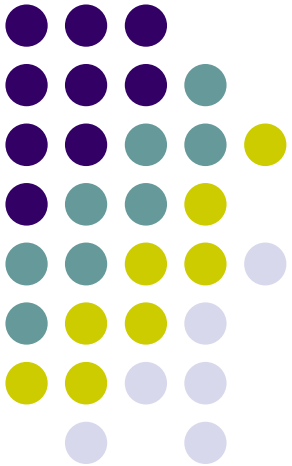
Evolučné algoritmy

Optimalizačný problém



Spracované podľa knihy

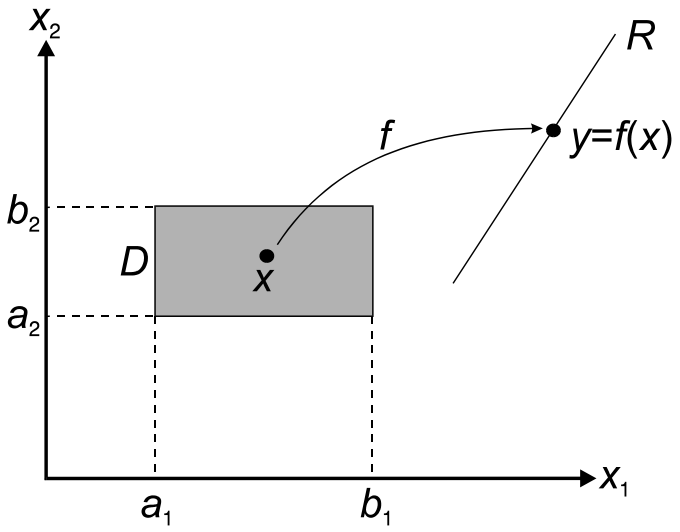
V. Kvasničku a J. Pospíchala

Optimalizačný problém

- Nech funkcia $f : D \to R$

zobrazuje n-rozmernú kocku D (karteziánsky súčin uzavretých intervalov $[a_i,b_i]$) na reálne čísla $y \in R$,

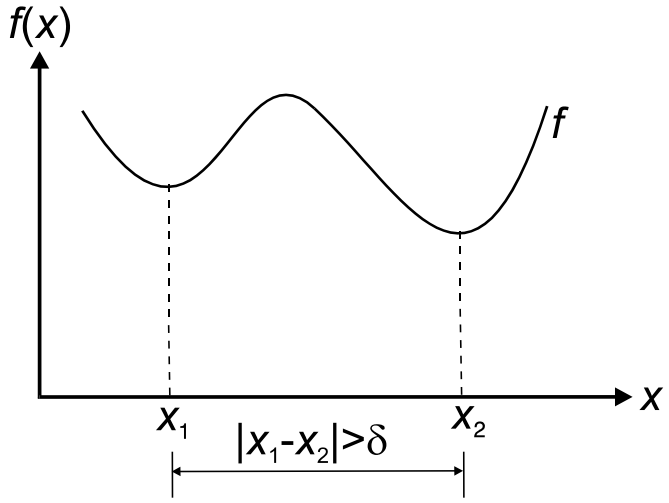
$$D = \prod_{i=1}^{n} [a_i, b_i] = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_n, b_n]$$



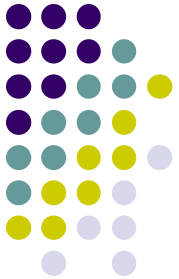
Táto funkcia je ohraničená dvomi podmienkami:

(1) Existuje taký algoritmus, ktorý funkciu *f* "vypočíta dostatočne rýchlo" s požadovanou presnosťou pre každé $\boldsymbol{x} \in D$ (hovoríme, že funkcia *f* je *dobre vypočítateľná*).

(2) Pre každú dvojicu lokálnych miním $\boldsymbol{x}1, \boldsymbol{x}2 \in D$ je vzdialenosť $|\boldsymbol{x}1-\boldsymbol{x}2|$ väčšia ako dané kladné číslo $\delta>0$, $|\boldsymbol{x}1-\boldsymbol{x}2|>\delta$.

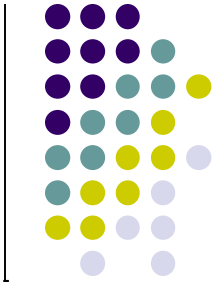


Optimalizačný problém



- Podmienka zhora ohraničuje počet lokálnych miním funkcie *f*, ktoré sa vyskytujú na kocke *D.*
- Nie je možné, aby sa v ľubovoľnom okolí minima funkcie vyskytovalo iné minimum, pre určité malé okolie minima funkcie by vyššie uvedená podmienka $|\boldsymbol{x}1-\boldsymbol{x}2|>\delta$ prestala platiť.
- Podmienka automaticky vylučuje z triedy prípustných funkcií tie funkcie, ktoré sú "fraktálového" typu, t. j. v každom okolí nejakého minima sa nachádza aspoň jedno iné minimum.

