ДИНАМИЧНО ОПРЕДЕЛЯНЕ НА ПЕРСОНАЛИЗИРАНИ УЧЕБНИ ПЪТИЩА

Георги Пашев, Георги Тотков

ПУ „Паисий Хилендарски"
georgepashev@gmail.com, totkov@uni-plovdiv.bg

***Резюме**: В работата се представя софтуерно приложение, осигуряващо персонално и адаптивно електронно обучение. За всеки учебен план приложението съхранява и поддържа данни – примери за неговото изпълнение (спец. пътищата за постигане на учебните цели от различни обучавани). На конкретен обучаван се предлага учебен път за по-ефективно усвояване на учебното съдържание на базата на откриване на сходство с данните за вече приключилите своето обучение. Реализацията използва авторска система за управление на графови бази данни.*

***Ключови думи**: план за обучение, предсказване, път в граф, адаптивно обучение, машинно обучение*

1. Въведение

При моделиране на е-обучение се налага въвеждане на различни модели – изучавана предметна област (ИПО), учебна програма и учебни цели, учебни ресурси (курсове, текстови, мултимедийни и тестови материали), субекти на обучението (обучавани, преподаватели, консултанти, проектанти, автори и др.) и т.н. [2].

Нека S е произволно крайно множество. Семейството от множества $\mathbf{T} = \{T_s : s \in S\}$ се нарича S – множество. То е крайно, ако $|T_s| < \infty$ за всяко $s \in S$ и едноелементно, ако $|T_s| = 1$ за всяко $s \in S$.

Нека $N = \{N_v\}_{v=1}^{n}$ е множество от имена на учебни материали от различен тип, $L = \{L_j\}_{j=1}^{m}$ е множество от всички слоеве, $R = \{r_k\}_{k=1}^{u}$ е множество от всички ресурси.

Модел M_d на ИПО е ориентиран граф $G_d = (V_d, E_d)$, където елементите на множеството V_d са възли, представящи понятията от дадена предметна област (ПО), а елементите на множеството E_d са дъги, определящи връзките от тип „предшественик на" между понятията.

Модел на учебен курс в M_d е ориентиран граф $G_c = (V_c, E_c)$, където: а) $V_c = V_c' \cup B$, множеството от възли V_c' $(V_c' \subseteq V_d, B \cap V_c' = \emptyset)$ представя понятията, изучавани в M_c, а множеството B се състои от четири типа възли –

and, or, yes, not; множеството от дъги $\mathbf{E_C}$ представя връзките между понятията, включени в M_c.

Планът за обучение P по даден учебен курс представлява ориентиран граф $G_p = (V_p, E_p)$, където $V_p = \{v_1, \ldots, v_k\}$ е крайно множество от списъци на учебни материали, а $E_p \subseteq E_c$.[2]

Мрежа на Петри е насочен биграф, в който възлите представляват позиции/преходи. Ребрата са насочени дъги, които описват кои позиции за кои преходи са съответно предусловия и постусловия. Една дъга свързва позиция и преход, или преход и позиция, но не и еднотипни възли. Планът за обучение по даден учебен курс може да се представи като мрежа на Петри, при която възлите ‘позиция’ са тестови единици или друга (вложена) мрежа на Петри (в частност – план за обучение, учебен материали за преглеждане и усвояване и т.н.). При зададени предусловия и постусловия, обучението може да протече по различни пътища на мрежата.

