Rješavanje optimizacijskih problema algoritmima evolucijskog računanja u Javi
Uvod

dr.sc. Marko Čupić

Fakultet elektrotehnike i računarstva
Sveučilište u Zagrebu
Akademska godina 2013./2014.

03. listopada 2013.

O čemu se tu radi?

Što je u imenu?

- Rješavanje:

O čemu se tu radi?

Što je u imenu?

- Rješavanje: suprotno od: načuo sam nešto o tome, nemam pojma što bih s time radio

O čemu se tu radi?

Što je u imenu?

- Rješavanje: suprotno od: načuo sam nešto o tome, nemam pojma što bih s time radio
- Optimizacijski problema:

O čemu se tu radi?

Što je u imenu?

- Rješavanje: suprotno od: načuo sam nešto o tome, nemam pojma što bih s time radio
- Optimizacijski problema: puno stvari koje susrećemo u životu može se svesti na problem optimizacije – traženje ekstrema funkcije: minimalna kazna, maksimalna dobit

O čemu se tu radi?

Što je u imenu?

- Rješavanje: suprotno od: načuo sam nešto o tome, nemam pojma što bih s time radio
- Optimizacijski problema: puno stvari koje susrećemo u životu može se svesti na problem optimizacije – traženje ekstrema funkcije: minimalna kazna, maksimalna dobit
- Algoritmima evolucijskog računanja:

O čemu se tu radi?

Što je u imenu?

- Rješavanje: suprotno od: načuo sam nešto o tome, nemam pojma što bih s time radio
- Optimizacijski problema: puno stvari koje susrećemo u životu može se svesti na problem optimizacije – traženje ekstrema funkcije: minimalna kazna, maksimalna dobit
- Algoritmima evolucijskog računanja: široka porodica algoritama, najčešće populacijskih no ne nužno, kojima upomoć može priskočiti još širi skup algoritama

O čemu se tu radi?

Što je u imenu?

- Rješavanje: suprotno od: načuo sam nešto o tome, nemam pojma što bih s time radio
- Optimizacijski problema: puno stvari koje susrećemo u životu može se svesti na problem optimizacije – traženje ekstrema funkcije: minimalna kazna, maksimalna dobit
- Algoritmima evolucijskog računanja: široka porodica algoritama, najčešće populacijskih no ne nužno, kojima upomoć može priskočiti još širi skup algoritama
- U Javi:

O čemu se tu radi?

Što je u imenu?

- Rješavanje: suprotno od: načuo sam nešto o tome, nemam pojma što bih s time radio
- Optimizacijski problema: puno stvari koje susrećemo u životu može se svesti na problem optimizacije – traženje ekstrema funkcije: minimalna kazna, maksimalna dobit
- Algoritmima evolucijskog računanja: široka porodica algoritama, najčešće populacijskih no ne nužno, kojima upomoć može priskočiti još širi skup algoritama
- U Javi: programski jezik u kojem ćete raditi sve implementacije i GUI gdje bude potrebno

Organizacija

- Termini predavanja
- Domaće zadaće
- Predaja domaćih zadaća

Osoblje

- Marko Čupić – predavanja i sve grozno vezano uz vještinu
- TA
  - … to be found yet …

Što želimo rješavati?

Optimizacijske probleme koji su teški!

- Ako su optimizacijski problemi lagani, postoje jednostavniji algoritmi.
- Jednostavniji algoritmi su efikasniji!
- Evolucijsko računanje nije optimalno rješenje za svaki optimizacijski problem.
- Populacijski algoritmi su posebno zahtjevni – barataju s čitavim populacijama mogućih kandidata za rješenje
  - Visok računski trošak
  - Memorijski mogu biti vrlo zahtjevni

Što je optimizacijski problem?

Imamo podjele po nekoliko kriterija:

- Prema domeni nad kojom je problem definiran:
  - Kontinuirana domena (skup realnih brojeva)
  - Diskretna domena (kombinatorički problemi)

Što je optimizacijski problem?

