Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Optimiranje evolucijskim računanjem

Paralelizacija optimizacijskih algoritama

Marko Čupić

Zagreb, 2023.

Zašto paraleliziramo? Općenito

- Tri su osnovna razloga

1) Paralelizacija zbog velikih količina podataka
   - ne stane sve u memoriju jednog računala
2) Paralelizacija zbog skraćivanja vremena izvođenja
   - fiksni obim posla → kraće vrijeme izvođenje
3) Paralelizacija zbog povećanja obima poslova
   - fiksno vrijeme izvođenja → veći obim posla

Zašto paraleliziramo? Kod EA

- Primjena paralelizacije kod algoritama evolucijskog računanja osigurava sljedeće:

  1) ubrzavanje algoritma kako bi se prije došlo do prihvatljivih rješenja

  2) povećanje kvalitete pronađenih rješenja jer algoritam može istražiti veći prostor rješenja

  3) povećanje robusnosti algoritma jer će se smanjiti vjerojatnost ostanka u lošijim područjima zbog mogućnosti istraživanja većeg prostora rješnja

Zašto paraleliziramo? Kod EA

- Primjena paralelizacije kod algoritama evolucijskog računanja osigurava sljedeće:

  4) Rješavanje velikih (engl. *Large scale*) problema gdje je podataka previše za jedno računalo

     - Idemo prema distribuiranom računanju

Vrste paralelizacije

- Uobičajno je paralelizaciju promatrati na tri razine
  1) paralelizacija na razini algoritama
  2) paralelizacija na razini populacije
  3) paralelizacija na razini jedne iteracije algoritma

Paralelizacija na razini algoritama

- To je svaki oblik paralelizacije kod kojeg se istovremeno izvodi više primjeraka algoritma
- Dijeli se na:
  - Nesuradnu (nema interakcije između primjeraka algoritama)
  - Suradnu (postoji interakcija između primjeraka algoritama)

Paralelizacija na razini algoritama: nesuradna

- Trivijalna paralelizacija
  - Je nesuradna paralelizacija gdje čovjek ručno pokrene više primjeraka algoritma na istom ili različitim računalima, sačeka da završe, pogleda dobivena rješenja svakog od algoritama i odabere najbolje
- Svojstva
  - Povećana robusnost zbog nezavisnih pretraživanja, a vrijeme izvođenja ostaje isto

Paralelizacija na razini algoritama: nesuradna

- Uz uporabu prikladnih biblioteka, ovaj se posao može automatizirati
  - Okruženje može automatski kopirati te pokretati algoritme
  - Ako se definira vremensko ograničenje izvođenja jednog algoritma, okruženje može automatski slijedno pokretati nove algoritme, sve dok ne istekne ukupno definirano vremensko ograničenje
    (npr: 10 minuta po algoritmu, 1 sat ukupno, 10 računala → 600 pokretanja)

Paralelizacija na razini algoritama: suradna

- **Suradna** paralelizacija na razini algoritama
  - podrazumijeva pokretanje više primjeraka algoritama koji međusobno razmjenjuju informacije
- Treba odgovoriti na pitanja
  - Što se razmjenjuje
  - Kada se razmjenjuje
  - Tko s kime razmjenjuje

Paralelizacija na razini algoritama: suradna

- Što se razmjenjuje?
  - Rješenja
    - Koje: najbolje ikada pronađeno, najbolje trenutno u populaciji, …
    - Koliko se rješenja razmjenjuje: fiksno, postotak populacije, neki mehanizam odabira, …

Paralelizacija na razini algoritama: suradna

- Što se razmjenjuje?
  - Dijelovi memorije algoritma
    - Dobri parametri (adaptivni GA: koja se vjerojatnost križanja i mutacije pokazala dobrom)
    - Kod mravljih algoritama: mapa feromonskih tragova