GDBMS е авторска система за управление на граф-базирани системи за управление на бази от данни (ГСУБД), която поддържа насочен хипергграф (с атрибути и етикети), управление на сесии, транзакции, потребителски права върху графа и др. Освен това притежава средства за дефиниране на типове данни; създаване на графи; въвеждане, промяна и премахване на възли и ребра в графи; заявки за данни от даден граф (напр. намиране на най-кратък път, най-дълъг път, намиране на всички пътища между дадени два възела, намиране на съседни възли/ребра на текущ възел/ребро и др.). GDBMS има вграден собствен скриптов интерпретатор на авторски език от процедурен тип. Възможно е да се дефинират различни тригери преди определени събития, например стартиране на сесия или на транзакция върху граф, въвеждане на възел/ребро, промяна на възел/ребро и изтриване на възел/ребро. Потребителските тригери могат да модифицират данни в тези възли/ребра, както и да откажат действието (с изпълнение на команда *reset*), ако това е необходимо в съответното приложение. Езикът има собствен оператор „потоков идентификатор”, който в комбинация с оператор *foreach* се използва за писане на заявки към GDBMS. Последната версия на GDBMS поддържа режими на комуникация с клиентски приложения SOAP service и CGI. Възможно е приложението да бъде извикано и локално – без отдалечена комуникация. Разработчиците на приложения върху GDBMS могат да избират дали да използват директно уеб методи на ГСУБД или да използват уеб метод на интерпретатора. Реализацията е в C/C++, програмна среда Qt, Linux [4].

В работата се предлага методика за определяне и предлагане на най-подходящ път в плана на даден обучаван чрез използване на акумулирани към момента данни за учебни пътища на вече завършили обучавани, близки до него по определени параметри.

2. Използван подход

Тук ще изложим основните ключови моменти в предложената методология за прогностично групиране на обучавани на базата на учебни пътища.

2.1. Моделиране на план на обучение

В GDBMS дефиницията на типовете данни на възли/ребра на графа се осъществява с езиковата конструкция за дефиниция на клас [4].

```
class clLectures
    int id
    string displName
    string displId
    int numCredits
    string urlResources
endclass
```

Фигура 1. Примерна декларация на клас данни за възли

По аналогичен начин се дефинират класовете данни за упражнения, задачи, тестове. Възлите, които отговарят на възли – преходи в мрежа на Петри са с тип данни, дефиниран по следния начин [4]:

```
class clTransition
    fun preCondition
    fun postCondition
endclass
```

Фигура 2. Декларация на клас данни за възли от тип 'Преход'

Вградената в езика на GDBMS ключова дума *fun* указва, че членовете за предусловие и постусловие са променливи от тип 'делегати' към дефинирани вече подпроцедури в скриптовия файл, които обработват условията за преходите от една позиция към друга в мрежата на Петри в този етап на изпълнение, когато вече е създадена инстанция на крайния автомат и се извършва преход.

Зареждането на данните за лекции, упражнения, задачи и тестове в програмната среда става чрез HTTP GET заявки към уеб сървърни скриптове, които предоставят JSON масиви от данни с намиращите се във външната База от данни (БД) лекции, тестови единици за даден курс и период на обучение.

```
   arrayLectures=json_loadArray(http_get(ulr_array_script
+"?discpl_id="+str(discpl_id)+"&semester_id="+str(semest
er_id)))
```

```
      ;arrayLectures ••••••• ••••• • •••••• •• ••• 
clLectures
      ;•••••• ••••••••• •• ••••• •• •••••••••• • ••••••• 
••••••
      i=0
      while(i<count(arrayLectures))
            insert_node(arrayLectures[i])
            i=i+1
      endwhile
```

Фигура 3: Извличане на данни за лекции от БД и създаване на възли за тях в графа

По аналогичен метод се зареждат и данни за другите типове позиции в мрежата на Петри.

Въвеждане на ребра от тип ‘преход’ и връзките им с възлите от типовете на позиции дооформя учебния план. Зареждането на тези данни в програмната среда отново става чрез HTTP GET извикване на специално подготвен за целта уеб сървърно приложение – скрипт, който извлича тези данни от оригиналната БД и създава удобен за целта json форматиран изход.

```
      arrayLinks=json_loadArray(http_get(ulr_array_links
+"?spec_id="+str(spec_id)+"&semester_id="+str(semester_i
d)))
      ;arrayLinks ••••••• ••••• • ••••••
      ;•••••• ••••••••• •• ••••• • ••••• (••••) •• 
••••••••• • ••••••• ••••••
      i=0
      while(i<count(arrayLinks))
            nodeFrom=getnode("*.id",
arrayLinks[i].idFrom)
            nodeTo=getnode("*.id", arrayLinks[i].idTo)
            if ^(exists(nodeFrom) and exists(nodeTo))
                  ;errormessage
                  reset
            endif
            clTransition transition
            ;•••••••••••••• preCondSub •
postConditionSub •• •• ••••••••• • ••••••••• ••
•••••••••••• •• ••••••••• • ••••••• •••••••
            transition.preCondition="preCondSub"
            transition.postCondition="postConditionSub"
            insert_edge(transition, nodeFrom, nodeTo)
      endwhile
```

