Implementacija Regex Golf igre pomoću Genetskog programiranja

Projekat u okviru kursa Računarska inteligencija
Matematički fakultet
Univerzitet u Beogradu

Anđela Ilić
mi17105@alas.matf.bg.ac.rs
Mina Milošević
mi17081@alas.matf.bg.ac.rs

Februar 2021

Sadržaj

1 Opis problema

Data su dva skupa reči - M i U, ne nužno istih dimenzija. Cilj *Regex Golf* igre je pronaći najkraći regularni izraz kojim se mogu zapisati sve reči iz skupa M, ali kojim se ne može zapisati nijedna reč skupa U.
Za date skupove M i U ne možemo sa sigurnošću da tvrdimo da postoji rešenje koje zadovoljava prethodne uslove. Takođe, ako dobijemo regularni izraz koji zadovoljava navedene uslove, ne možemo za svaki primer znati da li postoji i bolje rešenje tj. kraći regularni izraz.
U nastavku je dat opis naše implementacije rešenja pomoću Genetskog programiranja [1].

2 Implementacija

Svaka jedinka u Genetskom programiranju će biti predstavljena kao drvo. U listovima nalaze elementi koje ćemo jednim imenom zvati *Terminali* (terminal set), a u unutrašnjim čvorovima su elementi koje nazivamo *Funkcije* (function set).
Skup funkcija sadrži operatore koji se mogu javiti u regularnim izrazima. Primeri takvih operatora su: .∗, .+, .?, .{.,.}+, (.), [.], [^.], .., .|.. Tačka . je mesto na kome se nalaze deca u drvetu.
Skup terminala čine elementi koji zavise i koji ne zavise od ulaznih skupova M i U. Elementi koji su nezavisni - opsezi malih i velikih slova (a-z, A-Z), opsezi brojeva u regularnim izrazima (0-9), karakteri ^ i $, wildcard karakter '%' (kasnije se transformiše u .). Elementi skupa terminala koji su zavisni - skup svih karaktera iz M, parcijalni opsezi karaktera iz M i n-grami iz skupova U i M.

2.1 Obrada ulaznih podataka

Bez obzira na veličinu ulaznih skupova M i U i njihov sadržaj, implementacija prethodno navedenih zavisnih terminala je ista.
Skup karaktera iz M sadrži sve karaktere koji se mogu naći u rečima iz M, bez ponavljanja i sortirani po engleskom alfabetu.
Parcijalni opsezi se prave na osnovu skupa karaktera iz M. Potrebno je naći maksimalne podskupove tog skupa karaktera tako da se svi karakteri iz intervala $[c_f, c_l]$ nalaze u skupu karaktera iz M. c_f je prvi karakter, a c_l je poslednji karakter iz podskupa. Kao rezultat se vraćaju parcijalni opsezi u formatu $c_f - c_l$.
n-grami. Pravimo skup svih n-grama dužine $2 \leq n \leq 4$ koji se mogu naći u M ili u U (ili oba). Svakom dobijenom n-gramu se dodelju vrednost koja predstavlja njegov *score*. Za svaku reč iz M koja sadrži dati n-gram, njegov *score* se uvećava za 1, a za svaku reč iz U koja ga sadrži, *score* se umanjuje za 1. Nakon formiranja svih n-grama i određivanja njihovih *score*-ova, sortiraju se opadajuće po *score*-u. Potrebno je uzeti najmanji podskup n-grama tako da njihova ukupna vrednost (*score*) bude jednaka bar dužini skupa M ($|M|$).

2.2 Genetsko programiranje

2.2.1 Implementacija jedinke

Svaka jedinka se predstavlja preko *stabla*. U korenu stabla se nalazi karakter '.' i koren uvek ima dva deteta. Elementi u unutrašnjim čvorovima se biraju nasumično iz skupa *Function*, a elementi u listovima su nasumično izabrani iz skupa *Terminal*.

Na osnovu izabranog elementa za unutrašnji čvor dobijamo informaciju koliko dece će taj čvor imati - .∗, .+, .?, (.), [.], [ˆ.] će imati jedno dete; .., .|. dva deteta; .{.,.}+ ima tri deteta.
Od ovako kreiranog drveta se dobija niska koja predstavlja validan regularni izraz. Za svaku jedinku računamo *fitnes* funkciju po formuli:

$$f(x) = w_i * (n_m - n_u) - length(r)$$

gde je w_i statistički određena konstanta (kod nas ima vrednost 10), n_m i n_u brojevi reči iz skupova M i U, redom, koje su opisane dobijenim regularnim izrazom, a $length(r)$ je dužina regularnog izraza. Pošto želimo što kraći regularni izraz koji obuhvata sve reči iz M i nijednu reč iz U, ovako definisanu funkciju *fitnes* maksimizujemo.

