POUŽITÍ GENETICKÝCH ALGORITMŮ PRO ANALÝZU NÁVŠTĚVNOSTI WWW PORTÁLU

O. Popelka, J. Šťastný

Došlo: 8. července 2009

Abstract

POPELKA, O., ŠŤASTNÝ, J.: *WWW portal usage analysis using genetic algorithms*. Acta univ. agric. et silvic. Mendel. Brun., 2009, LVII, No. 6, pp. 201–208

The article proposes a new method suitable for advanced analysis of web portal visits. This is part of retrieving information and knowledge from web usage data (web usage mining). Such information is necessary in order to gain better insight into visitor's needs and generally consumer behaviour. By leveraging this information a company can optimize the organization of its internet presentations and offer a better end-user experience. The proposed approach is using Grammatical evolution which is computational method based on genetic algorithms. Grammatical evolution is using a context-free grammar in order to generate the solution in arbitrary reusable form. This allows us to describe visitors' behaviour in different manners depending on desired further processing. In this article we use description with a procedural programming language. Web server access log files are used as source data.

The extraction of behaviour patterns can currently be solved using statistical analysis – specifically sequential analysis based methods. Our objective is to develop an alternative algorithm.

The article further describes the basic algorithms of two-level grammatical evolution; this involves basic Grammatical Evolution and Differential Evolution, which forms the second phase of the computation. Grammatical evolution is used to generate the basic structure of the solution – in form of a part of application code. Differential evolution is used to find optimal parameters for this solution – the specific pages visited by a random visitor. The grammar used to conduct experiments is described along with explanations of the links to the actual implementation of the algorithm. Furthermore the fitness function is described and reasons which yield to its' current shape. Finally the process of analyzing and filtering the raw input data is described as it is vital part in obtaining reasonable results.

genetic algorithms, data mining, behaviour patterns, www portal

ÚVOD A CÍL PRÁCE

V současnosti probíhá značné množství komunikace mezi dodavatelem a zákazníkem prostřednictvím internetu. Jedna z klíčových služeb sítě internet jsou i webové prezentace, a tedy i webové stránky komerčních společností. Tyto prezentace by měly být navrženy a optimalizovány tak, aby jak stávající, tak noví zákazníci mohli efektivně nalézat požadované informace. Tyto optimalizace není možné provádět bez dostatečných informací o požadavcích zákazníků.

Důležitým zdrojem informací je analýza použití webové stránky (web usage mining). Jedná se o proces, jehož cílem je získání lepšího přehledu o požadavcích návštěvníka webové prezentace a všeobecně získání informací o chování zákazníka – spotřebi-

tele. Tato úloha je v současnosti řešitelná pomocí metod založených na sekvenční analýze, např. (Munk a Balogh, 2009). Cílem této práce je vytvoření alternativní automatizované metody schopné identifikovat v zadaných datech vzory chování.

METODY

Výchozí použitá metoda je dvoufázová gramatická evoluce (TLGE) (Ošmera, Popelka a Pivoňka; 2008). Hlavní výhodou této metody je schopnost generovat funkce v libovolném formálním jazyce zpracovatelném bezkontextovou gramatikou. Kromě toho umožňuje dvoufázová gramatická evoluce optimalizaci parametrů z oboru celých nebo reálných čísel, což není možné efektivně dosáhnout se standardní

Gramatickou evolucí (Dempsey, 2007) a (O'Neill a kol., 2003). Popsaná metoda byla již úspěšně aplikována k řešení automatického generování matematických modelů (Popelka a Štastný, 2007).

Všeobecně je modelování chování uživatele složitá úloha, k jejímuž řešení byla vyvinuta řada různých metod. Často je nutné pracovat s neurčitostí, a proto velkou skupinu těchto metod tvoří stochastické nebo pravděpodobnostní modely (Iwata, Saito a Yamada; 2007). Tyto metody vyžadují vytvoření empirického modelu, který může být reprezentován různými způsoby. V této práci je pro reprezentaci použit zvolený procedurální programovací jazyk. Volba je výhodná zejména s ohledem na další plánované využití výsledků.

Gramatická evoluce

Gramatická evoluce (O'Neill and Ryan, 2003) je modifikace genetického algoritmu (GA) (Goldberg, 2002) a má tedy shodné základní vlastnosti i operace. Kromě standardních operací, kterými jsou selekce, křížení a mutace (Mitchell, 1999), využívá gramatická evoluce překladač bezkontextové gramatiky. S jeho využitím je možné generovat řetězce v zadaném formálním jazyce.

