to select intelligently the best task from the set of relevant ones. The details of the selection algorithm are given below. If the student solves the problem successfully the student model complexities are set equal to the maximum of the solved task complexities and the current student model complexities. If the student did not solve the problem, the student model complexities are decreased.

5. A network for knowledge sequencing and task sequencing

The curriculum network and the advanced-overlay student model serve as the basis of the framework for knowledge and task sequencing. The following four kinds of sequencing can be built upon this framework: conceptual knowledge sequencing, procedural knowledge (skills) sequencing, simple task sequencing, and complex task sequencing. Two kinds of curriculum were used and studied in previous work: skills networks, in which KEs represent skills (pieces of procedural knowledge), and conceptual networks, in which KEs represent concepts (pieces of conceptual knowledge). Note that distinction between procedural and conceptual knowledge is not clear-cut [16]. A skills network was used for procedural knowledge sequencing, simple task sequencing, and complex task sequencing (BIP, BIP-II, WUSOR, REGIS, AT, ITBA, TOBIE, SCENT-3). Conceptual network was used for conceptual knowledge sequencing and simple task sequencing (GCAI, QUADBASE).

A system capable of all the listed kinds of sequencing should be built upon both conceptual and skills net. Two projects BIP-II and SCENT-3 [2], employ both kinds of nets. Both systems use conceptual network to build the skills network beforehand and use the skills network for task sequencing.

The domain model in ITEM/IP is a joint network that contains interconnected elements of conceptual and procedural knowledge. Really we used three kinds of nodes in the network: high level programming concepts (for example: loop), constructs of the programming language studied (for example *while* statement) and skills of using the programming constructs in a context (for example using a negative condition in a *while*

loop). The nodes are linked by the following relations: general/specific ("is a"), component ("part of"), and usage. The latter relationship links a programming construct usage skill with the construct itself (for example: the skill mentioned is linked to *while* by an "is a" link and is linked to the "negative condition" construct by a usage link.

It should be noted that the spectra of the ITEM/IP complex tasks contain references to skills only. A simple task is linked either to a construct or a skill. None of the tasks are linked to any concept. The special strategy is used to diffuse the changes in the student model along the links. It allows the system to re-evaluate the student's knowledge of the related KEs and to support the actual state of the student model, though most of the tasks have skills only in their spectra.

The joint network is a good basis for knowledge sequencing and task sequencing. The general view of the guided tutoring process supported by ITEM/IP is as follows. The system selects an optimal knowledge element and presents it to the student. Then the system poses a number of tasks (either simple or complex) of different kinds which serves to explain the element to the student, to force him to use this element and to check the student's knowledge. Diffusion in the student model enables the system to control the process of task sequencing. Then the new ready-to-be-learnt knowledge element is selected.

The teacher can limit the system's choice of next knowledge element by setting the individual teaching order for the student. The teaching order is just a sequence of subsets of knowledge elements. These subsets should be learnt sequentially subset by subset. If the teaching order is set, then the system makes the choice of the next element to be presented within the current subset of elements. When all the elements of the current subset have been learnt the system moves to the next subset. The teaching order enables the teacher to tune the system to his or her preferred order of teaching the course. It makes the system more flexible.

6. A strategy for sequencing tasks of different kinds

As noted, ITEM/IP is able to select the best task among the set of tasks of different kinds. The problem of sequencing different kinds of tasks is difficult in general. It was solved successfully by using a plan-based approach for simple tasks in SCENT-3 system.

Sequencing of complex multiple-kind tasks is more difficult. All the systems BIP, BIP-II, and ITBA that provide complex-task sequencing are only able to do single-kind task sequencing. In the future, ITBA may provide multiple-kind task sequencing. We use a pragmatic approach to multiple-element multiple-kind task sequencing in ITEM/IP to overcome general difficulties. ITEM/IP is able to select from five kinds of learning tasks: presentation, demonstration, test, programming problem to analyse, and programming problem to solve.

- **Presentation** tasks are simple tasks that introduce (or remind) the student of a piece of conceptual knowledge: a programming concept or construct.
- **Demonstration** tasks explain constructs to the student by visual demonstration of examples "in action".
- **Test** tasks check the student's understanding of a given programming construct. The student is presented with an example of the given construct in a context and with input data. He should mentally execute the example and enter the output data. Both demonstration and test are simple tasks related to the skill tested. These kinds of tasks have several skills in spectra. One of them is the key-skill, which is demonstrated or tested by this task.
- **Programming problem to solve** is the most important kind of tasks. The student is presented with the problem to be solved by developing a program.

