УДК 004.8

А.Г. ЧУХРАЙ

Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Украина

МОДЕЛИ И МЕТОДЫ АДАПТИВНЫХ КОМПЬЮТЕРНЫХ СИСТЕМ ПОДДЕРЖКИ ПРИОБРЕТЕНИЯ ЗНАНИЙ И УМЕНИЙ ПРИ РЕШЕНИИ АЛГОРИТМИЧЕСКИХ УЧЕБНЫХ ЗАДАЧ

На основе подхода к рациональному управлению объектами в нештатных режимах, созданного профессором А.С. Куликом, разработаны модели и методы адаптивных компьютерных систем поддержки приобретения знаний и умений при решении многошаговых алгоритмических учебных задач. Предложены три режима работы таких систем: «демо», решения с подсказками, тестовый, а также два этапа обучения – расчет по алгоритму и написание алгоритма. Описаны модели алгоритмической учебной задачи, метод генерации и расчета объектов, модели обучаемого, метод автоматического построения диагностических моделей пропуска операндов, модель данных, модель обучения, методы диагностирования программ, составленных обучаемым.

Ключевые слова: *алгоритмические учебные задачи*, *адаптивная компьютерная поддержка приобретения знаний умений*

Введение

С позиций системного управления процессами обучение знаниям и умениям в школе или в вузе характеризуется рядом факторов, негативно влияющих на качество подготовки обучаемых. Среди них: возмущающие воздействия, действующие на обучаемых и обучающих, слабая профессиональная и педагогическая подготовка отдельных обучающих, низкий стартовый уровень знаний и умений и недостаточная мотивация отдельных обучаемых. Кроме того, в условиях массового производства традиционное обучение не может быть адаптивным. Это следует из объективно существующей закономерности Миллера или магического числа 7 плюс-минус 2. Так, из-за ограниченности кратковременной памяти учитель (преподаватель) не может адаптивно управлять обучением каждого ученика (студента) в классе (группе) из 20-30 человек. Если учесть, что в рамках каждого учебного предмета количество компонентов знаний и умений, которое необходимо освоить каждому обучаемому, как правило, больше 30, то проблема становится еще острее.

Перспективными средствами решения проблемы являются адаптивные компьютерные системы-поддержки приобретения знаний и умений (АКСППЗУ). Такие системы могут обладать практически неограниченной памятью и иметь встроенные интеллектуальные функции для моделирования деятельности лучших учителей (преподавателей).

В Национальном аэрокосмическом университете им. Н.Е. Жуковского «ХАИ» на кафедре систем управления летательных аппаратов вопросами разработки и внедрения таких систем занимаются с 2004 года. Разработки ведутся в рамках созданного профессором А.С. Куликом подхода к рациональному управлению объектами в нештатных режимах [1].

Рассмотрим научные основы таких систем на примере алгоритмических учебных задач (АУЗ). Каждая такая задача является многошаговой, на каждом шаге расчеты производятся по некоторому алгоритму, в котором задействованы различные компоненты знаний и умений.

Постановка задач. Используя подход к рациональному управлению объектами в нештатных режимах, разработать модели и методы АКСППЗУ.

Модели и методы АКСППЗУ

Системно моделируя эффективную педагогическую деятельность обучающих, получим три режима работы программ: «демо», режим решения с подсказками, тестовый режим.

В режиме «демо» система демонстрирует, как нужно решать задачу, автоматически заполняя соответствующие поля ввода на экранной форме. При этом каждый раз система решает новую задачу, генерируя данные для условия, рассчитывая промежуточные данные и решение. Такой подход должен быть реализован для того, чтобы, во-первых, продемонстрировать особенности решения разных задач, а, во-вторых, для того, чтобы у обучаемого не сложилось впечатление о константном характере тех

или иных данных либо количества определенных операций при решении.

Во втором и третьем режимах задача формулируется программой, а решить ее по шагам должен обучаемый. Отличия заключаются в том, что во втором режиме система помогает обучаемому при необходимости, а в третьем – нет.

Далее рассмотрим первые два режима.

Обобщенную модель АУЗ можно представить упорядоченной n-кой:

ModelOfCT1=(condct1, objct1$_1$, objct1$_2$,…,objct1$_n$), (1)

где condct1 – условие задачи, objct1$_i$– это объект, который представим упорядоченной семеркой компонентов (uid, name, type, value, format, alg, note), где uid– уникальный идентификатор объекта, name – имя объекта, type – тип значения, value – значение, format – формат значения (например, с плавающей точкой), alg –алгоритм расчета значенияvalue, note – обозначение объекта.

Таким образом, для решения АУЗ обучаемый должен сформировать кортеж из n объектов, причем objct1$_{n-l}$, objct1$_{n-l+1}$,…,objct1$_n$– решение задачи, l є {1,2,…, k}, k<n, остальные объекты – промежуточные. Следует отметить, что одно и то же обозначение note может присутствовать у разных, но соседних объектов одной и той же задачи, например, в случае, когда рассчитываются значения коллекции данных (массива, матрицы и т.д.). Кроме того для данных в коллекции также может быть характерен один и тот же алгоритм расчета с параметрами.

Введем множество элементов ввода IE_CT1_D={iect1d$_1$, iect1d$_2$,…, iect1d$_a$} на экранной форме. Также сформируем множество из objct1$_i$ OBJ = {objct1$_1$, objct1$_2$,…,objct1$_n$} обобщенной модели АУЗModelOfCT1 (1). Пусть отображение F1 между множествами OBJ и IE_CT1_D будет сюръективным, инъективным и, следовательно, биективным. В режиме «демо» в каждый элемент iect1d$_j$ множества IE_CT1_D система заносит соответствующее значение objct1$_i$.

Для реализации возможности генерации условия и решения в режиме «демо» необходимо специфицировать отношения между объектами objct1$_i$. Так, каждому objct1$_i$ может предшествовать ноль, один или более objct1$_j$, значения которых должны рассчитываться по заданному алгоритму ранее. Для формализации таких связей воспользуемся теорией графов и построим ориентированный граф G=(E,Γ).

Пусть каждой вершине поставлено в соответствие порядковую функцию, определяемую для графа без контуров.

Определим подмножества $B_0, B_1, B_2, ..., B_r$ для вершин графа G:

$$B_0 = \{X_i \mid X_i \in E, \Gamma^{-1}X_i = \varnothing\},$$

$$B_1 = \{X_i \mid X_i \in E - B_0, \Gamma^{-1}X_i \subset B_0\},$$

$$B_2 = \{X_i \mid X_i \in E - (B_0 \cup B_1), \Gamma^{-1}X_i \subset B_0 \cup B_1\}, \quad (2)$$

...

$$B_r = \{X_i \mid X_i \in E - \bigcup_{k=0}^{r-1} B_k, \Gamma^{-1}X_i \subset \bigcup_{k=0}^{r-1} B_k\},$$

где r – такое наименьшее целое число, что $\forall X_i \in B_r : \Gamma X_i = \varnothing$. Тогда функция $O(X)$, которая определяется выражением $X_i \in B_k \Rightarrow O(X_i) = k$ будет порядковой функцией графа без контуров, а подмножества $B_k, k = \overline{0,r}$, которые образуют разбиение исходного множества вершин графа G, будут уровнями. Для нахождения уровней графа G можно воспользоваться методом Демукрона [2].

