UVOD U OPTIMIZACIJE

⇒ zašto heuristike?

↳ zbog prekomplicranih problema za egzaktno rješavanje (npr. SAT problem)

⇒ modeliranje problema može biti teže od optimizacije

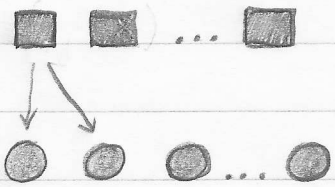
↳ pojednostavljenje stvarnosti

↳ dva pristupa:

1) $\text{MODEL}_{\text{SIMPLE}} \longrightarrow \text{SOLUTION}_{\text{EXACT}}(\text{MODEL}_{\text{SIMPLE}})$

2) $\text{MODEL}_{\text{PRECISE}} \longrightarrow \text{SOLUTION}_{\text{APPROX}}(\text{MODEL}_{\text{PRECISE}})$

PR: Želimo prevesti pakete od skladišta sk_i, $i = 1, \dots, m$ do ureda ur_j, $j = 1, \dots, k$ ▽. Minimiziraj cijenu ▽!



x_{ij} → prevezeno od sk_i do sk_j

$f_{ij}(x)$ → cijena prijevoza x paketa od skladišta sk_i do ureda ur_j

sklad(i) → u skladištu sk_i

ured(j) → potražnja ur_j

$$\min \sum_{i=1}^{n} \sum_{j=1}^{m} f_{ij}(x_{ij})$$

↳ uzeti: $x_{ij} \geq 0$

$$\sum_{j=1}^{k} x_{ij} \leq \text{sklad}(i) \quad \forall i = 1, \dots, m$$

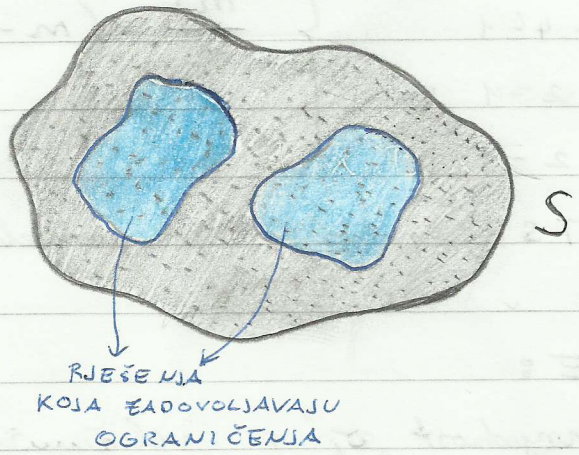
$$\sum_{i=1}^{m} x_{ij} \geq \text{ured}(j) \quad \forall j = 1, \dots, k$$

⇒ OPTIMIZACIJSKI PROBLEMI

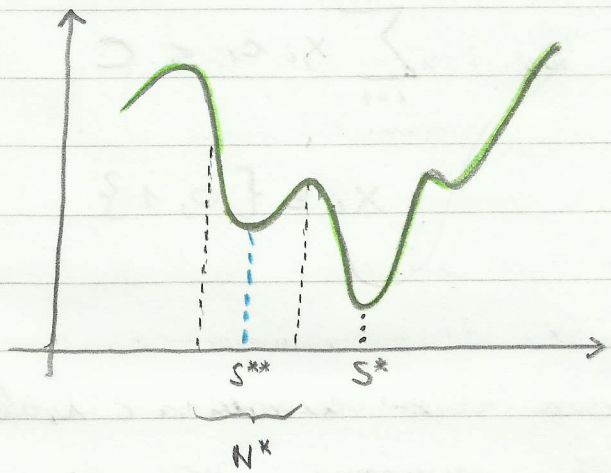
↳ uređeni par (S, f):

S → skup mogućih rješenja / prostor pretraživanja

f → funkcija cilja koju optimiziramo



⇒ lokalni i globalni optimumi:



⇒ lokalni optimum: (S^{**})

- najbolje rješenje u
u susjedstvu (N^*)

⇒ da bi metoda optimalna linearnim programiranjem, i funkcija cilja i ograničenja trebaju biti linearna

↳ ti problemi nisu teški za rješiti (npr. simpleks metodom)

⇒ za neke nelinearne probleme (npr. kvadratične i konveksne) postoje egzaktne probleme, no u općenitosti ovdje koristimo heuristike

⇒ PROBLEM TRGOVAČKOG PUTNIKA:

- imamo istih „n" rješenja (razlikuju se samo u početnoj točki):

$$\left.\begin{array}{l} 1-2-3-4 \\ 2-3-4-1 \\ 3-4-2-1 \\ 4-3-2-1 \end{array}\right\} \quad \frac{n!}{n} = (n-1)!$$

- za simetričan TSP, potrebno je podijelit sa 2: $\frac{(n-1)!}{2}$

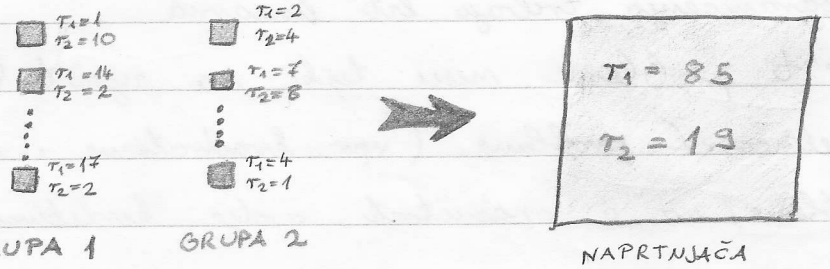
⇒ PROBLEM NAPRTNJAČE:

- predmet ima vrijednost „v_i" i veličinu „c_i"
- treba maksimizirat vrijednost koja stane u naprtnjaču ukupnog kapaciteta C:

$$\max \sum_{i=1}^{n} x_i v_i \quad \text{uz} \quad \sum_{i=1}^{n} x_i c_i \leq C$$

$$x_i \in \{0, 1\}$$

- druga formulacija:
  - naprtnjača ima više resursa, i svaki predmet troši više resursa
  - imamo „n" grupa predmeta i iz svake treba odabrati točno 1 predmet (uz maksimizaciju vrijednosti)



- ovo se naziva MMKP

- notacija:

v_{ij} → vrijednost „j"-tg predmeta u grupi „i"

$\vec{r}_{ij} = (r_{ij1}, \ldots, r_{ijm})$ $\quad \vec{r}_{ij}$ → rahtjev na resurse „j"-tog predmeta u grupi „i"

$\vec{R} = (R_1, \ldots, R_m)$ $\quad \vec{R}$ → raspoloživi resursi

- grupa „i" ima „l_i" predmeta
- svaki predmet ima „m" rahtjeva na resurse

$$\max \sum_{i=1}^{n} \sum_{j=1}^{l_i} x_{ij} v_{ij}$$

uz

$$\sum_{i=1}^{n} \sum_{j=1}^{l_i} x_{ij} \cdot r_{ijk} \leq R_k \quad ; \quad k = 1, \ldots, m$$

$$\sum_{j=1}^{l_i} x_{ij} = 1 \quad ; \quad i = 1, \ldots, n$$

$$x_{ij} \in \{0, 1\}$$

SLOŽENOST

⇒ ponovi formalne definicije za asimptotske složenosti

⇒ KLASIFIKACIJA PROBLEMA (generalno):

a) optimizacijski (nađi najbolji)

b) evaluacijski (odredi postoji li nešto)

↳ postoj li put sa pet skokova?

↳ problem odluke („da ili ne")

⇒ RAZREDI SLOŽENOSTI PROBLEMA:

POLYNOMIAL

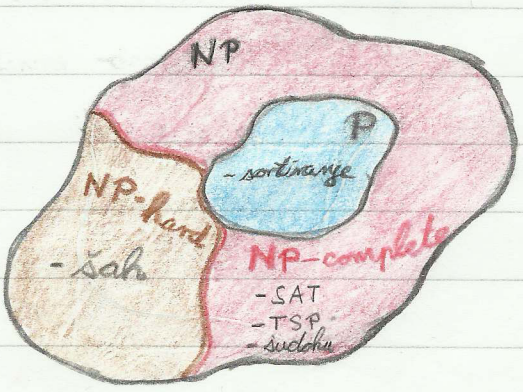
a) P ⇒ deterministički algoritam rješava ga u polinomijalnom vremenu

NON-DETERMINISTIC POLYNOMIAL

b) NP ⇒ nedeterministički algoritam rješava ga u polinomijalnom vremenu (predloženo rješenje moguće je verificirati u polinomijalnom vremenu)

⇒ NP - teški ⇒ ne može se u polinomijalnom vremenu provjeriti ponuđeno rješenje

⇒ NP - kompletno ⇒ svi problemi koji su NP - kompletni mogu se u polinomijalnom vremenu svesti jedni na druge



METODE OPTIMIZACIJE

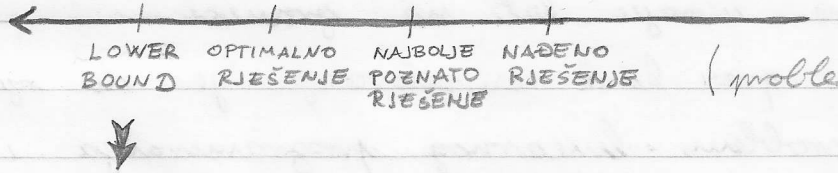
⇒ vidi podjelu na slajdovima!

⇒ **algoritmi aproksimacije:**

- faktor aproksimacije: ε
- omjer aproksimacije: r $\quad r(s) = \frac{s}{s^*}$ (minimizacija)
  $r(s) = \frac{s^*}{s}$ (maksimizacija)
- kad rješenje s ima omjer aproksimacije $r(s)$ kažemo da je to ε-aproksimacija ako je $\varepsilon = r(s) - 1$
  ↳ npr. globalni maksimum 120, a našli smo 100:
  - $\varepsilon = \frac{120}{100} - 1 = 0.2 = 20\%$
- nisu nam zanimljivi jer su specifični za problem

⇒ **heuristički algoritmi:**

- ne znamo koliko smo daleko od optimuma
- tipične „točke" na funkciji cilja:



(problem minimizacije)

ovo smo dobili tako da smo ograničenja olabavili

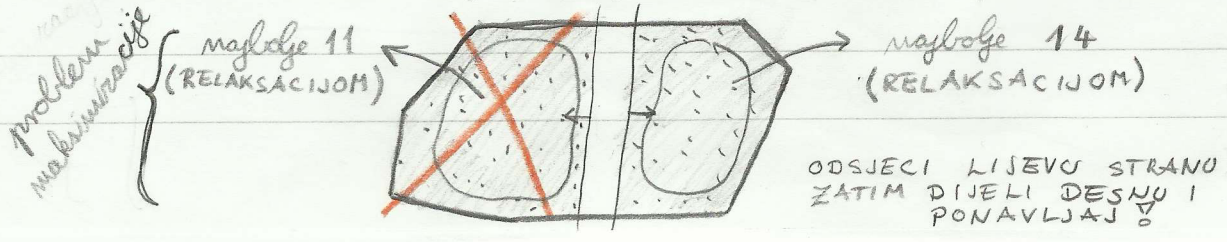
PREGLED EGZAKTNIH METODA

⇒ LINEARNO PROGRAMIRANJE: (LP)

- sva ograničenja moraju biti linearna
- rješenje je na nekoj granici konveksnog područja u kojem vrijede ograničenja
- rješava se:
  - a) simplex postupkom
    - ↳ skači od jedne do druge točke po vrhovima i traži najbolju
  - b) metoda unutarnje točke
    - ↳ kreni od točke koja je unutar ograničenja i idi do najbližeg ugla

⇒ CJELOBROJNO LINEARNO PROGRAMIRANJE: (ILP)

- teže od linearnog programiranja jer rješenja više ne moraju biti na granici
- prvi korak u rješavanju je da riješimo obični problem linearnog programiranja i time nađemo gornju granicu (u slučaju maksimizacije) - optimum ILP problema je sigurno lošij
- „BRANCH AND BOUND": (ISPIT - VIDI SLAJDOVE!)
  - ↳ razbijamo problem na više podproblema



(BITNO!)

↳ testovi eliminacije za granu - ako bilo koj vrijedi, razmatranjemo granu

1) LP - relaksacija nema rješenja

2) Gornja granica čvora je lošija od najboljeg pronađenog rješenja

3) Pronađeno je optimalno rješenje

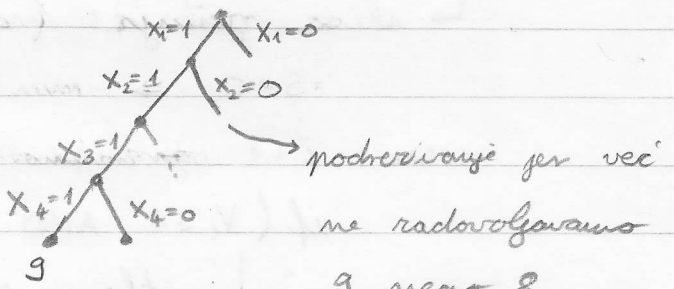
↳ NAPOMENA: relaksacija binarne varijable:

$$X \geq 0 \quad \wedge \quad X \leq 1$$

↳ primjer za branch-and-bound:

- max SAT problem u kojem je cilj zadovoljit najveći broj klauzula

PR: $f(x_1, x_2, x_3, x_4) = (x_1 \vee \bar{x}_2) \wedge \ldots$



DINAMIČKO PROGRAMIRANJE

⇒ ovo je generalizacija rekurzivnog razmišljanja
↳ veći problem svodimo na manje probleme iste vrste („OD-DNA-PREMA-GORE")

⇒ u danom stanju, naredne odluke ne ovise o prethodnima nego samo o trenutnom stanju

PR: Traženje minimalnog broja novčića za sumu $S = 11$. Vrijednosti novčića: $V_1 = 1$, $V_2 = 3$, $V_3 = 5$ ∇

↳ skica rješenja: (algoritam na slajdu 5)

$S = 0 \Rightarrow min[0] = 0$

KORAK: $i = 1$ → tražimo $min[1]$

if ($V_1 \leq 1$ AND $min[0] + 1 < \infty$)
then $min[1] = min[0] + 1 = 1$

KORAK: $i = 2$ → tražimo $min[2]$

$j = 1 \Rightarrow$ if ($V_1 \leq 2$ AND $min[2-1] + 1 < \infty$)
then $min[2] = 2$

KORAK: $i = 3$ → tražimo $min[3]$

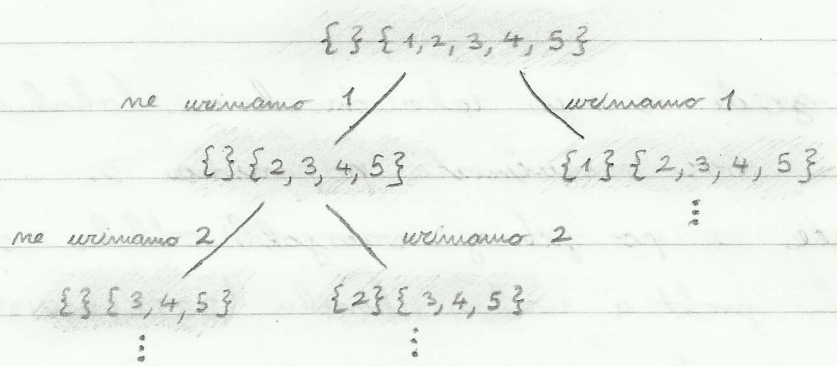
$j = 1 \Rightarrow$ if ($V_1 \leq 3$ AND $min[3-1] + 1 < \infty$)
then $min[3] = min[2] + 1 = 3$

$j = 2 \Rightarrow$ if ($V_2 \leq 3$ AND $min[3-3] + 1 < \infty$)
then $min[3] = 1$

⇒ IDEJA: pamti prethodnu vrijednost i koristi je za sljedeću

PR: Problem naprtnjače sa 5 predmeta!