Фигура 4: Извличане на данни за преходите в графа

Подпрограмите, които се грижат за определяне на допустимостта на даден преход в зависимост от текущия възел от тип позиция в мрежата на Петри се дефинират на езика на GDBMS и са с имена, посочени в предусловието и постусловието на преходите.

Новите функции, реализирани във връзка с това в GDBMS са http_get, http_post, json_loadArray, като е вградена и променлива от тип ‘делегат’ в скриптовия език, а командата за извикване на подпрограма call извиква подпрограма през неин делегат.

2.2. Създаване на мрежи на Петри с памет върху план за обучение

Създаването на мрежа на Петри с памет върху граф е нова функционалност за GDBMS. Във връзка с това е въведена нова езикова конструкция в скриптовия език на GDBMS, която описва работата на мрежа на Петри с памет (МПП) върху граф. В конкретното приложение за електронно обучение, МПП се създава и съхранява за всеки обучаван.

МПП върху граф, в контекста на GDBMS, се дефинира с обект от потребителски дефиниран тип данни, както и идентификатор на текущ възел, идентификатор на стартов възел, множество от идентификатори на крайни възли, делегат на подпрограма, която се извиква преди събитието на преход, и делегат на подпрограма, която се извиква след събитието на прехода. Подпрограмата, която се извиква преди прехода може да откаже прехода, ако конкретните условия за неговото изпълнение, разписани в подпрограмата, не са изпълнени. При това положение се запазва старият текущ възел на мрежата на Петри.

В конкретното приложение обектът на мрежата на Петри съдържа атрибути, които се извличат от оригиналната БД и служат за променливи, които впоследствие ще участват в класификация на обучавания в най-подходящата за него група от обучавани.

```
   class studentFsm
            int id
            int facnum
            int idDept   ;1. ••••••• •• ••••••••
            int gradeHighSchool ;2. ••••••••••••••••
•••••• •• ••••••• •• •••••• ••••••••••• {2..6}
            int semesterAverageGrade  ;3.
•••••••••••••••• •••••• •• ••••••••• •• •••• ••••••
{2..6}
            int labAverageGrade ;4. ••••••••••••••••
•••••• •• ••••••••••• ••••••••• {2..6}
            int seminarAverageGrade   ;5.
•••••••••••••••• •••••• •• ••••••••• ••••••••• {2..6}
```

```
            int lecturesPresence      ;6. ••••••••••••
•• •••••• {1 .. 4}
            int homeworkAverageGrade  ;7.
•••••••••••••••• •••••• •• ••••••• •••••• {2..6}
            int discpAverageRepetitions      ;8.
•••••••••••••••• •••• ••••••••• •• •••••••••
      endclass
```

Фигура 5: Пример. декларация на тип данни клас за обект в Мрежа на Петри

Извличането на данни за атрибутите на класа от фиг. 5., голяма част от които са обобщаващи функции върху записи в оригиналната БД се извършва чрез динамично извикване на специално създадено за целта сървърно приложение – скрипт, който връща отговора в json формат на клас със структура, отговаряща на структурата на класа, дефиниран на фиг. 5. След това, с извикване на специално създадената интерпретаторска функция json_LoadClass, json низовият входен параметър се десериализира и се връща като променлива.