Paralelizacija na razini algoritama: suradna

- Što se razmjenjuje?
  - Ugradnja (što napraviti s primljenim podatcima)
    - GA: dobivena rješenja nekako ugraditi u populaciju, primljene parametre interpolirati sa svojima
    - ACO: spojiti primljene feromonske mape, ako je primljeno rješenje, tim putem deponirati novu količinu feromonskih tragova

Paralelizacija na razini populacije: suradna

- Kada se razmjenjuje? Razmjena informacija može biti:
  - Slijepa: periodički, vjerojatnosno, …
  - Inteligentna: adaptivno, primjerice pronalaskom novog najboljeg rješenja

Paralelizacija na razini populacije: suradna

- Tko s kime razmjenjuje?
- Primjerci algoritma bit će organizirani u neku topologiju koja definira susjedstvo
  - Prsten
  - 2D rešetka
  - 3D kocka
  - Potpuno povezana

Paralelizacija na razini populacije: suradna

- Tko s kime razmjenjuje?
- Za topologiju definiramo:
  - Stupanj svakog čvora: broj susjeda (npr. kod prstena to je 2)
  - Broj veza (npr. kod prstena to je N)
  - Prosječna udaljenost do drugih čvorova (utječe na vrijeme širenja novih rješenja kroz sustav; kod prstena to je N/2)

Paralelizacija: Ubrzanje

- Razmotrimo s teorijskog stajališta izvođenje jednog posla koji se sastoji od dva dijela: S i P
  - S je dio posla koji se ne može paralelizirati
  - P je dio posla koji bi se mogao paralelizirati
- Ako izvođenje ovog posla pokrenemo na jednom (slijednom) računalu, vrijeme izvođenja posla bit će:

$$T = T_S + T_P$$

Paralelizacija: Ubrzanje

- Ako dio posla koji se može paralelizirati podijelimo na optimalan način na *k* radnika, vrijeme izvođenja čitavog posla tada će biti:

$$T' = T_S + T'_P = T_S + \frac{T_P}{k}$$

- Ubrzanje S (engl. *Speed-up*) definirano je kao:

$$S = \frac{T}{T'} = \frac{T_S + T_P}{T_S + \frac{T_P}{k}}$$

Paralelizacija: Ubrzanje

- Uvođenjem:

$$r_S = \frac{T_S}{T} \qquad r_P = \frac{T_P}{T} \qquad r_S + r_P = 1$$

- slijedi:

$$S = \frac{T}{T'} = \frac{T_S + T_P}{T_S + \frac{T_P}{k}} = \frac{T}{r_S \cdot T + \frac{r_P \cdot T}{k}} = \frac{1}{(1 - r_P) + \frac{r_P}{k}}$$

Paralelizacija: Ubrzanje

- Uočimo da je limes ubrzanja kada broj radnika teži u beskonačnost:

$$\lim_{k \to \infty} S = \lim_{k \to \infty} \frac{1}{(1 - r_P) + \frac{r_P}{k}} = \frac{1}{(1 - r_P)}$$

- I to je poznati Amdahlov zakon koji kaže da je postoji ograda na maksimalno ubrzanje koje možemo postići

Paralelizacija: Ubrzanje

- Pogledajmo primjer. Neka je vrijeme provođenja jedne iteracije algoritma zbroj vremena križanja i vremena vrednovanja jednike: Tk=4ms, Te=6ms, T = Tk+Te=10ms

k	S
1	1
2	1.43
3	1.67
4	1.82
inf	2.5

1.43/1 = 1.43; 43% povećanje

1.67/1.43 = 1.17; 17% povećanje

1.82/1.67 = 1.09; 9% povećanje

Uočiti smanjivanje dobiti dodavanjem svakog novog radnika!