Optimalizačný problém



Globálne minimum funkcie *f* na kocke *D* je určené vzťahom

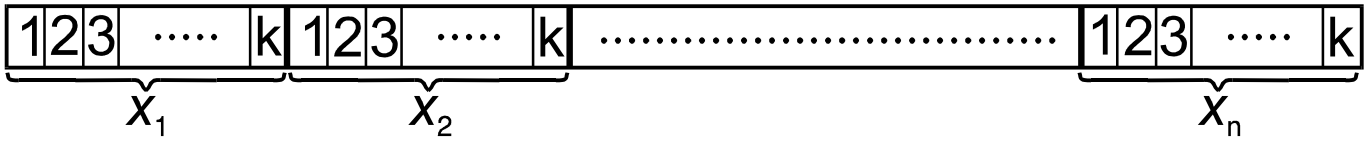
$$x_{opt} = \arg\min_{\boldsymbol{x} \in D} f(\boldsymbol{x})$$

Nájdenie globálneho minima použitím klasických optimalizačných metód (gradientových a negradientových) patrí medzi ťažko riešiteľné problémy pre funkcie, ktoré nie sú ohraničené ďalšími podmienkami (napr. že *f* (***x***) je konvexná funkcia v oblasti *D*).

Z týchto dôvodov sa v súčasnosti pri riešení problému často používajú **tzv. *evolučné optimalizačné algoritmy*,** ktoré poskytujú riešenia blízke globálnemu alebo s ním totožné.

Transformácia spojitého optimalizačného problému na binárny optimalizačný problém

- Predpokladajme, že každá z *n* premenných vektora $\boldsymbol{x} \in D$ je vyjadrená v binárnej reprezentácii bitovým vektorom dĺžky *k*. To znamená, že vektor $\boldsymbol{x} \in D$ je v binárnej reprezentácii vyjadrený bitovým vektorom dĺžky *kn.*



Nech funkcia *f* vyhovuje druhej podmienke pre danú kladnú konštantu δ. Budeme predpokladať, že dĺžka binárnej reprezentácie *k* je zvolená tak, že platí

$$\delta >> \frac{(b_i - a_i)}{(2^k - 1)} \text{ (pre } i = 1,2,...,n)$$

Táto podmienka vyžaduje, aby minimálna vzdialenosť medzi dvoma minimami funkcie *f* (***x***) nad oblasťou *D* bola omnoho väčšia ako "presnosť" binárnej reprezentácie pre každú premennú vektora $\boldsymbol{x} \in D$.

Optimalizačný problém

- Prechod od binárneho vektora $\boldsymbol{\alpha}=(\alpha_1,\alpha_2, ..., \alpha_{kn})\in\{0,1\}^{kn}$ k spojitému vektoru $\boldsymbol{x}=(x_1,x_2,..., x_n)\in D$ sa môže formálne chápať ako transformácia

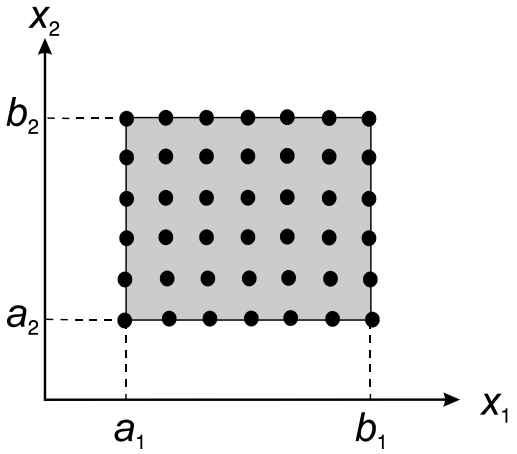
$$\Gamma : \{0,1\}^{kn} \rightarrow D$$

$$\boldsymbol{x} = \Gamma(\boldsymbol{\alpha})$$

ktorá zobrazuje množinu binárnych vektorov dĺžky *kn* na body - *n*-tice reálnych čísel *z* kocky *D.*

Ináč povedané, konečná množina (2^{kn}) binárnych vektorov dĺžky *kn* je reprezentovaná pomocou zobrazenia Γ bodmi, ktoré môžu byť v oblasti *D* usporiadané do ortogonálnej mriežky

Optimalizačný problém



Minimalizačný problém pri použití binárnej reprezentácie *n*-rozmerných vektorov ***x*** sa teda realizuje na konečnej množine diskrétnych bodov.