Imamo podjele po nekoliko kriterija:

- Prema domeni nad kojom je problem definiran:
  - Kontinuirana domena (skup realnih brojeva)
  - Diskretna domena (kombinatorički problemi)
- Prema ograničenjima koja su postavljena na domenu:
  - Bez ograničenja (npr. čitav skup realnih brojeva)
  - Postoje ograničenja (primjerice, mora vrijediti $\sin(x_1) + e^{x_2 \cdot \cos x_3} \cdot x_4 > 0$)

Što je optimizacijski problem?

Imamo podjele po nekoliko kriterija:

- Prema domeni nad kojom je problem definiran:
  - Kontinuirana domena (skup realnih brojeva)
  - Diskretna domena (kombinatorički problemi)
- Prema ograničenjima koja su postavljena na domenu:
  - Bez ograničenja (npr. čitav skup realnih brojeva)
  - Postoje ograničenja (primjerice, mora vrijediti $\sin(x_1) + e^{x_2 \cdot \cos x_3} \cdot x_4 > 0$)
- Prema broju oprečnih kriterija koje pokušavamo zadovoljiti:
  - Jednokriterijska optimizacija (TSP: nađi najkraći Hamiltonov ciklus)
  - Višekriterijska optimizacija (TSP, ali dodatno minimiziraj broj preleta preko rijeka, i minimiziraj broj preleta preko planina; vjerojatno – više planina $\leftrightarrow$ manje rijeka)

Vrste ograničenja

Razlikujemo dvije vrste ograničenja:

- Tvrda ograničenja (engl. *hard constraints*)
  rješenje koje ih ne zadovoljava je neprihvatljivo; tvrda ograničenja dijele prostor rješenja u dva podprostora: prostor prihvatljivih rješenja (engl. *feasible solutions*) te u prostor neprihvatljivih rješenja (engl. *unfeasible solutions*).

Vrste ograničenja

Razlikujemo dvije vrste ograničenja:

- Tvrda ograničenja (engl. *hard constraints*)
  rješenje koje ih ne zadovoljava je neprihvatljivo; tvrda ograničenja dijele prostor rješenja u dva podprostora: prostor prihvatljivih rješenja (engl. *feasible solutions*) te u prostor neprihvatljivih rješenja (engl. *unfeasible solutions*).
- Meka ograničenja (engl. *soft constraints*)
  rješenje koje ih ne zadovoljava i dalje je prihvatljivo; no što su ova ograničenja manje prekršena, to je rješenje bolje; ova ograničenja govore o kvaliteti rješenja (raspored u kojem studenti imaju pauzu za ručak *vs.* raspored u kojem studenti nemaju pauzu za ručak *vs.* raspored u kojem studenti imaju puno pauza za ručak)

Jednokriterijska optimizacija

Definition (Problem jednokriterijske optimizacije.)

Općeniti problem jednokriterijske optimizacije definiran je na sljedeći način.

$$\begin{array}{rlr} \text{Minimiziraj / maksimiziraj} & f(\vec{x}), & \\ \text{uz zadovoljenje ograničenja} & g_j(\vec{x}) \geq 0, & j = 1, 2, \ldots, J \\ & h_k(\vec{x}) = 0, & k = 1, 2, \ldots, K \\ & x_i^L \leq x_i \leq x_i^U, & i = 1, 2, \ldots, n. \end{array}$$

Rješenje $\vec{x}$ je vektor decizijskih varijabli $\vec{x} = \{x_1, x_2, \ldots, x_n\}$. Prvi i drugi skup ograničenja prestavljaju skup nejednakosti odnosno jednakosti koje moraju biti zadovoljene za sva prihvatljiva rješenja. Treći skup ograničenja definira donju i gornju granicu na vrijednosti koje decizijske varijable smiju poprimiti.

Globalni i lokalni optimumi

Definition (Globalni optimum.)

