Motivacija → No Free Lunch teorem → zato različiti algoritmi

NFL (Wolpert, Macready): na skupu svih mogućih problema svi su algoritmi pretrage jednaki (neprepoznatljivi); zaključak: ako algoritam daje bolje rezultate na nekim problemima, morat će davati gore rezultate na drugim problemima

Kodiranje stanja; genotip; fenotip; dekodiranje

$$x_{min} = \frac{n}{2^l - 1} \cdot (x_{max} - x_{min}) + x_{min}$$

Problem interpretacije → slučajna izmjena → nevažeća rješenja (npr. TSP, rješenja su permutacije)

Mala promjena genotipa → moguća velika promjena fenotipa → binarno vs. gray

Problem simetričnosti → puno genotipski različitih rješenja koje predstavljaju identično rješenje; npr. TSP → rotacije permutacija; bitno povećava prostor pretraživanja i negativno utječe na operatore algoritama; kanonski prikaz?

Lokalne pretrage → svrha, kada ih primjeniti? Heuristike – problemski specifične:

- konstrukcijski algoritmi
- algoritmi lokalne pretrage

Metaheuristike – skup algoritamskih koncepata koji koristimo za definiranje heurističkih metoda primjenjivih na širok skup problema; možemo reći da je metaheuristika heuristika opće namjene čiji je zadatak usmjeravanje problemski specifičnih heuristika prema području u prostoru rješenja u kojem se nalaze dobra rješenja.

Memetički algoritmi: evolucijski algoritmi + lokalna pretraga (iako kažu da nije :-))

Podjela problema:

1. kontinuirani prostor
2. diskretni prostor
    - sa semantikom broja (postoji poredak; prethodnih, sljedebenik); redne (engl. ordinal)
    - bez semantike broja (jagode, kruške, …); nominalne, kategoričke

Funkcija cilja, evaluacijska funkcija (objective function)

- skalar (single-objective optimization)
- vektor (multi-objective optimization)

Vrednovanje rješenja

- jednokriterijsko (single-objective)
    - veće je bolje (maksimizacija)
    - manje je bolje (minimizacija)
- višekriterijsko (multi-objective) → ? → rješenja mogu biti neusporediva; dominacija, Pareto fronta, gustoća rješenja, …

Lokalni optimum
rješenje iz kojega se algoritam svojim operatorima ne može izvući i nastaviti dalje pretragu; oko njega su rješenja lošije kvalitete pa on pretragu povlači prema sebi

Faze pretrage

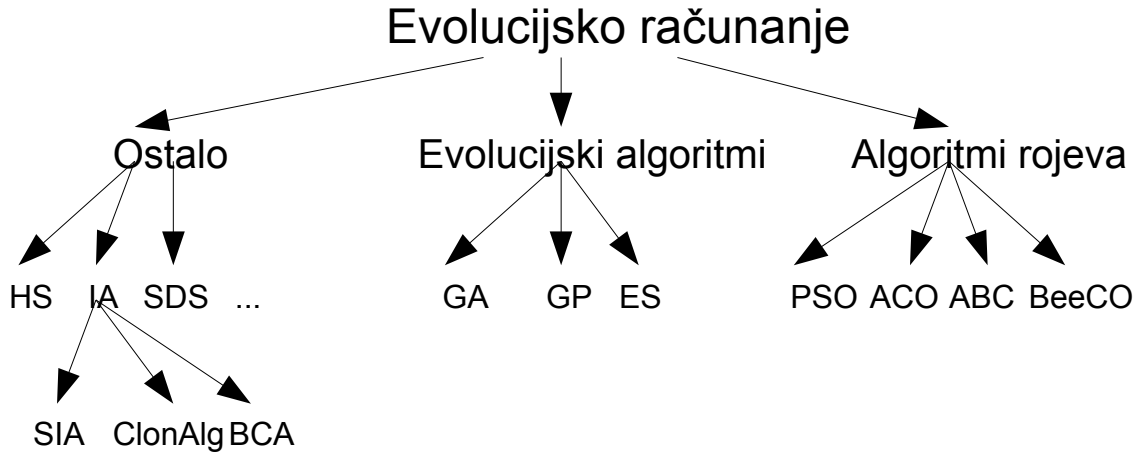
- gruba pretraga, veliki prostor, konvergencijom prelazi u:
- fina pretraga, mali prostor

Prerana konvergencija → zapinjanje u lokalnom optimumu

Populacije → problem duplikata → vodi ka preranoj konvergenciji

Generacijski vs. steady-state algoritmi

Operatori
- odabir jedinki roditelja
- križanje
- mutacija
- diferencijacija
- hipermutacija, hipermakromutacija
- odabir jedinke za zamjenu
- starenja
- …



