Ján Suchal

Kompresia CSS genetickým algoritmom

Prípadová štúdia predmetu

Evolučné algoritmy

FIIT STU

jún 2006

Zadanie

Pomocou genetického algoritmu sa pokúste minimalizovať (kompresovať) veľkosť štýlových predpisov v jazyku CSS, pričom vnútorná sémantika štýlového predpisu sa nezmení. Experimentujte s parametrami ako elitárstvo, pravdepodobnosť mutácie a kríženia, či veľkosť populácie. Dosiahnuté výsledky experimentov analyzujte a zdokumentujte.

Úvod

Táto práca sa venuje minimalizácii (kompresii) štýlových predpisov jazyka CSS, ktorý je okrem iného určený na popis vizuálnej štruktúry HTML dokumentov. Vzhľad je v jazyku CSS je definovaný v tzv. štýlových predpisoch sadou pravidiel a konštrukcii. Tie je možné väčšinou zapísať rôznymi spôsobmi s rovnakou sémantikou. Táto práca pomocou genetického algoritmu rieši problém hľadania zápisu daného štýlového predpisu s minimálnou veľkosťou[1] a rovnakou sémantikou.

Možnosti kompresie jazyka CSS

Jazyk CSS slúži okrem iného na definovanie vizuálneho vzhľadu HTML dokumentov. Konkrétne časti HTML dokumentu sú identifikované takzvanými selektormi. Pre tieto selektory sú následné definované pravidlá definujúce vizuálne vlastnosti.

```
.normal {
     font-family: serif;
     background-color: white;
     color: black;
}

.extra {
     font-family: serif;
     background-color: white;
     color: red;
}
```

V predchádzajúcej ukážke krátkeho štýlového predpisu je možné vidieť dva selektory (`.normal` a `.extra`), tri rôzne vlastnosti (`font-family`, `color` a `background-color`) a štyri rôzne hodnoty vlastností (`serif`, `black`, `white` a `red`). Tento štýlový predpis má dĺžku 155 znakov.

Tento štýlový predpis je možné zapísať aj kratším spôsobom – využitím zoskupenia selektorov. Vlastnosti (`font-family` a `background-color`) a ich hodnoty sú pre oba selektory rovnaké a tak sa ich zoskupením ušetrí časť znakov. Nasledujúci štýlový predpis s rovnakou sémantikou má už len 132 znakov.

1 Veľkosťou štýlového predpisu rozumieme počet znakov, ktorými je zapísaný.

```
.normal, .extra {
     font-family: serif;
     background-color: white;
}

.normal {
     color: black;
}

.extra {
     color: red;
}
```

Aj predchádzajúci predpis je možné ešte skompresovať. Využitím prekrývania selektorov dostávame zápis s dĺžkou len 114 znakov.

```
.normal, .extra {
     font-family: serif;
     background-color: white;
     color: black;
}

.extra {
     color: red;
}
```

Pri prekrývajúcich sa selektoroch platí zásada, že neskoršie zapísané vlastnosti majú väčšiu prioritu[2]. Preto bude mať selektor `.extra` nakoniec vlastnosť `color` rovnú `red` a nie `black`.

Genetický algoritmus

V genetickom algoritme je množstvo voliteľných parametrov. Najdôležitejšie z nich sú pravdepodobnosti mutácie a kríženia ako aj forma výberu jedincov a veľkosť populácie. V implementácii sú použité aj ďalší koncept: elitárstvo. Elitárstvo je percentuálny podiel najlepších jedincov prenášaných automaticky do ďalšej generácie.

Ako forma výberu bola v implementácii použitá „ranking“ metóda, kde sú jedinci najprv usporiadaný podľa zostupne podľa fitnes a potom očíslovaný postupne od N po 1. Pravdepodobnosť výberu jedinca ruletou je priamo úmerná tomuto očíslovaniu. Takéto hodnotenie umožňuje aj urýchlenie výberu z algoritmického hľadiska a je názorne vysvetlené v prílohe A.

2 V špecifikácii jazyka CSS sa priorita počíta zložitejšie cez tzv. špecifičnosť selektora. V tejto práci sa však používa iba jeden typ selektoru a tak je rozhodujúce len poradie.

Reprezentácia CSS v genóme

Štýlový predpis CSS je v implementácii reprezentovaný ako postupnosť vlastností (vlastnosť a jeho hodnota) a príznaky, na ktoré selektory sa má dané pravidlo aplikovať. Ukážkový štýlový predpis je potom možné znázorniť tabuľkou nasledovne:

```
.normal, .extra {
     font-family: serif;
     background-color: white;
     color: black;
}
.extra {
     color: red;
}
```

Vlastnosť	`.normal`	`.extra`
`font-family: serif;`	1	1
`background-color: white;`	1	1
`color: black;`	1	1
`color: red;`	0	1

Každý riadok tabuľky takto zodpovedá riadku štýlového predpisu. Ak sú príznaky pre dva nasledujúce riadky rovnaké je možné tieto dva riadky zoskupiť. V ukážkovom príklade je možné zoskupiť prvé tri riadky.