Bezkontextová gramatika G je definována jako čtveřice $G = (II, \Sigma, P, S)$, kde II je množina neterminálů, Σ je množina terminálů, S je počáteční neterminál a P je tabulka přepisovacích pravidel. Příklad gramatiky použité při řešení úlohy ukazuje tabulka I. Přepisovací pravidla jsou uvedena v tabulce II a III.

Vzhledem k tomu, že je použita modifikace gramatické evoluce – gramatická evoluce se zpětným překladem (Ošmera, Popelka a Panáček; 2005) – musí použitá gramatika splňovat jisté požadavky. K jejich popisu je nutné rozdělit neterminály do dvou skupin následujícím způsobem.

I: Definice gramatiky

$\Pi = \{cond, dummyt, ifthen, ifthent, dummy, op, var, visitedt\}$
$\Sigma = \{or, and, visited, if, inert, c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9\}$
$S = dummy$

II: Přepisovací pravidla typu T

$\langle var \rangle ::=$	$c_0 \mid c_1 \mid c_2 \mid c_3 \mid c_4 \mid c_5 \mid c_6 \mid c_7 \mid c_8 \mid c_9 \mid$	$\langle op \rangle ::=$	$and \mid or$
		$\langle ifthen \rangle ::=$	if
		$\langle visitedt \rangle ::=$	$visited$
		$\langle dummy \rangle ::=$	$inert$

Neterminál typu T je neterminál, který se může být substituován pouze terminály. Odpovídající pravidla použitelná k překladu tohoto terminálu jsou T pravidla. T pravidlo má na pravé straně pouze terminály a to ve všech možných substitucích (Tabulka II).

Neterminál typu N je neterminál, který může být přepsán pouze na další neterminály. N pravidlo je pravidlo, které má na pravé straně pouze neterminály (Tabulka III).

III: Přepisovací pravidla typu N

$\langle ifthen \rangle ::=$	$\langle ifthen \rangle \langle var \rangle \langle cond \rangle$
$\langle cond \rangle ::=$	$\langle op \rangle \langle cond \rangle \langle cond \rangle \mid \langle visitedt \rangle \langle var \rangle$
$\langle dummy \rangle ::=$	$\langle dummyt \rangle \langle dummy \rangle \langle dummy \rangle \mid \langle dummyt \rangle \langle dummy \rangle \langle ifthen \rangle \mid \langle dummyt \rangle \langle ifthen \rangle \langle ifthen \rangle \mid$

Gramatika vhodná pro zpracování algoritmem zpětného překladu smí obsahovat *pouze* pravidla typu (a neterminály) T nebo N. Toto omezení znamená, že smíšená pravidla, nebo pravidla obsahující literály nejsou pro algoritmus použitelná – například:

$\langle ifthen \rangle ::= if \langle cond \rangle then \langle expr \rangle$

Tato pravidla je však možné přepsat na dvě nebo více pravidel s využitím dalších neterminálů a pravidel s jedinou substitucí. Použití takto omezené gramatiky umožňuje využití progresivního algoritmu křížení (Ošmera, Popelka a Panáček; 2005), který má významný pozitivní vliv na výkon gramatické evoluce.

Příklad a generování řešení

Produkční pravidla jsou klíčovou součástí definice gramatiky, protože definují strukturu generovaného řešení. Je zřejmé, že jednotlivé neterminály mohou být přepsány mnoha různými způsoby. Například neterminál $\langle op \rangle$ může být přepsán na dva různé řetězce, které reprezentují logické operátory *and*, a *or*. Vlastní proces překladu je tedy nejednoznačný a je řízen chromozomem jedince. V gramatické evoluci je chromozom jedince reprezentován náhodným vektorem celých čísel. Proces překladu začíná zpracováním počátečního neterminálu (S).