Mutacija → povećavanje raznolikosti; exploration
Križanje → fina pretraga, smanjuje raznolikost; exploitation
Nužan balans: premala raznolikost → prerana konvergencija; prevelika raznolikost → nasumična pretraga

Ograničenja
- tvrda (hard constraints), rješenje koje daje algoritam ga ne smije prekršiti
  - međutim, algoritam interno može dopuštati takva rješenja
- meka (soft constraints), poželjno je da budu zadovoljena, ali nije nužno

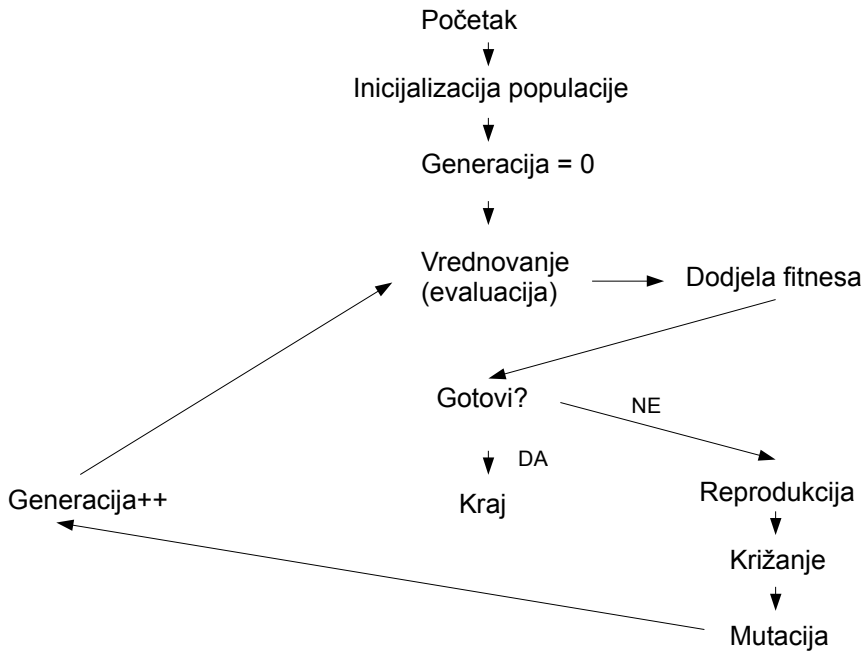
Složenost problema:
- razlikovati problem odluke (postoji li u TSP-u ciklus kraći ili duljine jednake 173 km) vs. optimizacijski problem (koji je najkraći put u TSP-u)
- problemi odluke imaju svoju klasifikaciju: P vs NP vs NP Complete vs NP Hard
- za optimizacijske probleme ovo direktno ne vrijedi, iako se često koristi jer se problemi odluke mogu u određenom smislu preslikati u optimizacijske probleme

Elitizam → algoritam čuva najbolje pronađeno rješenje
Selekcijski pritisak → u kojoj mjeri bolja rješenja imaju veću šansu biti izabrana?
Prevelik selekcijski pritisak → prerana konvergencija
Premali selekcijski pritisak → slučajna pretraga
Smanjenju selekcijskog pritiska može doprinjeti višepopulacijski algoritam; umjesto jedne populacije veličine N možemo koristiti k populacija veličine N/k uz prikladnu razmjenu jedinki → korak prema paralelizaciji i distribuiranim algoritmima.

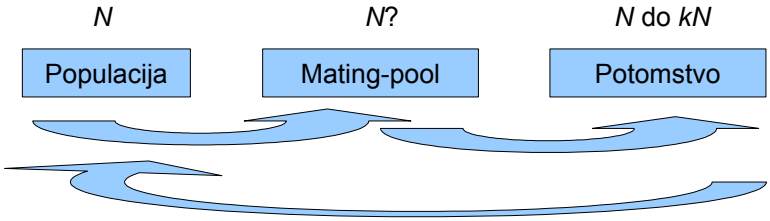
Primjeri paralelizacije unutar jednog algoritma:

- paralelno računanje fitnesa
- paralelno stvaranje / računanje čitave jedinke
- u slučaju više populacija, paralelizacija na razini populacije (bilo dretva po populaciji, bilo fiksni broj dretvi jednak broju procesora uz red poslova koje treba odraditi)

Genetski algoritam – "opći" oblik



Reprodukcija: zadatak je stvoriti "mating-pool" u kojem bolja rješenja imaju duplikate a gora rješenja su eliminirana; primjeri operatora: turnirska selekcija, proporcionalna selekcija, rangirajuća selekcija; potom se iz mating-pool-a slučajno biraju po dva roditelja, radi se križanje i mutacija i popunjava nova generacija.