Предположим, что объекты 0-го уровня, являющиеся независимыми, должны генерироваться, а объекты всех остальных уровней, кроме 0-го, т.е. зависимые объекты, должны рассчитываться. Тогда можно описать метод для генерации и расчета объектов. В общем виде такой метод будет выглядеть следующим образом.

Для каждого объекта Y 0-го уровня
Начало цикла
Повторять
  Сгенерировать значение V для Y;
До тех пор, пока значение V станет допустимым для Y
Конец цикла
Для каждого уровня Rот 1 доN
Начало цикла
Для каждого объекта X уровня R
Начало цикла
Если значение V для X – не рассчитанное, То
  Рассчитать значение V для X;
Если значение V – недопустимое для X, То
Начало
$N_{попыток}$:=0;
Повторять
КонъюнктивнаяДопустимость:=Правда;
  Выбрать случайно объект 0-го уровня Y, от которого зависит X;
  Сгенерировать для Y новое значение;
  Рассчитать значения всех зависимых от Y объектов и накопить значение КонъюнктивнаяДопустимость:=
    КонъюнктивнаяДопустимость^ Допустимость(зависимого объекта);
$N_{попыток}$:=$N_{попыток}$+1;
До тех пор, покаКонъюнктивнаяДопустимость или$N_{попыток}$=$N_{вых}$
Конец
Конец цикла
Конец цикла

Представим более детально часть метода, связанную с циклом До тех пор, пока Конъюнктивная Допустимость или $N_{попыток}=N_{вых}$.

1) Выбрать случайно объект 0-го уровня Y, от которого зависит X. Сформируем A – множество объектов 0-го уровня Y, от которых зависит X как непосредственно, так и транзитивно:

$$\text{À} = \left\{ Y \middle| Y \in B_0, X \subset \Gamma^g Y \right\},$$

$g \in \overline{1,N}, \Gamma^1 Y = \Gamma Y, \Gamma^2 Y = \Gamma\{\Gamma Y\}, \Gamma^3 Y = \Gamma\{\Gamma\{\Gamma Y\}\}, \ldots$. Теперь из множества A необходимо случайно выбрать один объект: Y=random(A).

2) Сгенерировать для Y новое значение $\Pi_{value}Y$ по алгоритму $\Pi_{alg}Y$.

3) Рассчитать значения всех зависимых от Y объектов, т.е. $\forall z \in Z$ рассчитать $\Pi_{value}z$ по алгоритмам $\Pi_{alg}z$ и Конъюнктивная Допустимость:= Конъюнктивная Допустимость ^ Допустимость(z).

Следует отметить, что метод генерации и расчета объектов должен применяться до демонстрации программой решения задачи для установления уверенности в том, что значение для каждого объекта рассчитано, и оно является допустимым, т.е. задача имеет решение. В случае же использования метода во время демонстрации, программа может прийти в состояние фрустрации и к необходимости начинать демонстрацию заново, породив у обучаемого сомнения в своей способности обучать.

В режиме «демо» также можно реализовать возможность объяснения тех или иных шагов, сделанных программой. Для этого программа может экипироваться кнопкой «Объясни», при нажатии на которую раскрываются особенности расчета некоторого значения. Для этого из упорядоченного графа объектов G выбираются значения всех объектов, от которых непосредственно зависит данный объект, т.е. необходимо для объекта Yвыбрать все такие объекты, которые составляют множество $\Gamma^{-1}Y$. Кроме значений каждого объекта, принадлежащего множеству $\Gamma^{-1}Y$, также программа может выдать на экран алгоритм расчета значения Y.

Модель обучаемого для АУЗ в режиме «демо».

В режиме «демо» для АУЗ в модели обучаемого можно собрать некоторые данные, привязанные к уникальному коду обучаемого. Для каждого типа задачи обучаемый может осуществить несколько попыток просмотра демонстрации решения. Для каждой попытки следует сохранить общее время управляемой обучаемым демонстрации.

Пусть обучаемый управляет демонстрацией решения задачи посредством кнопки «Следующий шаг» и, уже указанной выше, кнопки «Объясни». Условимся, что нажатие на кнопку «Следующий шаг» возможно только тогда, когда программа закончила ввод значения в предыдущий элемент ввода $iect1d_j$ и обучаемый осмыслил этот шаг. При нажатии этой кнопки программа заносит значение в следующий элемент ввода $iect1d_k$. Нажатие на кнопку «Объясни» связано с активным элементом ввода, в который уже было занесено значение, что, таким образом, определяет возможность ее нажатия в интервале между двумя последовательными нажатиями кнопки «Следующий шаг», при этом ее действие распространяется только на текущий шаг, заключенный в этом интервале. Временная последовательность нажатия кнопок может быть представлена графически на рис. 1.

Рис. 1. Временная последовательность нажатия кнопок обучаемым в режиме «демо»

Как показано на рис. 1 текущий шаг программа выполняет с момента времени t_j до момента времени t_{j+1}, заполняя соответствующий элемент ввода. Именно к этому текущему шагу и относится нажатие кнопки «Объясни» в момент времени t_{j+2}. С момента времени t_{j+1} до момента времени t_{j+3} происходит осмысление обучаемым текущего шага, которое может быть дополнено объяснением программой этого шага с момента времени t_{j+2} до момента времени t_{j+3}.

Таким образом, в модели обучаемого фиксируются все указанные моменты времени с соответствующими ассоциациями: для любого обучаемого для любого класса АУЗ для любой попытки обучаемого освоить этот класс задач в рамках режима «демо» для любого объекта, храним кортеж $T_i=(t_j, t_{j+1},$

t_{j+2}, t_{j+3}). Если шаг понятен обучаемому и без дополнительной помощи, тогда вместо t_{j+2} заносим специальное значение, например, NULL. Следует учесть, что если мы будем хранить два кортежа времен, связанные с двумя последовательными шагами, то в этом случае последний элемент первого кортежа будет равен первому элементу второго кортежа, т.е. присутствует некоторая избыточность. Тем не менее, последний элемент в таком кортеже нужен для уникального хранения времени окончания демонстрации, а первый – для уникального хранения времени ее начала.

В дальнейшем, исходя из накопленных кортежей T_i можно выявлять обучаемых, которые не работают, а играются с программой (это будет в том случае, если $(t_{j+3} - t_j < \sigma_{i1}) \wedge (t_{j+3} - t_{j+2} < \sigma_{i2})$, где σ_{i1}, σ_{i2} – некоторые пороги для так называемого "гейминга"), обучаемых, которые не мотивированы к самостоятельным размышлениям (при $(t_{j+2} - t_{j+1} < \sigma_{iz}) \wedge (t_{j+3} - t_{j+2} >> t_{j+2} - t_{j+1})$,где σ_{iz} – порог несамостоятельности). Последнюю формулу можно переписать в виде $(t_{j+2} - t_{j+1} < \sigma_{iz}) \wedge \frac{r_{j-3} - r_{j-2}}{r_{j-2} - r_{j-1}} > \sigma_{i4}$, где σ_{i4} – порог для отношения несамостоятельности обучаемого к его самостоятельности.