Kod jednokriterijske optimizacije, rješenje $\vec{x}^*$ naziva se globalnim optimumom ako i samo ako rješenje $\vec{x}^*$ pripada prostoru prihvatljivih rješenja i ako za svako drugo rješenje $\vec{x}$ iz prostora prihvatljivih rješenja vrijedi $f(\vec{x}^*) \geq f(\vec{x})$, gdje je f funkcija dobrote rješenja.

Nije nužno jedinstven!

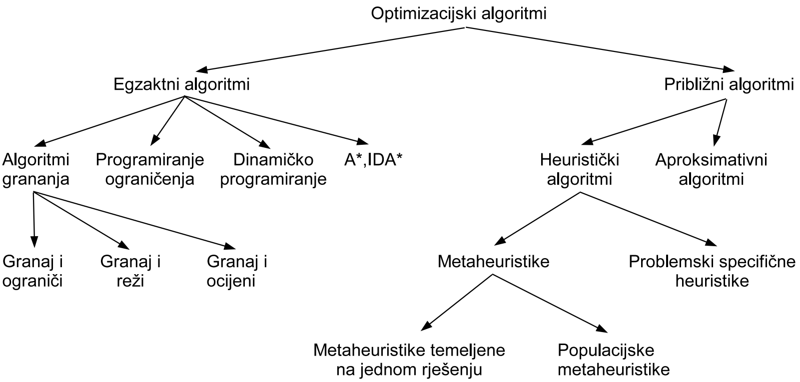
Globalni i lokalni optimumi

Definition (Lokalni optimum.)

Kod jednokriterijske optimizacije, rješenje $\vec{x}^*$ naziva se lokalnim optimumom ako i samo ako rješenje $\vec{x}^*$ pripada prostoru prihvatljivih rješenja i ako za svako drugo rješenje $\vec{x}$ iz δ-okoline od $\vec{x}^*$, tj. uz $\delta > 0$ i $|\vec{x} - \vec{x}^*| \leq \delta$ vrijedi $f(\vec{x}^*) \geq f(\vec{x})$, gdje je f funkcija dobrote rješenja.

Globalni optimum jest lokalni optimum!

Optimizacijski postupci



Slika: Podjela optimizacijskih algoritama

*ristika

Višestruke moguće interpretacije; smatrat ćemo da vrijedi sljedeće:

- Heuristika:

*ristika

Višestruke moguće interpretacije; smatrat ćemo da vrijedi sljedeće:

- Heuristika: jednostavna iskustvena pravila koja su specifična za pojedine probleme i pomažu u njegovom efikasnijem rješavanju; na prethodnom slideu – problemski specifične heuristike

*ristika

Višestruke moguće interpretacije; smatrat ćemo da vrijedi sljedeće:

- Heuristika: jednostavna iskustvena pravila koja su specifična za pojedine probleme i pomažu u njegovom efikasnijem rješavanju; na prethodnom slideu – problemski specifične heuristike
- Metaheuristika:

*ristika

Višestruke moguće interpretacije; smatrat ćemo da vrijedi sljedeće:

- Heuristika: jednostavna iskustvena pravila koja su specifična za pojedine probleme i pomažu u njegovom efikasnijem rješavanju; na prethodnom slideu – problemski specifične heuristike
- Metaheuristika: skup algoritamskih koncepata koji koristimo za definiranje heurističkih metoda primjenjivih na širok skup problema; heuristika opće namjene čiji je zadatak usmjeravanje problemski specifičnih heuristika prema području u prostoru rješenja u kojem se nalaze dobra rješenja

*ristika

Višestruke moguće interpretacije; smatrat ćemo da vrijedi sljedeće:

- Heuristika: jednostavna iskustvena pravila koja su specifična za pojedine probleme i pomažu u njegovom efikasnijem rješavanju; na prethodnom slideu – problemski specifične heuristike
- Metaheuristika: skup algoritamskih koncepata koji koristimo za definiranje heurističkih metoda primjenjivih na širok skup problema; heuristika opće namjene čiji je zadatak usmjeravanje problemski specifičnih heuristika prema području u prostoru rješenja u kojem se nalaze dobra rješenja
- Hiperheuristika:

