УДК 004.4

Интеллектуальная диагностика ошибок и маршрутизация заданий дистанционной самоподготовки студентов по программированию

Е.Р. Пантелеев, А.Л. Архипов, А.В. Второв
ФГБОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»,
г. Иваново, Российская Федерация
E-mail: erp@poks.ispu.ru

Аннотация

Состояние вопроса: Актуальность инноваций в области самоподготовки студентов энергетических специальностей и направлений по программированию определяется двумя факторами. Это, с одной стороны, возрастание роли информационных технологий в энергетике, а с другой – увеличение доли самостоятельных занятий в учебных планах и ориентация самоподготовки на формирование навыков решения профессиональных задач. Наиболее перспективной инновацией в этой области является использование методов дистанционного обучения. Однако формирование профессиональных навыков требует от системы дистанционного обучения наличия элементов интеллекта для преобразования учебных целей в персональные учебные воздействия. Большинство работ в области компьютерного обучения программированию специализируют интеллект на обслуживании процесса написания программ.
Материалы и методы: Математическую основу разработки составляет аппарат сетей Байеса, который использован для формализации вероятностных зависимостей между исходами автоматизированного тестирования, диагностированными ошибками и локальными учебными воздействиями. Стратегия самоподготовки строится на использовании сетевой модели предметных знаний в контексте текущих достижений студента..
Результаты: Предложен альтернативный подход, основанный на интеллектуальном анализе окончательного результата – компьютерной программы. На платформе СДО ГИПЕРТЕСТ разработана архитектура системы дистанционной самоподготовки студентов по программированию и выполнена ее распределенная программная реализация.
Выводы: Опыт практического использования системы позволяет сделать вывод об адекватности автоматически формируемых ею локальных обучающих воздействий и рекомендаций по выбору очередного задания. Это дает основания утверждать, что при равных затратах времени на взаимодействие студента с системой достигнутые им учебные результаты будут более высокими по сравнению с базовым вариантом использования автоматизированного тестирования, не предусматривающим интеллектуальной поддержки.

Ключевые слова: дистанционное обучение, интеллектуальная обучающая система, диагностика ошибок программирования, сеть Байеса.

Intelligent Diagnostics of Errors and Routing of Students' Assignments in Distant Self-training in Programming

E.R. Panteleev, A.L. Arhipov, A.V. Vtorov
Ivanovo State Power Engineering University, Ivanovo, Russian Federation
E-mail: erp@poks.ispu.ru

Abstract

Background: The topicality of innovation in power-engineering students' self-training in programming is twofold. It comes, on the one hand, from the growing role of IT in power engineering, and on the other – from the increased proportion of self-training that focuses on problem solving in the engineering curricula. The most promising innovation in this field is distance learning. However, the build-up of professional skills requires the presence of intelligence to transform training goals into personal learning impacts. Most of intelligent tutoring systems specialize in supporting the computer program development process. This paper proposes an alternative approach that supports intellectual analysis of black-box testing of a computer program itself.
Materials and methods: Bayesian network apparatus, which is used to formalize the probabilistic relationships between automated testing outcomes, detected errors and local learning impacts forms the mathematical basis of the study. The self-training strategy is based on the use of subject knowledge in the context of the personal learning achievements.
Results: The research has resulted in a distributed internet application that supports the above self-training tactics and strategy and is implemented as the extension for HYPERTEST learning management system.
Conclusions: Practical use of the system permits us to conclude that automatically generated local learning impacts and next task recommendations are quite relevant to personal results and training goals. Thus, it is reasonable to state that the proposed approach to self-training is more efficient (i.e. needs less time to obtain the same result) compared to conventional self-training with no intellectual support available.

Key words: distant learning, intelligent tutoring system, programming errors diagnosis, Bayesian network.

Проблема организации эффективной самоподготовки студентов энергетических специальностей и направлений в области информационных технологий (ИТ) становится все более актуальной. Этому способствуют два фактора. Во-первых, роль ИТ в обеспечении жизненного цикла объектов и систем энергетики неуклонно возрастает. ИТ являются основным инструментом при выполнении научных исследований и проектирования [1], реализации задач оперативного контроля и управления [2], энергетического мониторинга и аудита [3]. Целесообразное использование этих инструментов требует, как минимум, понимания алгоритмической природы процессов, реализуемых на базе ИТ, а в ряде случаев – и навыков их практической реализации. Во-вторых, стандарты высшего профессионального образования третьего поколения не только увеличивают объем часов, отводимых на самоподготовку, но и переносят акцент в этом виде учебной деятельности с выполнения предписаний заданного преподавателем сценария на активную познавательную деятельность, направленную на решение профессиональных задач. Указанные факторы требуют внедрения инновационных форм самоподготовки в области алгоритмизации и программирования.