Označme binárny vektor dĺžky *kn*, ktorý bol získaný riešením daného optimalizačného problému, pričom funkcia *f* tohto problému je totožná s funkciou *f* v optimalizačnom probléme

$$\tilde{\boldsymbol{\alpha}}_{opt} = \arg \min_{\boldsymbol{\alpha} \in \{0,1\}^{kn}} f\left(\Gamma\left(\boldsymbol{\alpha}\right)\right)$$

$$\boldsymbol{x}_{opt} \approx \tilde{\boldsymbol{x}}_{opt} = \Gamma\left(\tilde{\boldsymbol{\alpha}}_{opt}\right)$$

Binárna reprezentácia reálnych čísel

- Nech binárny vektor α dĺžky k je interpretovaný ako celé číslo

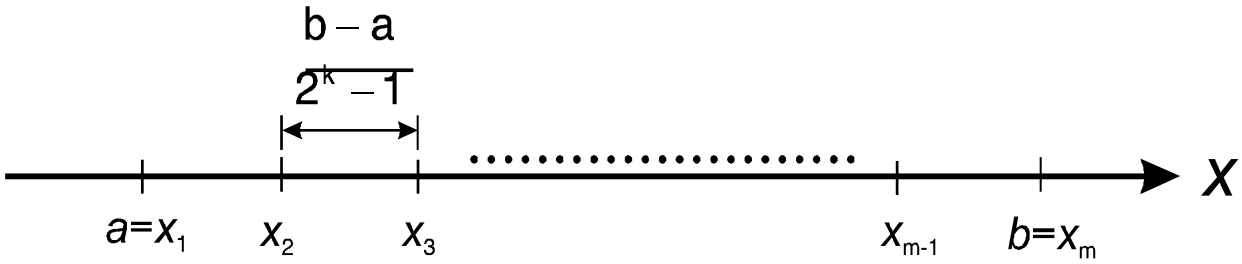
$$int(\alpha) = \sum_{i=1}^{k} \alpha_i 2^{k-i} = \alpha_1 2^{k-1} + \alpha_2 2^{k-2} + \ldots + \alpha_{k-1} 2 + \alpha_k$$

- K tomuto celému číslu jednoduchým spôsobom priradíme reálne číslo, ktoré môže byť chápané ako aproximácia reálneho čísla $x \in [a, b]$

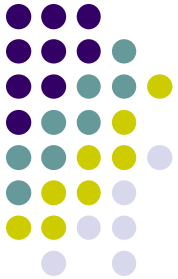
$$x \approx real(\alpha) = a + \frac{b-a}{2^k - 1} int(\alpha)$$

Binárna reprezentácia reálnych čísel

- Táto konštrukcia racionálneho čísla $a \le real(\alpha) \le b$ z binárneho reťazca α dĺžky k sa formálne interpretuje ako "transformácia" binárnej reprezentácie na "reálnu" reprezentáciu,
- zostrojené racionálne číslo real(α) aproximuje požadované reálne číslo x s presnosťou $(b-a)/(2k-1)$. Interval $[a, b]$ obsahuje $m=2k$ bodov $x_1=a$, $x_2=a+(b-a)/(2k-1)$, ..., $x_i = a+(i-1)(b--a)/(2k-1)$, ..., $x_n=b$,



Binárna reprezentácia reálnych čísel



- Inverzná transformácia má tvar

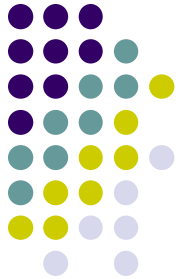
$$\mathrm{int}(\boldsymbol{\alpha}) = \left\lceil \frac{x-a}{b-a}\left(2^{k}-1\right) \right\rceil$$

- Prechod od binárneho vektora $\boldsymbol{\alpha}=(\alpha 1,\alpha 2, ..., \alpha kn)\in\{0,1\}$ *kn* k spojitému vektoru $\boldsymbol{x}=(x1,x2,..., xn)\in D$ sa môže formálne chápať ako transformácia

$$\Gamma : \{0,1\}^{kn} \to D \qquad \boldsymbol{x} = \Gamma(\boldsymbol{\alpha})$$

- ktorá zobrazuje množinu binárnych vektorov dĺžky *kn* na body - *n*-tice reálnych čísel *z* kocky *D.* Ináč povedané, konečná množina (2*kn*) binárnych vektorov dĺžky *kn* je reprezentovaná pomocou zobrazenia Γ bodmi, ktoré môžu byť v oblasti *D* usporiadané do ortogonálnej mriežky.