*ristika

Višestruke moguće interpretacije; smatrat ćemo da vrijedi sljedeće:

- Heuristika: jednostavna iskustvena pravila koja su specifična za pojedine probleme i pomažu u njegovom efikasnijem rješavanju; na prethodnom slideu – problemski specifične heuristike
- Metaheuristika: skup algoritamskih koncepata koji koristimo za definiranje heurističkih metoda primjenjivih na širok skup problema; heuristika opće namjene čiji je zadatak usmjeravanje problemski specifičnih heuristika prema području u prostoru rješenja u kojem se nalaze dobra rješenja
- Hiperheuristika: heuristika čija je zadaća odabrati i podesiti prikladnu metaheuristiku

*ristika

Višestruke moguće interpretacije; smatrat ćemo da vrijedi sljedeće:

- Heuristika: jednostavna iskustvena pravila koja su specifična za pojedine probleme i pomažu u njegovom efikasnijem rješavanju; na prethodnom slideu – problemski specifične heuristike
- Metaheuristika: skup algoritamskih koncepata koji koristimo za definiranje heurističkih metoda primjenjivih na širok skup problema; heuristika opće namjene čiji je zadatak usmjeravanje problemski specifičnih heuristika prema području u prostoru rješenja u kojem se nalaze dobra rješenja
- Hiperheuristika: heuristika čija je zadaća odabrati i podesiti prikladnu metaheuristiku

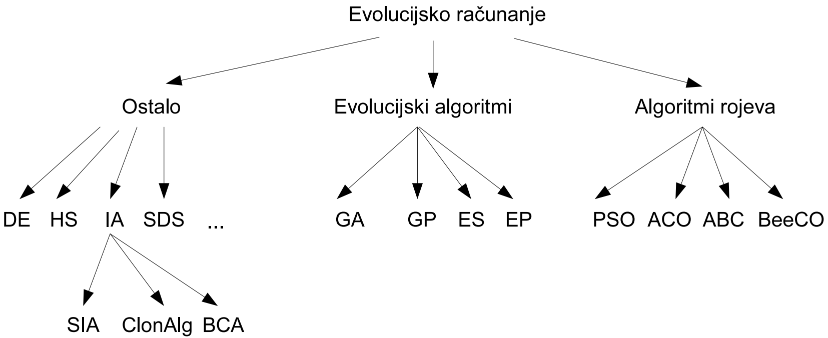
Možemo li optimirati parametre metaheuristike?

Heuristike

Heuristike dijelimo na:

- Konstrukcijske algoritme
- Algoritme lokalne pretrage (tj. algoritme iterativnog poboljšavanja)

Evolucijsko računanje: podskup metaheuristika



Slika: Podjela evolucijskog računanja na glavne grane. Uz svaku granu prikazani su i odabrani algoritmi.

Proces pretraživanja

Pretraživanje prostora kreću od jednog ili više početnih rješenja i potom temeljem postojećih rješenja (ili njihovog indirektnog utjecaja) generiraju nova rješenja.

Da bi optimizacijski proces u takvim uvjetima mogao biti uspješan, postupak pretraživanja prostora rješenja provodi se u dvije faze.

Proces pretraživanja

Pretraživanje prostora kreću od jednog ili više početnih rješenja i potom temeljem postojećih rješenja (ili njihovog indirektnog utjecaja) generiraju nova rješenja.

Da bi optimizacijski proces u takvim uvjetima mogao biti uspješan, postupak pretraživanja prostora rješenja provodi se u dvije faze.

1. Gruba pretraga. U fazi grube pretrage algoritam nasumično uzorkuje rješenja iz (nadamo se) čitavog prostora pretraživanja kako bi pronašao podprostor koji sadrži obećavajuća rješenja.

Proces pretraživanja

Pretraživanje prostora kreću od jednog ili više početnih rješenja i potom temeljem postojećih rješenja (ili njihovog indirektnog utjecaja) generiraju nova rješenja.