↳ prikažimo problem grananjem



svak od čvorova je isto problem naprtnjače s manje predmeta i manje kapaciteta

PR: Računanje Fibonaccijevih brojeva uz memoizaciju!

```
Fib[i] = -1   za sve  i < m
Fib[0] = 0;  Fib[1] = 1;
int f(m) {
    if ( Fib[m] = -1 )
        Fib[m] = f(m-1) + f(m-2);
    return Fib[m];
}
```

PR: Floydov algoritam - dovraj najkraću udaljenost između svaka dva čvora!

↳ IDEJA: gledaj najprije direktne veze (D^0), pa dodaj međučvor ① između svaka dva (D^1)

$$D^0 = \begin{bmatrix} - & \infty & \infty & 5 \\ 4 & - & \infty & 9 \\ 2 & \infty & - & 7 \\ \infty & 3 & 2 & - \end{bmatrix}$$

POHLEPNI ALGORITMI

⇒ algoritmi koj grade rješenje tako da bira lokalno najpovoljniji izbor bez mogućnosti popravka

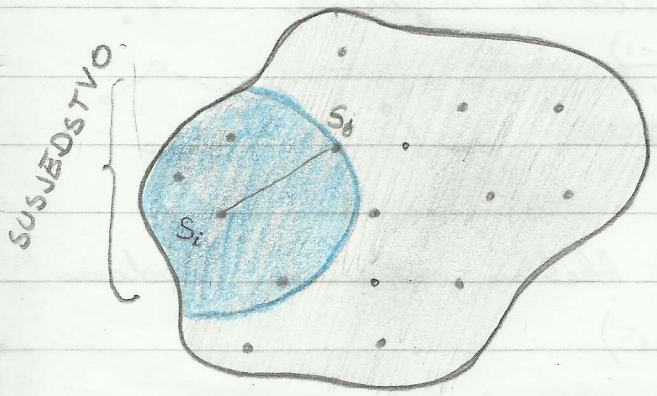
↳ kreće se po jednoj varijabli odluke i bira maksimalni profit u svakom koraku (DETERMINISTIČKA ODLUKA)

POBOLJŠAVAJUĆE METAHEURISTIKE

⇒ počinju od potencijalnog rješenja (ili skupa rješenja) i onda ga iterativno pokušavaju poboljšati

⇒ dva kontradiktorna kriterija:

1. DIVERZIFIKACIJA - istraživanje prostora rješenja
   - nasumično pretraživanje
2. INTENZIFIKACIJA - iskorištavanje najboljih pronađenih rješenja
   - lokalno pretraživanje



⇒ sami moramo definirati susjedstvo (operator koji vraća elemente iz susjedstva)

⇒ osnovne dvije podjele:

↳ stohastički - deterministički

↳ populacijski - pojedinačno rješenje

⇒ PRIKAZ RJEŠENJA:

- mora moći prikazati sva moguća rješenja
- iz svakog se rješenja mora moći doći direktno u svako drugo

LOKALNO PRETRAŽIVANJE

⇒ želimo neko rješenje iterativno poboljšati lokalnim transformacijama
↳ možemo (i vjerojatno hoćemo) zapeti u lokalnom optimumu

⇒ ključno je **definirati susjedstvo**
• susjedstvo je skup rješenja oko „s" gdje se svako to rješenje može dobiti primjenom elementarnih transformacija na „s"

⇒ vrlo često trebamo znati odrediti veličinu susjedstva 'ISPIT'
↳ kod zamjene parova: $\frac{n(n-1)}{2}$
↳ „two-opt-swap": $\frac{n(n-1)}{2} - n$

⇒ lokalno pretraživanje uvijek staje u lokalnom optimumu (nadamo se da je to ujedno i globalni)

⇒ UNAPRIJEĐENJA LOKALNOG PRETRAŽIVANJA:

OVI SU NAMA ZANIMLJIVI
↳ **multi-start**: više puta s različitim početnim pretraživanjem
↳ dozvoljavanje ne-poboljšavajućih rješenja
↳ promjene susjedstva tijekom pretraživanja
↳ promjene funkcije cilja tijekom pretraživanja

TABU SEARCH

⇒ poboljšanje lokalnog pretraživanja uz dopuštanje ne-poboljšavajućih rješenja

- lista koja zabranjuje posjećivanje rješenja kako bi spriječila „vrtnju u krug" (vraćanje na najbolje moguće)

↳ TABU LISTA

- svaki zapis u tabu listi vrijedi određeni broj iteracija
- u listi može biti zapisano koju promjenu nećemo raditi (npr. „nemoj flippati drugi bit") → ATRIBUTIVNA TABU LISTA

⇒ ideja je da izbjegava lokalne optimume

⇒ u osnovnoj izvedbi, algoritam je deterministički

⇒ KRITERIJ ASPIRACIJE:

↳ izračunamo i tabu susjede te ako je njihov fitness bolji od do sada najboljeg nađenog rješenja, onda zanemari **tabu pravilo** i uzmi to novo rješenje

⇒ 4 „dimenzije" tabu pretraživanja: VIDI NA SLAJDU?

a) kratkoročno pamćenje (~~svojstvima~~)

b) dugoročno pamćenje (~~učestalost~~) - moćemo zabranit česte poteze

c) kvaliteta

d) utjecaj

PR: Primjer knapsack problem tabu pretraživanjem.