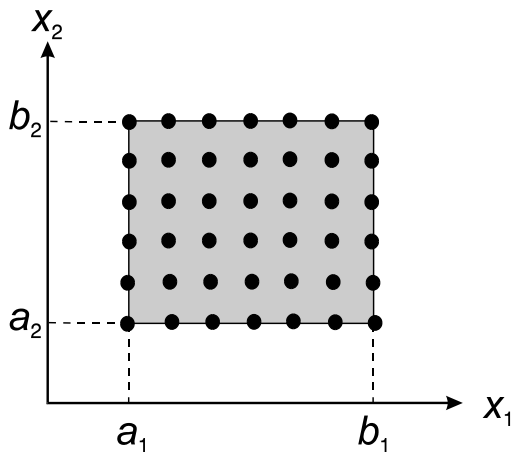
Základné stochastické optimalizačné algoritmy: *slepý algoritmus* a *horolezecký algoritmus*



- Dva základné typy stochastických optimalizačných algoritmov, ktoré aj keď neobsahujú evolučné rysy, budú slúžiť ako základ pre formuláciu evolučných optimalizačných algoritmov.
- *Slepý algoritmus* - základný stochastický algoritmus, ktorý opakovane generuje náhodne riešenie z oblasti *D* a zapamätá si ho len vtedy, ak bolo lepšie ako to riešenie, ktoré už bolo zaznamenané v predchádzajúcej histórii algoritmu. Z dôvodov kompatibility tohto algoritmu s evolučnými algoritmami uvedieme jeho implementáciu pre binárnu reprezentáciu vektorov – riešení.

Slepý algoritmus

procedure Blind_Algorithm(input: t_{max},k,n; output: α_{fin},f_{fin});
begin f_{fin}:=∞; t:=0;
    while t<t_{max} **do**
    begin t:=t+1;
        α:=náhodne generovaný binárny
            vektor dĺžky kn;
        if $f(\Gamma(\alpha))<f_{fin}$ **then**
        begin α_{fin}:=α; f_{fin}:=$f(\Gamma(\alpha))$ **end**;
    end;
end;



Slepý algoritmus

- Dá sa dokázať, že tento jednoduchý stochastický optimalizačný algoritmus poskytuje korektné globálne minimum optimalizačného problému realizovaného nad ortogonálnou mriežkou bodov z oblasti *D* za predpokladu, že parameter procedúry t_{max} asymptoticky rastie do nekonečna

$$\lim_{t_{\max}\to\infty} P\left(t_{\max} \mid \alpha_{fin} = \alpha_{opt}\right) = 1$$

kde $P(t_{max}|\alpha_{fin}=\alpha_{opt})$ je pravdepodobnosť toho, že slepý algoritmus po t_{max} iteračných krokoch poskytne výstupné riešenie, ktoré je totožné s presným riešením (globálne minimum).

Horolezecký algoritmus

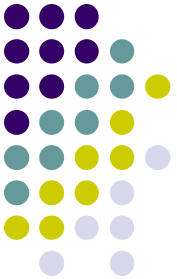
- ktorý iteračne hľadá najlepšie lokálne riešenie v určitom okolí a toto riešenie sa v ďalšom kroku použije ako "stred" novej oblasti
- operácia *mutácie* stochasticky transformuje binárny vektor α na nový binárny vektor α' , pričom stochastičnosť tohto procesu je určená pravdepodobnosťou P_{mut}

$$\alpha' = O_{mut}(\alpha)$$

- kde α a α' sú dva binárne vektory rovnakej dĺžky *kn,* kde jednotlivé komponenty α' sú určené takto

$$\alpha_i' = \begin{cases} 1-\alpha_i \ (\text{pre } random < P_{mut}) \\ \\ \alpha_i \ (\text{ostatné prípady}) \end{cases}$$

Horolezecký algoritmus



- Okolie $U(\alpha)$ binárneho vektora α sa zostrojí pomocou vektorov

$$U(\alpha) = \{\alpha' = O_{mut}(\alpha)\}$$

pričom budeme predpokladať, že kardinalita (počet elementov) sa rovná predpísanej hodnote $|U(\alpha)| = c_0$ kde c_0 je dané kladné celé číslo.

Poznamenajme, že v dôsledku stochastičnosti aplikácie operácie mutácie na daný binárny vektor α má zloženie okolia $U(\alpha)$ tiež stochastický charakter.