Da bi optimizacijski proces u takvim uvjetima mogao biti uspješan, postupak pretraživanja prostora rješenja provodi se u dvije faze.

1. Gruba pretraga. U fazi grube pretrage algoritam nasumično uzorkuje rješenja iz (nadamo se) čitavog prostora pretraživanja kako bi pronašao podprostor koji sadrži obećavajuća rješenja.
2. Fina pretraga. Nakon lociranja obećavajućeg podprostora nastupa faza fine pretrage u kojoj se pretraživanje fokusira. U ovom fazi istražuje se okolica pronađenog podprostora u potrazi za globalnim optimumom.

Proces pretraživanja

Pretraživanje prostora kreću od jednog ili više početnih rješenja i potom temeljem postojećih rješenja (ili njihovog indirektnog utjecaja) generiraju nova rješenja.

Da bi optimizacijski proces u takvim uvjetima mogao biti uspješan, postupak pretraživanja prostora rješenja provodi se u dvije faze.

1. Gruba pretraga. U fazi grube pretrage algoritam nasumično uzorkuje rješenja iz (nadamo se) čitavog prostora pretraživanja kako bi pronašao podprostor koji sadrži obećavajuća rješenja.
2. Fina pretraga. Nakon lociranja obećavajućeg podprostora nastupa faza fine pretrage u kojoj se pretraživanje fokusira. U ovom fazi istražuje se okolica pronađenog podprostora u potrazi za globalnim optimumom.

Lokalna pretraga, memetički algoritmi?

Proces pretraživanja

Genetski algoritam kao jedan od primjera algoritama evolucijskog računanja koristi dva operatora:

Proces pretraživanja

Genetski algoritam kao jedan od primjera algoritama evolucijskog računanja koristi dva operatora:

1. Operator križanja:

Proces pretraživanja

Genetski algoritam kao jedan od primjera algoritama evolucijskog računanja koristi dva operatora:

1. Operator križanja: primjerice, dijete je aritmetička sredina svojih roditelja

Proces pretraživanja

Genetski algoritam kao jedan od primjera algoritama evolucijskog računanja koristi dva operatora:

1. Operator križanja: primjerice, dijete je aritmetička sredina svojih roditelja; dovodi do kompresije populacije $\rightarrow$ lokalni optimum?

Proces pretraživanja

Genetski algoritam kao jedan od primjera algoritama evolucijskog računanja koristi dva operatora:

1. Operator križanja: primjerice, dijete je aritmetička sredina svojih roditelja; dovodi do kompresije populacije $\rightarrow$ lokalni optimum?
2. Operator mutacije:

Proces pretraživanja

Genetski algoritam kao jedan od primjera algoritama evolucijskog računanja koristi dva operatora:

1. Operator križanja: primjerice, dijete je aritmetička sredina svojih roditelja; dovodi do kompresije populacije $\rightarrow$ lokalni optimum?
2. Operator mutacije: uvođenje nasumične promjene u rješenje

Proces pretraživanja

Genetski algoritam kao jedan od primjera algoritama evolucijskog računanja koristi dva operatora:

1. Operator križanja: primjerice, dijete je aritmetička sredina svojih roditelja; dovodi do kompresije populacije $\rightarrow$ lokalni optimum?
2. Operator mutacije: uvođenje nasumične promjene u rješenje; dovodi do raspršenja populacije $\rightarrow$ uništava "naučeno"?

Proces pretraživanja

Genetski algoritam kao jedan od primjera algoritama evolucijskog računanja koristi dva operatora:

1. Operator križanja: primjerice, dijete je aritmetička sredina svojih roditelja; dovodi do kompresije populacije $\rightarrow$ lokalni optimum?
2. Operator mutacije: uvođenje nasumične promjene u rješenje; dovodi do raspršenja populacije $\rightarrow$ uništava "naučeno"?

Potrebno je pronaći dobar balans između ovih operatora!