Естественно, что все вышеуказанные пороги должны определяться экспериментально, например, на основании протоколирования событий при прохождении программы отличником. Кроме того, эти выводы могут быть сделаны только в том случае, если обучаемый во время демонстрации решения не теряет концентрации внимания и не отвлекается. Это можно отследить с помощью видеокамеры и специальных алгоритмов распознавания.

Модель задачи для АУЗ в режиме «решения с подсказками».

Модель задачи для АУЗ в режиме «решения с подсказками» строится на основе модели АУЗ в режиме «демо». В дополнение к модели АУЗ в режиме «демо» здесь совершается параллельный расчет: как обучаемый, так и программа рассчитывают значения объектов $object1_i$. Затем для каждого объекта эти значения сравниваются. В случае несовпадения должны быть выяснены причины такого несовпадения, чтобы определить адаптивную обучающую последовательность для конкретного обучаемого. Например, один обучаемый может неправильно округлять числа, второй – не знать, что в правильной формуле необходим множитель «двойка» в числителе, третий – потерять знак при вычислении и так далее. Соответственно первому обучаемому необходимы дополнительные знания и умения по округлению чисел, второму требуется запомнить правильную формулу, третьему – простая подсказка, о том, что вычисления – правильные, но потерян знак «минус».

Итак, причины – пробелы в знаниях или умениях обучаемого, а также неправильные знания или неправильные умения, следствия – неправильные результаты его расчетов в рамках конкретной задачи. Необходимо решить обратную задачу: по следствию найти причину с тем, чтобы отразить ее в модели обучаемого и выбрать правильную обучающую последовательность.

Особенности решения обратных задач в области интеллектуальных компьютерных обучающих программ следующие:

1. Заранее неизвестно множество причин – пробелов в знаниях или умениях, а также неправильных знаний или умений. Для каждого обучаемого они могут быть свои, особенные.

2. Множество следствий – бесконечное при решении обучаемым задачи на бумаге, а при использовании компьютерной программы ограничено лишь количеством различных значений типа данных в конкретном элементе ввода.

3. Одному и тому же следствию может соответствовать множество причин. Например, некоторое число $\widetilde{X}$, не равное эталонному значению $\widehat{X}$, может быть получено как в результате неправильного расчета, так и в результате правильного расчета, но неправильного последующего округления.

4. В одном и том же вычислении может быть совершено сразу несколько ошибок. Потенциально, каждое исходное значение и каждая операция могут быть ошибочными.

5. Даже, если обучаемый не знает и не умеет, он может правильно рассчитать, угадав или воспользовавшись подсказкой товарища[3].

6. Зная и умея, обучаемый может допустить ошибку из-за невнимательности.

Поэтому, необходим собственный путь решения обратных задач для выбранной предметной области. И первым шагом такого пути должны стать исследования, нацеленные на выявление причин и следствий ошибок обучаемых, допускаемых ими при решении конкретных классов задач.

Известны результаты исследований зарубежных ученых, описывающие наиболее распространенные ошибки, допускаемые обучаемыми в математических вычислениях. Например, в работе [4] представлена таксономия ошибочных представлений школьников о десятичных дробях и действиях над ними. Наряду с использованием результатов этих исследований необходимо провести собственные эксперименты, для выявления ошибок, наиболее типичных для рассматриваемых классов задач, и их причин. Такие эксперименты позволят построить специализированное математическое обеспечение,

которое будет служить для адекватного реагирования компьютерных программ на ошибки обучаемых и последующего адаптивного обучения. Вместе с тем, понятно, что невозможно, таким образом, проанализировать действия при решении какой-либо задачи всей генеральной совокупности обучаемых, ведь некоторые еще не родились, поэтому речь может быть только о наиболее распространенных ошибках. Тем не менее, компьютерные обучающие программы должны быть открытыми для внесения новых классов ошибок, а часть классов ошибок может быть получена автоматически, например, для случаев пропуска обучаемым операндов при вычислениях по формулам.

Наиболее распространенные ошибки студентов, допускаемые ими при решении АУЗ в предметной области «Теория автоматического управления», а также соответствующие диагностические модели описаны в работах [5,6]..

Метод автоматического построения диагностических моделей пропуска операндов

Условимся в рамках данного метода рассматривать наиболее распространенные бинарные операторы из множества Operators={'+','–','*','/','^'}. Это связано с тем, что с помощью именно этих операторов решаются большинство математических задач. Кроме того, к этим операторам могут быть сведены и другие, не бинарные операторы. Например, унарный минус легко может быть сведен к бинарному оператору минус, поскольку –a=0–a.

Результаты приведения пропусков, как левого, так и правого операндов к бинарным операторам для всех пяти исходных операторов приведены в табл.1.

Таблица 1

Результаты пропуска операндов при вычислениях

Выражение	Пропуск левого операнда	Пропуск правого операнда
x/y	1/y	x/1
a^b	1^b	a^1
c*d	1*d	c*1
f+g	0+g	f+0
k–z	0–z	k–0

Следует отметить, что операнды, приведенные в таблице, могут быть как простыми, так и составными. Тогда, метод автоматического построения диагностических моделей пропуска операндов должен для каждой возможной формулы, записанной с помощью выделенных пяти операторов, построить все такие возможные формулы, в которых для каждого оператора либо левый, либо правый операнд пропущен, причем ошибка – однократная.

Разработанный метод состоит из двух частей:

1) Формула переводится с помощью метода "сортировочная станция" Э. Дейкстры[7] в обратную польскую запись.

2) С помощью модифицированного вычисления значений в обратной польской записи происходит формирование требуемого множества диагностических моделей.

Рассмотрим вторую часть метода.

Пусть s – строка, которой соответствует формула в обратной польской записи. Например, для формулы 1–2^(g*5) в инфиксной записи соответствующая строка в постфиксной записи будет иметь вид s='1 2 g 5 * ^ –'.