Pro výběr jednoho konkrétního pravidla pro překlad neterminálu s použitím hodnoty genu se používá operátor modulo. Takto se získá jedno konkrétní pravidlo, které se použije k substituci neterminálu. Proces se opakuje tak dlouho, dokud se v generovaném řešení vyskytnou neterminály. Výsledkem procesu překladu je řetězec, který představuje část zdrojového kódu programu v daném procedurálním jazyce. Proces překladu tedy využívá chromozom jedince k určení jednoznačného postupu při generování řetězce, který vyhovuje za-

dané gramatice. Jakmile je tento proces dokončen, pokračuje gramatická evoluce jako standardní genetický algoritmus (Mitchell, 1999). Generované řešení je předáno kritériální funkci, která řešení vyhodnotí a přiřadí mu hodnotu fitness. Základní vývojový diagram dvoufázové gramatické evoluce zobrazuje obrázek 1.

- *visited* – *function visited(integer): boolean* – funkce, která vrátí hodnotu true v případě, že zadaná stránka (identifikovaná číslem) byla navštívena
- *if* – celá konstrukce if-then
- *inert* – *function inert(boolean): array of integer* – funkce, která slouží k seskupení více podmínek do jedné návratové hodnoty

1: Vývojový diagram dvoufázové gramatické evoluce

Cílovým stavem je následující forma popisu chování návštěvníka:

```

if userVisited(DocumentX) or
  userVisited(DocumentY) then
  userWillProbablyVisit(DocumentZ)
    
```

Jedná se o část zdrojového kódu programu, který je možné dále využít v rámci aplikace, která obsluhuje webový portál například k nabízení vhodných odkazů návštěvníkovi. V gramatice definované výše jsou terminálům přiřazeny následující významy:

- *or*, *and* – logické operátory pro sestavení podmínek

- $c_0 - c_9$ – parametry funkce *visited()* ve formě celočíselných konstant.

Funkce *inert()* slouží k seskupení dvou nezávislých podmínek. Jedná se o to, že k popisu chování návštěvníka nestačí pouze jedna sekvence, ale vzhledem k různorodosti návštěvníků se dá očekávat více různých vzorů. Aktuálně byl tedy zvolen přístup nezávislého sloučení pravidel, bez ohledu na časovou posloupnost návštěv.

Konstanty $c_0 - c_9$ jsou *abstraktní konstanty* – to znamená, že jejich hodnoty jsou neznámé v době, kdy je řešení generováno a jejich hodnota je určena až dodatečně. Současně však mají v rámci jednoho řešení konstantní hodnotu. Mezi různými jedinci se však jejich hodnota může měnit. V žádném jedinci

se nemůže vyskytovat více než 10 různých konstant. Naopak platí, že nemusí být použity všechny konstanty. Abstraktní konstanty je možné interpretovat jako parametry řešení, řešení bez těchto konstant pak reprezentuje obecný vzor chování, který by neměl být specifický pro konkrétní data.

Dvoufázová gramatická evoluce

Při práci s parametrizovaným řešením je vhodné proces optimalizace rozdělit do dvou kroků. Nejprve nalézt vhodnou (v ideálním případě optimální) strukturu, která celkově reprezentuje charakteristiky dat. Ve druhém kroku se pak hledají optimální hodnoty parametrů pro tuto strukturu tak, aby co nejlépe odpovídaly danému datovému souboru. Gramatická evoluce je sama o sobě schopná řešit obě části úlohy (O'Neill et al., 2003), ale jako efektivnější se ukázalo zpracovávat obě podúlohy odděleně s využitím dvoufázové gramatické evoluce (Ošmera, Popelka a Pivoňka; 2008). To je způsobeno tím, že gramatika nutná pro generování libovolného čísla (s více jak jednou cifrou) je relativně složitá (O'Neill et al., 2003) a často její složitost přesáhne složitost gramatiky pro generování celého modelu (Tabulky I–III). Pro druhou fázi optimalizace byla použita *Diferenciální evoluce* (DE) (Price, 1996).

Diferenciální evoluce je také modifikací genetického algoritmu, je to poměrně jednoduchá a rychlá metoda specificky navržená k optimalizaci parametrů v oboru reálných čísel (Price, Storn and Lampinen; 2005).

