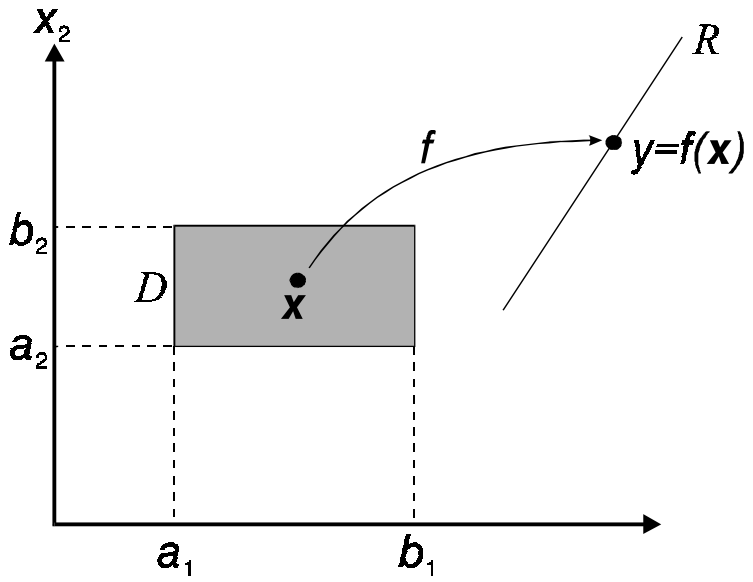
2. prednáška

Optimalizačný problém a kódovanie

$$f : D \rightarrow R$$

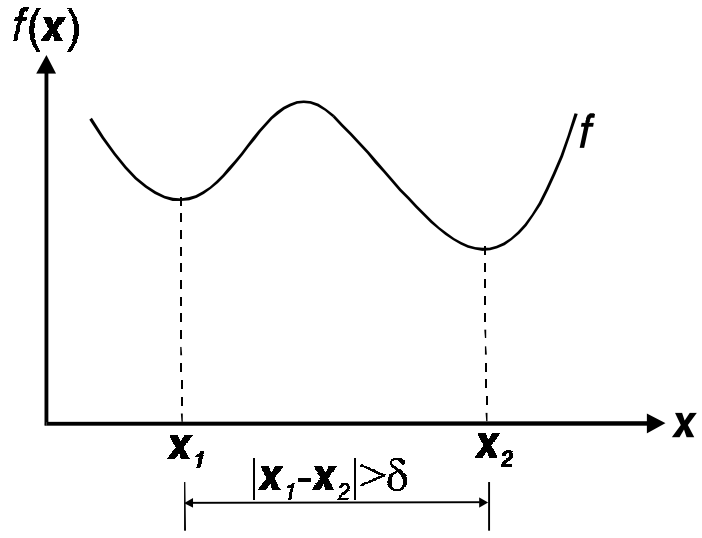
$$D = \prod_{i=1}^{n} [a_i, b_i] = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_n, b_n]$$



Funkcia *f* je ohraničená podmienkami

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

(2) Pre každú dvojicu lokálnych miním x_1, $x_2 \in D$ vzdialenosť $|x_1 - x_2|$ je väčšia ako dané kladné číslo $\delta > 0$, $|x_1 - x_2| > \delta$. 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.



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

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

Nájdenie globálneho minima použitím klasických optimalizačných metód patrí vo všeobecnosti medzi obtiažne numerické problémy pre funkcie, ktoré nie sú ohraničené ďalšími podmienkami

Z týchto dôvodov sa v súčasnosti [4,5] pri riešení globálneho optimaliyačného problému často používajú tzv. *evolučné optimalizačné algoritmy*, ktoré poskytujú riešenia blízke globálnemu, alebo s ním totožné.

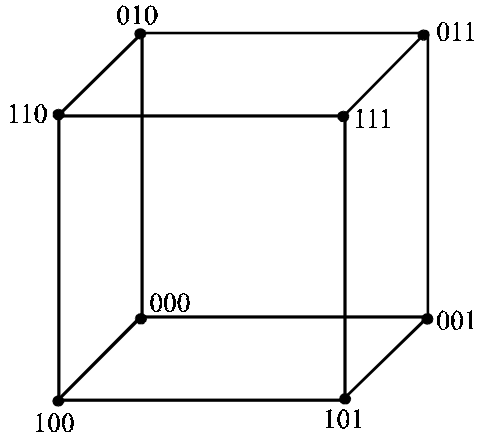
Binárna verzia funkcie má tvar

$$f:\{0,1\}^k \to R$$

$$y = f(\alpha)$$

Táto funkcia je definovaná nad množinou binárnych vektorov dĺžky *k*, kardinalita množiny binárnych vektorov dĺžky *k* je

