Minimal Sorting by Reversals

Vuk Stefanović 66/2019, Anja Čolić 231/2019

Septembar 2023

Seminarski rad u okviru kursa Računarska inteligencija

Matematički fakultet

Sadržaj

1 Uvod

Minimum Sorting by Reversals je pojednostavljena verzija problema preuredjivanja genoma koja pokušava otkriti evolutivni odnos izmedju različitih genoma i jedan je od mnogih izazovnih problema u bioinformatici. Rešavanje ovog problema optimalno je dokazano da je NP-teško, i stoga su razvijeni različiti aproksimativni algoritmi. Mi ćemo u radu za rešavanje koristiti genetski algoritam i uporediti rezultate dobijene i grubom silom.

2 Formulacija problema

Na ulazu imamo permutaciju brojeva od 1 do n. Potrebno je permutaciju dovesti do permutacije identiteta (permutacija sortirana rastuće), tako da broj operacija bude minimalan.

Dozvoljene operacije su operacije invertovanja intervala [i,j], 1<=i,j<=n.

Primer:
Ulazni niz: [2,1,4,3,5]
Koraci:
[2,1,4,3,5] → invertujemo [0,1]
[1,2,4,3,5] → invertujemo [2,3]
[1,2,3,4,5]

Na izlazu dobijamo sortiran niz, koristeći dve inverzije.

3 Ideja rešenja

Svaka jedinka u populaciji prestavlja niz inverzija koje primenjujemo na početnu permutaciju. Pošto broj inverzija nije unapred poznat svaka jedinka može biti različite dužine (uvek parne dužine).

3.1 Fitnes funkcija

Za izračunavanje fitnes funkcije, pored broja inverzija, bitan nam je i broj prelomnih tačaka niza (mesta na kojima niz prestaje da bude rastući).

Kažemo da između dva susedna elementa niza postoji prelomna tačka ako je apsolutna razlika tih elemenata različita od 1.

Primer:
[1,2,4,5,3] → prelomne tacke su između 2 i 4, kao i 5 i 3

Fitnes funkciju definišemo kao:

A*dužina(hromozoma)+B*brojPrelomnihTacaka

A i B su proizvoljni parametri koji određuju uticaj broja prelomnih tačaka i dužine hromozoma.

3.2 Populacija

Veličina populacije je n*log(n), gde je n dužina niza koji se sortira. Dužina hromozoma se kreće od broja tačaka preloma do 3n. Delimo populaciju na dva dela, na oba dela primenjujemo selekciju, mutaciju i ukrstanje, vodeci računa o tome da ne izgubimo najbolje jedinke u narednoj generaciji (Elitizam).

Komentar : Velicina populacije preuzeta iz rada [1]

3.3 Selekcija

Koristićemo turnirsku selekciju.

3.4 Ukrštanje

U zavisnosti da li su oba roditelja iste dužine, razlikovaćemo 2 vrste ukrštanja:

- Ako su iste dužine - Koristimo dvopoziciono ukrštanje.
- Ako su različite dužine - Koristimo takodje dvopoziciono ukrštanje ali gde se početna pozicija većeg hromozoma odabere nasumično i onda se ceo manji hromozom stavi na poziciju nakon nje (apsorpcija).

Apsorpcija primer:

- Roditelj 1 : [0, 1, 2, 4 , 0, 3, 1, 2]
- Roditelj 2 : [0, 3, 0, 2]
- Prva izabrana pozicija = 2
- Dete 1 : [0, 1, 0, 3, 0, 2, 1, 2]
- Dete 2 : [2, 4 , 0, 3]

3.5 Mutacija

Na svaku od jedinki primenjujemo jedan od 4 operatora mutacije koji random određujemo.

Operatori mutacije:

- Menjamo nasumično izabran gen
- Izaberemo 2 pozicije i zamenimo vrednosti na tim pozicijama novim random vrednostima u opsegu od 0 do veličina niza.
- Odaberemo 2 inverzije [x1, x2] i [y1, y2] i na pozicije nakon x2 i pre y1 umetnemo 2 random broja čime dobijamo dužu jedinku.
- Odaberemo 2 inverzije [x1, x2] i [y1, y2], obrisemo x2 i y1 i time dobijemo novu jedinku

Komentar: Ideja za poslednje dve mutacija uzeta iz rada [1]

4 Modifikacije genetskog algoritma

Uočili smo problem koji bi mutacije koje smo preuzeli iz rada teško rešile:

- U situaciji kad je niz skoro sortiran (kad je potrebno uraditi samo jednu ili dve inverzije) poslednje dve mutacije ne mogu da reše ovu situaciju, zato uvodimo mutaciju 1.

Zbog situacije da niz na kraju često nije bio sortiran, izvršavali smo ponovno pokretanje genetskog algoritma nad datim nizom. Nakon svakog pokretanja novi kod najbolje jedinke dodavali smo na stari.

Na primer: Početni niz: [3, 2, 1, 5, 4]
Algoritam vrati niz :[1, 2, 3, 5, 4]
Kod najbolje jedinke : [0, 2]
Ponovo pozovemo algoritam za niz [1 ,2 ,3, 5, 4], algoritam vrati kod najbolje jedinke [3, 4], i to nadovežemo na [0, 2]

5 Rezultati

Upoređujemo algoritam grube sile i genetski algoritam samo za nizove najviše dužine pet, zbog velike složenosti grube sile i činjenice da se genetski algoritam mnogo brže izvršava za duže nizove.

Nastavljajući, upotrebljavali smo tri različita genetska algoritma za dužine nizova najviše 15, jer smo primetili da je za duže nizove bilo potrebno znatno više vremena za izvršavanje.