Пусть lex=(lex_1, lex_2,…, lex_n) – это лексемы, выделенные из строки s. Например, lex=('1','2','g','5','*','^','–'). Пусть pos_lex =(pos_lex_1, pos_lex_2,…, pos_lex_n) – это позиции лексем из кортежа lex в исходной строке s. Для приведенного примера pos_lex = (1, 3, 5, 7, 9, 11, 13). Пусть pos_operat=(pos_operat_1, pos_operat_2,…, pos_operat_m) – позиции операторов в строке s. В нашем случае, pos_operat=(9, 11, 13). Тогда формирование требуемого множества диагностических моделей будет производиться по следующему методу.

```
i:=1;
Повторять
Начало цикла
Поз_иск_тек_операт:=pos_operat_i;
Длядот 1 доn
Начало цикла
Тек_лекс:= lex_j;
ЕслиТек_лекс∈Operators, То
Начало
    Поместить в Стек_иск_слеваТек_лекс;
    Поместить в Стек_иск_справаТек_лекс;
Конец
Иначе
Начало
Операнд_правый_иск_слева:=извлечь
Стек_иск_слева;
Операнд_левый_иск_слева:=извлечь
Стек_иск_слева;
Операнд_правый_иск_справа:=извлечь
Стек_иск_справа;
Операнд_левый_иск_справа:= извлечь
Стек_иск_справа;
ЕслиПоз_тек_лекс= Поз_иск_тек_операт,
То
   ВыборТек_лексИз
     '+', '–':
Начало
  Рез-т_иск_слева:='0'+Тек_лекс+
Операнд_правый_иск_слева;
  Рез-т_иск_справа:= Операнд_левый_
```

```
    иск_справа+Тек_лекс+’0’;
    Конец
        ‘*’,’/’,’^’:
    Начало
      Рез-т_иск_слева:=’1’+ Тек_лекс+
      Операнд_правый_иск_слева;
     Рез-т_иск_справа:= Операнд_левый_
    иск_справа+Тек_лекс+’1’;
    Конец
Конец выбора
    Иначе
    Начало
Рез-т_иск_слева:= Операнд_левый_
иск_слева + Тек_лекс+
                Операнд_правый_иск_слева;
Рез-т_иск_справа:= Операнд_левый_
иск_справа+Тек_лекс+
                Операнд_правый_иск_справа;
    Конец
    Поместить в Стек_иск_слева ‘(‘+
    Рез-т_иск_слева+’)’;
    Поместить в Стек_иск_справа ‘(‘+
    Рез-т_иск_справа+’)’;
    Конец
    Конец цикла
    ДМ_левая[i]:=извлечь Стек_иск_слева;
    ДМ_правая[i]:= извлечь Стек_иск_справа;
    i:=i+1;
    Конец цикла
    До тех пор, покаi=m+1;
```

Отметим, что возможны и кратные ошибки пропуска. Тем не менее, такие ошибки носят единичный характер, даже для не очень сложных формул с учетом многократных ошибок получается большое количество вариантов расчета, что значительно усложняет диагностическое обеспечение и, кроме того, вероятность двукратной ошибки можно считать как произведение вероятностей однократных ошибок, что меньше вероятности однократной ошибки. Поэтому диагностическое обеспечение для многократных ошибок разрабатывать нецелесообразно.

Выступая партнером (а не строгим экзаменатором) обучаемого в режиме «обучения с подсказками», в случае срабатывания одновременно нескольких различных диагностических моделей компьютерная программа должна попросить у обучаемого уточнить диагноз, предлагая ему несколько вариантов расчета неправильного значения по разным диагностическим моделям, а также возможность ввести собственный, не совпадающий с предлагаемыми вариант расчета, что послужит сигналом для поиска новой диагностической модели.

Предложение не согласиться и ввести свой вариант расчета должно формулироваться программой и в том случае, когда сработала единственная диагностическая модель, поскольку не исключено, что адекватная диагностическая модель для данного случая еще не построена.

Модель обучаемогодля АУЗ в режиме «решение с подсказками».

Сформулируем некоторые предположения относительно того, какой должна быть модель обучаемого.

Компоненты знаний и умений можно разделить на глобальные, не зависящие от конкретной задачи, и локальные, зависящие от задачи. Например, умение округлять не зависит от конкретной задачи, а знание конкретной формулы для расчета действительных корней по методу Лобачевского-Греффе-Данделена к задаче привязано. Компоненты знаний и умений могут составлять иерархии. Например, умение решать алгебраическое уравнение n-го порядка включает в себя умение рассчитывать коэффициент матрицы А, умение применять условие окончания квадрирования коэффициентов матрицы, умение рассчитывать действительные корни, умение рассчитывать комплексные корни. В свою очередь, умение рассчитывать коэффициент матрицы включает в себя умение определять исходные данные для расчета коэффициента, умение возводить в квадрат, умение умножать, умение вычитать, умение округлять.

Таким образом, выделение тех или иных компонентов знаний и умений и их иерархий целиком определяется сущностью изучаемых методов расчета.

Очевидно, что именно компоненты знаний и умений играют центральную роль в модели обучаемого. Именно от их значений зависит адаптивная обучающая последовательность для конкретного обучаемого. Вместе с тем возникает еще одна обратная задача, как по результатам работы обучаемого с программой определить значения компонентов знаний или какая должна быть связь между шагами метода – объектами $object1_i$, значения которых формирует обучаемый, и компонентами знаний и умений.

При этом нужно выбрать общий подход к моделированию обучаемого, который наложит ограничения, в том числе на типы значений компонентов знаний. Ясно, что в рамках подхода должны предлагаться инструментальные средства борьбы с неопределенностью при вводе обучаемым неправильных значений или чередовании его правильных и неправильных расчетов. Если обучаемый сделал правильный шаг, тогда все равно существует некоторая уверенность в том, что он угадал, либо ему подсказали. Если же обучаемый сделал неправильный шаг, необходимо оставлять долю уверенности в том, что он

владеет компонентом умений, а в данный момент ошибся из-за невнимательности. Модель обучаемого также должна учитывать результаты срабатывания диагностических моделей.

Рассмотрим возможные подходы к моделированию обучаемого с учетом неопределенности. Среди них: а) вероятностный подход; б) методы на основе правил; в) теорию Демпстера-Шефера; г) нечеткие множества и нечеткую логику [8].

Наиболее приемлемым подходом в силу его разработанности, наименьшего по сравнению с другими подходами числа недостатков, а также широты использования в научных целях будем считать вероятностный подход для моделирования обучаемого.

Полную картину для некоторой предметной области с неопределенностями можно получить, зная полное совместное распределение вероятностей. Вместе с тем, по мере увеличения количества переменных, оно может приобретать настолько большие размеры, что вычисления становятся невозможными. Более того, затруднительным может быть сам способ задания вероятностей для атомарных событий при отсутствии большого объема статистических данных. Уменьшить размер задачи, сократив количество вероятностей, которые должны быть заданы в целях определения полного совместного распределения, и облегчить вычисления можно с помощью введения связей независимости и условной независимости между переменными. Для этого вводятся специальные структуры, которые называются байесовскими сетями[8].

В известной работе американского исследователя Курта ВанЛена предлагается использовать не одну байесовскую сеть, а их совокупность для моделирования обучаемого [9]. Но поскольку мы владеем дополнительной информацией о компонентах знаний и умений обучаемых в виде множества сработавших диагностических моделей такая совокупность байесовских сетей может быть представлена рис. 2.

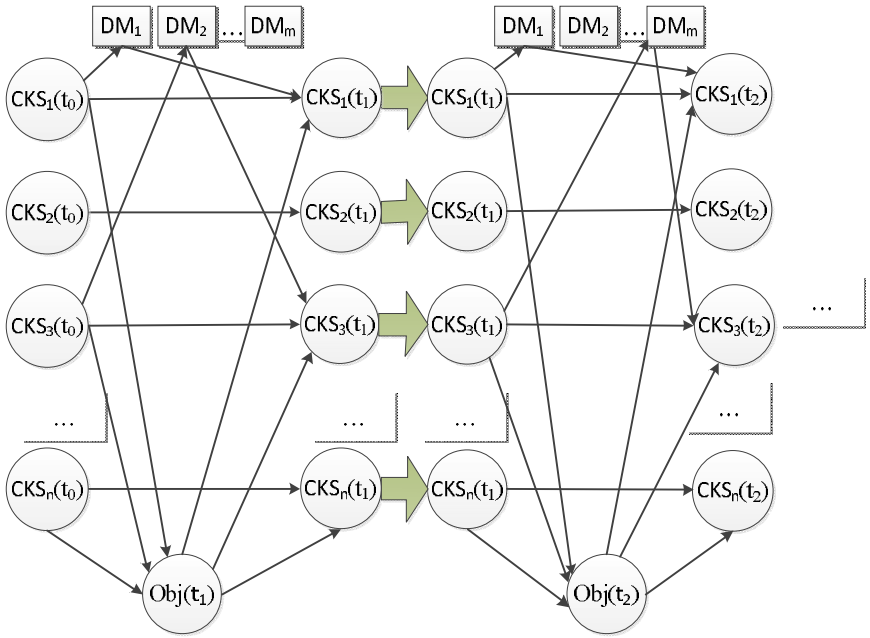

Рис. 2. Совокупность байесовских сетей
для моделирования преобразования знаний и умений обучаемого