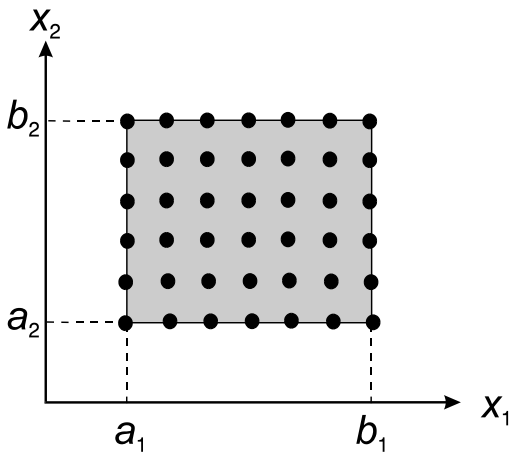
To, či nejaký vektor α' patrí alebo nepatrí do okolia $U(\alpha)$, je určené len pravdepodobnostne, a nie deterministicky.

Najlepšie riešenie v okolí $U(\alpha)$ je určené takto $\alpha^* = \arg \min_{\alpha' \in U(\alpha)} f(\Gamma(\alpha'))$

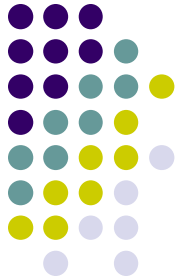
V horolezeckom algoritme sa takto získané riešenie $\alpha*$ použije ako "stred" v ďalšom iteračnom kroku algoritmu

Horolezecký algoritmus

procedure Hill_Climbing(input:t_{max},c_0,P_{mut}; output: f_{fin},α_{fin});
begin α:= náhodne generovaný binárny vektor dĺžky kn;
$f_{fin}:=\infty$; t:=0;
while $t<t_{max}$ **do**
begin t:=t+1; $\alpha^* = \arg\min_{\alpha' \in U(\alpha)} (\Gamma(\alpha))$;
if $f(\Gamma(\alpha^*)) < f_{fin}$ **then**
begin
$\alpha_{fin}:=\alpha^*$;
$f_{fin} := f(\Gamma(\alpha^*))$
end;
$\alpha:=\alpha^*$;
end;
end;



Metóda zakázaného hľadania (tabu search)

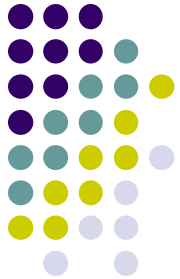


- Glover - koncom 80-tých rokov, ako určité zovšeobecnenie horolezeckého algoritmu (HA) na riešenie zložitých optimalizačných úloh predovšetkým z operačného výskumu.

- Základnou nevýhodou algoritmu HA je, že sa po určitom počte iteračných krokov vracia k lokálnemu optimálnemu riešeniu, ktoré sa vyskytlo už v jeho predchádzajúcom priebehu (problém zacyklenia).

- Glover navrhol jednoduchú heuristiku, ako tento problém odstrániť. Do horolezeckého algoritmu je zavedená tzv. *krátkodobá pamäť,* ktorá si pre určitý krátky interval predchádzajúcej histórie algoritmu pamätá inverzné transformácie k tým transformáciám riešení, ktoré poskytovali lokálne optimálne riešenia. Tieto inverzné transformácie **sú zakázané (tabu)** pri tvorbe nového okolia pre dané aktuálne riešenie.

Metóda zakázaného hľadania (tabu search)



Definujme si množinu prípustných transformácií $S = \{t_1, t_2, \ldots, t_p\}$

Transformácia $t \in S$ zobrazuje binárny vektor $\alpha \in \{0,1\}^{kn}$ na iný binárny vektor $\alpha' \in \{0,1\}^{kn}$

$$t : \{0,1\}^{kn} \rightarrow \{0,1\}^{kn}$$

pre $\forall t \in S$. Jednoduchá realizácia týchto transformácií je $t_i(\ldots\alpha_i\ldots) = (\ldots 1-\alpha_i \ldots)$

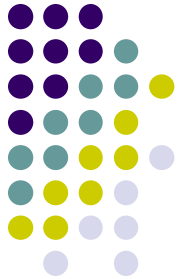
pre i=1,2,...,$p=kn$. Operátor t_i zmení v i-tej polohe binárnu hodnotu na jej komplement.

Vo všeobecnosti, transformácie z S sú ohraničené nasledujúcimi podmienkami:

- Nech $t_1, t_2 \in S$ sú dve rôzne transformácie, $t_1 \neq t_2$, potom pre $\forall \alpha \in \{0,1\}^{kn}$ platí $t_1\alpha \neq t_2\alpha$.
- Pre každú transformáciu $t \in S$ existuje taká transformácia $t^{-1} \in S$, ktorá je inverzná k t, $tt^{-1}\alpha = t^{-1}t\alpha = \alpha$, pre $\forall \alpha \in \{0,1\}^{kn}$.
- Pre každú dvojicu $\alpha 1, \alpha 2 \in \{0,1\}^{kn}$ rôznych binárnych vektorov, $\alpha 1 \neq \alpha 2$, existuje taká postupnosť $t_{i_1}, t_{i_2}, \ldots, t_{i_n} \in S$ transformácií , že "východzí" vektor $\alpha 1$ je postupne pretransformovaný na "konečný" vektor $\alpha 2$

$$\alpha_1 = \beta'_1 \xrightarrow{t_{i_1}} \beta'_2 \xrightarrow{t_{i_2}} \ldots \xrightarrow{t_{i_n}} \alpha_2 = \beta'_n$$