$$\left|\{0,1\}^k\right| = 2^k$$

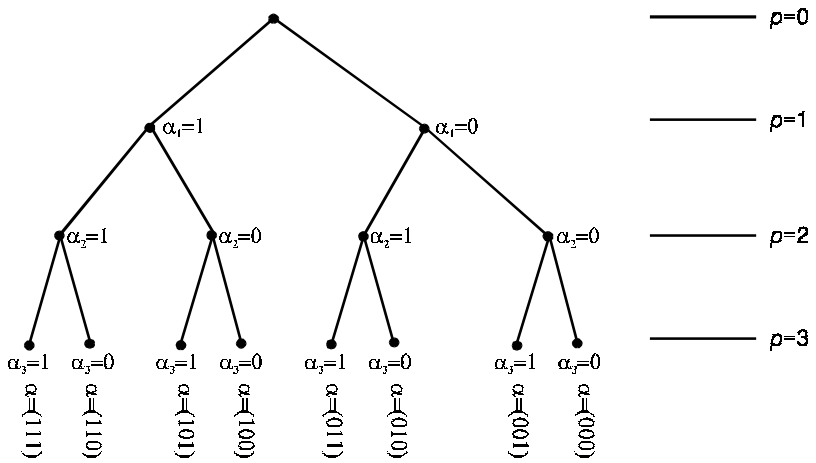


Optimalizačný problém pre binárne vektory má tvar

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

Vo všeobecnosti, globálne optimum $\boldsymbol{\alpha}_{opt}$ sa nájde po preskúšaní všetkých možných binárnych vektorov dĺžky k. Algoritmicky tento prístup môže byť implementovaný pomocou metódy spätného prehľadávania. CPU čas potrebný na riešenie optimalizačnej úlohy je potom úmerný kardinalite priestoru riešení

$$t_{CPU} \propto 2^k$$



Binárna reprezentácia reálnej premennej

Binárny vektor α dĺžky k

$$\alpha = (\alpha_1 \alpha_2 \ldots \alpha_k) \in \{0,1\}^k$$

je interpretovaný ako celé číslo

$$\text{int}(\alpha) = \sum_{i=1}^{k} \alpha_i 2^{k-i} =$$

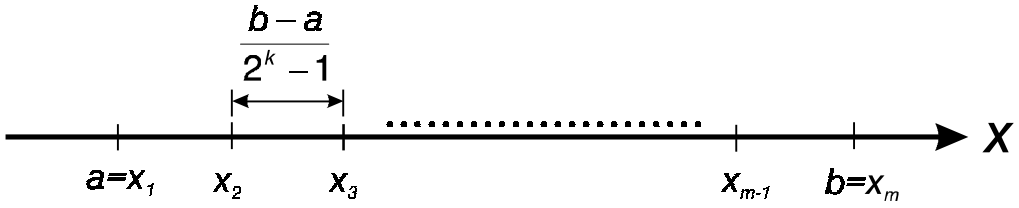
$$= \alpha_1 2^{k-1} + \alpha_2 2^{k-2} + \ldots + \alpha_{k-1} 2 + \alpha_k$$

K tomuto celému číslu priradíme racionálne číslo $x \in [a,b]$

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

real(α) aproximuje požadované reálne číslo x s presnosťou $(b-a)/2^k-1$.

Interval $[a,b]$ obsahuje $m=2^k$ bodov $x_1=a$, $x_2=a+(b-a)/(2^k-1)$, ..., $x_i=a+(i-1)(b-a)/(2^k-1)$, ..., $x_n=b$, pozri obr. 2.5 a tab. 2.1.



No.	α	int(α)	real(α)	$\tilde{\alpha}$ (Gray)
1	000	0	0	000
2	001	1	1/7	001
3	010	2	2/7	011
4	011	3	3/7	010
5	100	4	4/7	110
6	101	5	5/7	111
7	110	6	6/7	101
8	111	7	1	100

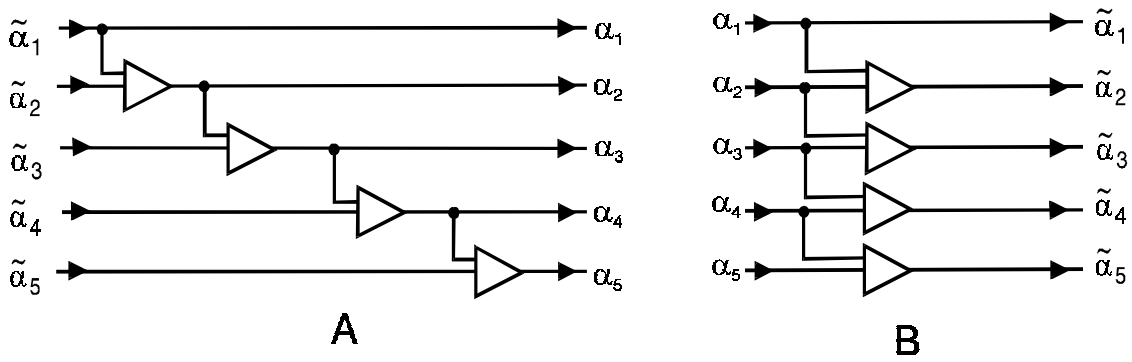
Grayova binárna reprezentácia

Dvojica binárnych reťazcov, ktoré sú odlišné vo všetkých polohách bitových premenných môže odpovedať dvom susedným celým číslam.