2.2.2 Selekcija, ukrštanje, mutacija, elitizam

Za *selekciju* koristimo turnirsku selekciju veličine 7. Jedinke za selekciju biramo nasumično i uzimamo najbolju jedinku tj. onu koja ima najveći fitnes među odabranim.
Koristimo *jednopoziciono ukrštanje*. Najpre obilazimo stabla roditelja BFS-om i numerišemo čvorove. Uzimamo broj iz intervala $[0, min(len(parent1), len(parent2))]$. Broj 0 odgovara korenu stabla, pa postavljamo *child1* na *parent2*, a *child2* na *parent1*. Inače, ako smo dobili broj veći od 0, tražimo podstabla u roditeljima koja treba da razmene mesta. Na kraju treba proveriti da li su regularni izrazi ovako dobijene dece validni. U slučaju da neko dete nije validno, vraćamo odgovarajućeg roditelja umesto njega.
Mutaciju radimo sa verovatnoćom 0.1. Biramo indeks čvora koji želimo da promenimo. Razlikujemo dva slučaja - kada je čvor iz skupa terminala i kada je iz skupa funkcija. U prvom slučaju, samo izaberemo nasumično novi terminal i ažuriramo vrednost čvora. Ukoliko je čvor bio iz skupa funkcija, biramo novu vrednost iz istog skupa ali sada treba obratiti pažnju da li treba menjati i njegovu decu. Ako smo izabrali element iste kardinalnosti (isti broj dece), onda samo promenimo vrednost u čvoru. Ako se kardinalnost smanjila, odgovarajuću decu brišemo, a ako se kardinalnost povećala pravimo novo poddrvo. Na kraju opet treba proveriti da li je dobijena jedinka validna. Ako nije, vraćamo jedinku za koju smo i pokrenuli mutaciju.
U našoj implementaciji koristimo i *elitizam* kako bismo sačuvali najbolje jedinke u populaciji. Elitizmom čuvamo 20% najboljih jedinki.

2.2.3 Parametri genetskog programiranja

Parametri su postavljeni na osnovu podataka iz [1]. Za većinu primera sa ovim parametrima se dobijao veoma dobar rezultat, dok je za neke bilo potrebno povećati broj generacija ili populacija.
U algoritmu imamo populaciju od 500 jedinki i pravimo 1000 generacija. Kao što je ranije rečeno, turnirska selekcija je dimenzije 7, a verovatnoća mutacija je 0.1. Za elitizam uzimamo 20% jedinki. Kako bismo dobili što bolje rešenje algoritam pozivamo za 32 populacije.

2.3 Izlaz programa

Iz svake populacije čuvamo sve jedinke kod kojih važi:

$$n_m - n_u = |M|$$

n_m i n_u su brojevi reči iz skupova M i U, redom, koje su opisane regularnim izrazom, a $|M|$ je veličina skupa M.

Dakle, želimo da sačuvamo sve jedinke kod kojih su ispunjena prva 2 uslova ovog problema. Ako u trenutnoj populaciji ne postoji takva jedinka, čuvamo samo najbolju jedinku na osnovu celokupnog fitnesa.
Na kraju rada algoritma biramo najbolju jedinku, od svih sačuvanih, po njenom ukupnom fitnesu i nju proglasavamo za rešenjem datog problema.

3 Rezultati

U sledećoj tabeli su zabeleženi naši rezultati za odabrane primere. Primeri su uglavnom preuzeti sa [2].

PRIMER	NAŠE REŠENJE	SCORE	NAJBOLJE REŠENJE	SCORE
Plain strings	foo	207	foo	207
Anchors	k$	208	k$	208
It never ends*	u$	28	u\b	27
Ranges	ff|.b|d[a-f]	178	^[a-f]*$	202
Abba	st|.+u|z	142	^(?!(.)+ \1)|ef	196
A man, a plan	^r|x|gg|oo	90	^(.)[^p].*$	177
Presidents**	Bu|am|Ta|Har|N+i|vel	120		390
Movies***	m |B?R|[AB?]	48	m | [tn]|b	40
Regions****	br|-|os|L|P|t?il	104		200

Vreme izvršavanja programa je nekoliko desetina minuta (uglavnom do sat vremena).

∗ U primeru na sajtu [2] nije dozvoljeno korišćenje karaktera $, pa su rešenja drugačija.
∗∗ Primer je preuzet sa [4]. Nije pronađen podatak o najboljem rešenju pa je zapisan idealan *score*.
∗ ∗ ∗ Primer i najbolje rešenje su preuzeti sa [5]. Njihovo rešenje je *case insensitive*, dok je naše *case sensitive*.
∗ ∗ ∗∗ Primer je preuzet sa [6]. Nije pronađen podatak o najboljem rešenju pa je zapisan idealan *score*.

4 Zaključak

Na osnovu rezultata iz prethodne tabele možemo da zaključimo da naša implementacija ovog problema, zasnovana na [1], daje zadovoljavajuće rezultate. Za većinu problema je moguće dobiti bolje rezultate povećavanjem broja populacija ili jedinki u populaciji, ali bi tada vreme izvršavanja bilo dosta veliko.
Takođe, u primeru Movies [5] možemo da primetimo da je naš program našao bolje rešenje u odnosu na rešenje koje su ljudi našli (bolje u smislu naše funkcije fitnes, oni su je možda definisali na drugi način), što je veoma ohrabrujuće.

Literatura

[1] Alberto Bartoli, Eric Medvet, Andrea De Lorenzo, and Fabiano Tarlao - *Playing Regex Golf with Genetic Programming* (2014)

[2] Regex Golf Examples

[3] Regex Golf Solutions

[4] USA Election winners and losers

[5] Star Wars and Star Trek titles

[6] Regions of Italy vs. States of USA