První fáze optimalizace je řešena pomocí gramatické evoluce, která generuje řešení podle zadané gramatiky (Tabulka I–III). Výstup může obsahovat několik podmíněných výrazů a současně také může obsahovat abstraktní konstanty ($c_0 - c_9$), jejichž hodnota není známa:

```
if userVisited(c0) or userVisited(c1)
then
    userWillProbablyVisit(c9)
```

Tato funkce nemůže být vyhodnocena a je nutné ji předat do druhé fáze optimalizace. Ve druhé fázi optimalizace je použita diferenciální evoluce. Vstupem pro druhou fázi je vektor proměnných, které odpovídají konstantám v gramatické evoluci (c_0, c_1, c_9). Diferenciální evoluce najde s využitím svého genetického algoritmu optimální hodnoty těchto proměnných tím, že z počátečního náhodného stavu pomocí operátorů křížení a mutace konverguje do stavu s lepším hodnocením řešení. Po zadaném počtu cyklů jsou hodnoty proměnných považovány za "vhodné" a jsou předány zpět gramatické evoluci a dosazeny jako hodnoty konstant v řešení. V optimalizaci parametrů je možné pokračovat v dalším cyklu gramatické evoluce, kdy u existujících jedinců nejsou počáteční hodnoty konstant voleny náhodně, ale jsou to právě „vhodné“ hodnoty z předchozího cyklu.

Vývojový diagram dvoufázové gramatické evoluce ukazuje Obrázek 1. Celý algoritmus se skládá ze dvou vnořených cyklů. Vnitřní cyklus imple-

mentuje genetický algoritmus diferenciální evoluce (schéma DE/rand/1) a vnější cyklus reprezentuje hlavní cyklus gramatické evoluce. To znamená, že pro vypočítání hodnoty kritériální funkce v gramatické evoluci je nutné vytvořit novou populaci v diferenciální evoluci, předat jí neznámé parametry jako vektor proměnných (každý jedinec může mít jiný počet proměnných). Tyto parametry jsou pak optimalizovány prostřednictvím diferenciální evoluce a následně vloženy zpět do jedince v gramatické evoluci. Tento algoritmus spojuje výhody obou metod – umožňuje jak generování složitých struktur, tak efektivní optimalizaci jejich parametrů.

Kritériální funkce

Použitý algoritmus umožňuje pracovat s vektorovou hodnotou kritériální funkce – vektorovým fitness (Popelka and Ošmera, 2008). Každá složka vektoru popisuje jinou kvalitativní stránku generovaného řešení. Pro určení pořadí jednotlivých hodnot fitness jsou tyto hodnoty porovnávány s využitím systému pravidel, který zajišťuje konvergenci algoritmu k požadovanému typu řešení. Použitá definice fitness n -tého jedince je následující:

$$F_n = \left(\sum_{k=1}^n \left(\sum_{j=1}^d M_{jk}^n \right), \sum_{k=1}^n \left(\sum_{j=1}^d H_{jk}^n \right), C^n, I^n \right). \quad (1)$$

Skládá se ze čtyř složek, kde d je počet položek v datovém souboru (počet zaznamenaných sezení), $M = 1$, pokud daná podmíněná konstrukce k aktuálně hodnoceného jedince n vrací správnou hodnotu pro položku datového souboru j , v opačném případě je $M = 0$. Tedy každá podmíněná konstrukce je testována na každé položce datového souboru (což je jedno uživatelské sezení, tedy skupina několika požadavků na webový server). Podobně $H = 1$, pokud je testovaná podmíněná konstrukce k aktuálně hodnoceného jedince n odpovídající pro položku datového souboru j , v opačném případě je $H = 0$. To znamená, že pokud je splněna podmínková část IF konstrukce, je vrácena nějaká konkrétní hodnota stránky, kterou by uživatel měl navštívit. Pokud podmíněná část splněna není, tak není pravidlo na daný datový soubor aplikovatelné. C^n odpovídá počtu uzlů v interní reprezentaci řešení; což je hodnota, která slouží k hodnocení složitosti daného řešení.

Porovnávací funkce může být zapsána v následujícím pseudokódu:

```
Compare(F[n][0]*F[n][1], F[n+1][0]*F[n+1][1])
if (Equals) then
    Compare(F[n][2], F[n+1][2])
    if (Equals) then
        Compare(F[n+1][3], F[n][3])
```

$F[n][0]$ je hodnota první složky vektoru fitness hodnoty n -tého jedince. Takto definovaná funkce vede genetický algoritmus k maximalizaci počtu správných hodnot, maximalizaci počtu položek, pro které je pravidlo aplikovatelné, maximalizace

počtu podmíněných konstrukcí a minimalizaci složitosti každé konstrukce.