На рис. 2 обозначены CKSi(tk) – i-й компонент знаний и умений в момент времени tk, Obj(ts) – объект, введенный обучаемым в систему в момент времени ts, DMj – j-тая диагностическая модель. Отметим, что не все компоненты знаний и умений, а также не все диагностические модели могут участвовать в связях с Obj(ts). Их совокупности определяются конкретной расчетной задачей и шагом в ее решении в момент времени ts.

При добавлении задачи и ее компонентов знаний и умений в систему следует проверить, существуют ли такие компоненты для ранее добавленных задач с тем, чтобы использовать в качестве априорных вероятностей владения тем или иным компонентом новой задачи апостериорные вероятности для задач, которые обучаемый уже решал. Поскольку компоненты знаний и умений отличаются друг от друга названиями и значениями вероятностей для Mastered (P(Nomastered)=1-P(Mastered)), то имеет смысл осуществлять поиск похожих компонентов знаний и умений на заданный по названию во избежание дубликатов и квазидубликатов компонентов в системе. Из-за возможных ошибок оператора, а также из-за возможной перестановки слов или исполь-

зуемых аббревиатур следует использовать одновременно несколько метрик для похожих строк. Например, с наиболее распространенными ошибками оператора справляется метрика В.И. Левенштейна [10] – минимальное расстояние редактирования между строками, с перестановками слов – метрика q-грамм – отношение количества общих q-грамм к количеству q-грамм в строке меньшей (большей либо средней) длины [11], для нахождения похожих строк на заданную с использованием и искажением аббревиатур можно использовать метрику, разработанную в трудах [12,13].

Для быстрого поиска строк похожих, на заданную можно использовать метод NearestHash [14] и его производные [13].

Прежде чем ответить на вопрос, как состыковуются между собой компоненты знаний и умений для разных задач, решаемых обучаемым, построим модель данных для сущностей «Обучаемый», «Задача», «Компонент знаний и умений», «Шаг», «Решение задач».

Вербальная модель данных

Пусть дано множество обучаемыхLearners. Каждый обучаемый характеризуется такими атрибутами, как уникальный идентификатор (id_learner), Фамилия Имя Отчество (name), Дата рождения (birthday), адрес (address), электронный адрес (email), мобильный телефон (mobile). Пусть дано множество задач Tasks. Каждая сущность «Задача» включает уникальный идентификатор (id_task) и наименование (name). Дано множество компонентов знаний и умений Components. Каждый компонент имеет уникальный идентификатор (id_cks) и имя (name). Дано множество шагов решения задач Steps с атрибутами уникальный идентификатор (id_step) и наименование (name). На каждом шаге решения задачи содержится множество компонентов знаний и умений. Обучаемый решает задачи, делая шаги и овладевая некоторыми компонентами знаний и умений. Для любого обучаемого для любой задачи на каждом шаге для каждого компонента знаний и умений проставляется соответствующее значение вероятности овладения этим компонентом знаний и умений (p), а также время фиксации шага (t). Кроме того, для каждого обучаемого и для каждой задачи фиксируется номер попытки решения такой задачи (num_try).

Графическая модель данных представлена на рис. 3.

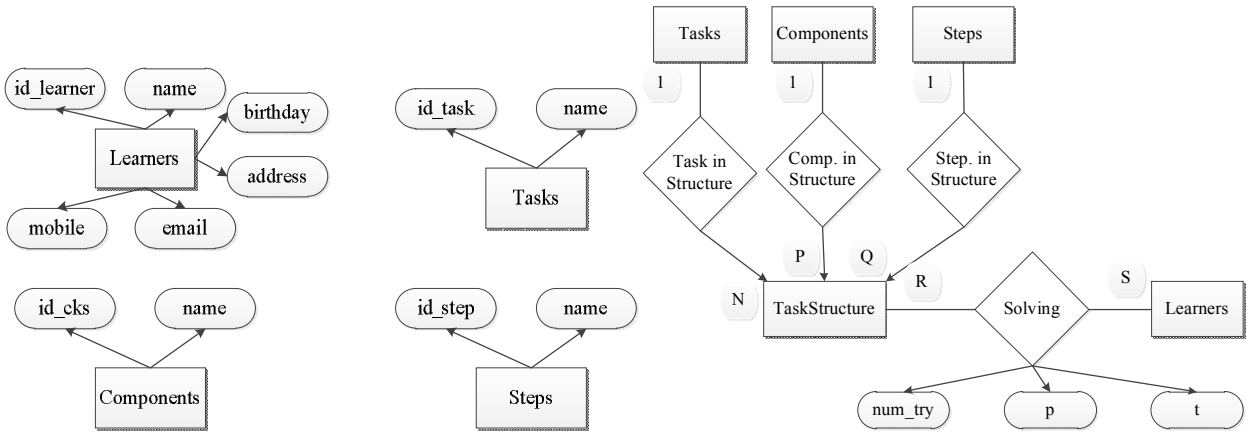

Рис. 3. Модель данных

На физическом уровне модель данных состоит из следующих таблиц: Learners (id_learner, name, birthday, address, email, mobile), Tasks(id_task, name), Steps(id_step,name), Components(id_cks,name), TaskStructure(id_task, id_step, id_cks), Solving(id_learner, id_task, id_step, id_cks, num_try, p, t). Кроме того сформируем два представления (view):

1) Tasks_components – компоненты знаний и умений, содержащиеся в задачах, – SELECTDISTINCTid_task,id_cksFROMTaskStructure;

2) Tasks_steps – шаги задач – SELECT DISTINCT id_task,id_step FROM TaskStructure.

Имея такую модель данных теперь можно отвечать на следующие вопросы, как, например, выбрать последнее по времени значение вероятности овладения конкретным компонентом знаний и умений для конкретного обучаемого или выбрать задачи, в которых есть один новый для конкретного обучаемого компонент знаний и умений или выбрать такие новые для обучаемого задачи, в которых нет новых для него компонентов знаний и умений.

Модель обучения для АУЗ в режиме «решение с подсказками».

Выбор обучающего действия должен производиться на основе информации, получаемой из модели обучаемого, и текущего шага, сделанного обучаемым при решении і-й задачи.

Рассмотрим различные условия, предшествующие выбору обучающего действия:

1. При решении j-й задачи обучаемый сделал неправильный i-й шаг. В этом случае возможны два варианта действия:

a. подсказка обучаемому на основании активизированных диагностических моделей;

b. переход к более простой задаче, содержащей проблемные компоненты знаний и умений.