Наиболее органичной инновацией в этой области является применение методов и технологий дистанционного обучения, поскольку в данном случае среда самоподготовки совпадает с целевой средой профессиональной деятельности. Однако возможностей широко распространенных сред дистанционного обучения (СДО), таких как Blackboard [4] и Moodle [5], недостаточно для того, чтобы самоподготовка в области алгоритмизации и программирования стала эффективной. Стандартные технологии контроля результатов обучения, встроенные в эти системы, обеспечивают в основном проверку знаний, т.е. способностей студента к воспроизведению и категоризации теоретического материала, тогда как целевой установкой самостоятельной работы, декларированной стандартами третьего поколения, является формирование умений и навыков решения задач. Для этого СДО должна уметь диагностировать ошибки, допущенные студентом в процессе самоподготовки, формировать обучающие воздействия и индивидуальный маршрут обучения в контексте учебной цели и персонального профиля результатов. Системы, располагающие необходимыми для решения этих задач знаниями о предметной области (эталонные результаты и способы их достижения), о студенте (персональные ошибки и результаты), о целях и способах обучения (интерфейс пользователя, оценка персональных результатов и формирование обучающих воздействий), относятся к классу интеллектуальных обучающих систем (ИОС). Известные реализации ИОС в области программирования [6–8] обеспечивают интеллектуальную поддержку валидации[1] кода программы (адаптивная навигация аннотированного кода [6], интерактивная разработка псевдокода на естественном языке [7]). Если все спецификации процесса разработки программы корректны, то заключительная спецификация правильно решает поставленную задачу. Это способствует формированию навыков грамотного использования языков программирования в качестве инструмента решения поставленной задачи. К сожалению, сложность валидации заставляет разработчиков ИОС ограничиваться поддержкой начальных этапов обучения программированию. Вместе с тем существует еще один способ убедиться в правильности программы – это верификация[2], самым эффективным методом которой является автоматизированное тестирование. Несмотря на недостатки (автоматизированное тестирование, как правило, рассматривает программу как «черный ящик» и к тому же принципиально не может гарантировать отсутствие ошибок), сравнительная простота процедуры и наличие готовых решений делают автоматизированное тестирование весьма привлекательным инструментом верификации программ.

Рассмотрим один из возможных подходов к формированию навыков программирования путем тестирования студенческих программ, интеллектуальной диагностики ошибок и маршрутизации заданий самоподготовки студентов по программированию. Данный подход реализован в СДО ГИПЕРТЕСТ [9]. ГИПЕРТЕСТ не относится к классу ИОС, однако располагает средствами расширения стандартного набора процедур контроля внешними процедурами (обработчиками) с унифицированным интерфейсом web-сервисов, включающим операции «выдать задание» и «оценить решение», что позволило реализовать архитектуру, которая помимо ГИПЕРТЕСТ включает в себя систему автоматизированного тестирования программ eJudge и интегрирующий их web-сервис [10]. Комплекс прошел апробацию при проведении региональных интернет-олимпиад по информатике, но алгоритм оценки студенческих решений был основан на достаточно серьезных допущениях, справедливых только для тривиальных программ:

- тестирование является исчерпывающим, т.е. проверяет наличие всех возможных ошибок программной реализации алгоритма;
- каждый из тестов детерминированно выявляет одну из ошибок.

[1] Валидация – это проверка соответствия спецификации программы на каждом этапе ее разработки исходной постановке задачи.

[2] Верификация – это проверка соответствия результатов исполнения программы ее исходной постановке.

Обсуждаемое ниже расширение описанной в [10] архитектуры пост-обработчиком протокола тестирования (рис. 1,а) позволило в значительной степени снять эти допущения и сделать процедуру оценки решений и управления выбором заданий более интеллектуальной. Функции диагностики ошибок и маршрутизации заданий в этой расширенной архитектуре реализованы с помощью сети Байеса (СБ) [11]. СБ – формализм, многократно доказавший свою эффективность в качестве инструмента интеллектуального сопровождения разработки программ [12, 13]. СБ – это направленный ациклический граф, вершинам которого соответствуют случайные величины с конечным числом взаимоисключающих исходов, а дуги отражают вероятностную зависимость этих случайных величин так, что любой дочерней вершине B соответствует распределение условных вероятностей исходов: $P(B|A_1,A_2,\ldots,A_i,\ldots A_n)$, где A_i – родительские вершины, а вершинам, не имеющим родителей, соответствует распределение маргинальных вероятностей исходов. СБ используются для расчета распределения вероятностей с явным учетом условной независимости переменных, что позволяет уменьшить вычислительную сложность этого расчета по сравнению с прямым применением формулы Байеса. В рамках предложенной архитектуры вычисления на СБ выполняет SMILE (Structural Modeling, Inference, and Learning Engine) – компонент ядра свободно распространяемой среды графической разработки СБ GeNIe [14], упакованный для унификации взаимодействия с другими компонентами в оболочку web-сервиса.

Рассмотрим интеллектуальное наполнение нового компонента этой архитектуры – пост-обработчика протокола тестирования, выполняющего функции диагностики ошибок и маршрутизации заданий. Оно включает модель предметной области (МПО), модель обучения (МО), модель студента (МС) и модель интерфейса (МИ). Информационные связи этих моделей показаны на рис. 1,б.