Proces pretraživanja

Genetski algoritam kao jedan od primjera algoritama evolucijskog računanja koristi dva operatora:

1. Operator križanja: primjerice, dijete je aritmetička sredina svojih roditelja; dovodi do kompresije populacije $\rightarrow$ lokalni optimum?
2. Operator mutacije: uvođenje nasumične promjene u rješenje; dovodi do raspršenja populacije $\rightarrow$ uništava "naučeno"?

Potrebno je pronaći dobar balans između ovih operatora!
Većina algoritama su parametrizirani i za prilagodbu problemu koji rješavaju potrebno je pronaći povoljan skup parametara.

Najbolji optimizacijski algoritam?

Matematički je dokazano: ništa od toga...

Najbolji optimizacijski algoritam?

Matematički je dokazano: ništa od toga...

Wolpert i Macready dokazali da nema najboljeg algoritma: *no-free-lunch* teorem

> *All algorithms that search for an extremum of a cost function perform exactly the same, according to any performance measure, when averaged over all possible cost functions. In particular, if algorithm A outperforms algorithm B on some cost functions, then loosely speaking there must exist exactly as many other functions where B outperforms A.*

Najbolji optimizacijski algoritam?

Matematički je dokazano: ništa od toga...

Wolpert i Macready dokazali da nema najboljeg algoritma: *no-free-lunch* teorem

> *All algorithms that search for an extremum of a cost function perform exactly the same, according to any performance measure, when averaged over all possible cost functions. In particular, if algorithm A outperforms algorithm B on some cost functions, then loosely speaking there must exist exactly as many other functions where B outperforms A.*

To je upravo razlog za upoznavanje većeg broja optimizacijskih algoritama – treba naučiti kako radi algoritam te na koju je vrstu problema primjenjiv!

Prikaz rješenja

Ovisno o domeni nad kojom je definiran optimizacijski problem, rješenja možemo u računalu prikazivati na različite načine.

- Prikaz nizom bitova.
- Prikaz poljem ili vektorom brojeva.
- Prikaz permutacijama i matricama.
- Prikaz složenijim strukturama podataka.
- Prikaz stablima.
- Ostali prikazi.

Prikaz nizom bitova – direktno

Ponekad se kao rješenje traži lista (popis) zastavica što se trivijalno preslikava u polje bitova.

Primjerice, neka je s $\mathcal{O}$ označen skup objekata $\{o_1, o_2, \ldots, o_n\}$; zadatak je pronaći podskup skupa $o \subseteq \mathcal{O}$ nad kojim funkcija $f(o)$ poprima maksimum; funkcija $f(o)$ definirana je kao funkcija koja za svaki podskup $o \in \mathcal{O}$ vraća mjeru kvalitete tog podskupa.

Polje bitova može direktno kodirati svaki podskup; npr. rješenje 0011 bi predstavljalo podskup $\{o_3, o_4\}$.

Prikaz nizom bitova – kodiranjem

Ako su rješenja realni brojevi, može se koristiti neka vrsta kodiranja (primjerice, prirodni binarni kod, Grayev kod i slično).

- Pretpostavimo da koristimo binarni prikaz rješenja koji se sastoji od n bitova.
- Neka su legalne vrijednosti varijable x vrijednosti iz raspona $[x_{min}, x_{max}]$.
- Koristit ćemo prirodni binarni kod.

Prikaz nizom bitova – kodiranjem

Uz pretpostavke s prethodnog slidea:

1. Binarni niz se protumači kao cijeli broj. Označimo njegovu vrijednost s k. Ako se koriste n-bitovni binarni nizovi, tada će k poprimiti vrijednost iz skupa cijelih brojeva $\{0, 1, \ldots, 2^n - 1\}$.
2. Vrijednost k preslika se na interval $[x_{min}, x_{max}]$. Pri tome se najčešće $k = 0$ preslika u vrijednost x_{min}, $k = 2^n - 1$ preslika u vrijednost x_{max} a ostale vrijednosti se linearno preslikaju na interval $[x_{min}, x_{max}]$. U tom slučaju se koristi izraz:

$$x = x_{min} + \frac{k}{2^n - 1} \cdot (x_{max} - x_{min}). \tag{1}$$

Preciznost pretrage? Proširivost na prikaz više varijabli?