Príznaky a vlastnosti sú v implementácii reprezentované v konečnom dôsledku ako reťazec čísel, je možné tento reťazec použiť na optimalizáciu genetickým algoritmom.

Genetický algoritmus pre kompresiu CSS

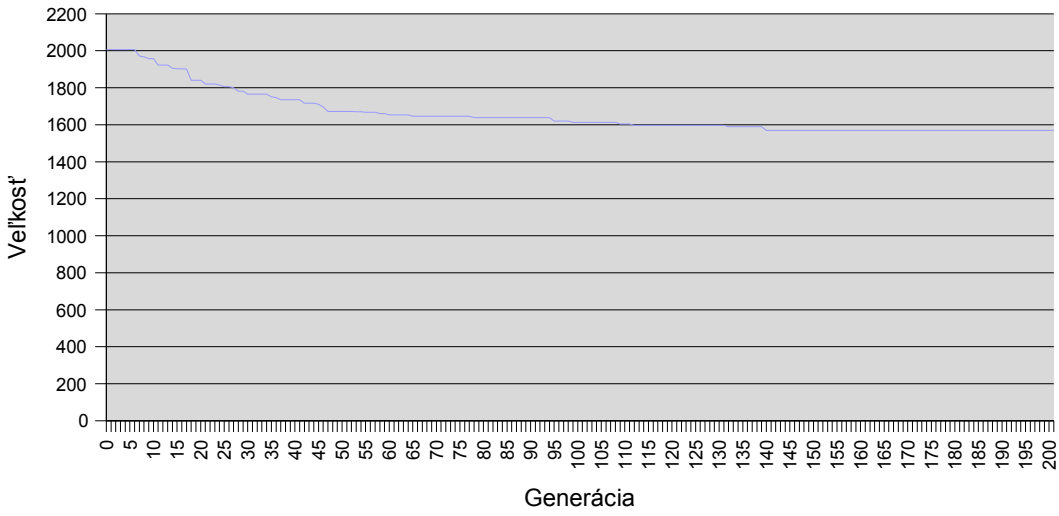
Problém, ktorý sa vyskytol pri opísanej reprezentácii bolo zabezpečenie rovnakej sémantiky náhodne pozmenených (mutovaných a krížených) reťazcov. Pri mutácii sa mohlo úplne vytratiť pravidlo `color:red` na selektore `.extra` a tak sa síce výsledok zdal byť lepší lebo bol v konečnom dôsledku kratší avšak nemal rovnakú sémantiku. Tento problém je možné riešiť dvoma spôsobmi. Zavedením penalizačnej funkcie do ohodnotenia jedincov alebo upravením reťazca, tak aby mal vždy rovnakú sémantiku. V implementácii sa používa druhý spôsob, ktorý bol nazvaný validácia genómu.

Validácia genómu funguje na jednoduchom princípe. Do každého reťazca sú pred ohodnotením doplnené chýbajúce vlastnosti so selektormi zo vzorového štýlového predpisu. Taktiež však môže nastať prípad, že vo vygenerovanom reťazci sa budú nachádzať pravidlá na niektorých selektoroch navyše, t.j. také, ktoré na danom selektore byť nemajú. Analogickým spôsobom sú teda tie vlastnosti, ktoré sa vo vzorovom predpise nenachádzajú, ale v generovanom áno vynechané.

Výsledky experimentov

Priebeh kompresie CSS

V tomto experimente bol skúmaný priebeh kompresie CSS v závislosti od čísla generácie (iterácie). Účelom tohto experimentu bolo zistiť, či priebeh kompresie pripomína známe priebehy z iných aplikácii genetického algoritmu a tak overiť možnosť jeho uplatnenia aj v doméne CSS.