Модель предметной области представлена:

- сетевым графом $G = (S,F)$, вершинам которого соответствуют навыки $S = \{s_1, s_2, \ldots, s_i, s_n\}$, формируемые в процессе самоподготовки, а дугам – отношения F, отражающие последовательность освоения этих навыков;
- целевым профилем результатов самоподготовки, представляющим собой вектор эталонных вероятностей успешного применения навыков $P = (p(s_1), p(s_2), \ldots, p(s_i), p(s_n))$;
- банком заданий $Q = \{q_1, q_2, \ldots, q_i, q_m\}$ и их профилей, определенных как ребра $q \in Q$ гиперграфа $H = (S,Q)$[3], а также банком эталонных решений $R = \{r_1, r_2, \ldots, r_i, r_m\}$ и диагностических тестов $T = U^m_{i=1} T_i$.

Модель обучения представлена следующими компонентами:

1. Для каждого из заданий по программированию q определена сеть Байеса, которая отражает вероятностную зависимость между множеством двоичных результатов (маской) проверочных тестов $T = \{t_1, t_2, \ldots, t_k\}$, множеством выявленных ими профильных ошибок задания q: $E \leftrightarrow q$, персональным профилем студента и видами локальных учебных воздействий $\{c_1, c_2\}$: комментарий, пример – для каждой ошибки.

2. Так как в общем случае построить исчерпывающую систему тестов невозможно, для каждого задания существует подмножество нераспознаваемых ошибок E_{undef}. Пусть T_e – подмножество тестов, диагностирующих профильную ошибку, а T_{Eq} – множество всех подмножеств T_e. Тогда $T_{E\text{undef}} = T_E \backslash T_{Eq}$, где T_E – множество всех подмножеств тестов задания. Таким образом, мощность множества $T_{E\text{undef}}$ определяет количество нераспознанных ошибок, а каждое подмножество в $T_{E\text{undef}}$ есть маска одной из нераспознаваемых ошибок.

3. Определены правила стратегии обучения, задающие условия начала и завершения процесса самоподготовки, а также тактику выбора следующего задания. Самоподготовка может быть начата любым заданием, навыки профиля которого не требуют предусловий: $\forall s_i \in q \left\{ \left(s, s_i\right) \mid s \in S \setminus \left\{s_i\right\} \right\} = 0$. Самоподготовка завершается успехом, если освоены все навыки: $\forall i \in \overline{1,n} : \bar{p}(s_i) \geq p(s_i)$, или неудачей, если предыдущее условие не выполнено и нет доступных заданий для продолжения самоподготовки. Задание доступно в точке i индивидуального маршрута самоподготовки, если его профиль q^{i+1}, определенный на гиперграфе H, включает непустое подмножество преемников $S^{i+1} = U_{Sx \in Si\,(+)}\{S_y \mid (S_x,S_y) \in F\}$ освоенных навыков из q^i: $S^{i(+)} = \{s_j \in q^i \mid \bar{\tilde{p}}\,(s_j) \geq p(s_j)\}$ и все неосвоенные навыки из q^i: $S^{i(-)} = S^i \setminus S^{i(+)}$, а множество неиспользованных вариантов локальных учебных воздействий не пустое. Из множества доступных заданий выбирается такое, для которого $|S^{i+1}| \rightarrow \min$.

Модель студента представлена:

- траекторией обучения $(q^0, q^1, \ldots, q^i)$;
- глобальным профилем (персональным вектором вероятностей успешного применения навыков) $\tilde{P} = \left\{\tilde{p}(s_1), \tilde{p}(s_2), \ldots, \tilde{p}(s_i), \ldots, \tilde{p}(s_n)\right\}$;
- профилем локальных результатов (ошибок $E^{i(-)} \subseteq E^i$, допущенных при решении i-й задачи).

[3] Ребро q гиперграфа – это подмножество инцидентных ему вершин, в данном случае навыков S, формируемых заданием q, называемых профилем задания.