$C = 10$ $c_1 = 5$ $v_1 = 5$ $c_4 = 4$ $v_4 = 3$

$c_2 = 4$ $v_2 = 6$ $c_5 = 3$ $v_5 = 5$

$c_3 = 2$ $v_3 = 3$

$\max \sum_{i=1}^{5} x_i v_i$ $x_i \in \{0, 1\}$ $\Rightarrow$ definicija problema

$\sum_{i=1}^{5} x_i v$ if $\sum x_i c_i \leq C$ else $-\infty$ } ovo je fitness funkcija

↳ inicijalizacija:

- greedy: uzmi najvrednije sve dok možeš

[1 1 0 0 0]

↳ susjedstvo:

- odredujemo „1-bit-flip" metodom

↳ provedba algoritma:

(1) curr = [1 1 0 0 0] f=11

najbolje → incum = [1 1 0 0 0] f=11

TL = [0 0 0 0 0]

TOGA UZIMAMO ZA SLJEDEĆU ITERACIJU UZ TABU LISTU:
TL = [3 0 0 0 0]

solution	$\sum x_i v_i$	$\sum x_i c_i$	f
1 1 0 0 0	11	9	11
0 1 0 0 0	6	4	6
1 0 0 0 0	5	5	5
1 1 1 0 0	14	11	$-\infty$
1 1 0 1 0	14	13	$-\infty$
1 1 0 0 1	16	12	$-\infty$

(2) curr = 01000

incum = 11000

TL = [30000]

solution	$\sum x_i v_i$	$\sum x_i c_i$	f
01000	6	4	6
11000	TABU		
00000	0	0	0
01100	9	6	9
01010	9	8	9
01001	11	7	11

TOGA UZIMAMO ZA SLJEDEĆU ITERACIJU UZ TABU LISTU: TL = [20003]

PR: Problem raspoređivanja posada po letovima!

Dano je 5 posada i 5 letova! Minimizirati ukupni trošak. Svaka posada je dodijeljena točno jednom letu.

↳ reprezentacija rješenja: [A B C D E]

↑ ↑ ↑ ↑ ↑

L_1 L_2 L_3 L_4 L_5

} posada A ima let L_1
posada B ima let L_2
⋮

↳ optimizacijski problem (funkcija cilja):

c_{ij} ⇒ dodjela posade „i" na let „j"

$$\min \sum_{i=1}^{5} c_{x_i i}$$

↳ početno rješenje:

• greedy: prvu posadu najeftinij, drugu najeftinij, ...

[C E A B D] ⇒ f = 30

↳ susjedstvo:

• radimo „2-swap" metodu: veličina $\binom{5}{2} = 10$

↳ iteracije algoritma:

① curr = incub = [C E A B D]

TL = [-, -, -]

} najbolje dobivamo za zamjenu (A, C)
↳ zamjenu (A, C) stavljamo u tabu listu:

TL = [(A,C), -, -]

još dvije zamjene ponuditi

⇒ dodatne memorije ra pomoć kod tabu pretraživanja:

B	C	D	E	
0	2	4	0	A
	1	3	10	B
		8	2	C
			0	D
				E

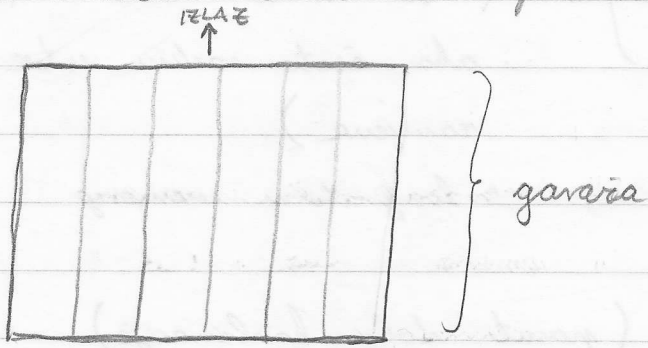
FREQENCY BASED MEMORY:
- bilježi koliko često radim neke rangove (može služit ra diversifikaciju ako često radim iste rangove)
- long-term memory

RECENCY BASED MEMORY:
- služi ra intenzifikaciju (pamt dobre kombinacije)
- medium-term memory

TABU LIST:
- služi ra prevenciju okl'avanja na roditelje
- short-term memory

PARKIRANJE VOZILA JAVNOG PRIJEVOZA U SPREMIŠTU
(projekt)

↳ imamo vozila u nekom parkiralištu koj se parkiraju u neke trake u parkiralištu



↳ za svako vozilo je definiran može li se parkirati u svaku pojedinu traku

- dodatno, dano je kada vozilo izlazi

↳ VIDI OSTALA OGRANIČENJA NA SLAJDU!