- **Programming problem analysis** tasks are inverted tasks. The student is presented step by step with a solution to a programming problem. These two kinds of tasks are complex. They serve to develop mastery of a number of skills, and these skills are listed in the task's spectra.

Thus we have five distinct kinds of tasks to be used in order. First we present the programming construct to the student, then we demonstrate how it works, then we test the student's understanding, then we show by example how to use this construct to solve the programming problem and finally we force the student to use the construct to solve a programming problem.

Thus the process of mastering the given construct is divided into five stages corresponding to five kinds of tasks, and each stage implies the use of the corresponding kind of task to contribute to the learning of the construct. The use of the next kind of tasks is not allowed, because the student is not ready to it. The use of the previous kinds of tasks is allowed but these tasks do not contribute to the learning of the construct at this stage. For example, if testing is the current learning stage for the given construct, we can't use programming problems to master this construct, since it is not clear yet whether the student has understood the semantics of the construct. Presentations, demonstrations and tests are allowed, but only tests will contribute to the learning of the construct at this stage.

We use the student model to reflect and control the stages of learning KEs. For each domain KE the student model contains an positive-integer counter. The interval of possible values for this counter is divided into subintervals by five thresholds (figure 1). Each interval corresponds to a stage of KE learning. Thus the value of the counter tells us the stage of the corresponding concept and what kinds of tasks are optimal, allowed, or not allowed to teach the concept. The current stage for a skill is computed as the minimum stage of concepts related to the skill. If the counter value is zero, then the concept is not ready to be learnt.

The pragmatic approach to the process of teaching and learning the programming concepts and constructs restricts the choice of the next task and enables us to build a strategy for the best choice among dozens of tasks of five kinds. The key idea of this strategy is that solving programming problems is the most important student activity. It is during the process of solving these problems that one can thoroughly understand various programming concepts and constructs and learn to use them properly in order to achieve the goals posed before him. This idea steams from the Soviet psychology. An algorithm for the strategy is described briefly in figure 2.

Threshold No		1	2	3	4	5	
Interval No	1	2	3	4	5	6	
0							>
Presentation	-	+!	+	+	+	+	+
Demonstration	-	-	+!	+	+	+	+
Test	-	-	-	+!	+	+	+
Problem to analyse	-	-	-	-	+!	+	+
Problem to solve	-	-	-	-	-	+!	+

- not allowed to select
+ allowed to select
! optimal kind for this stage

Figure 1. Thresholds and intervals for construct's counter values in the student model define constraints for the kinds of task selected.

Step 1. Trying to select a programming problem to solve.
- (a) Compile a list of all relevant programming problems. (The problem is relevant if all the skills from the spectrum are on the fifth or sixth stage (see figure 1) and at least one of skills is right on the fifth stage.)
  If the list is empty compile a list of "not ready-5" skills and go to step 2.
  (If the problem has all but one or two skills on the fifth or sixth stage and these one or two on the fourth stage, add these one or to skills to "not ready-5" list. These skills protect the problem from being relevant)
- (b) Select the best task from the list of relevant. For each relevant task we count the weighted polynomial value. (The parts of the polygon are: squared differences between structural complexity of the task and student's optimal structural complexity; the same differences for the conceptual complexity and for the number of "erroneous" skills in the task's spectrum.) The best problem is those one that have minimal value.

Step 2. Trying to select a programming problem to analyze.
- (a) Compile a list of all relevant problems. If the list is empty than compile "not ready-4" list and go to step 3.
- (b) Select the best task from the list of relevant. (Here one more part is added to polygon: a number of "not ready-5" skills in the problem's spectrum.)

Step 3. Trying to select a test.
.....

Step 4. Trying to select a demonstration
.....

Step 5. Trying to select a presentation
- (a) Compile a list of all ready-to-be-learned concepts. If list is empty go to step 6.
- (b) .....

Step 6. Mark the next subset of KE as ready-to-be-learned. If we were working with last subset of the teaching order, the guided teaching would be finished.

Figure 2. ITEM/IP algorithm for choice of the best teaching operation

7. Conclusion

We have built a framework for knowledge sequencing and task sequencing, including multiple-concept multiple-kind task sequencing. The framework consists of the domain network (a combination of conceptual network and skills network), the advanced-overlay student model (a set of counters, a set of thresholds and optimal complexities), and a set of learning tasks of different kinds, augmented with spectra and complexities. Some parts of this framework were studied in the previous work of several authors, other parts were suggested and designed by us.