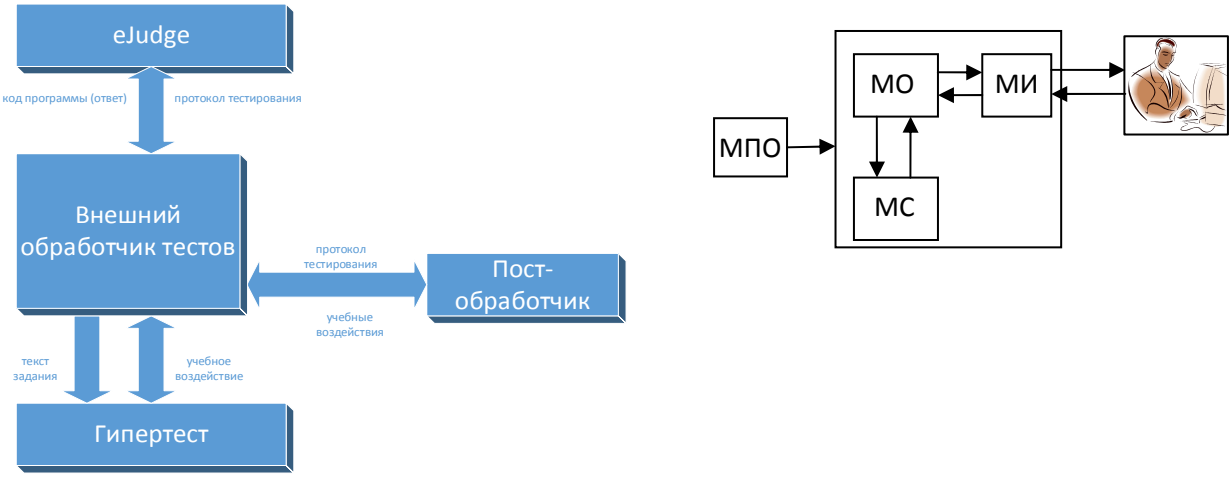


Рис. 1. Комплекс самоподготовки и интеллектуальный компонент пост-обработки: а – архитектура комплекса самоподготовки; б – структура пост-обработчика

Модель интерфейса представлена:

- правилом формирования начального состояния персонального профиля $\forall s \in S : \tilde{p}(s) = \frac{1}{2}$;
- правилом пересчета профиля по результатам решения очередного задания: пусть $\tilde{p}(s) = \frac{a}{b}$, тогда после решения *b* автоматически инкрементируется, а *a* инкрементируется, только если вероятность ошибки не превышает заданного порогового значения;
- правилом формирования локального учебного воздействия (определяется соответствием между ошибками, с одной стороны, и комментариями и примерами – с другой).

Рассмотрим пример *модели предметной области* для раздела самоподготовки «Циклы». Множество *S* включает в себя навыки, ошибки формирования которых можно выявить с помощью тестов:

1) организация цикла с фиксированным числом переменных;
2) организация итерационного цикла;
3) организация сложного цикла;
4) организация цикла для работы с массивом;
5) выбор управляющей переменной цикла;
6) эффективная организация циклов.

В дальнейшем для ссылки на навык будем использовать его индекс в вышеприведенном списке.

Множество дуг F = {(s_1,s_3), (s_1,s_4), (s_1,s_5), (s_1,s_6), (s_2,s_3), (s_2,s_4), (s_3,s_6), (s_4,s_6)} определяет последовательность освоения этих навыков (рис. 2,а). Пусть навык считается освоенным, если он применяется безошибочно не менее чем в 80 % случаев. Тогда эталонный профиль имеет вид P = (0.8, 0.8,0.8,0.8,0.8, 0.8). Пусть теперь в банке заданий присутствует элемент q_1. Ему соответствует задание: «*Даны натуральные числа a и b, причем a< b; найдите все простые числа в интервале от a до b*». Профиль навыков этого задания показан на рис. 2,б. Чтобы не перегружать изложение, опустим код эталонной программы, однако при разработке тестов будем исходить из того, что для решения данной задачи необходимо реализовать сложный цикл. Внешний цикл используется для перебора чисел от *a* до *b*, а внутренний проверяет, является ли это значение простым. И внешний и внутренний циклы допускают оптимизацию. Внешний оптимизируется за счет исключения перебора четных чисел и применения алгоритма 6*k* + 1, а внутренний – за счет предварительного формирования массива простых чисел.

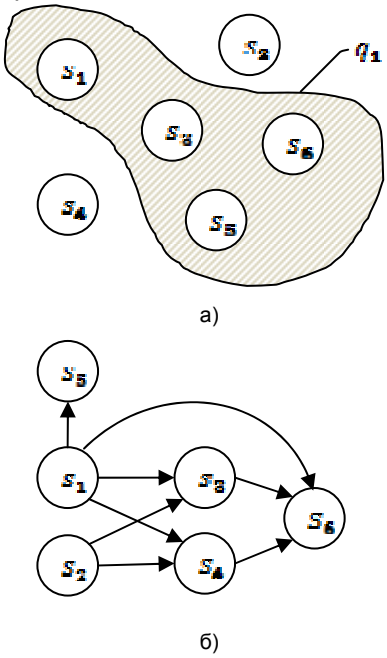


Рис. 2. Фрагменты модели предметной области «Циклы»: а – Сетевой граф навыков для предметной области «Циклы»; б – профиль задания q_1

Кроме того, при разработке тестов исходили из того, что каждая ошибка диагностируется, как минимум, одним тестом, так что количество тестов должно быть не меньше количества элементов профиля задания. Для более надежного диагностирования ошибок рекомендуется, чтобы количество тестов превышало количество элементов профиля.

1. a = 1, b = 2 ... 100. Данная группа тестов проверяет корректность организации внутреннего цикла. Небольшое значение b гарантирует работоспособность любого правильного (пусть и неэффективного) алгоритма, а значение a = 1 позволяет снизить влияние неправильной реализации внешнего цикла (хотя и не исключает ошибку именно в нем).