2. Задача решена обучаемым правильно. Тогда возможны тоже два варианта развития событий:

a. переход к следующей задаче того же класса;

b. переход к задаче другого класса.

Все указанные переходы к следующей задаче могут осуществляться двумя способами:

a. задача выбирается программой;

b. задача выбирается обучаемым.

Итого получается семь различных сценариев выбора дальнейшего обучения. Рассмотрим подробнее каждый сценарий автоматического выбора.

В случае неправильного i-го шага обучаемого при решении им j-й задачи и последующей подсказки на основании активизированных диагностических моделей происходит следующее: в модель обучаемого после байесовской сети для i-го шага вставляется столько временных слоев – байесовских сетей, сколько будет сделано неправильных шагов. При этом если представить первый слева такой слой как граф G=(E,Г), то все слои дубликаты будут его полностью повторять и по вершинам и по дугам, т.е. будут изоморфными исходному графу, при этом априорные значения вероятностей овладения компонентами знаний и умений будут апостериорными значениями вероятностей для предыдущего слоя.

На рис. 4 приведен пример совокупности байесовских сетей для случая когда объект О1 введен правильно с s+1 попытки и после этого обучаемый перешел на следующий шаг – ввод объекта О2.

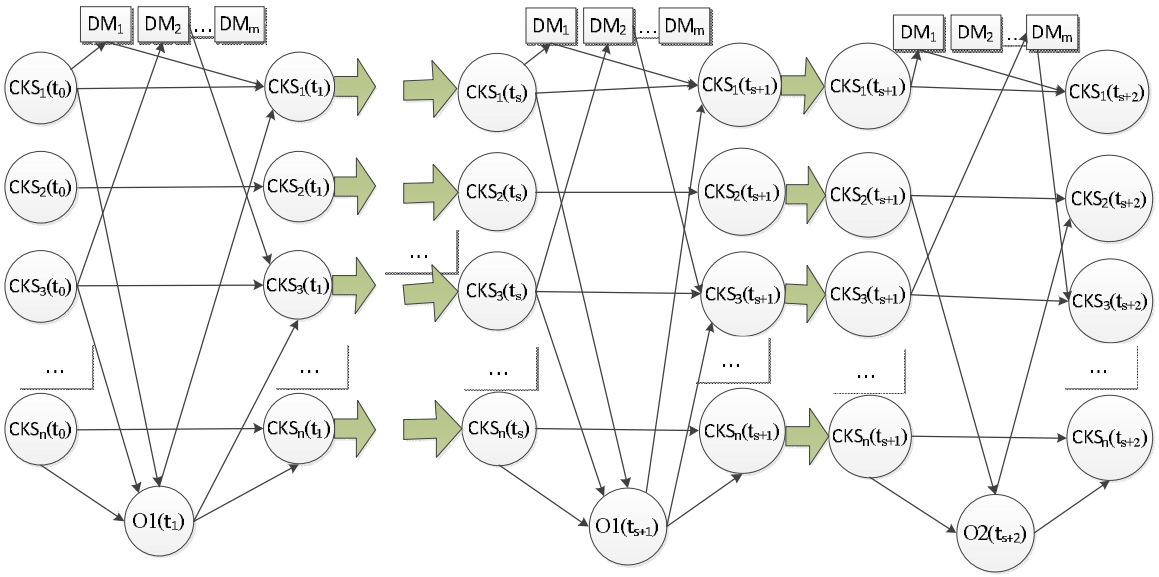

Рис. 4. Модель обучаемого в случае его неправильных шагов

Ясно, что в случае ряда неправильных повторных вводов одного и того же объекта вероятности овладения соответствующих компонентов знаний и умений должны монотонно снижаться.

1. При решении j-й задачи обучаемый сделал i-й неправильный шаг. При этом осуществляется автоматический переход к более простой задаче, содержащей проблемные компоненты знаний и умений. Этот переход может быть сделан к задаче, в которой содержится один проблемный компонент либо к любой задаче, содержащей меньшее количество компонентов знаний и умений, чем в текущей задаче. Кроме того, может быть осуществлен переход к задачам, содержащим меньше проблемных для конкретного обучаемого компонентов знаний и умений, чем в текущей задаче. Каждый такой переход может быть выбран посредством единственного SQL-запроса.

2. Задача решена обучаемым правильно и осуществляется переход к следующей задаче того же класса. Это может быть сделано с помощью метода генерации и расчета объектов. Поскольку при работе с классом задач компоненты знаний и умений одни и те же, то априорные значения вероятностей их овладения будут апостериорными вероятностями для последней решенной задачи этого класса. Как показали проведенные расчеты в случае безошибочного решения ряда задач одного и того же

класса значения вероятностей овладения соответствующими компонентами знаний и умений будут возрастать и достигнут некоторого порога «мастерства овладения», например 0.95.

3. Задача решена обучаемым правильно. Осуществляется переход к задаче другого класса. Такая задача – это:

a) новая задача, в которой присутствует хотя бы один новый компонент знаний и умений для конкретного обучаемого либо

b) новая задача, в которой присутствуют в другой комбинации только те компоненты знаний и умений, которые обучаемый уже проходил. Другими словами, необходимо выбрать такие новые для обучаемого задачи, в которых нет новых для него компонентов знаний и умений.

Эти переходы также легко формируются с помощью SQL-запросов.

Альтернативным сценарием обучения может быть сценарий с использованием кластеризации задач. Задачи можно сгруппировать по кластерам так, чтобы в каждом кластере располагались похожие друг на друга задачи, а кластеры упорядочить по сложности, например, по среднему количеству компонентов знаний и умений в задачах кластера или же по средней сложности задач, где сложность задачи определяется как суммарная сложность составляющих ее компонентов знаний и умений. В таком случае обучаемый должен начинать обучение с самого легкого по сложности кластера и не выходить из него до тех пор, пока не будут решены все задачи в этом кластере. В качестве представления задачи для кластеризации можно использовать кортеж $(x_1, x_2,\ldots, x_n)$, где n – общее количество компонентов знаний и умений в предметной области, $x_i \epsilon \{0,1\}$ – компонент кортежа, указывающий на присутствие или отсутствие i-го компонента знаний и умений в задаче, в качестве метрики для сравнения различных кортежей – метрику Хэмминга, а для кластеризации задач – один из известных методов кластеризации, например, k-means [15]. Но поскольку один из главных недостатков метода k-means заключается в том, что нужно заранее знать число кластеров, то для решения задачи кластеризации можно использовать и другие методы, например, модифицированный метод NearestHash. Модификация NearestHash заключается в том, что для объединения похожих записей используется специальная структура данных – система непересекающихся множеств [16].

Кроме того, каждой задаче можно поставить в соответствие иерархию компонентов, а каждой иерархии ее строковое представление. Например, «(a(b(e)(f))(c)(d(g)))» соответствует иерархии, представленной на рис. 5.