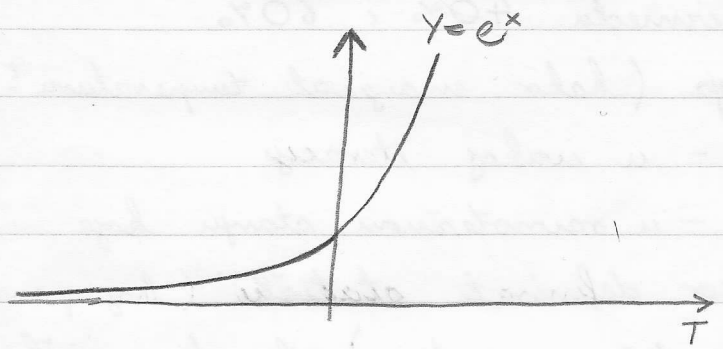
SIMULIRANO HLAĐENJE

⇒ iterativna metaheuristika razvijena na jednom rješenju

⇒ primjenjuje principe iz statističke mehanike
↳ simulira energetske promjene u sustavu tijekom hlađenja

$$P(\Delta E, T) = e^{-\frac{\Delta E}{k \cdot T}}$$

} vjerojatnost promjene energije na nekoj temperaturi (veća temperatura, veća vjerojatnost)



⇒ viša temperatura – veća stohastičnost
– prihvaćamo lošija rješenja (često)
– istražujemo prostor rješenja

⇒ OSNOVNA IDEJA:
- na početku, imaj visoku temperaturu
- ako susjed poboljšava, prihvati ga
- ako susjed ne poboljšava, prihvati ga s vjerojatnošću koja ovisi o temperaturi

⇒ stoga, vjerojatnost biranja ne poboljšavajućeg susjeda je:

MAKSIMIZACIJA

$$P(\Delta f, T) = e^{-\frac{\Delta f}{T}}$$

NAPOMENA: $\Delta f = 0$ ako je susjed bolji od trenutnog rješenja

⇒ druga strategija je: $P = \frac{1}{1 + \frac{\Delta f}{T}}$

⇒ odabir strategije ovisi o problemu

⇒ ODABIR HIPERPARAMETARA:

a) početna temperatura
- tipično između 40% i 60%

b) raspored hlađenja (kako smanjivat temperaturu?)
1) homogeni - u svakoj iteraciji
2) nehomogeni - u ravnotežnom stanju koje možemo definirat **statički** (broj poraka do ravnoteže se radaje) il **dinamički** (npr. kada nađemo poboljšavajućeg susjeda)

c) funkcija hlađenja (za koliko smanjit temperaturu?)
- tipično geometrijski

d) kriterij zaustavljanja
- postizanje konačne temperature
- broj iteracija bez poboljšanja

⇒ NAPREDNIJE TEHNIKE:

- ponovno zagrijavanje, ograničenje susjedstva, sistematično (redoslijedno) pretraživanje

ISPIT V°

ZAD: Raspoređivanje posada A, B, C, D, E na letove L_1 - L_5.

Zadani su troškovi raspoređivanja posada po letovima:

	L_1	L_2	L_3	L_4	L_5
A	10	8	6	5	12
B	12	7	3	5	9
C	4	8	5	8	10
D	12	10	8	10	8
E	10	8	4	5	5

Treba dodijeliti svakom letu točno jednu posadu!

[A B C D E] ⇒ mjesto u vektoru označava let
(reprezentacija je vektor od 5 članova)

c_{ix_i} → cijena ako je na letu „i" posada „x_i"

↳ funkcija cilja: $\min \sum_{i=1}^{5} c_{ix_i}$

↳ da su bila radana ograničenja, onda kada nije svako radovoljeno pišemo $c_{ix_i} = \infty$

↳ početno rješenje: [B A E C D] (nasumično)

↳ susjed: zamjena dva člana

↳ temperatura: $T = \lambda T$, $\lambda = 0.9$

• homogeno (jednoliko) hlađenje

OKRENI →

$f(\text{početno}) = 40$

T	sosed x'	zamjena indeksa	f(x')	Δf	P	novi x	f(x
100	EABCD	1, 3	37	-3	1	EABCD	37
90	DABCE	1, 5	36	-1	1	DABCE	36
81	DACBE	3, 4	35	-1	1	DACBE	35
72.9	DECBA	2, 5	42	7	0.908	DECBA	35
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

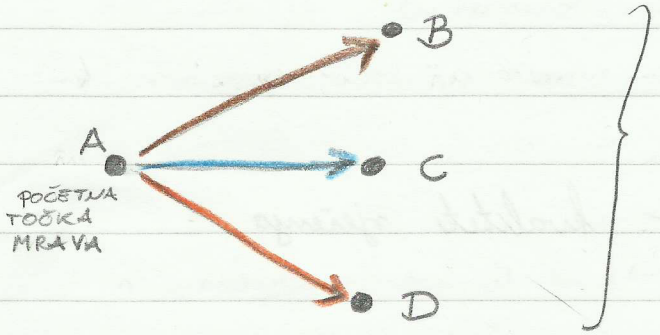
↳ dalje je lagano ?

· samo pazi da ne mijenjaš $f(x_{best})$ zato s
prihvatio bolje rješenje

OPTIMIZACIJA KOLONIJOM MRAVA

⇒ mravi idu s većom vjerojatnosti tamo gdje osjete veću koncentraciju feromona

- oni sami ispuštaju feromone



odluka o putu ovisi o:
- duljini puta
- količini feromona

⇒ feromonski trag pojačavamo proporcionalno dobroti rješenja koje smo našli na tom putu

⇒ mravi komuniciraju preko okoline (nema centraliziranog znanja nego svi doprinose globalnom znanju)

Pr: Za TSP, T_{ij} je količina feromona između gradova „i" i „j"!
↳ ovo je jedna tablica
↳ druga tablica je tablica udaljenosti

⇒ feromonski tragovi s vremenom isparavaju - ako ih dugo mrav nije prošao, oni će nestati!