Metóda zakázaného hľadania (tabu search)

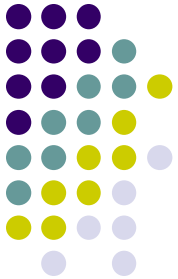


- Okolie U(α) obsahuje obrazy α vytvorené transformáciami $t \in S$

$$U(\alpha) = \{t\alpha; \forall t \in S\}$$

- Pôvodný horolezecký algoritmus bude teraz modifikovaný tak, že namiesto okolia generovaného náhodne pomocou stochastického operátora mutácie, použijeme deterministicky nové definované okolie, generované pomocou prípustných transformácií z množiny S.
- Hlavná myšlienka tejto heuristiky je ***zakázaný zoznam T* (tabu list),** majúci vlastnosť krátkodobej pamäti, ktorý dočasne obsahuje inverzné transformácie k použitým transformáciám v predchádzajúcich iteráciách. Zakázaný zoznam transformácií $T \subseteq S$, maximálnej kardinality s, $0 \leq |T| \leq s$, je zostrojený a systematicky obnovovaný v priebehu celého algoritmu

Metóda zakázaného hľadania (tabu search)

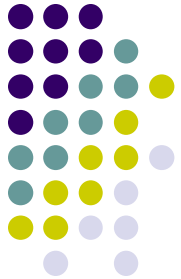


- Ak transformácia *t* patrí pre danú iteráciu do zakázaného zoznamu, $t \in T$, potom sa nemôže používať v lokálnej minimalizácii v rámci okolia aktuálneho riešenia α.

- Pri inicializácii algoritmu je zakázaný zoznam prázdny, po každej iterácii sa do zakázaného zoznamu dodá transformácia, ktorá poskytla lokálne optimálne riešenie zostrojené z riešenia z predchádzajúcej iterácie.

- Po *s* iteráciách zakázaný zoznam už obsahuje ***s*** transformácií, zo zakázaného zoznamu sa vylúči transformácia, ktorá tam bola dodaná pred *s* iteráciami.

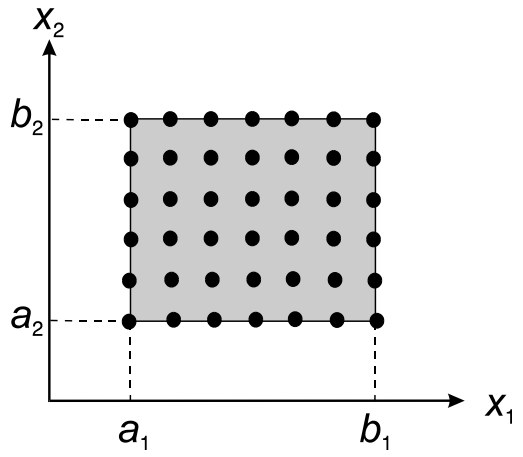
Metóda zakázaného hľadania (tabu search)



- Uvedieme pseudopascalovskú procedúru pre metódu zakázaného hľadania.
- Vstupnými parametrami sú $time_{max}$ a s, ktoré určujú maximálny počet iteračných krokov resp. veľkosť zakázaného zoznamu T.
- Procedúra obsahuje dva cykly: vonkajší while-cyklus realizuje iteračné kroky zakázaného hľadania, zatiaľ čo
- vnútorný for-cyklus slúži pre konštrukciu okolia $U(\alpha)$. Poznamenajme, že toto okolie nie je explicitne zostrojené, generujú sa len jeho elementy a tie sa hneď testujú, či ich funkčná hodnota nie je menšia ako lokálne najlepšia funkčná hodnota $f_{fin\text{-}loc}$.
- Vonkajší cyklus je ukončený operáciou obnovy zakázaného zoznamu.