$$r_P = \frac{6}{10} = 0.6$$

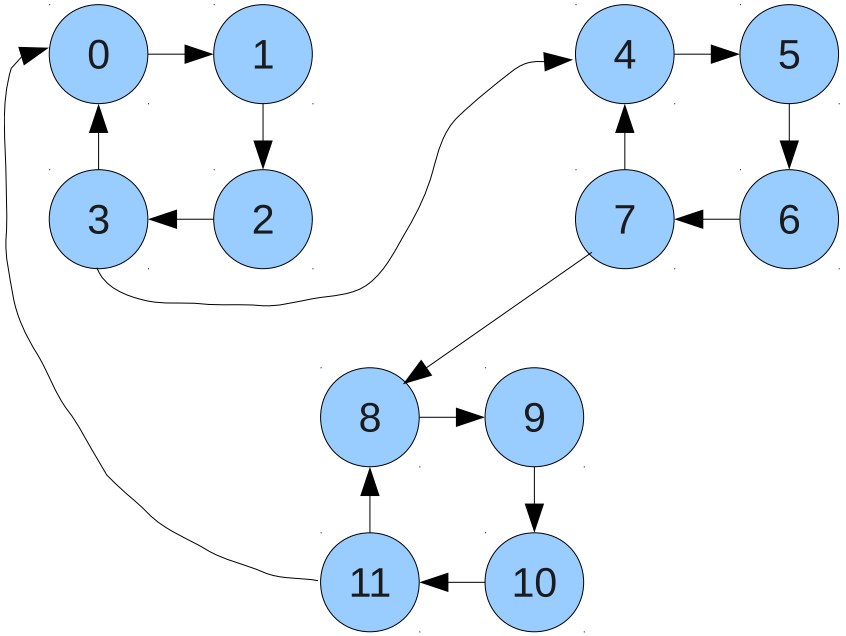
$$\lim_{k \to \infty} \frac{6}{1-0.6} = \frac{1}{0.4} = 2.5$$

Modeli paralelizacije

- Na razini algoritama: Otočki model
  - Primjer suradne paralelizacije
  - Algoritmi mogu biti raspodijeljeni na više računala
  - Sustav može biti:
    - Homogen: svi algoritmi su iste vrste (npr. GA)
    - Heterogen: pojedini primjerci algoritama mogu biti različiti (npr. GA, DE, ACO, ClonAlg, ...)