⇒ za TSP vjerojatnost prelaska iz grada „i" u grad „j":

$$p_{ij} = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{k \in S} \tau_{ik}^{\alpha} \cdot \eta_{ik}^{\beta}} \quad \forall j \in S$$

$\alpha \Rightarrow$ utjecaj feromona

$\beta \Rightarrow$ utjecaj dobrote rješenja

⇒ koeficijent pojačavanja Δ:

- obično ovisi o kvaliteti rješenja

HIBRIDNI ALGORITMI

⇒ u praksi, rijetko se koriste algoritmi zasebno
↳ najčešće se heuristike kombiniraju

⇒ PROBLEM 1: multicast - routing
↳ minimalno Steinerovo stablo
↳ u reprezentaciji rješenja su neravniši čvorovi
- ako je čvor uključen u stablo, u vektoru se na tom mjestu nalazi 1 (inače 0)
↳ na slajdovima 9 i 10 se pokazuju načini prilagođavanja problemu - izbacivanje čvorova i pojednostavljenje veza (DRAMATIČNO SMANJENJE PROSTORA RJEŠENJA)
↳ cilj je samo pronaći koji neravniši čvorovi tvore rješenje - kada to znamo, pokrenemo deterministički algoritam koj nađe minimalno razapinjajuće stablo (pohlepni algoritam)

⇒ GRASP: „Greedy Randomized Adaptive Search Procedure"
↳ multi - start heuristika
↳ <u>faza izgradnje početnog rješenja:</u>
• koristi nasumični pohlepni algoritam (svaki put daje drugo rješenje) - bira vrijednost varijable odluke tako da izgradi listu kandidata i ratim bira jednu vrijednost s te liste (ponavlja sve dok ne izgradi cjelokupno rješenje)

- veličina liste kandidata je hiperparametar (vidi varijante liste kandidata na slajdu 23)
  - možemo se ograničiti po broju elemenata u listi ili se možemo ograničiti po vrijednosti funkcije dobrote

↳ <u>fara lokalnog pretraživanja:</u>

- standardno lokalno pretraživanje

⇒ PROBLEM 2: routing wavelength assignment

↳ svod se na problem bojanja stabla (slajd 42)

EVOLUCIJSKI ALGORITMI

⇒ populacijske metaheuristike:

1) temeljene na evoluciji (npr. genetski algoritmi)
2) „blackboard" based
   ↳ sva rješenja pišu po istoj ploči (memoriji) slično kako i mravi obnavljaju feromonske tragove

⇒ IDEJA EVOLUCIJSKIH ALGORITAMA:

↳ koriste se operatori odabira, križanja i mutacije kako bi iz trenutne populacije dobili sljedeću

↳ različiti pristupi:

- genetički algoritmi
- evolucijske strategije
- evolucijsko programiranje
- genetičko programiranje

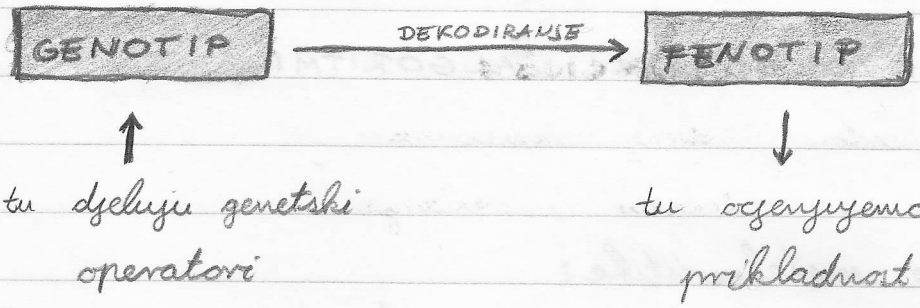
} GLAVNE IDEJE SU ZAJEDNIČKE

⇒ specifične odluke vezane za evolucijske algoritme u odnosu na sve heurističke metode:

- odabir (selekcija) roditelja
- strategija reprodukcije

⇒ GENOTIP (kromosom) ⇒ kodni prikaz rješenja

⇒ FENOTIP ⇒ dekodirana vrijednost rješenja



⇒ POČETNA POPULACIJA:

- ima nekoliko strategija (vidi slajd 12)
- želimo da bude dovoljno raznolika (to uvelike utječe na performanse evolucijskog algoritma)

↓

vidi kvalitativnu ocjenu na slajdu 13

⇒ ODABIR (SELEKCIJA) RODITELJA:

- kotač ra rulet („roulette wheel")
  - ↳ problem skale, pa često skaliramo:

$$P(i) = \frac{2-S}{n} + \frac{2 \cdot r(i) \cdot (S-1)}{n(n-1)}$$

$n \Rightarrow$ veličina populacije

$r(i) \Rightarrow$ rang rješenja

$S \Rightarrow$ selekcijski pritisak $\quad S \in [1, 2]$

Pr:

R_i	A	B	C
f	1	5	4

standardni rulet →

$P(i)$	0.1	0.5	0.4

↓ rang

$r(i)$	1	3	2

S=1.5 →

$K(i)$	0.17	0.5	0.33

S=2 →

$K(i)$	0	0.67	0.33

⇒ STRATEGIJA REPRODUKCIJE:

a) jedan roditelj - mutacija

b) dva roditelja - križanje

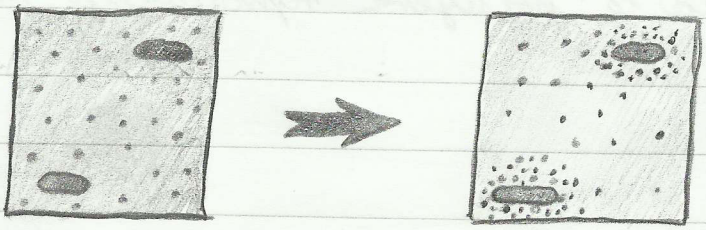
⇓

vrlo često se koriste obje strategije istovremeno

↳ vidi razlike te primjere reprodukcije na slajdovima

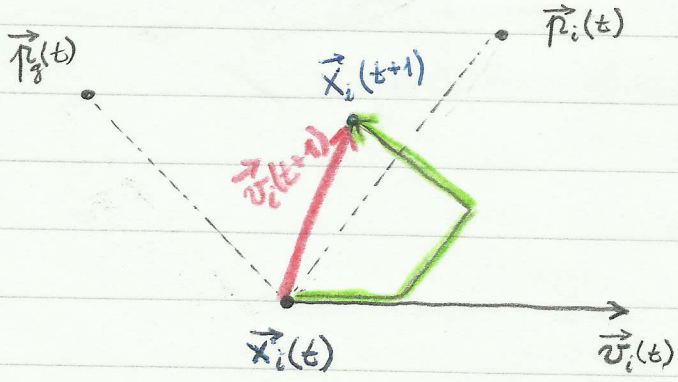
ALGORITAM ROJA ČESTICA

⇒ kolektivno ponašanje roja je bolje od pojedinačnog
↳ rješenja se ovdje ne bore za opstanak kao kod genetskog algoritma već međusobno surađuju



⇒ čestica se kreće na temelju tri stvari:

a) svojeg vektora brzine (inercija) ⇒ $\vec{v}_i(t)$
b) svojeg najboljeg rješenja ⇒ $\vec{p}_i(t)$
c) najboljeg rješenja cijelog roja ⇒ $\vec{p}_g(t)$



pretpostavimo da je $\vec{v}_i(t+1)$ linearna kombinacija vektora $\vec{v}_i(t)$, $\vec{p}_i(t)$ i $\vec{p}_g(t)$

↓

VIDI FORMULU NA SLAJDU!

⇒ ako više slijediš $\vec{v}_i(t)$, onda si hrabar jer dalje istražuješ
⇒ ako više slijediš $\vec{p}_i(t)$, onda si tvrdoglav jer ideš u svoje najbolje rješenje
⇒ ako više slijediš $\vec{p}_g(t)$, onda si kukavica jer ideš u društveno najbolje rješenje

⇒ napomena: globalno najbolje rješenje može biti definirano samo unutar nekog susjedstva rješenje, pa dobijemo lokalni utjecaj

⇒ ovaj algoritam se većinom primjenjuje u kontinuiranoj optimizaciji, ali ga je moguće i primjeniti na diskretne probleme

↳ vidi primjenu na TSP na slajdovima, ali nije jako bitno ?

⇒ ovdje nemamo križanje i mutacije već imamo jednu populaciju u kojoj svaku jedinku micemo kroz prostor rješenja

$$v_i^{t+1} = v_i^t + \varphi_1 C_1 \cdot \left(p_i - x_i^t \right) + \varphi_2 C_2 \cdot \left(p_g - x_i^t \right)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

ZAD: Problem naprtnjače s 5 vrsti predmeta. Imamo po dva predmeta svake vrste. Predmet 2 i 3 ne smiju biti zajedno u naprtnjači. Rješiti uz ACO!

$$[x_1, \dots, x_5] \qquad x_i \in \{0, 1, 2\}$$

↳ broj predmeta vrste „i", $i = 1, \dots, 5$

$$\max \sum_{i=1}^{5} x_i v_i \quad \text{tako da} \quad \sum_{i=1}^{5} x_i c_i \leq 15$$

↳ feromonski tragovi: $\tau_i = [0.5 \dots 0.5]$

↳ „eta" vrijednost: $\eta_i = \left[\frac{v_1}{c_1} \quad \frac{v_2}{c_2} \dots \frac{v_5}{c_5}\right]$

↳ ponašanje mrava (gradimo vrijednost):

$$p_i = \frac{\tau_i^{\alpha} \eta_i^{\beta}}{\sum_{i=1}^{5} \tau_i^{\alpha} \eta_i^{\beta}}$$

ZAD: Problem kapacitativnog usmjeravanja vozila uz jednaku potražnju svih 5 kupaca i danom matricom udaljenosti. Riješi uz ACO!

[A B C D (S) E]

skladište jer imamo 5 kupaca a kapacitet je 40 ⇒ sigurno idemo jednom u skladište!

↳ „eta" vrijednosti: isto kao i za TSP: $1/d_{ij}$

↳ feromonski tragovi: svi na 0,5 (na dijagonali nula)

↳ obnavljanje feromona: isto kao za TSP na slajdu

ZAD: Imamo graf i treba naći vrh koj ima minimalnu ukupnu težinu (vrh dijeli graf na dva dijela).

$[0\ 1\ 1\ 0\ 1\ 1]$ } varijanta 1: grana je u vrhu ili ne

varijanta 2: čvor je u jednom ili drugom grafu (dobro samo za podjelu na dva dijela)

↳ idemo s varijantom 1:

$$\min \sum_{i=1}^{6} x_i e_i$$