**OPTIMIRANJE EVOLUCIJSKIM RAČUNANJEM**

Podjela optimizacijskih problema prema:

* Domeni nad kojom je problem definiran
* Ograničenjima koja su postavljena na domenu
* Prema broju oprečnih kriterija koje pokušavamo zadovoljiti

**Tvrda ograničenja** (*engl. hard constraints*) – rješenje koje ih ne zadovoljava je neprihvatljivo. Dijeli prostor rješenja u **prostor prihvatljivih rješenja** (*engl. feasible solutions*) te u **prostor neprihvatljivih rješenja** (*engl. unfeasible solutions).*

**Meka ograničenja** (*engl. soft constraints*) – rješenje koje ih ne zadovoljava je i dalje prihvatljivo. Ograničenja govore o kvaliteti rješenja.

**Problem jednokriterijske optimizacije**

Rješenje je **vektor decizijskih varijabli** .

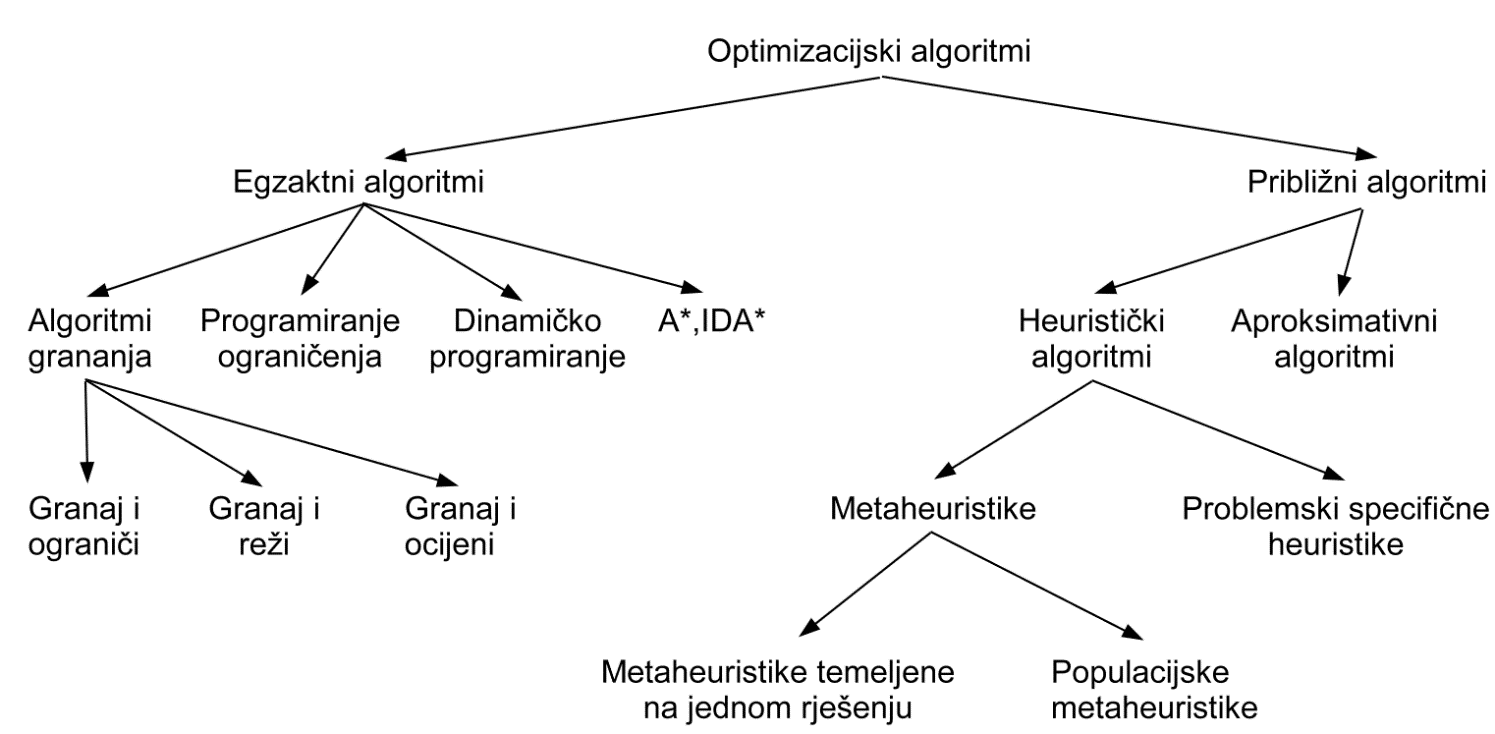
i predstavljaju skup rješenja koja **moraju** biti zadovoljena za sva prihvatljiva rješenja.

predstavlja **donju** i **gornju** **granicu** na vrijednosti koje decizijske varijable smiju poprimiti.

* Minimiziraj / maksimiziraj
* Uz zadovoljenje ograničenja

**Globalni optimum –** rješenje naziva se globalnim optimumom ako i samo ako rješenje pripada prostoru prihvatljivih rješenja i ako za svako drugo rješenje iz prostora prihvatljivih rješenja vrijedi . Nije nužno jedinstven.

**Lokalni optimum** - rješenje naziva se globalnim optimumom ako i samo ako rješenje pripada prostoru prihvatljivih rješenja i ako za svako drugo rješenje iz prostora -okoline od  vrijedi .