Modeli paralelizacije

- Na razini algoritama: Otočki model
  - Primjer jedne topologije

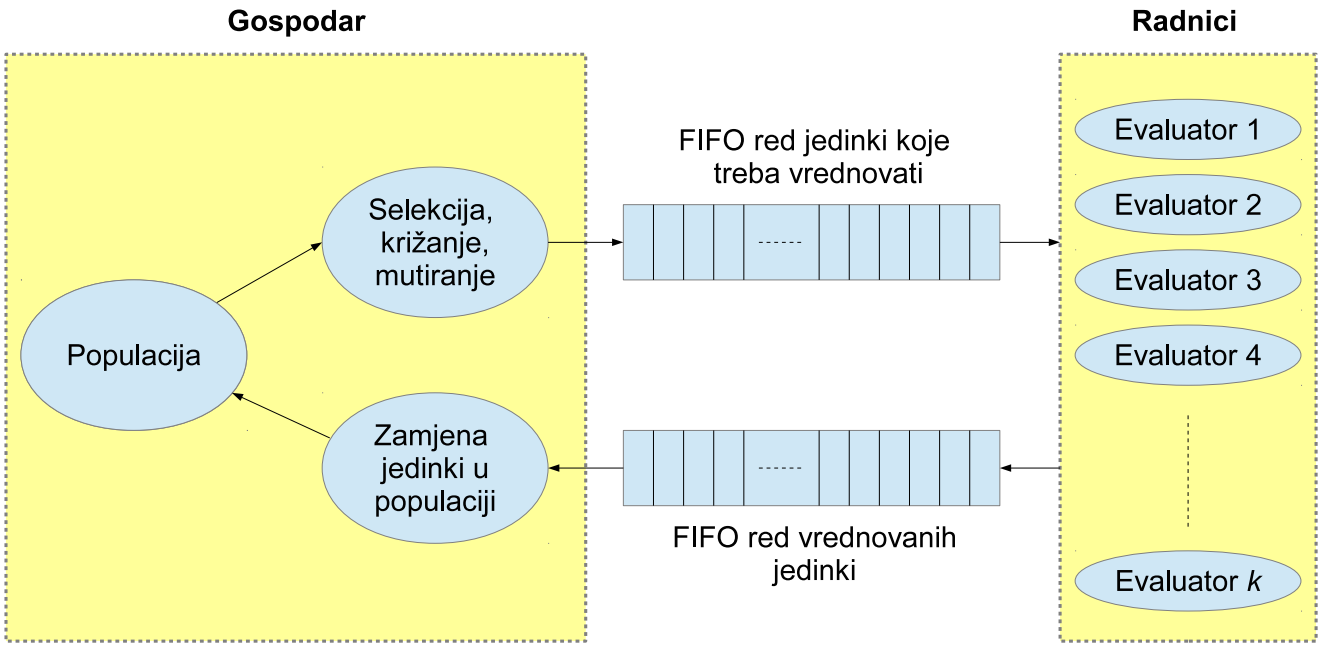


0,4,8: GA
1,5,9: ACO
2,6,10: DE
3,7,11: ClonAlg

Migracijski intenzitet unutar manjih prstena može biti veći, između prstena manji.