2. $b = a + 1$, a и b – не являются простыми числами. Данная группа тестов позволяет проверить реализацию внешнего цикла. В интервале от a до b нет простых чисел, это позволяет снизить вероятность ошибки во внутреннем цикле.

3. a = 1, b = 10000, 20000, 30000. Данная группа тестов проверяет эффективность организации внутреннего цикла. С их помощью можно определить правильные, но неэффективные алгоритмы (полный перебор или перебор до $N/2$). Эти тесты также можно использовать для определения ошибок в организации циклов.

4. a = 1, b = 70000, 80000, 90000. Данная группа тестов определяет корректное использование 32-х разрядных переменных (вместо 16-ти разрядных).

5. a = 1, b = 1000000, 2000000, 3000000. Группа тестов с большим верхним пределом позволяет проверить корректность реализации эффективного алгоритма поиска простых чисел типа $6k \pm 1$ с проверкой числа делением только на простые числа, найденные ранее.

6. a = 1000000, 2000000, 3000000, $b = a + 1000000$. Данная группа тестов проверяет корректную реализацию алгоритма $6k \pm 1$ независимо от значения нижней границы интервала, поиск делителей следует начинать с 2.

Центральным компонентом *модели обучения* является сеть Байеса. Для рассмотренного выше задания она имеет вид, показанный на рис. 3,а. Ее топология отражает связь между исходами тестов (пройден/провален), диагностированными ошибками, профилем студента и локальными учебными воздействиями.

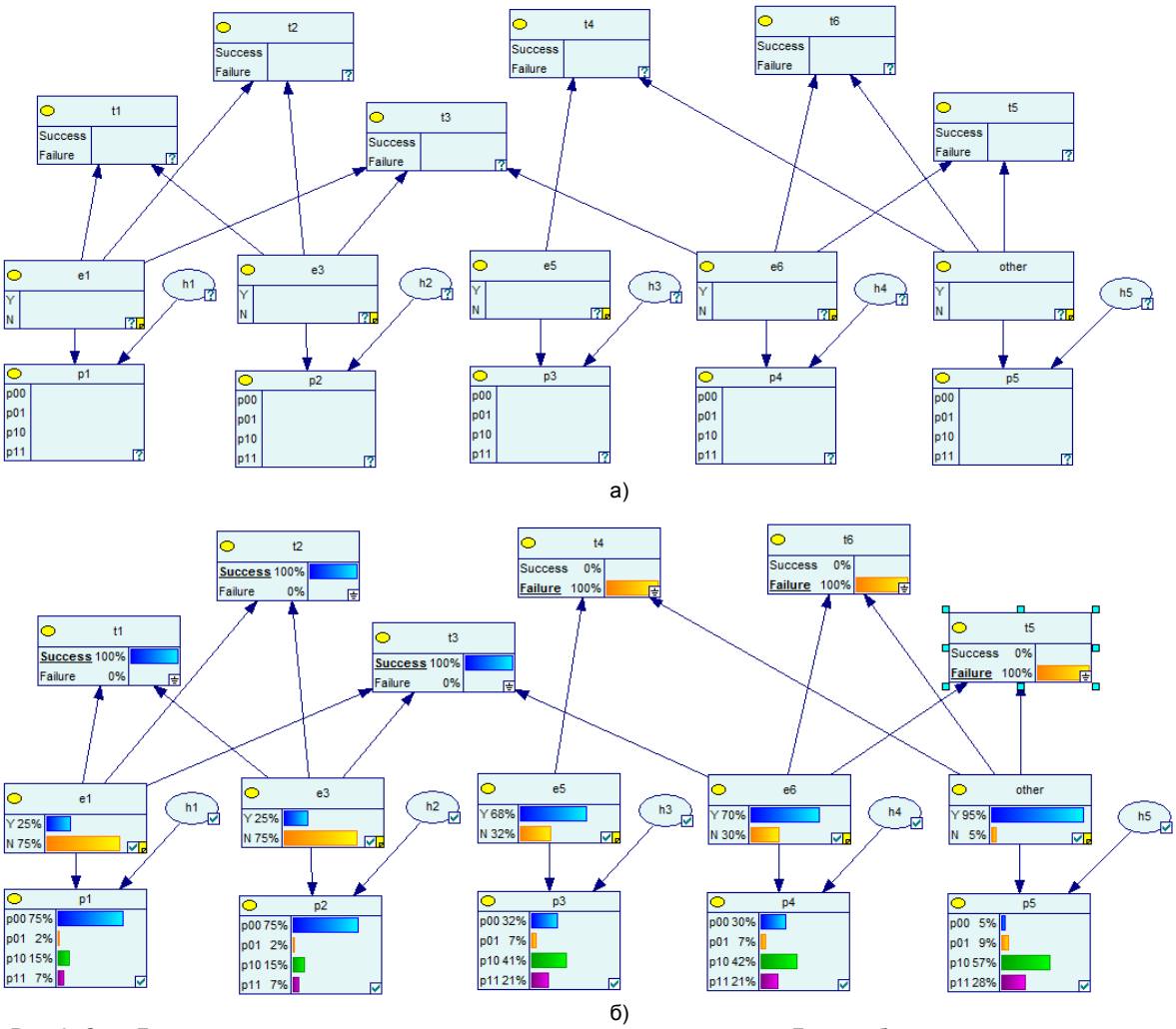


Рис. 3. Сеть Байеса для задачи о простых числах: а – неактивированная сеть Байеса; б – расчет на активированной сети

