Univerzitet u Novom Sadu

Fakultet tehničkih nauka

Dokumentacija za projektni zadatak

Studenti: Popov Vladimir, SV29/2021

Bulatović Balša, SV37/2021

Predmet: Paralelno programiranje

Tema projektnog zadatka: Genetski algoritam, problem N – kraljica

# Opis problema

Problem N kraljica predstavlja klasičan kombinatorni problem u oblasti veštačke inteligencije. Problem ima jednostavnu strukturu i definisan je na sledeći način. Na šahovsku tablu, veličine NxN, treba postaviti N kraljica, tako da se nijedan par kraljica medjusobno ne napada. Dve kraljice se napadaju ukoliko se nalaze u istom redu, koloni ili na dijagonali.

# Uvod

Najjednostaviji primer ove vrste problema, za koji postoji rešenje, predstavlja problem 4 kraljice. U opštem slučaju, postoji različitih načina da se postavi N kraljica na tablu veličine NxN, tako da za relativno mali problem od 10 kraljica postoji više od 1.73 \* mogućnosti, što predstavlja velik prostor rešenja za pretraživanje, kod tako malog problema.

# Prikaz jedinke

Svaka jedinka predstavlja niz od N elemenata, u kojem je vrednost svakog elementa tačno jedan broj od 0 do N-1. Na ovaj način rešen je problem konflikta kraljica u istom redu. Element jedinke predstavlja broj kolone gde se nalazi kraljica koja se nalazi u *i*-tom redu, gde je *i* broj člana jedinke.



Slika 1Prikaz jedinke

# Ocena kvaliteta jedinke

Kvalitet jedinke (*fittness)* predstavlja ukupan broj kraljica koje se medjusobno napadaju. Zbog samog prikaza jedinke, jedini sporan konflikt je na dijagonalama i u kolonama, pa kvalitet jedinke neposredno predstavlja broj kraljica koje se nalaze na istim dijagonalama i u istim kolonama. Za svaku kraljicu koja je u konfliktu sa nekom drugom, kvalitet opada za 1. Kako je u implementaciji programa cilj dostići nula konflikata, najkvalitetnija jedinka ce imati ocenu nula.

# Ukrštanje i mutacija

Odabir jedinki za ukrštanje vrši se ruletskom selekcijom. Samo ukrštanje dve jedinke počinje nasumičnim biranjem broja *x* [0, N-1]. Nakon toga, roditelji se polove na *x* – toj poziciji, i nastaju deca spajanjem prvog dela prvog roditelja sa drugim delom drugog roditelja i obrnuto. Nakon toga postoji 70% šanse da će doći do mutacije kod dece, koja započinje ponovnim nasumičnim biranjem broja *x* u istom domenu. Mutacija se vrši postavljanjem nasumične vrednosti od 0 do N-1 na poziciji *x.*

# Selekcija

Nakon N ukrštanja, populacija se povećala za broj novonastale dece. S obzirom da je neophodno da kardinalnost populacije ostane ista, potrebno je izvršiti selekciju jedinki. Primenjuje se elitizam, gde se dopušta da 5% najboljih iz prethodne populacije predje u sledeću generaciju. Sortiranjem populacije na osnovu ocene kvaliteta svake jedinke, u narednu generaciju prelazi prva polovina trenutne populacije.

# Odabir parametara algoritma

Kako bismo sprečili da algoritam ostane u “lokalnom minimumu” šansa za mutaciju je poprilično velika i iznosi 70%, dok je stopa elitizma 5%. Na ovaj način omogućavamo algoritmu da u svaku generaciju uvodi veći broj noviteta i time brže konvergira. Podrazumevano, svaki od ovih parametara je podložan promenama i ukoliko primetimo da ne dolazi do promena u generacijama duži vremenski period, program se može zaustaviti i neki od parametara se može izmeniti u cilju brže kovergencije za dati slučaj.

# Kriterijum zaustavljanja

Postoje dve mogućnosti kod zaustavljanja programa. Prva je pronalazak rasporeda kraljica kod kojeg ne postoji medjusobno napadanje bilo koje dve kraljice, to jest postojanje jedinke čija je ocena kvaliteta jednaka nuli. Zbog mogućnosti upadanja u lokalni minimum, postoji mogućnost beskonačnog izvršavanja programa. Kako bi se to sprečilo, poželjno je pratiti rezultate dobijene nakon svake nove generacije. Ukoliko ne dolazi do promena duži vremenski period, može se ranije prekinuti traženje rešenja. Alternativno, prekid programa je moguće izvršiti nakon unapred zadatog broja generacija.

# Paralelizacija zadatka

Paralelizacija je primenjena u…

# Rezultati

Nakon testiranja programa uz konfiguraciju veličine šahovske table i populacije,zabeleženi su rezultati prikazani na grafikonima u nastavku. Testovi su izvršeni na AMD Ryzen 5 3400G procesoru, sa 4 jezgra, koji se zasniva na Zen+ mikroarhitekturi.

Naredni grafikon predstavlja vreme serijskog i paralelnog izvršavanja po generacijama. U proseku se paralelni program izvršava 3.85 puta brže, čime se dostiglo linearno ubrzanje.

# Analiza rezultata

Kao što je i poznato, genetski algoritam nema garancije pronalaženja rešenja, a razlog te nepredvidljivosti su u velikoj meri mutacije i ukrštanja. Takodje, empirijskom proverom zaključeno je da program koji radi sa manjom kardinalnosti populacije brže konvergira ka traženom rešenju, prilikom serijskog izvršavanja. Paralelizacijom dobijamo mogućnost da povećamo veličinu populacije, čime trošimo više resursa, ali vreme izvršavanja je kraće. S obzirom na nepredvidivost završetka rada programa, relevantnije je meriti izvršavanje zadatatka na nivou jedne generacije, gde je moguće dostići ubrzanje i do 20 puta.

# Zaključak

Paralelizacija programa pruža mogućnost efikasnijeg korišćenja dostupnih hardverskih resursa, što može rezultovati poboljšanjem performansi i smanjenjem vremena izvršavanja. Upotrebom odgovarajućih tehnika, moguće je rasporediti radne zadatke na više procesora ili jezgara i ostvariti istovremeno izvršavanje, čime se postiže ubrzanje izvršavanja. Ograničenja koja paraleliyacija nosi su trke do podataka i veća kompleksnost koda, kao i teže otkrivanje grešaka. Pravilan odabir algoritama i strategija, kao i pažljivo upravljanje podacima i sinhronizaija istih predstavljaju ključne korake u postizanju optimalnih rezultata.

Uprkos navedenim izazovima, paralelizacija se pokazuje kao snažan alat za poboljšanje performansi programa. Ispravno implementirana paralelizacija može doneti značajno kraće vreme izvršavanja, ali i bolju upotrebu raspoloživih resursa.