Следва примерно използване в скриптовия език на GDBMS на езиковата конструкция за мрежа на Петри с памет.

```
   entry load_pnt
            ;;••••••••• •••
            pnt("pnt_stud_"+str(stud_id),
currentStudent,       id_start_node,       ids_end_nodes,
id_cur_node)
                  ;••••••••••    •••••••••    this    •
••••••• • •••••• • ••••••• ••••••••
                  ;•••••••••••          id_start_node,
ids_end_nodes, id_cur_node •• •••••••••••• • •••••• ••
•• ••••• • •• ••••••••• •• endfsm
                  ;••• • •••• •••• ••••••••• ••••• ••
••••• • ••••• • •••• ••• (••••• •••••• •••••••••), ••
••••    ••••••••••    ••••••    ••    ••    ••••••••••••
••••••••••••••, •• •• •• ••••••• ••••••• •••••••
                  ;•••••••••••   ••   •••••••••   •••••
••••••• •• •••••••••• • •••• •••••••
                  beforeChangeState("preCondSub")
                  afterChangeState("postConditionSub")
                  id_next_node = .... ;•••• •• ••••• ••
••••••••••••
                  tryChangeState(id_next_node)

            endpnt

      stop
```

Фигура 6: Пример. Използване на езикова конструкция за мрежа на Петри с памет

2.3. Класификация на обучаван в подходяща група

За да се подберат възможно най-информационните променливи (параметри) за всеки обучаван, които да се ползват при тяхната класификация, най-напред трябва да се направи оценка на информационната печалба за всеки един от тях, като се използват величините ентропия и информационна печалба [1].

Класификация на обучаваните се извършва след като се определят групите, които ще се различават при класификация. След като графът е създаден се определят класификационните групи. Групите са толкова на брой, колкото са възможните пътища в графа – от началния стартов възел, до всеки краен възел. Всички пътища в графа се определят по стандартен начин с вид заявка на GDBMS, която използва алгоритъма allDFS за намиране на всички пътища в граф [4].

```
   while i<count(ids_end_nodes)
              curEnd=ids_end_nodes[i]
              dbconn=sqlite_open("EMSG.db")
              foreach @0[id_start_node]a[curEnd] as
currentPath ;итериране за всички открити пътища,
запаметени в променливата currentPath като масив от
възли
              ;sqlite_query изпълнение на insert заявка
за текущия открит път
                    endforeach
                    sqlite_close(dbconn)

                    i=i+1
              endwhile
```

Фигура 7. GDBMS код (фрагмент) за запазване на информация за пътищата в графа

Данните за пътищата в графа са запазени в релация с атрибут за идентификатор на пътя и съставен атрибут с идентификаторите на възлите за този път. Съставният атрибут не е подложен на нормализация, с цел избягване на ресурсоемки релационни join заявки, генериращи списък от идентификатори на възли в пътя, които заявки биха се изпълнявали често.

В началния етап, когато все още няма достатъчно обучителни примери и нито една МПП на обучаван няма текущ възел, обявен за краен (не е приключил етапа на обучение), класификация не е възможна. Това се случва

едва след като по всеки възможен учебен път е преминал поне един обучаван (за всяка категория има поне по един пример).

Приключване на етапа на обучение, за всеки обучаван се извършва при успешен преход към текущ възел, обявен за краен при дефиницията на мрежата на Петри. Съответният алгоритъм е реализиран в подпрограма *PostConditionSub*., като за целта се използва отново ненормализирана релация със същите атрибути, които има класът от фиг. 5., и отделен съставен атрибут, съдържащ идентификаторите на възлите, през които е преминал обучавания в процеса си на обучение. По този начин за всеки възел от тип ‘място’ в МПП ще има примери на обучавани, преминали през него.

Процедурата за предлагане на най-подходящ преход за текущата МПП е приложима, когато от текущия възел – позиция е възможен повече от един преход и алгоритмът от подпрограмата за предусловие на преходите *preCondSub* не ги забранява.

Нека е наредена n-торка. Ще наричаме X – обучителен пример, когато $x_i, i \geq 1, i \leq n,$ е валидна стойност на атрибут v_i, участващ в множеството от атрибути $V = \{v_i\}_{i=1}^{n}$, определени като атрибути за класификация. $n \in \mathbb{N}$ е броят на тези атрибути.