Следует заметить, что ошибка в организации сложного цикла диагностируется, если провалены тесты групп 1 и 2 одновременно. В основу параметризации сети положены следующие допущения:

1. Тесты считаются независимыми испытаниями.

2. Вероятность профильной ошибки e_i зависит от исхода детектирующих ее тестов $T_{i,,}$ так что:

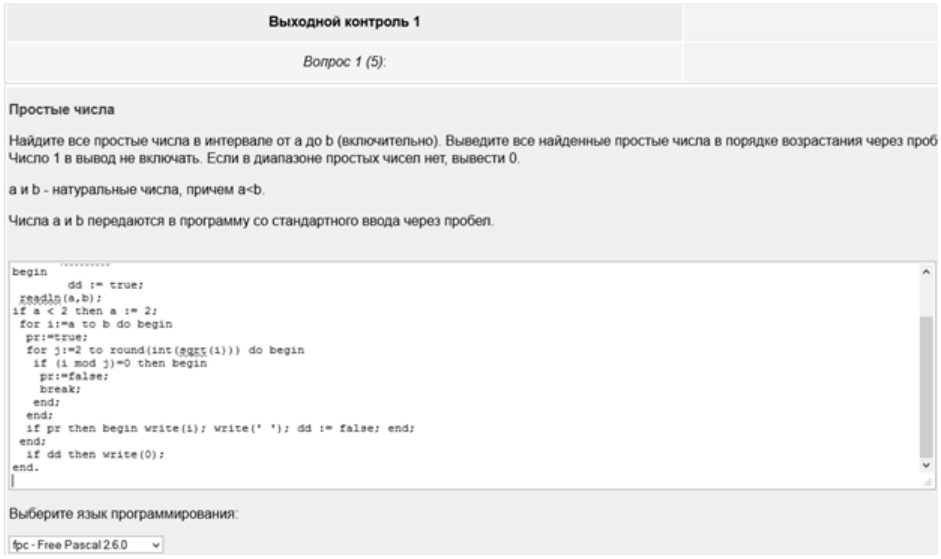
а) условная вероятность выявления ошибки тестами $T_k \subseteq T_i$ равна сумме условных вероятностей выявления ошибки каждым из них: $\left(p(e_i \mid \forall t \in T_k : t = 0) = \sum_{t \in Tk} p(e_i \mid t = 0)\right)$

б) при неудаче всех тестов ошибка абсолютно достоверна: $\left(p(e_i \mid \forall t \in T_i : t = 0) = 1\right)$.

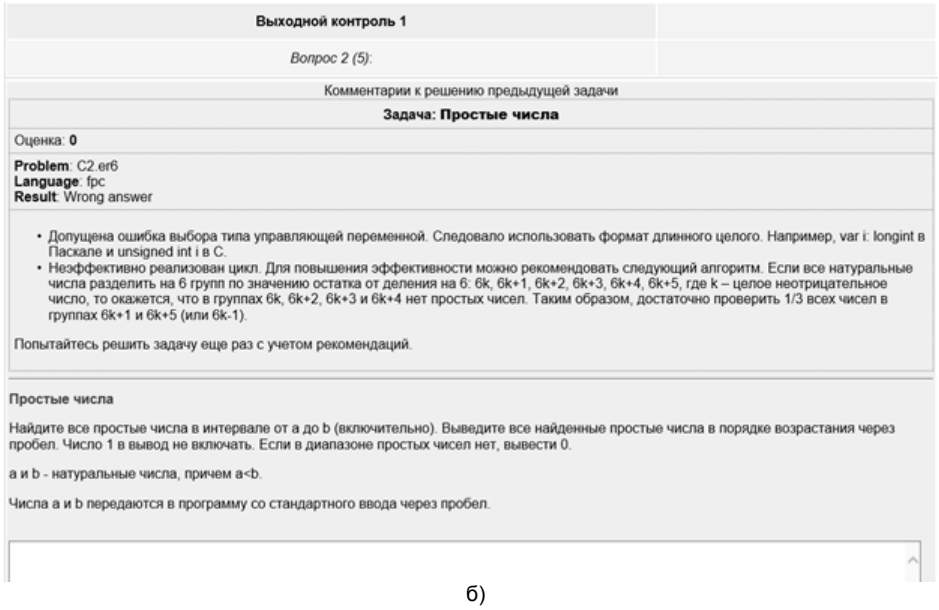
3. При выборе локального обучающего воздействия вероятностное предпочтение отдается комментарию, если студент делает ошибку впервые, и примеру – в противном случае.