Metóda zakázaného hľadania (tabu search)

procedure Tabu_Search(input:time$_{max}$, s, S, k, n ; output: f_{fin},α_{fin});
begin α:=náhodne generovaný binárny vektor dĺžky kn;
f_{fin}:=∞; time:=0; T:=$\varnothing$;
while time<time$_{max}$ **do**
begin time:=time+1; $f_{fin\text{-}loc}$:=∞;
for t$\in$S **do** {S je množina povolených transformácií }
begin α':=tα;
if (t$\notin$T and f(Γ(α')<$f_{fin\text{-}loc}$) or f(Γ(α')<f_{fin} **then**
begin α*:=α';
t*:=t;
$f_{fin\text{-}loc}$:=f(Γ(α'))
end;
end;
if $f_{fin\text{-}loc}$<f_{fin} **then** **begin** f_{fin}:=$f_{fin\text{-}loc}$; α_{fin}:=α* **end**;
α:=α*;
if |T|<s **then** T:=T$\cup${t^{*-1}} **else** T:=(T$\cup${t^{*-1}})\{ t };
end; { z T je odstránená najstaršia transformácia }
end;



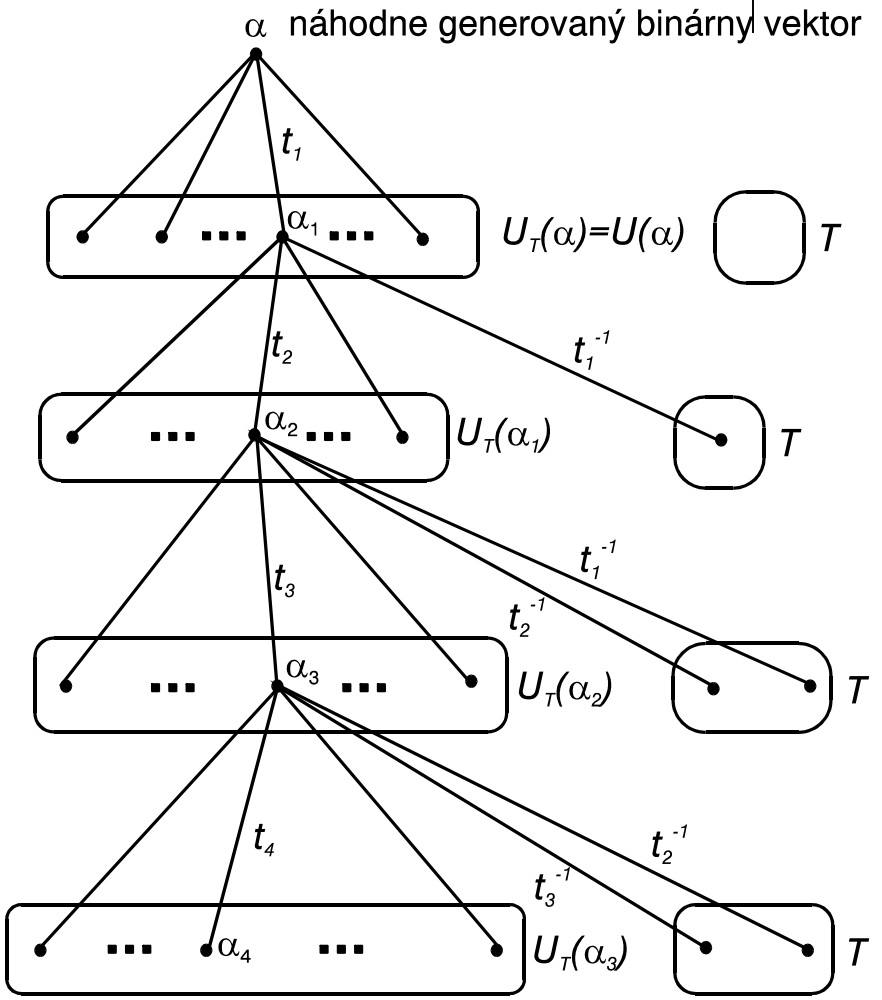
Metóda zakázaného hľadania (tabu search)

Veľkosť zakázaného zoznamu s=2.

Metóda je inicializovaná náhodne generovaným vektorom α.

Pomocou transformácií t z množiny prípustných transformácií S je zostrojené prvé okolie $U(\alpha)$.

Najlepšie riešenie z tohto okolia je označené $\alpha 1$, v nasledujúcom iteračnom kroku je toto riešenie použité ako "stred" pre konštrukciu nového okolia. Inverzná transformácia je zavedená do zakázaného zoznamu T, ktorý bol pôvodne prázdny. Menšie bloky v prav2424om stĺpci odpovedajú zakázaným zoznamom pre jednotlivé iteračné kroky.