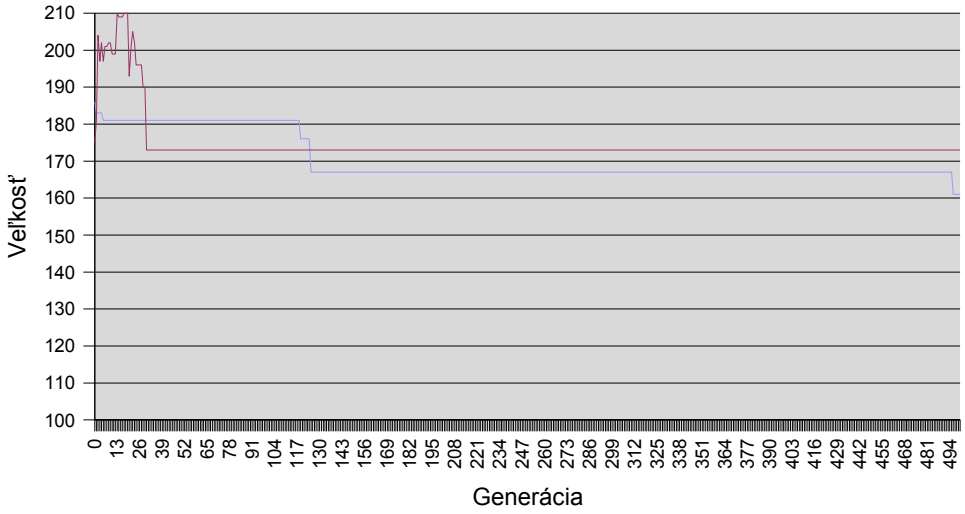


Obrázok 1: *Priebeh kompresie CSS*

Z obrázka 1, ktorý vyobrazuje vývoj veľkosti štýlového predpisu CSS je zrejmé, že dochádza k postupnej optimalizácii. Táto závislosť zodpovedá priebehom z iných problémových oblastí a je teda možné konštatovať, že genetický algoritmus sa správa aj v tejto doméne podľa predpokladov.

Vplyv elitárstva

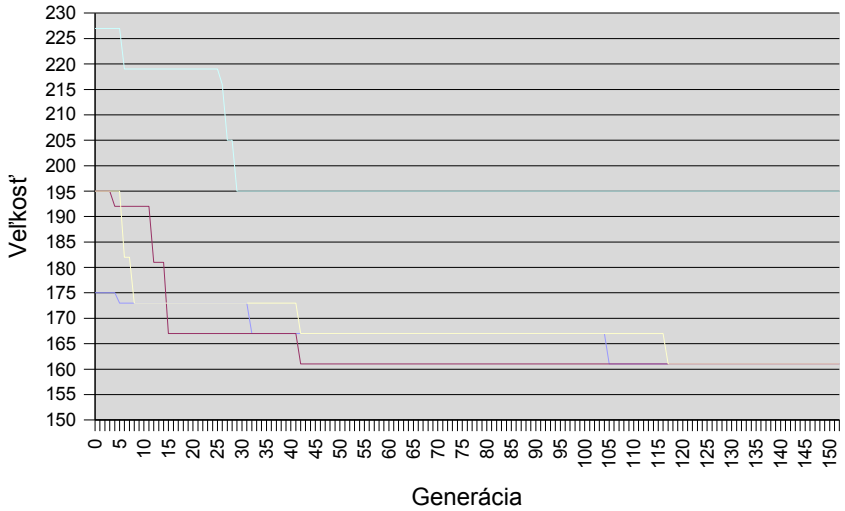
Elitárstvo je spôsobom ako do istej miery urýchliť konvergenciu populácie k lepším riešeniam tým, že isté percento najlepších jedincov z jednotlivých generácií automaticky kopírujeme do ďalšej generácie. Na obrázku 2 sú znázornené priebehy optimalizácie štýlového predpisu pre pokus s elitárstvom (modrá čiara) a bez neho (červená čiara). Je názorne vidieť, že pri elitárstve nemožno stratiť najlepšieho jedinca a bez elitárstva je začiatok vývoja populácie veľmi dramatický, čo má za následok aj prvotné zhoršovanie najlepších jedincov generácií. Na druhej strane príliš veľká miera elitárstva môže spôsobiť, že zdanlivo kvalitní jedinci sa stanú slepou uličkou, ktorá sa už nemá šancu zlepšovať avšak v rámci elitárstva nie je šanca, že zaniknú. Toto sa do istej miery potvrdilo, kde v niektorých pokusoch s vysokou mierou elitárstva dochádzalo k veľmi rýchlemu uviaznutiu v lokálnom extréme.



Obrázok 2: *Priebeh kompresie CSS bez elitátrstva (červená) a s elitárstvom (modrá)*

Vplyv veľkosti populácie

Experimenty boli vykonané na rôznych veľkostiach populácii jedincov. Ukázalo sa, že zatiaľ čo veľkosti populácii 100, 500 a 1000 dávali približne rovnaké výsledky. Pri menších počtoch, napríklad 20 sa však konvergencia buď značne spomalila alebo populácia uviazla v lokálnom extréme veľmi vzdialenom tomu, ktoré dosiahli väčšie populácie. Na obrázku 3 je možné pozorovať tento efekt pre dané veĺkosti populácie 1000 – tmavo modrá, 500 – červená, 100 – žltá a 20 – svetlo modrá.