Počet podmíněných konstrukcí je maximalizován proto, aby bylo podpořeno hledání co nejsložitějších vzorů. Na druhou stranu je ale žádoucí, aby jednotlivá pravidla v těchto vzorech byla reprezentována co nejjednodušeji, a proto je celková složitost řešení minimalizována. Všechny procesy probíhají v populaci paralelně.

VÝSLEDKY A DISKUSE

Jako testovací zdrojová data byly použity záznamy přístupů na webový server známého časopisu z oboru IT (registrováno přibližně 100 tis. čtenářů). Surová data představuje kompletní výpis přístupů ze webového serveru Apache za celý náhodně vybraný pracovní den. To představuje celkem 156819 HTTP požadavků. Tyto záznamy byly vloženy do relační databáze, kde jsou zpřístupněny gramatické evoluci. Před vlastním zpracováním byla data normalizována následujícím způsobem.

5453 požadavků bylo odstraněno, protože HTTP status kód byl jiný než "200 OK", jedná se tedy o požadavky na neexistující nebo nepřístupné stránky. 5354 záznamů bylo odstraněno, protože pocházejí od automatických robotů (googlebot, yahoo, seznam, iSilo) procházejících celé webové stránky. Následně byly identifikovány sezení uživatele na základě kombinace hlavičky User-Agent (UA) (identifikace prohlížeče, operačního systému, včetně verzí a případných dalších komponent) a IP adresy (nebo reverzního DNS záznamu), přičemž musí být splněno, že mezi požadavky neexistuje prodleva delší než 20 minut. Tento čas byl stanoven na základě předběžné analýzy dat, která ukázala, že prodleva mezi požadavky s danou kombinací UA a IP je vždy buď menší než 20 minut, nebo větší než několik hodin. Existuje tedy zřejmá hranice konce sezení uživatele. Po identifikaci sezení byly odstraněny přístupy ke stránkám, které neobsahují články a jiné dokumenty, jedná se o: obrázky (98790), kaskádové styly (13634), skripty (19535), a speciální stránky, jako jsou výsledky hledání, formuláře, RSS kanály, favicon, etc. (~3500)). Dále bylo odstraněno 4479 záznamů, protože vytvářely sezení pouze s jedním požadavkem, tedy sezení nepoužitelné k identifikaci vzoru chování. Výsledný soubor dat tedy obsahuje 5896 požadavků na stránky s 1309 unikátními navštívenými dokumenty a 992 sezeními. Průměrná

délka sezení je 5.94 požadavků a maximální délka sezení je 179 požadavků.

Jedno sezení uživatele reprezentuje jednu položku datového souboru, která je použita pro výpočet fitness podle předchozí kapitoly. Kromě vstupních dat je nutné do algoritmu zadat také gramatiku (viz tabulky I–III) a také implementovat funkci pro porovnávání fitness. Kromě těchto vstupů pracuje systém zcela automaticky. Níže jsou uvedeny náhodně vybrané vzory z testovací populace (a – j odpovídá konstantám c_0 – c_9).

```
(M) if visited(i[1]) then i[1]
(X) if visited(c[2044]) or
    visited(b[1575]) then i[1]
(X) if visited(i[1]) and
    visited(c[2044]) then d[1646]
(M) if visited(b[1575]) then b[1575]
(X) if visited(g[1748]) or
    visited(a[2276]) then c[2044]
(M) if visited(j[2326]) or
    (visited(i[1]) and
    visited(c[2044])) then g[1748]
(X) if visited(d[1646]) and
    visited(d[1646]) then h[1405]
```

Podmínky s označením (X) nevrací správnou hodnotu. Zbýlé podmínky jsou z velké části triviální zjištění interpretovatelné jako "Pokud uživatel navštívil domovskou stránku, tak uživatel jistě navštíví domovskou stránku." (pozn: domovská stránka má ID = 0). Pouze šestá uvedená podmínka je netriviální a současně platná a identifikuje nějaký vzor chování. Zahlcení algoritmu triviálními řešeními je způsobené nedostatkem v implementaci i v normalizaci dat, kdy nebyly odstraněny shodné požadavky v rámci jednoho sezení a současně je algoritmu umožněno tato řešení generovat.