Evolučné programovanie

- patrí medzi tie stochastické optimalizačné algoritmy, ktoré možno chápať ako jednoduché zovšeobecnenie horolezeckého algoritmu
- Základná úloha spočíva v riešení nasledujúceho optimalizačného problému

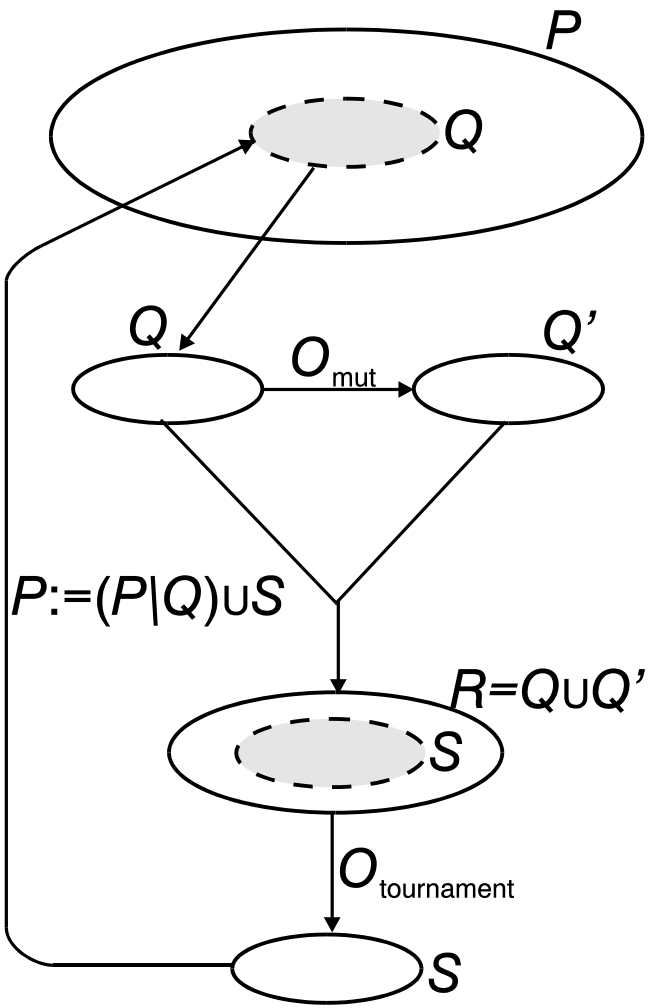
$$\alpha_{opt} = \arg \min_{\alpha \in \{0,1\}^k} f(\alpha)$$

- kde f , $f:\{0,1\}^k \to R$ je funkcia, ktorá zobrazuje binárne vektory dĺžky *k* na reálne čísla. Nech *P* je množina - populácia riešení tvaru

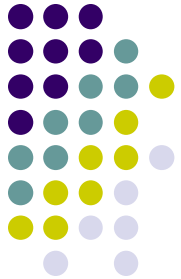
$$P = \{\alpha_1, \alpha_2, ..., \alpha_p\} \subseteq \{0,1\}^k$$

Z populácie *P* sa náhodne vyberie podpopulácia rodičov *Q*, ktorá je upravená pomocou operátora mutácie na podpopuláciu potomkov *Q'*. *Z* týchto dvoch podpopulácií sa vytvorí zjednotená podpopulácia *R*. Aplikovaním turnaja na túto podpopuláciu *R* dostaneme podpopuláciu nasledovníkov *S*. Podpopulácia nasledovníkov sa vracia do pôvodnej populácie *P* tak, že sa pôvodná rodičovská podpopulácia *Q* odstráni.

Evolučné programovanie

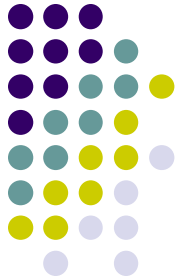


Evolučné programovanie



```
procedure Evolutionary_Programmning(output: P);
begin P:= náhodne generovaná populácia chromozómov;
      stop_criterion:=false;
      while not stop_criterion do
      begin Q:= náhodne vybraná podpopulácia z P;
            Q':=Omut(Q); R:=Q∪Q'; S:=Otournament(R);
            P:=(P\Q)∪S;
            if konvergenčné kritériá sú splnené then
            stop_criterion:=true;
      end;
end;
{Výstupom je populácia splňujúca konvergenčné kritéria}
```

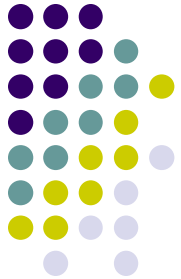
Genetické programovanie



- Na prelome 80-tých a 90-tých rokov americký informatik John Koza (Stanford University) navrhol originálnu modifikáciu genetického algoritmu, ktorú nazval ***genetické programovanie***.

- V tomto prístupe sú chromozómy - znakové reťazce nahradené zložitejšími štruktúrami - funkciami.

- Čo sa rozumie pod pojmom "funkcia"?



Jednoduchá funkcia "$x*(1+x)$" je reprezentovaná pomocou syntaktického stromu (parse tree)