Obrázok 3: *Priebeh kompresie CSS pre rôzne veľkosti populácie.*

Zhodnotenie

Táto práca ukázala, že genetický algoritmus je možné použiť na kompresiu CSS využitím presúvania vlastností a zoskupovaním selektorov. Z praktického hľadiska je však takáto optimalizácia časovo náročná a preto je jej nasadenie v širšom meradle skeptického charakteru.

V experimentoch sa ukázalo, že elitárstvo má zásadný vplyv na rýchlosť optimalizácie avšak môže spôsobovať rýchlejšie uviaznutie v lokálnom extréme.

Experimentálne sa tiež ukázalo, že malá veľkosť populácie znižuje rýchlosť konvergencie a zvyšuje rýchlosť uviaznutia v lokálnom extréme. Naopak príliš veľké populácie nemajú takmer žiadny vplyv na rýchlosť konvergencie alebo rýchlosť uviaznutia v lokálnom extréme. Veľké populácie sa tak z výpočtového hľadiska zdajú byť neefektívne, pretože stoja viac výpočtového času a porovnateľné výsledky je možné dosiahnuť aj s menšou populáciou.

V tejto práci sa taktiež nachádza návrh a implementácia nového a z algoritmického hľadiska rýchleho výberu ruletou. Tento prístup k výberu môže byť výhodný pri problémoch vyžadujúcich veľké populácie a vysoké počty generácii v experimentoch.

Použitá literatúra

[1] Špecifikácia jazyka CSS (CSS2 Specification) - http://www.w3.org/TR/REC-CSS2/

[2] Java™ 2 Platform Standard Edition 5.0 API Specification - http://java.sun.com/j2se/1.5.0/docs/api/

[3] Kvasnička, Pospíchal, Tiňo: Evolučné algoritmy, STU Bratislava, 2000

Príloha A – Výber z rulety zložitosťou O(1)

Klasický prístup k výberu jedincov ruletou spočíva v tom, že sa najprv vygeneruje číslo $x \in (0, \Omega)$, kde Ω je súčet fitnes všetkých jedincov a následne sa poľom jedincov lineárne prechádza postupne cez jedincov a akumuluje fitnes až kým nepresiahne hodnotu x. Takto sa zachová distribúcia náhodného rozdelenia výberu jedincov, avšak takéto riešenie má zložitosť $O(n)$, kde n je počet jedincov.
Pri použití metódy „ranking“ sú jedinci najskôr usporiadaní zostupne podľa fitnes a očíslovaný od n po 1. Takto vzniká aritmetická postupnosť $n, n-1, \ldots, 2, 1$, ktorú je možné zapísať ako

$$a_i = n - i + 1 \tag{1}$$

Takáto aritmetická postupnosť má prvý člen $a_1 = n$ a diferenciu $d = -1$ a teda pre súčet prvých i členov platí

$$s_i = \frac{i}{2}(a_1 + a_i) = \frac{i}{2}(n + n + (i-1)(-1)) = \frac{i}{2}(2n + 1 - i) \tag{2}$$

z čoho po úprave pre daný súčet $s_i = x$

$$2in + i - i^2 - x = 0 \tag{3}$$

$$i^2 - i(2n + 1) + x = 0 \tag{4}$$

Potom pre hľadané i dostávame

$$i = \frac{(1 + 2n) - \sqrt{(1 + 2n)^2 - 8x}}{2} \tag{5}$$

Index hľadaného jedinca je možné nájsť so zložitosťou $O(1)$ tak, že sa najprv vygeneruje číslo x z intervalu $x \in (0, \frac{(n-1)n}{2})$ a zo vzťahu 5 sa určí index do poľa jedincov. Pravá strana intervalu je súčet všetkých čisle od 1 po *n*.

Je potrebné si uvedomiť fakt, že pole jedincov treba na rozdiel od klasického riešenia lineárnym hľadaním najprv utriediť, čo taktiež stojí výpočtový čas. Pre porovnanie zoberme v úvahu, že je potrebné vygenerovať celú populáciu n jedincov za pomoci rulety. Pri klasickom výbere to znamená n-krát lineárne vyhľadať jedinca čiže v konečnom dôsledku $O(n^2)$.

Pri využití vlastností aritmetického radu je potrebné pole najprv usporiadať, čo je pri štandardnom quicksorte možné so zložitosťou $O(n \log(n))$. Čiže v konečnom dôsledku je výsledná zložitosť taktiež $O(n \log(n))$. Pri použití distribučného (radixového) usporiadania by bolo možné túto celkovú zložitosť znížiť až na $O(n)$.