Svi drugi prikazi

Pogledati u knjizi poglavlje 3 te načine modificiranja rješenja i kombiniranja rješenja.

- operatori modificiranja rješenja uobičajeno imaju ulogu diverzifikacije odnosno bijega iz lokalnog optimuma
- operatori kombiniranja rješenja uzimaju u obzir dva ili više postojećih rješenja te temeljem njih generiraju novo rješenje koje (nadamo se) zadržava i unaprijeđuje dobre karakteristike rješenja iz kojih je nastalo

Pogledati u knjizi diskusiju "Genotipski i fenotipski prikaz rješenja" na kraju poglavlja 3.

Klasika na djelu

Prije no što krenemo na algoritme evolucijskog računanja, pogledat ćemo nekoliko klasičnih optimizacijskih algoritama koji se često koriste kao lokalne pretrage (odnosno kao dopuna) algoritmima evolucijskog računanja.

- Iterativni algoritam pretraživanja
- Pohlepni algoritam uspona na vrh
- Višekratno pokretanje lokalne pretrage
- Iterativna lokalna pretraga
- Pohlepna randomizirana adaptivna procedura pretrage

Susjedstvo

Definition (Susjedstvo)

Neka je $x \in \mathcal{X}$ neko rješenje iz skupa mogućih rješenja $\mathcal{X}$. Susjedstvo rješenja x, oznaka $\mathcal{N}(x)$, definirano je funkcijom susjedstva $\mathcal{N}$ koja predstavlja preslikavanje $\mathcal{N} : \mathcal{X} \rightarrow 2^{\mathcal{X}}$. Ova funkcija svakom rješenju x pridružuje podskup skupa $\mathcal{X}$ koji čini skup svih rješenja koja zovemo susjedima od rješenja x.

Kod problema optimizacije nad kontinuiranim prostorom, susjedstvo rješenja x najčešće se definira kao skup svih rješenja x' koja su od rješenja x udaljena za ne više od nekog fiksnog iznosa ϵ; u tom slučaju susjedstvo je definirano kao:

$$\mathcal{N}(x) = \{x' : |x' - x| \leq \epsilon, \;\; x' \in \mathcal{X}\}.$$

Susjedstvo

Kod kombinatoričkih problema definiramo funkciju pomaka $\pi(x)$. Primjerice, ako kao prikaz rješenja koristimo prikaz nizom bitova, operator $\pi(x)$ možemo definirati kao operator koji na najviše jednom slučajno odabranom mjestu mijenja vrijednost bita. U tom slučaju, susjedstvo možemo definirati kao skup svih riječi koje je moguće dobiti uporabom operatora π nad zadanim rješenjem x.

Iterativni algoritam pretraživanja

```
x(0) ... početno rješenje
t = 0
ponavljaj
  Generiraj N(x(t)) tj. susjedstvo rješenja x(t)
  x(t+1) = odaberi neko rješenje iz generiranog
           susjedstva N(x(t))
  t = t+1
dok nije zadovoljen uvjet zaustavljanja
vrati x(t)
```

Pohlepni algoritam uspona na vrh

```
x(0) ... početno rješenje
t = 0
ponavljaj
  Generiraj N(x(t)) tj. susjedstvo rješenja x(t)
  x(t+1) = odaberi najbolje rješenje iz generiranog
           susjedstva N(x(t))
  t = t+1
  ako je x(t)=x(t-1) prekini petlju
dok nije zadovoljen uvjet zaustavljanja
vrati x(t)
```

Pohlepni algoritam uspona na vrh

U prethodnom algoritmu kompletno susjedstvo može se, ali i ne mora se generirati. Uobičajene su tri inačice.