В соответствии с сетевой структурой навыков предметной области «Циклы» (рис. 2,а), в качестве стартового задания может быть выбрано любое задание из банка, профиль которого включает только навыки из $\{s_1, s_2\}$. Например, доступно задание «Определить, является ли m натуральной степенью n, где m и n – заданные натуральные числа», которое оценивает навык s_2 (организация итерационных циклов).

Рассмотрим последовательность постобработки результатов автоматизированного тестирования в следующей ситуации. Постобработчик получает от eJudge маску $T = \{t_1 = 1, t_2 = 1, t_3 = 1, t_4 = 0, t_5 = 0, t_6 = 0\}$ результатов тестирования программы «Найти простые числа…» (рис. 4,а).



а)



б)

Рис. 4. Интерфейс пользователя: а – форма ввода задания; б – форма вывода учебных воздействий

Сопоставление маски с топологией СБ показывает, что все допущенные в решении ошибки являются профильными (в противном случае пост-обработчик формирует сообщение о нераспознаваемой ошибке). Далее, используя маску, пост-обработчик выполняет расчет СБ для текущего задания с учетом состояния глобального профиля студента $\tilde{P} = \{\tilde{p}(s_1) = 0.8, \tilde{p}(s_3) = 0.75, \tilde{p}(s_5) = 0.75, \tilde{p}(s_6) = 0.66\}$. В результате расчета (рис. 3,б) формируется профиль локальных результатов (текущих ошибок) студента (ошибка выбора управляющей переменной цикла – e_3, неэффективное управление циклом – e_4) и набор локальных учебных воздействий (рис. 4,б). Оценка локальных результатов путем сравнения вероятностей ошибок с заданными граничными значениями активизирует пересчет глобального профиля студента $\tilde{P} = \{\tilde{p}(s_1) = 0.83, \tilde{p}(s_3) = 0.8, \tilde{p}(s_5) = 0.6, \tilde{p}(s_6) = 0.5\}$. Анализ глобального профиля студента показывает, что навыкs_1им освоен, и поэтому в профиль следующего задания можно включать навыки-преемники s_1: {s_3, s_4, s_6}, а также неосвоенные навыки текущего задания {s_3, s_5, s_6}. Но так как при выполнении задания студентом были допущены ошибки и пост-обработчик сформировал в функции этих ошибок локальные учебные воздействия, это задание будет предложено ему повторно.

Таким образом, в результате выполненных исследований разработана архитектура и реализована система дистанционной самоподготовки студентов по программированию. Самоподготовка направлена на формирование навыков программирования путем автоматизированного тестирования решений и интеллектуальной диагностики результатов тестирования. Целью диагностики является формирование локальных обучающих воздействий и общей стратегии самоподготовки. Модельную базу диагностики составляет СБ, которая на основании анализа результатов тестирования формирует вероятностные гипотезы о допущенных ошибках и способах формирования локальных обучающих воздействий. Результаты диагностики используются также для формирования стратегии в контексте учебных целей, доступных ресурсов и персонального уровня подготовки студента.

Система дистанционной самоподготовки прошла апробацию на контрольной группе студентов.

Список литературы

1. **Целищев Е.С.** Новый подход к построению универсальной структуры информационного обеспечения процесса проектирования систем контроля // САПР и графика. – 2009. – № 5. – С. 103–106.

2. **Анисимов А.А., Тарарыкин С.В., Аполонский В.В.** Параметрическая оптимизация и настройка цифровых регуляторов состояния // Проблемы разработки перспективных микро-и наноэлектронных систем-2010 (МЭС-2010). – 2010. – С. 482–487.

3. **Ратманова И.Д.** Информационно-аналитическое сопровождение энергетической политики на региональном уровне // Вестник ИГЭУ. – 2012. – Вып. 5. – С. 63–67.

4. **Liaw S.S.** Investigating students' perceived satisfaction, behavioral intention, and effectiveness of e-learning: A case study of the Blackboard system // Computers & Education. – 2008. – Т. 51, № 2. – С. 864–873.

5. **Brandl K.** Are you ready to «Moodle» // Language Learning & Technology. – 2005. – Т. 9, № 2. – С. 16–23.

6. **Yudelson M., Brusilovsky P.** NavEx: Providing Navigation Support for Adaptive Browsing of Annotated Code Examples // AIED. – 2005. – Т. 5. – С. 710–717.

7. **Lane H. C., VanLehn K.** A Dialogue-Based Tutoring System for Beginning Programming // FLAIRS Conference. – 2004. – С. 449–454.

8. **Butz C.J., Hua S., Maguire R.B.** A web-based intelligent tutoring system for computer programming // Web Intelligence, 2004. WI 2004. Proceedings. IEEE/WIC/ACM International Conference on IEEE. – 2004. – С. 159–165.

9. **Пантелеев Е.Р.** Среда разработки программ дистанционного обучения и профильного тестирования ГИПЕРТЕСТ: логистическая модель и архитектура // Информационные технологии. – 2001. – № 5. – С. 30–36.