Предлагането на най-подходящ преход става с намиране на множество $Y = \{X_1, ..X_n\}$ от обучителни примери, с мощност $k = |Y|$, зададена като настройка, които са вече запаметени за текущия възел, за които функцията за изчисление на разстоянията $D(\overrightarrow{X_\tau} - \overrightarrow{X_\mu})$ е минимална, $\tau \geq 1, \tau \leq n, \tau \neq \mu$, където X_μ е обучителен пример на текущ обучаван. От Y се избира този обучителен пример $X_l, i \geq 1, i \leq n$, за който даден целеви атрибут v_i, избран за мярка за успешност на обучаван, има максимална стойност. В множеството V има и подмножество от атрибути $\{v_\kappa\}_\kappa$, $\kappa \geq l, \kappa \leq t, t \geq 1, t \leq n, l \geq 1, l \leq n, l < t$, които имат булев тип данни. $|\{v_\kappa\}|$ е броят на възлите в графа, които в контекста на мрежата на Петри са позиции. За всяка позиция има такъв булев атрибут със стойност true, ако процесът на обучение, на който отговаря обучителният пример, е преминал през тази позиция и false, ако не е преминал през нея, до момента на достигане на текущата позиция на МПП за текущия обучаван.

Функцията за изчисление на разстоянията на радиус-векторите е от вида:

$$D = \sqrt[2]{\Sigma_i (a_i x_i)^2},$$

Необходими условия тази функция да бъде функция на разстояние са следните: $D(x_1,\dots,x_n) \geq 0$ (**неотрицателност**); $D(x_1,\dots,x_n) = 0 \Leftrightarrow x_1 = x_2 = \dots = x_n$ (идентичност на неразличимите); симетрия: $D(X)$ да бъде инвариантно за всички пермутации на X; за всички радиус-вектори в $\mathbb{R}_n$ $\overrightarrow{X_i}$, отговарящи на обучителни примери X_i, трябва да е изпълнено следното: $\sqrt{\sum_{i=1}^{n} D(X_i)^2} \leq \sum_{i=1}^{n} |\overrightarrow{X_i}|$ (триъгълно неравенство).

След като е определен кой е най-успешният обучаван, се предлага този преход, който той е извършил когато неговата МПП е била в тази позиция, в която е МПП за текущия обучаван.

Поради необходимостта от бързи изчисления на близостта на радиус-вектори, същите са реализирани под формата на подходящи паралелни изчисления и са вградени в интерпретатора на GDBMS.

Заключение

С разширение на функционалностите на съществуваща авторска среда и проектирано приложение на нейна основа се предлага решение за подпомагане на избора на обучавани между алтернативи в процеса на тяхното обучение. По този начин чрез метод, подобен на използваните в областта на машинното обучение, се осъществява адаптивно е-обучение, което дава възможност на обучаваните да направят по-добър и информиран избор, а на преподавателите – възможност за проследяване на успеваемостта при следване на определени учебни пътища за обучение с цел усъвършенстване.

Литература

1. Abeer Badr El Din Ahmed, Ibrahim Sayed Elaraby: Data Mining: A prediction for Student's Performance Using Classification Method, World Journal of Computer Application and Technology 2(2): 2014, 43-47
2. Е-обучението в информационното общество: технологии, модели, системи, достъпност и качество (под ред. на Г. Тотков), Пловдивско университетско издателство, Пловдив, 387 стр., 2010, ISBN 978-954-423-651-9.
3. P.M.B. Vitanyi: Information distance in multiples, IEEE Trans. Inform. Theory, 57:4(2011), 2451-2456
4. Georgi Pashev, Dilyana Budakova: SOFTWARE PLATFORM FOR EXECUTION OF GRAPH DATABASE APPLICATIONS, ARTTE Vol. 1, No. 2, 2013, ISSN 1314-8788 (print), ISSN 1314-8796 (online)

DYNAMIC DETERMINATION OF PERSONALIZED EDUCATIONAL PATHS

Abstract: *In this paper a software application, which provides personalized and adaptive e-learning is presented. The application stores and supports data for each educational plan – examples for its execution (especially the paths for completion of educative goals by various students). A specific educated person is proposed a specific path in order to achieve more effective learning, based on a discovery of a similarity within the data of those who already graduated. The implementation makes use of a developed by the author Graph Database Management System.*