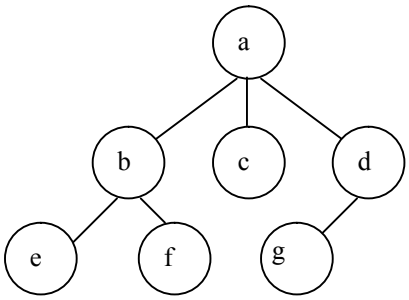

Рис. 5. Пример иерархии

Тогда для кластеризации задач можно использовать NearestHash с системой непересекающихся множеств и метрикой – минимальное расстояние редактирования между деревьями [17].

Поскольку при решении рассмотренных выше расчетных задач для осуществления расчетов используются формулы и алгоритмы, которые покрывают не одну конкретную задачу, а целый класс задач, то необходимо отдельно обучать знанию этих формул и алгоритмов. Тогда можно быть уверенным, что учащийся решит любую задачу из заданного класса. Проведем такую аналогию. В настоящее время нет искусственных моделей человеческого разума лучше, чем нейронные сети. Искусственная нейронная сеть обучается на примерах, но при этом нет гарантии того, что она правильно сработает при любых исходных данных за пределами обучающей выборки. Точно также и при обучении человека только лишь на примерах нет гарантии того, что им усвоено общее правило, алгоритм либо формула, покрывающие все возможные случаи. Поэтому закономерна постановка научной задачи обучения алгоритмам.

Для ее решения можно воспользоваться методом автоматической проверки правильности построенного учащимся алгоритма, который описан в работе [18]. Программа, написанная обучаемым, может в точности совпадать с эталонной программой, например, в случае небольших задач. Тогда, если тексты этих двух программ совпадают, то тестирование не нужно. Если же тексты не совпадают, то необходимо тестирование. Общий классификатор для программы обучаемого может быть представлен рис. 6. Оптимальность программ можно оценивать по таким критериям, как скорость работы, используемая память, лаконизм. Если программа обучаемого, прошедшая через тестирование превосходит существующие эталонные программы (а для одной и той же задачи можно составить целый ряд эталонных программ), то система может внести эту программу как новую оптимальную.

Таким образом, реализуется самообучение системы.

Рис. 6. Классификатор программ обучаемого

Если программа обучаемого не проходит хотя бы один тест, тогда решаем следующую диагностическую задачу – поиск места несовпадения.

Считаем каждую программу (эталонную и написанную обучаемым) в отдельную строку. Тогда можно рассчитать расстояние Левенштейна – минимальное расстояние редактирования между строками и след – необходимые операции для превращения одной строки в другую.

Известно что время расчета расстояния Левенштейна для двух строк по методу Вагнера-Фишера [19] и используемая при этом память могут быть записаны как O(mn), где m– длина первой строки, n – длина второй строки.

Тем не менее, проведем вычислительные эксперименты, чтобы понять сегодняшние ограничения.

При проведении экспериментов использовался компьютер IntelCoreI5-3210, 2.5GHz, ОЗУ – 4 Гб. Программа на языке Java запускалась в среде Eclipse SDK, Version: 3.5.1.

Среднее время расчета в зависимости от размера файла (примем, что файлы совпадают по размеру) представлено в табл. 2.

Таблица 2

Зависимость времени расчета от размера файлов

Размер файла	Среднее время расчета
28байт	443109 нс
2895 байт	135536867 нс = 0,1 сек
7251 байт	1250641880 нс =1,25 сек

Для размера файла 10701 байт – сообщение об ошибке при выполнении из-за переполнения памяти.

Пусть обе программы – эталонная и реальная – прошли лексический анализ и для каждой хранится массив лексем. Тогда можно ввести другую метрику – минимальное расстояние редактирование между массивами лексем двух программ. В этом случае для файла из 28 байт среднее время расчета составило 386295 нс.

Таким образом, объем памяти и объем вычислений значительно сокращается. Кроме того, из такого следа можно получить больше информации о самой программе, а не о составляющих ее символах.

Также можно получить не только информацию о пропущенных, неправильных или лишних лексемах, но и диагностическую информацию структурного характера. Для этого нужно сопоставить два абстрактных синтаксических дерева (АСД), полученных в результате синтаксического разбора эталонной программы и программы, составленной обучаемым. Первый способ сопоставления АСД – использование метода поиска в ширину, а второй – вычисление расстояния между деревьями по метрике Shasha-Zhang [17] и следа редактирования. При этом узлами деревьев должны быть не символы, а операнды и операторы. Таким образом, можно вначале указать обучаемому, что его ошибка – в таком то блоке программы.

Вывод

На основе разработанных моделей и методов создано и успешно эксплуатируется на кафедре программное обеспечение для обучения студентов [20-22]. Кроме того, создано и передано в эксплуатацию программное обеспечение по заказу Харьковского регионального центра оценивания качества

образования [23], университета Прикладных наук Верхней Австрии [24] и Британского центра инноваций в обучении математике [25, 26].

Дальнейшие исследования будут нацелены на разработку моделей и методов адаптивной компьютерной поддержки приобретения знаний и умений при решении задач, требующих рассуждений.

Литература

1. Кулик, А.С. О научной школе рационального управления объектами в нештатных режимах [Текст] / А.С. Кулик // Авиационно-космическая техника и технология. – 2010. – № 2 (69). – С. 20 – 26.

2. Сироджа, И.Б. Комбинаторика и теория графов [Текст]: учебное пособие по курсу «Основы дискретной математики»/ И.Б. Сироджа. – Х.:ХАИ. – 1998. – 197 с.

3. Baker, R.S. More Accurate Student Modeling Through Contextual Estimation of Slip and Guess Probabilities in Bayesian Knowledge Tracing [Текст] / R.S. Baker, A.T. Corbett, V. Aleven // Proceedings of the 9th International Conference on Intelligent Tutoring Systems. – 2008. – P. 406 – 415.

4. Isotani, S. Towards intelligent tutoring with erroneous examples: A taxonomy of decimal misconceptions [Текст] / S. Isotani, B. McLaren, M. Altman, // Proceedings of the 10th International Conference on Intelligent Tutoring Systems. Lecture Notes in Computer Science. – 6094. – Berlin: Springer, 2010. – P. 346 – 348.

5. Kulik, A. Diagnostic models of intelligent tutor system for teaching skills to solve algebraic equations [Электронныйресурс] / A. Kulik, A. Chukhray, M. Chukhray // International Journal of Emerging Technologies in Learning. – Vol 2, No 1. – 2007. – Режим доступа: http://online-journals.org/i-jet/article/view/68.–9.04.2013 г.

6. Kulik, A. Diagnostic models of intelligent tutor system for teaching skills to construct control object frequency characteristics [Электронный ресурс]/ A.Kulik, A.Chukhray // Proceedings of Interactive Aided Learning Conference.– 2007, Villach, Austria. – Режим доступа: http://www.icl-conference.org/dl/proceedings/2007/paper/52_Final_Paper.pdf.– 9.04.2013 г.

7. Dijkstra, E.W. Algol Bulletin, Supplement № 10, 1961 [Электронный ресурс] / E.W. Dijkstra. – Режим доступа: http://www.cs.utexas.edu /~EWD/MCReps/MR35.PDF.– 9.04.2013 г.

8. Рассел, С. Искусственный интеллект. Современный подход [Текст] / С. Рассел, П. Норвиг. – М.: Вильямс, 2006. – 1408 с.