10. **Пантелеев Е.Р., Второв А.В., Архипов А.Л., Ильина Е.В.** Интеграция инструментов контроля навыков программирования в среду интернет-обучения // Вестник ИГЭУ. – 2010. – №. 3. – С. 104–108.

11. **Jain A.** Software Defect Content Estimation: A Bayesian Approach // Electrical and Computer Engineering, 2006. CCECE'06. Canadian Conference on IEEE. – 2006. – С. 2449–2454.

12. **Ziv H., Richardson D.J.** Constructing Bayesian-network models of software testing and maintenance uncertainties // Software Maintenance, 1997. Proceedings. International Conference on IEEE. – 1997. – С. 100–109.

13. **Druzdzel M.J.** SMILE: Structural Modeling, Inference, and Learning Engine and GeNIe: a development environment for graphical decision-theoretic models // AAAI/IAAI. – 1999. – С. 902–903.

14. **Heckerman D.** A tutorial on learning with Bayesian networks. – Springer Berlin Heidelberg, 2008. – С. 33–82.

References

1.Tselishchev, E.S. Novyy podkhod k postroeniyu universal'noy struktury informatsionnogo obespecheniya protsessa proektirovaniya sistem kontrolya [A new approach to developing a universal structure of information support of control system design]. *SAPR i grafika*, 2009, no. 5, pp. 103–106.

2.Anisimov, A.A., Tararykin, S.V., Apolonskiy, V.V. Parametricheskaya optimizatsiya i nastroyka tsifrovykh regulyatorov sostoyaniya [Parameter optimization and digital state regulator adjustment]. *Problemy razrabotki perspektivnykh mikro- i nanoelektronnykh sistem-2010 (MES-2010),* 2010, pp. 482–487.

3.Ratmanova, I.D. Informatsionno-analiticheskoe sopровozhdenie energeticheskoy politiki na regional'nom urovne [Research and information support of energy policy at a regional level]. *Vestnik IGEU*, 2012, no. 5, pp. 63–67.

4. Liaw, S.S. Investigating students' perceived satisfaction, behavioral intention, and effectiveness of e-learning: A case study of the Blackboard system. Computers & Education, 2008, vol. 51, no. 2, pp. 864–873.

5. Brandl, K. Are you ready to «Moodle». Language Learning & Technology, 2005, vol. 9, no. 2, pp. 16–23.

6. Yudelson, M., Brusilovsky, P. NavEx: Providing Navigation Support for Adaptive Browsing of Annotated Code Examples. AIED, 2005, vol. 5, pp. 710–717.

7. Lane, H.C., VanLehn, K. A Dialogue-Based Tutoring System for Beginning Programming. FLAIRS Conference, 2004, pp. 449–454.

8. Butz, C.J., Hua, S., Maguire, R.B. A web-based intelligent tutoring system for computer programming. Web Intelligence, 2004. WI 2004. Proceedings. IEEE/WIC/ACM International Conference on. IEEE, 2004, pp. 159–165.

9. Panteleev, E.R. Sreda razrabotki programm distantsionnogo obucheniya i profil'nogo testirovaniya GIPERTEST: logisticheskaya model' i arkhitektura [Development environment of distance learning and subject-oriented testing HYPERTEXT: logistic model and architecture]. *Informatsionnye tekhnologii*, 2001, no. 5, pp. 30–36.

10. Panteleev, E.R., Vtorov, A.V., Arkhipov, A.L., Il'ina, E.V. Integratsiya instrumentov kontrolya navykov programmirovaniya v sredu internet-obucheniya [Integration of programming skills control tools into e-learning environment]. *Vestnik IGEU*, 2010, no. 3, pp. 104–108.

11. Jain, A. et al. Software Defect Content Estimation: A Bayesian Approach. Electrical and Computer Engineering, 2006. CCECE'06. Canadian Conference on. IEEE, 2006, pp. 2449–2454.

12. Ziv, H., Richardson, D.J. Constructing Bayesian-network models of software testing and maintenance uncertainties. Software Maintenance, 1997. Proceedings, International Conference on. IEEE, 1997, pp. 100–109.

13. Druzdzel, M.J. SMILE: Structural Modeling, Inference, and Learning Engine and GeNIe: a development environment for graphical decision-theoretic models. AAAI/IAAI, 1999, pp. 902–903.

14. Heckerman, D. A tutorial on learning with Bayesian networks. Springer Berlin Heidelberg, 2008, pp. 33–82.

Пантелеев Евгений Рафаилович,
ФГБОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»,
доктор технических наук, профессор кафедры программного обеспечения компьютерных систем,
e-mail: erp@poks.ispu.ru

Архипов Ален Леонидович,
ФГБОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»,
Программист,
e-mail: alain@cmko.ispu.ru

Второв Алексей Владимирович,
ФГБОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»,
преподаватель кафедры программного обеспечения компьютерных систем,
e-mail: wert@vc.ispu.ru