Na slici: veći *k* daje veću šansu roditeljima da ispolje svoje dobre karakteristike i prenesu ih na potomstvo.

Definira se vjerojatnost križanja p_c; uz vjerojatnost $(1-p_c)$ roditelji se samo kopiraju u novu generaciju. Elitizam → npr. napraviti uniju populacije i stvorenog potomstva, i iz tog skupa odabrati *N* najboljih jedinki za novu populaciju.

Primjer: imamo populaciju od *N* jedinki. Slučajno odaberemo dvije jedinke, napravimo turnir, bolju gurnemo u mating pool; od preostalih biramo opet dvije, radimo turnir, bolju gurnemo u mating pool; ... Ponavljamo dok ne popunimo mating-pool; ako smo potrošili sve, ponovno krenemo iz čitave populacije. Na ovaj način će svaka jedinka dobiti 0, 1 ili 2 kopije.
k-turnirska selekcija: biramo *k* jedinki, uzimamo najbolju; što je k veći, selekcijski pritisak je veći

Proporcionalna selekcija: svakoj jedinki dodjeljujemo broj:

$$l_i = \frac{fitness(i)}{\sum_{j=1}^{N} fitness(j)}$$

Uočimo,

$$\sum_{i=1}^{N} l_i = 1$$

Biramo random broj *r* iz [0,1], i uzimamo prvu jedinku *k* za koju vrijedi:

$$\sum_{i=1}^{k} l_i \leqslant r$$

Izvlačenje ponavljamo onoliko puta koliko trebamo roditelja.

Postoji problem skale!

Rangirajuća selekcija: sortiramo populaciju; najgoroj jedinki damo fitness 1, sljedećoj 2, ..., sve do najbolje koja dobije N. Taj fitness koristimo dalje za proporcionalnu selekciju.

Stohastičko univerzalno uzorkovanje (stohastic universal sampling): generirati jedan slučajni broj *r* iz [0,1]. Njime je određeno svih *N* roditelja koje biram, na sljedeći način. Stvorim skup *R* od *N* brojeva:

$$R = \{ r, r + \frac{1}{N}, r + \frac{2}{N}, \ldots, N - \frac{1}{N} \} mod\ 1$$

Svaki element ovog skupa određuje jednog roditelja, baš kao u proporcionalnoj selekciji. Prednost: odmah se dobije svih *N* roditelja, također bolje "ponašanje" od klasične proporcionalne selekcije.

Križanje

- izvedba ovisi o prikazu kromosoma;
  - binarni prikaz (genotip):
    - s jednom točkom prijeloma, poopćenje na k-točaka prijeloma
    - uniformno križanje
  - broj / polje brojeva
    - aritmetička sredina
    - težinska sredina

Mutacija

- opet ovisi o prikazu kromosoma
  - binarni prikaz
    - definirana vjerojatnost mutacije bita p_m; za svaki bit izvlačimo slučajan broj iz [0,1]; ako je manji od p_m, taj bit okrećemo
  - decimalni broj
    - broju dodajemo slučajno generiran broj iz normalne distribucije s parametrima (0,sigma); koliki je sigma prikladan?

DZ. Napraviti GA koji rješava problem poopćene Rastiginove funkcije (poglavlje 6.1).
a) steady-state izvedba: koristiti troturnirsku selekciju (slučajno se odaberu tri jednike iz populacije; dvije bolje postaju roditelji; dijete se umeće u populaciju umjesto treće – najgore – jedinke)
b) generacijski s elitizmom: koristiti mating-pool jednake veličine kao i populacija; puniti ga turnirski; stvoriti k*N djece (k=2) i iz unije s populacijom odabrati N za sljedeću generaciju

http://aule.zemris.fer.hr/nastava/ui/Cupic2009-PrirodomInspiriraniOptimizacijskiAlgoritmi.pdf

Teme:

1. Vrste optimizacijskih problema.
2. Genetski algoritam primjenjen na problem optimizacije kontinuirane funkcije.
3. Genetski algoritam primjenjen na problem kombinatoričke optimizacije.
4. Algoritam diferencijalne evolucije (DE).
5. Algoritam mravlje kolonije.
6. Algoritam roja čestica.
7. Imunološki algoritmi.
8. Višekriterijska optimizacija.
9. Genetski algoritmi za višekriterijsku optimizaciju.
10. Imunološki algoritmi za višekriterijsku optimizaciju.
11. Paralelizacija kod evolucijskih algoritama.