We have also designed a strategy for multiple-kind multiple-concept task sequencing that is based on a pragmatic approach to the teaching of programming concepts and constructs. We have tested the strategy and the system itself in the learning process among first-year students of the Moscow State University and 14-year-old students of Moscow schools. The students found that the task-sequencing strategy seemed intelligent and they usually agreed with the system's choice. Nevertheless advanced students in the second part of the course preferred to select the next task on their own.

We consider the framework described above as a general framework for knowledge sequencing and task sequencing. Different intelligent strategies can be built upon this framework. We do not consider our pragmatic approach and the strategy described above as general ones. We have tried to apply it to multiple-kind multiple-concept task sequencing in other domains such as geography, but we have failed. We are now designing an

authoring tool that will support the design of learning strategies in the form of production rules operating within the the framework described above. We plan to use this tool to compare various guided tutoring strategies in the classroom.

References

1. A.Barr, M.Beard, R.C.Atkinson: The computer as tutorial laboratory: the Stanford BIP project. *International Journal on the Man-Machine Studies*, 8(5), 1975, 567-596.
2. B.J.Brecht: *Determining the focus of instruction: Content planning for intelligent tutoring systems*. PhD Thesis, University of Saskatchewan, 1990
3. B.J.Brecht, G.I.McCalla, J.E.Greer: Planning the content of instruction. *Proceedings of the 4-th International Conference on AI and Education*, Amsterdam: IOS, 1989
4. P.L.Brusilovsky: The intelligent tutor, environment and manual for introductory programming. *Educational and Training Technology International*, 29(1), 1992
5. J.R.Carbonell: AI in CAI: An artificial intelligence approach to computer aided instruction. *IEEE Transactions on Man-Machine Systems*, MMS-11(4), 1970, 190-202
6. C.Dede: A review and synthesis of recent research in intelligent computer - assisted instruction. *International Journal on the Man-Machine Studies*, 24, 1986, 329-353
7. I.P.Goldstein: The Genetic graph: a representation for the evolution of procedural knowledge. *International Journal on the Man-Machine Studies*, 11(1), 1979, 51-77
8. I.P.Goldstein, B.Carr: The computer as coach: an athletic paradigm for intelligent education. *Proceedings of the ACM*, 1977
9. J.Heines, T.O'Shea: The design of a rule-based CAI tutorial. *International Journal on the Man-Machine Studies*, 23(1), 1985, 1-25
10. P.V.Hudson, J.A.Self: A dialogue system to teach database concepts. *The Computer Journal*, 25(1), 1982, 135-139
11. E.B.Koffman, J.M.Perry: A model for generative CAI and concept selection. *International Journal on the Man-Machine Studies*, 8, 1976, 397-410
12. A.Lesgold: Towards a theory of curriculum for use in designing intelligent instructional systems. H.Mandl, A.Lesgold (eds.), *Learning issues for intelligent tutoring systems*, Berlin: Springer-Verlag, 1988
13. D.McArthur et al.: Skill-oriented task sequencing in an intelligent tutor for basic algebra. *Instructional Science*, 17(4), 1988, 281-307
14. G.I.McCalla, et al.: SCENT-3: An architecture for intelligent advising in problem-solving domains. C.Frasson, G.Gauthier (eds.), *Intelligent Tutoring Systems: At the crossroads of artificial intelligence and education*, Norwood: Ablex, 1990
15. J.Raats, A.Tolmacheva: *Adaptive training system for Analytic Functions Differentiation Teaching Automation and Computing Technique*, (4), 1980, 65-69
16. J.Self: Bypassing the intractable problem of student modelling. C.Frasson, G.Gauthier (eds.), *Intelligent Tutoring Systems: At the crossroads of artificial intelligence and education*, Norwood: Ablex, 1990
17. K.M.Swigger, D.Evans: A Computer - Based Tutor for Assembly language. *Journal of Computer-Based Instruction*, 14(1), 1987, 35-38
18. J.Vassileva: An architecture and methodology for creating a domain-independent, plan-based intelligent tutoring system. *Educational and Training Technology International*, 27(4), 1990, 386-397
19. J.Vassileva, R.Radev, B.Dimchev, J.Madjarova: TOBIE: An experimental ICAI-software in mathematics. *Proceedings of International conference on computer-aided learning and instruction in science and engineering*, Lausanne, 1991
20. K.T.Wescourt, M.Beard and L.Gould: Knowledge-based adaptive curriculum sequencing for CAI: Application of a network representation. *Proceedings of the ACM*, 1977