Práce popisuje použití dvoufázové gramatické evoluce k identifikaci vzorů chování v návštěvách webových stránek. Byla navržena gramatika a implementovány dílčí složky algoritmu a systém byl otestován na skutečných datech. Z analýzy výsledků vyplynula řada významných poznatků a informací týkajících se implementace algoritmu. Konkrétně je nutné zejména upravit systém tak, aby se nezahlocoval triviálními řešeními. Současně však bylo dosaženo dílčích úspěchů, které ukazují, že algoritmus je v zásadě schopen požadované řešení nalézt. Jedná se samozřejmě zatím pouze o dílčí část řešení, kterou je do budoucna nutné rozšířit zejména o práci s časovou posloupností požadavků.

SOUHRN

Článek popisuje novou metodu navrženou pro pokročilou analýzu návštěv webového portálu. Jedná se o část procesu získávání informací a znalostí z dat o použití webové prezentace. Tyto informace jsou nezbytné k lepšímu poznání potřeb a požadavků návštěvníka, všeobecně tedy k poznání zákazníka. S využitím těchto informací může organizace optimalizovat svoje webové prezentace a nabídnout tak koncovým uživatelům větší pohodlí a snadnější přístup k informacím. Navrhovaná metoda využívá gramatickou evoluci, což je výpočetní metoda založená na genetickém algoritmu.

Popsaná úloha je řešitelná pomocí statistické analýzy – konkrétně metodami založenými na sekvenci analýze. Cílem této práce je popsat alternativní metodu pro identifikaci vzorů chování.

Tato práce používá reprezentaci s využitím procedurálního programovacího jazyka, která je vhodná pro další použití jako součást obslužné aplikace webového portálu. To je významná motivace pro vývoj alternativní metody a použití právě gramatické evoluce. Výhoda použití gramatické evoluce je v tom, že vyhovorené řešení již není na výpočetním systému nijak závislé.

Gramatická evoluce je genetický algoritmus rozšířený o překládací vrstvu, vloženou mezi chromozom jedince a skutečné řešení úlohy. Tato vrstva je tvořena překladačem bezkontextové gramatiky. Hlavní výhodou tohoto rozšíření je schopnost generovat řešení v dále použitelném formátu v libovolném formálním jazyce.

Algoritmus používá tzv. vektorové fitness, které umožňuje implementaci i velmi složitých podmínek hodnocení řešení. Každá složka vektoru hodnoty kritériální funkce popisuje jinou kvalitativní složku řešení. Tyto vektory jsou pak porovnávány s využitím definovaných pravidel.

Vstupní data do systému jsou záznamy o jednotlivých HTTP požadavcích na webový server. V těchto záznamech není jednoznačně identifikován konkrétní návštěvník, pro identifikaci je tedy použita kombinace IP adresy a identifikace webového prohlížeče. Jako testovací zdrojová data byly použity záznamy o přístupech na webový server známého IT časopisu. Surová data představují záznamy za jeden konkrétní náhodně vybraný pracovní den. Implementace algoritmu a testování na těchto datech vedlo k získání mnoha poznatků. Zejména je nutné upravit implementaci tak, aby se zabránilo generování triviálních řešení. Dále bude zřejmě vhodnější použít alternativní postup, kdy je genetickým v rámci jednoho řešení generována pouze jedna podmínka a následně tyto podmínky seskupit až dodatečně.

genetické algoritmy, dolování dat, vzory chování, webový portál

SUMMARY

The article describes new method usable for advanced analysis of web portal visits. This is part of retrieving information and knowledge from web usage data – web usage mining. Obtaining such information is necessary in order to gain better insight into visitor's needs and generally consumer behaviour. By leveraging this information a supplier can optimize its internet presentations and offer a better end-user experience.

The proposed approach is using Grammatical evolution which is stochastic computational method based on genetic algorithms. Grammatical evolution is using a context-free grammar in order to generate the solution in arbitrary reusable form. This allows us to describe visitors' behaviour in different manners depending on desired further processing

Grammatical evolution is basically a genetic algorithm extended with a translation layer inserted between the chromosome and the actual solution which is formed by a processor of context-free grammar. The main advantage of such extension is the ability to create generic structures defined in a formal language and retrieve them in reusable format.

The fitness function is key part of grammatical evolution since it is the main source of convergence information for the genetic algorithm. There are several options how to compute fitness and compare different hypotheses produced by genetic algorithm. The algorithm allows the use of vector fitness. Instead of using a scalar real value for fitness representation a vector is used.