Modifikacije genetskih algoritama koje smo upotrebljavali su:

- "MSR1234- svaka jedinka izvršava jednu random mutaciju od prethodno opisane četiri.
- "MSR234- svaka jedinka izvršava jednu random mutaciju od tri koje predstavljaju unapred izabran podskup prethodno četiri.
- "MSR2- svaka jedinka izvršava treću po redu opisanu mutaciju sa odredjenom verovatnoćom.

Svi rezultati su prikazani u narednoj tabeli.

N	Array	Brute Force	"MSR1234"	"MSR234"	"MSR2"	Time
2	[1, 2]	0	0	0	0	0.05 min
3	[3, 1, 2]	2	2	2	2	0.05 min
4	[4, 3, 2, 1]	1	1	1	1	0.1 min
5	[1, 2, 3, 5, 4]	1	1	1	1	0.1 min
6	[5, 4, 1, 2, 3, 6]	-	2	2	2	0.2 min
7	[2, 6, 1, 4, 7, 5, 3]	-	6	6	6	0.2 min
8	[6, 2, 1, 5, 3, 8, 4, 7]	-	5	6	-	0.2 min
9	[9, 5, 4, 2, 1, 3, 6, 7, 8]	-	5	6	5	0.5 min
10	[8, 3, 6, 1, 9, 10, 4, 2, 7, 5]	-	7	8	7	0.5 min
11	[3, 10, 5, 4, 11, 7, 2, 6, 9, 1, 8]	-	9	9	11	1 min
12	[5, 7, 11, 3, 9, 2, 12, 4, 10, 1, 6, 8]	-	10	9	11	1 min
13	[4, 1, 2, 5, 3, 6, 7, 10, 8, 9, 11, 12, 13]	-	7	5	5	1 min
14	[4, 1, 2, 5, 3, 6, 7, 9, 8, 10, 11, 14, 12, 13]	-	10	6	14	1 min
15	[4, 1, 2, 5, 3, 6, 7, 9, 8, 10, 11, 12 , 15, 14, 13]	-	5	5	7	1 min

Tabela 1: Rezultati koji prikazuju broj inverzija dobijenih primenom svakog algoritma na random generesine permutacije dužine od 2-15

6 Zaključak

U navedenom radu smo predstavili pristup za rešavanje problema minimalnog sortiranja inverzijom intervala. Implementirali smo ovaj pristup pomoću genetskog algoritma sa promenljivom dužinom hromozoma.

Važno je napomenuti da naše rešenje ne može garantovati optimalno rešenje problema. Na kraju izvođenja algoritma, dobijamo broj inverzija potrebnih za sortiranje niza, kao i listu tih inverzija koje treba primeniti da bi se postigao sortirani niz.

Moguće je pokrenuti program više puta za istu instancu problema (isti početni niz), i ovo može rezultirati različitim rešenjima. Međutim, najmanji broj inverzija među svim dobijenim rezultatima smatramo najboljim rešenjem koje smo do sada pronašli.
Važno je naglasiti da nije moguće precizno odrediti koliko se naše rešenje razlikuje od optimalnog. Praktično, da bismo saznali koliko smo daleko od optimalnog rešenja, morali bismo da pronađemo baš to optimalno rešenje, što može biti izazovno ili nemoguće.

Objasnjenje za prethodno Ako program vrati broj inverzija n i istovremeno imamo informaciju da je naše rešenje udaljeno za k od optimalnog rešenja, tada možemo zaključiti da je optimalno rešenje n - k inverzija. Ovo je zato što znamo da je naše rešenje od n inverzija bolje od rešenja od n + k inverzija (jer je bliže optimalnom rešenju za k).

Za rešenje od n + k inverzija, možemo pretpostaviti da ono nije optimalno, jer već imamo rešenje od n inverzija koje je "optimalnije"(bliže optimalnom rešenju). Dakle, možemo zaključiti da rešenje od n + k inverzija ne može biti optimalno, jer imamo bolje rešenje sa samo n inverzija.

Ova analiza nam pomaže da relativno procenimo kvalitet našeg rešenja u odnosu na optimalno rešenje, bez potrebe da stvarno pronađemo optimalno rešenje.

Dalji tok unapredjivanja U daljem razvoju algoritma, razmotrenje različitih operacija mutacije i ukrštanja svakako može doprineti poboljšanju performansi algoritma. Promenom ovih operatora možemo eksperimentisati sa različitim strategijama i prilagoditi ih specificiranom problemu kako bismo postigli bolje rezultate.
Takođe, razmatranje različitih strategija selekcije, kao što je selekcija rangiranjem, takođe može biti korisno. Različite strategije selekcije mogu imati različite efekte na diverzitet populacije i konvergenciju algoritma.

Ako je vreme izvršavanja ključni faktor, prelazak na programski jezik poput C ili C++ može ubrzati izvođenje algoritma. Međutim, važno je napomenuti da ovo neće promeniti asimptotsku složenost algoritma. Takođe, optimizacija algoritma i struktura podataka može takođe doprineti poboljšanju brzine izvršavanja.

U svakom slučaju, iterativno eksperimentisanje sa različitim komponentama algoritma i evaluacija njihovog uticaja na performanse može dovesti do boljeg rešenja za dati problem.

Literatura

[1] Sorting Unsigned Permutations by Reversals using Multi-Objective EvolutionaryAlgorithms with Variable Size Individuals, Ahmadreza Ghaffarizadeh, Kamilia Ahmadi and Nicholas S. Flann