**Podjela optimizacijskih algoritama**

**Heuristika** –jednostavna iskustvena pravila specifična za pojedine probleme i pomažu u njegovom efikasnijem rješavanju.

**Metaheuristika** – skup algoritamskih koncepata koji koristimo za definiranje heurističkih metoda primjenjivih na širok skup problema. **Heuristika opće namjene**.

**Hiperheuristika** –heuristika čija je zadaća odabrati i podesiti prikladnu metaheuristiku.

Heuristike djelimo na **konstrukcijske algoritme** i **algoritme lokalne pretrage**

Dvije faze pretraživanja prostora optimizacijskog procesa su:

1. **Gruba pretraga** – nasumično uzorkovanje rješenja iz (nadamo se) čitavog prostora pretraživanja kako bi se pronašao obećavajući podprostor.
2. **Fina pretraga** – fokusiranje pretraživanja obećavajućeg podprostora. Potraga za globalnim optimumom

Genetski algoritam koristi dva operatora: **križanje** i **mutaciju**

**No-free-lunch teorem** – ne postoji najbolji optimizacijski algoritam.

---------------------------

POGLAVLJE 3. U KNJIZI

---------------------------

Operatori modificiranja rješenja – imaju ulogu diverzifikacije odnosno bijega iz lokalnog optimuma.

Operatori kombiniranja rješenja uzimaju u obzir 2 ili više postojećih rješenja te temeljem njih generiraju novo rješenje koje zadržava dobre karakteristike

**Algoritmi lokalne pretrage** – dopuna algoritmima evolucijskog računanja

1. **Iterativni algoritam pretraživanja**

x(0) // početno rješenje

t = 0

**DO** {

Generiraj N(x(t)) tj. susjedstvo rješenja x(t)

x(t+1) = odaberi neko rješenje iz generiranog susjedstva N(x(t))

t = t + 1

} **WHILE** (uvjet zaustavljanja nije zadovoljen)

**RETURN** x(t)

1. **Pohlepni algoitam uspona na vrh**

X(0) // početno rješenje

t = 0

**DO** {

Generiraj N(x(t)) tj. susjedstvo rješenja x(t)

x(t+1) = odaberi neko rješenje iz generiranog susjedstva N(x(t))

t = t + 1

**IF** (x(t) == x(t+1)) {

**BREAK**

}

} **WHILE** (uvjet zaustavljanja nije zadovoljen)

Tri inačice algoritma:

1. Generira se cjelokupno susjedstvo te se pronalazi najbolje rješenje (računski zahtjevno)
2. Susjedstvo se generira rješenje po rješenje i postupak se zaustavlja čim se pronađe prvo bolje rješenje. Ako je rješenje lokalni optimum prethodni slučaj iscrpnog pretraživanja
3. Slučajnim mehanizmom odabire se neko od rješenja iz susjedstva koja su bolja od trenutnog

Zaglavljivanje u lokalnom optimumu pokušavamo riješiti sljedećim strategijama :

1. **Iterativno pretraživanje algoritma s različitih početnih rješenja**
2. **Prihvaćanje rješenja koja nisu bolja od trenutnog**
3. **Promjena susjedstva**
4. **Višekratno pokretanje lokalne pretrage** (*engl. Multistart local search*)

x'' = null // još nema najboljeg rješenja

**DO** {

x = generiraj slučajno najbolje rješenje

x' = primjeni lokalnu pretragu na rješenje x

**IF** (x'' == null || x' bolji od x'') {

x'' = x'

kraj

} **WHILE** (uvjet zaustavljanja nije zadovoljen)

**RETURN** x''

1. **Iterativna lokalna pretraga** – poboljšanje prethodnog algoritma

x0 = generiraj slučajno početno rješenje ili ga preuzmi izvana

x = primjeni lokalnu pretragu na x0

**DO** {

x' = **perturbiraj**(x, povijesni podatci)

x'' = primjeni lokalnu pretragu na x'

x = **prihvati**(x, x'', memorija)

} **WHILE** (uvjet zaustavljanja nije zadovoljen)

**RETURN** najbolje pronađeno rješenje

1. **Pohlepna randomizirana adaptivna procedura pretrage** – GRASP (*engl. Greedy randomized adaptive search procedure)*

x = null

**DO** {

x' = pohlepnim slučajnim algoritmom konstruiraj novo početno rješenje

x'' = primjeni lokalnu pretragu na x'

x = prihvati(x, x'')

} **WHILE** (uvjet zaustavljanja nije zadovoljen)

**RETURN** najbolje rješenje tj. x

* Početna rješenja se stvaraju element po element koristeći pohlepnu nasumičnu proceduru.
  + Rješenja nisu skroz slučajna, već u sebi sadrže dijelove rješenja koja su vjerojatno na putu globalnog optimuma.
  + Tipično se koriste **ograničene liste kandidata, RCL** (*engl. restricted candidate list*).

**SUSJEDSTVO** – neka je neko rješenje iz skupa mogućih rješenja. Susjedstvo rješenja , oznaka , definirano je funkcijom susjedstva koja predstavlja preslikavanje . Ova funkcija svakom rješenju pridružuje podskup skupa koji čini skup svih rješenja koja zovemo **susjedima od rješenja** .

1. Problem optimizacije nad kontinuiranim prostorom susjedstvo je područje čija je udaljenost od točke x manja od .
2. Kod kombinatoričkih problema definiramo funkciju pomaka skup svih riječi (bitova) koje je moguće dobiti uporabom operatora nad zadanim rješenjem .