Web server access log files are as source data. It is possible to use for example the combination of IP address and user-agent string to identify a single visitor. This task can currently be solved using sequential analysis based methods. In this paper another algorithm to identify access patterns is proposed.

As a source data we used web server access log from a website of a well known IT and computer science magazine. Raw data is represented by a complete access log from a randomly chosen working day.

Testing runs and analysis of the results yielded a lot of important information. Namely the implementation of helper functions must be modified so that trivial solutions are not generated and so that they do not clutter the result set.

Tento článek vznikl v rámci řešení projektu 116/2102/IG190651 – Webové rozhraní pro aplikace genetického algoritmu a výzkumného záměru – MSM 6215648904/03.

LITERATURA

DEMPSEY, I., O'NEILL, M., BRABAZON, A., 2007: Constant creation in grammatical evolution, Int. J. Innovative Computing and Applications, Vol. 1, No. 1, pp. 23–38.

GOLDBERG, D. E., 2002: The Design of Innovation: Lessons from and for Competent Genetic Algorithms. Boston: Kluwer Academic Publishers. 272 p. ISBN 1-4020-7098-5.

- IWATA, T., SAITO, K., YAMADA, T., 2007: Modeling User Behavior in Recommender Systems based on Maximum Entropy. In proceedings of the Sixteenth International World Wide Web Conference (WWW2007), Banff, Alberta, Canada. pp. 1281–1282, ACM Press, ISBN:978-1-59593-654-7.
- MITCHELL, M., 1999: An Introduction to Genetic Algorithms. MIT Press, Cambridge MA. 162 p. ISBN 0-262-63185-7.
- MUNK, M., BALOGH, Z., 2009: Possibilities of web optimization based on discovered user patterns. In: Žufan, P. Firma a konkurenční prostředí 2009 – 5. část. Brno: MSD, s. r. o., s. 71–78. ISBN 978-80-7392-088-3.
- O'NEILL, M., DEMPSEY, I., BRABAZON, A., RYAN, C., 2003: Analysis of a Digit Concatenation Approach to Constant Creation In proceedings of the European Conference on Genetic Programming, (EuroGP), Essex, UK. p. 173–182. ISBN 3-540-00971-X.
- O'NEILL, M., RYAN, C., 2003: Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language, Kluwer Academic publishers. 160 p, ISBN 1-4020-7444-1.
- OŠMERA, P., POPELKA, O., PANÁČEK, 2005: T. Grammatical Evolution with backward processing. In proceedings of 12th Zittau East-West Fuzzy Colloquium. Zittau. p. 235–244. ISBN 3-9808089-6-3.
- OŠMERA, P., POPELKA, O., PIVOŇKA, P., 2008: Two Level Parallel Grammatical Evolution. In: 2008 IEEE World Congress on Computational Intelligence. 6. vyd. Hong Kong: IEEE Press, ISBN 978-1-4244-1823-7.
- POPELKA, O., OŠMERA P., 2008: Parallel Grammatical Evolution for Circuit Optimization In proceedings of The 8th International Conference on Evolvable Systems: From Biology to Hardware (ICES 2008) September 21–24. Prague, Czech Republic.
- POPELKA, O., ŠŤASTNÝ, J., 2007: Generation of mathematic models for enviromental data analysis. Management si Ingerie Economica. sv. 6, č. 2A, s. 61–66. ISSN 1583-624X.
- PRICE, K., 1996: Differential evolution: a fast and simple numerical optimizer, In proceedings of 1996 Biennial Conference of the North American Fuzzy Information Processing Society, NAFIPS, pp. 524–527, IEEE Press, New York, NY, 1996. ISBN: 0-7803-3225-3.
- PRICE, K. V., STORN, R. M., LAMPINEN, J. A., 2005: Differential evolution – a practical approach to Global Optimization. NCS Springer. 543 p., ISBN 3-540-20950-6.

Adresa

doc. RNDr. Ing. Jiří Šťastný, CSc., Ing. Ondřej Popelka, Ústav informatiky, Mendelova zemědělská a lesnická univerzita v Brně, Zemědělská 1, 613 00 Brno, Česká republika, e-mail: jiri.stastny@node.mendelu.cz, xpopelka@node.mendelu.cz