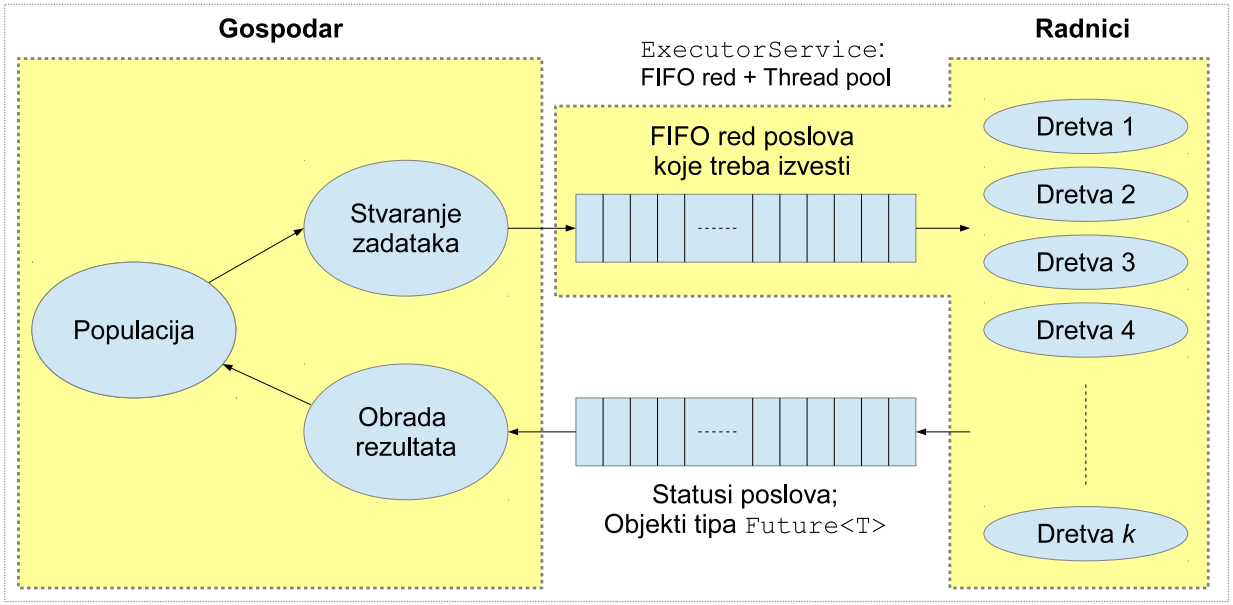
Modeli paralelizacije

- Paralelizacija na razini populacije
  - konceptualno



Modeli paralelizacije

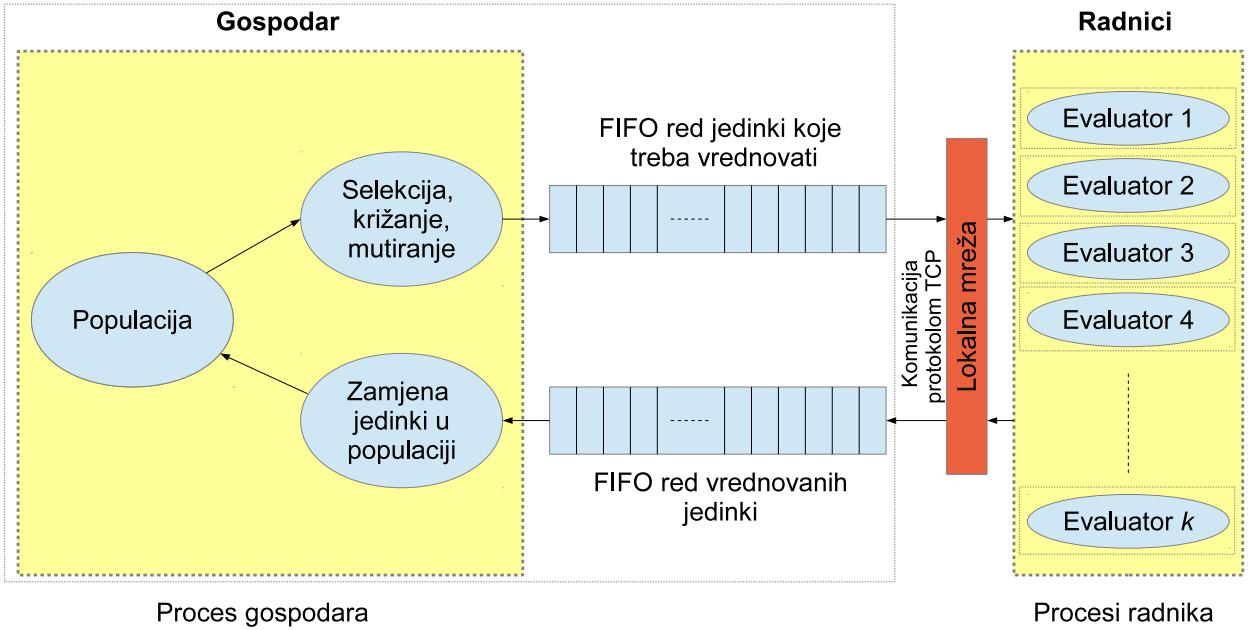
- Paralelizacija na razini populacije
  - Višedretvena implementacija u Javi



Jedan adresni prostor (dretve mogu pristupiti jedinkama iz populacije).

Modeli paralelizacije

- Paralelizacija na razini populacije
  - Raspodijeljena implementacija na više računala



Modeli paralelizacije

- Na razini jedne iteracije
  - Primjerice, klasičan genetski algoritam, kod kojeg se metoda vrednovanja jednog rješenje interno paralelizira
  - Ovaj model je skup u smislu da ga ručno treba raditi za svaki problem koji se rješava, pa se u praksi rijetko koristi

Zaključak

- Paralelizacija na razini algoritama odnosno na razini populacije ne ovise o problemu koji se rješava
  - Moguće pripremiti i iskoristiti gotova radna okruženja koji nude ove modele paralelizacije
  - Programer često samo mora definirati kako se vrednuje rješenje

Zaključak

- Dosta toga nismo spominjali
- Neke specifične modele, poput staničnog genetskog algoritma (engl. *Cellular GA*)
- Uporabu specijaliziranog sklopovlja (tipa grafičke kartice) za paralelizaciju