9. VanLehn, K. Intelligent tutoring systems for continuous embedded assessment [Текст] / K. VanLehnIn C. A. Dwyer (Ed.), The future of assessment: Shaping teaching and learning. NewYork, NY: Erlbaum.– 2008. – P. 113-138.

10. Левенштейн, В.И. Двоичные коды с исправлением выпадений, вставок и замещений символов [Текст] / В.И. Левенштейн // Доклады Академий Наук СССР . – 1965. – № 163 (4). – С. 845 –848.

11. Using q-grams in a DBMS for Approximate String [Текст] / L. Gravano, P. Ipeirotis, H. VisvesvarayaJagadish, N. Koudas, S. Muthukrishnan, L. Pietarinen, D. Srivastava. – Venue: IEEE Data Engineering Bulletin, Vol. 24. – No. 4, December 2001. –7 p.

12. Кулик, А.С. Нечеткий поиск похожих строк в системах повышения качества данных автоматизированных систем организационного управления [Текст] / А.С. Кулик, А.Г. Чухрай, А.Ю Завгородний // Радиоэлектронные и компьютерные системы. – 2006. – № 7 (19). – С. 17-22.

13. Kulik, A. Similar strings detecting methods [Текст] / A. Kulik, A. Chukhray, A. Zavgorodniy // Proceedings of the East-West Fuzzy Colloquium, Zittau, Germany, IPM. – 2005. – P. 38 – 47.

14. Чухрай А. Г. Метод нечеткого поиска объектов [Текст] / А.Г. Чухрай // Радиоэлектронные и компьютерные системы. – 2012. – № 5 (57). – С. 171 – 177.

15. MacQueen, J. Some methods for classification and analysis of multivariate observations. [Текст] / JMacQueen // Proc. 5th Berkeley Symp. on Math. Statistics and Probability. – 1967. – P. 281 – 297.

16. Кормэн, Т. Алгоритмы: Построение и анализ. [Текст] / Т. Кормэн, Ч. Лейзерсон, Р. Ривест. – М. МЦНМО, 2001. – 960 с.

17. Zhang, K. Simplefastalgorithmsfortheeditingdistancebetweentreesandrelatedproblems [Текст] / K. Zhang, D. Shasha, Societyf or Industrial and Applied Mathematics Journalon Computing. – 1989. – Vol. 18, No. 6. – P. 1245 – 1262.

18. Чухрай, А.Г. Об одномметодепроверкипрофессиональныхуменийалгоритмизации. [Текст] / А.Г. Чухрай // Радиоэлектронные и компьютерные системы. – 2009. – № 4 (38). – С. 84 – 86.

19. Wagner, R.A. The String-to-String Correction Problem [Текст] /R.A. Wagner, M.J. Fischer. – 1974. – Journal of the ACM, 21(1). –P. 168–173.

20. Development of the universal environment for creation and translation of intelligent tutoring programs. [Текст] / A.Kulik, A. Chukhray, S Pedan, T. Kulik // Proceedings of the International Conference of "Interactive computer aided learning" ICL 2009 : EPortofolio and Quality in e-Learning. – 2009. – Austria, Villach. – 6 p.

21. Implementation of Computer Tutoring Program Adaptation Methods [Текст]/ A .Kulik, S .Pedan, A. Chukhray, S. Hall //Journal of Higher Education Theory and Practice. – 2012. – Vol. 12 (1). – P. 39 – 55.

22. Development of the automated laboratory practical work at the course «Modeling of systems» [Текст] / A. Kulik, S. Pedan, A. Chukhray, P. Anzenberger // Conference ICL, September 24 -26. – 2008. – Villach, Austria. – 11 p.

23. Компьютерная система интерактивного тестирования знаний и умений учащихся [Текст] /

А.Л. Сидоренко, С.А. Раков , А.С. Кулик , А.Г. Чухрай. – Вестник ТИМО. – 2008. – № 4. – С. 15 – 19.

24. Chukhray, A An approach to development of intelligent computer system for SQL tutoring [Текст] / A. Chukhray, V. Kalinichenko // Proceedings of the East-West Fuzzy Colloquium, Germany, Zittau, IPM. – 2010. – P. 251 – 256.

25. Чухрай, А.Г. Разработка комплекса интерактивных web-тестов по математике [Текст] А.Г. Чухрай, С.И. Педан, Е.С. Вагин // Радиоэлектронные и компьютерные системы. – 2010. – №1 (42). – С. 103 – 107.

26. Chukhray, A.G. Formalization of tasks generation for complex of interactive web-tests on math [Текст] / A.G. Chukhray, Ie.S. Vagin , S.I Pedan // Proceedings of the East-West Fuzzy Colloquium, Germany, Zittau, IPM. – 2010. – P. 264 – 270.

Поступила в редакцию 28.02.2013, рассмотрена на редколлегии 20.03.2013

Рецензент: д-р техн. наук, проф., зав. каф. авиационных приборов и измерений Н.Д. Кошевой, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.

МОДЕЛІ ТА МЕТОДИ АДАПТИВНИХ КОМПЬ'ЮТЕРНИХ СИСТЕМ ПІДТРИМКИ НАДБАННЯ ЗНАНЬ ТА ВМІНЬ ПРИ ВИРІШЕННІ АЛГОРИТМІЧНИХ НАВЧАЛЬНИХ ЗАДАЧ

А.Г. Чухрай

На основі підходу до раціонального управління об'єктами в позаштатних режимах, що створено професором А.С. Куліком, розроблено моделі та методи адаптивних комп'ютерних систем підтримки надбання знань та вмінь при вирішенні багатокрокових алгоритмічних навчальних задач. Запропоновано три режими роботи таких систем: «демо», вирішення з підказами, тестовий, а також два етапи навчання – розрахунок за алгоритмом та написання алгоритму. Описано моделі алгоритмічної навчальної задачі, метод генерації і розрахунку об'єктів, моделі особи, яка навчається, метод автоматичної побудови діагностичних моделей пропуску операндів, модель даних, модель навчання, методи діагностування програм, що складені особою, яка навчається.

Ключові слова: алгоритмічні навчальні задачі, адаптивна комп'ютерна підтримка надбання знань та вмінь.

MODELS AND METHODS OF KNOWLEDGE AND SKILLS ACQUISITION ADAPTIVE COMPUTER SUPPORT IN SOLVING OF ALGORITHMIC EDUCATIONAL PROBLEMS

A.G. Chukhray

Based on the approach to rational control of objects in abnormal modes, created by Professor A. Kulik, models and methods of knowledge and skills acquisition adaptive computer support in solving of algorithmic educational problems were developed. Three modes of operation of such systems: "demo", solutions with hints, test, and two stages of training - calculation on the base of algorithm and writing of algorithm were supposed. The models of algorithmic educational problem, the method of calculation and generation of objects, the student model, the method of automatic generation of diagnostic models pass the operands, the data model, the model of teaching, methods of diagnosis of programs written by the trainees are described.

Key words: algorithmic educational problems, knowledge and skills acquisition adaptive computer support.

Чухрай Андрей Григорьевич– докторант каф. систем управления летательных аппаратов, канд. техн. наук, доцент, Национальный аэрокосмический университет им. Н.Е. Жуковского «Харьковский авиационный институт», Харьков, Украина.