- Generira se cjelokupno susjedstvo te se pronalazi najbolje rješenje. U slučaju da je susjedstvo veliko, ovaj pristup će biti računski izuzetno zahtjevan.
- Susjedstvo se generira rješenje po rješenje i postupak se zaustavlja čim se pronađe prvo bolje rješenje. U slučaju da je trenutno rješenje lokalni optimum, postupak se pretvara u prethodni slučaj u kojem se radi iscrpna pretraga cjelokupnog susjedstva.
- Posredstvom slučajnog mehanizma odabire se neko od rješenja iz susjedstva koja su bolja od trenutnog (dakle, nije nužno da će biti odabrano najbolje rješenje iz susjedstva).

Povećanje robusnosti algoritama

Jedan od problema koji je preostao jest zaglavljivanje u lokalnom optimumu. Strategije kojima se borimo protiv toga su:

Iterativno pokretanje algoritma s različitih početnih rješenja.
: Pronađeni lokalni optimum ovisi o početnom rješenju. Ideja: ponavljati postupak puno puta s različitih početnih rješenja.

Prihvaćanje i rješenja koja nisu bolja od trenutnog.
: Ideja je sama po sebi jasna?

Promjena susjedstva.
: I opet, ideja je jasna: lokalni optimum koji je rezultat načina kako smo definirali susjedstvo može nestati u nekom drugom susjedstvu.

Višekratno pokretanje lokalne pretrage

Algoritam je poznat pod nazivom engl. *Multistart local search*.

```
x'' = null ... još nema najboljeg rješenja
ponavljaj
  x = generiraj slučajno početno rješenje
  x' = primjeni lokalnu pretragu na rješenje x
  ako je x''=null ili ako je x' bolji od x'' tada
   x'' = x'
  kraj
dok nije zadovoljen uvjet zaustavljanja
vrati x''
```

Iterativna lokalna pretraga

Poboljšanje prethodnog algoritma je algoritam poznat pod nazivom engl. *Iterated local search*.

```
x0 = generiraj slučajno početno rješenje
     ili ga preuzmi izvana
x = primjeni lokalnu pretragu na x0
ponavljaj
  x’ = perturbiraj(x, povijesni podatci)
  x’’ = primjeni lokalnu pretragu na x’
  x = prihvati(x, x’’, memorija)
dok nije zadovoljen uvjet zaustavljanja
vrati najbolje pronađeno rješenje
```

Pohlepna randomizirana adaptivna procedura pretrage

Algoritam *pohlepna randomizirana adaptivna procedura pretrage* poznat pod nazivom GRASP, što je kratica od engl. *Greedy randomized adaptive search procedure* - GRASP.

```
x = null
ponavljaj
  x' = pohlepnim slučajnim algoritmom konstruiraj
       novo početno rješenje
  x'' = primjeni lokalnu pretragu na x'
  x = prihvati(x, x'')
dok nije zadovoljen uvjet zaustavljanja
vrati najbolje pronađeno rješenje, tj. x
```

Pohlepna randomizirana adaptivna procedura pretrage

Umjesto da se početna rješenja stvaraju sasvim slučajno, početna se rješenja konstruiraju element po element koristeći neku pohlepnu (i po mogućnosti randomiziranu) proceduru. Ovime se osigurava da rješenja od kojih se kreće nisu skroz slučajna, već u sebi sadrže dijelove rješenja koja se vjerojatno nalaze i u rješenju koje je globalni optimum, a također zbog načina na koji su konstruirana, već imaju kvalitetu koja je bitno bolja od tipičnih nasumično stvorenih rješenja.
Tipično se koriste ograničene liste kandidata, RCL, engl. *rescricted candidate list*.

Sljedeći puta...

- Numeričke optimizacije
- Popravljanje uspješnosti generiranja početnih rješenja
- ...

Što nas sve čeka?

- Numeričke optimizacije
- Algoritam simuliranog kaljenja
- Genetski algoritam
- Algoritam diferencijske evolucije
- Mravlji algoritmi
- Algoritam roja čestica
- Umjetni imunološki algoritmi
- Višekriterijska optimizacija
- Genetski algoritam za višekriterijsku optimizaciju
- Paralelizacija algoritama