α_1 =(011) a α_2=(100) sú interpretované ako int(α_1)=4 resp. int(α_2)=5

Táto nevýhoda štandardného binárneho kódu je odstránená použitím tzv**. Grayovho kódu**. Jeho základná myšlienka spočíva v tom, že kóduje binárne čísla tak, že dve susedné celé čísla sú binárne reprezentované reťazcami, ktoré sú rôzne len v jednej polohe binárneho reťazca

$\tilde{\alpha}_1$=(010) a $\tilde{\alpha}_2$=(110) sú interpretované ako celé čísla int($\tilde{\alpha}_1$)=4 resp. int($\tilde{\alpha}_2$)=5



$$\tilde{\alpha}_1 = \alpha_1$$
$$\tilde{\alpha}_2 = \alpha_1 \oplus \alpha_2$$
$$\tilde{\alpha}_3 = \alpha_2 \oplus \alpha_3$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$\tilde{\alpha}_k = \alpha_{k-1} \oplus \alpha_k$$

$$\alpha_1 = \tilde{\alpha}_1$$
$$\alpha_2 = \tilde{\alpha}_1 \oplus \tilde{\alpha}_2 = \alpha_1 \oplus \tilde{\alpha}_2$$
$$\alpha_3 = \tilde{\alpha}_1 \oplus \tilde{\alpha}_2 \oplus \tilde{\alpha}_3 = \alpha_2 \oplus \tilde{\alpha}_3$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$\alpha_k = \tilde{\alpha}_1 \oplus \tilde{\alpha}_2 \oplus \ldots \oplus \tilde{\alpha}_k = \alpha_{k-1} \oplus \tilde{\alpha}_k$$

Transformácia spojitého optimalizačného problému na lineárny optimalizačný problém

Predpokladajme, že premenná $\boldsymbol{x} \in D$ je vyjadrená v binárnej reprezentácii bitovým vektorom dĺžky k. Budeme predpokladať, že dĺžka binárnej reprezentácie k je zvolená tak, že platí

$$\delta >> \frac{(b - a)}{(2^k - 1)}$$

Minimálna vzdialenosť medzi dvoma minimami funkcie $f(\boldsymbol{x})$ na oblasti D musí byť o mnoho väčšia ako "presnosť" binárnej reprezentácie.

Transformácia binernej reprezentácie na reálne číslo je formálne chápaná ako zobrazenie Γ

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

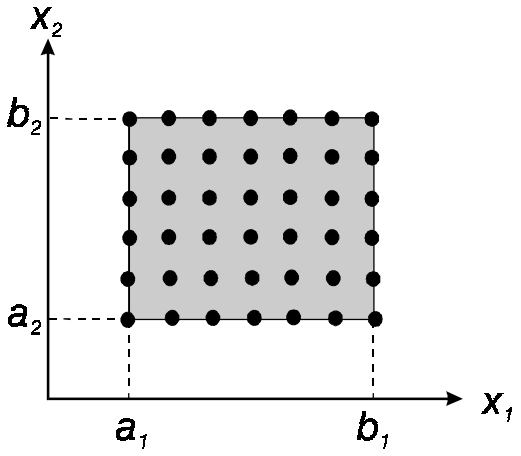
$$\boldsymbol{x} = \Gamma(\boldsymbol{\alpha})$$

Nech $\tilde{\boldsymbol{\alpha}}_{opt}$ je binárny vektor dĺžky *k*, ktorý bol získaný riešením optimalizačného problému

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

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

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



Závery

- Presnosť riešenia optimalizačného problému (2.2) pri prechode zo spojitej reprezentácie k binárnej reprezentácii závisí na konštante *k*, ktorá určuje dĺžku binárnych vektorov reprezentujúcich jednotlivé reálne premenné.
- Ak funkcia *f* obsahuje málo miním, ktoré sú dostatočne navzájom izolované (konštanta δ je veľká) a "široké", konštanta k nemusí byť veľká.
- Avšak, ak funkcia *f* obsahuje množstvo miním, ktoré ležia blízko seba (konštanta δ musí byť malá), potom konštanta *k* musí byt pomerne veľká.
- Ortogonálna mriežka bodov nad oblasťou *D*, ktoré sú generované zvolenou binárnou reprezentáciou reálnych premenných, musí byť dostatočne jemná, aby sa postihli a odlíšili blízko seba ležiace minimá funkcie *f*.

3. prednáška

Genetický algoritmus (GA)

Použitie metafory darvinovskej evolúcie v informatike pre konštrukciu optimalizačných algoritmov

Formalizácie základných pojmov evolúcie

Populácia P je množina týchto chromozómov

$$P = \{\alpha_1, \alpha_2, \ldots, \alpha_p\} \subseteq \{a, b, \ldots\}^k$$

Každý chromozóm $\alpha \in P$ je ohodnotený **silou** (fitness)

$$F: P \to R_+$$

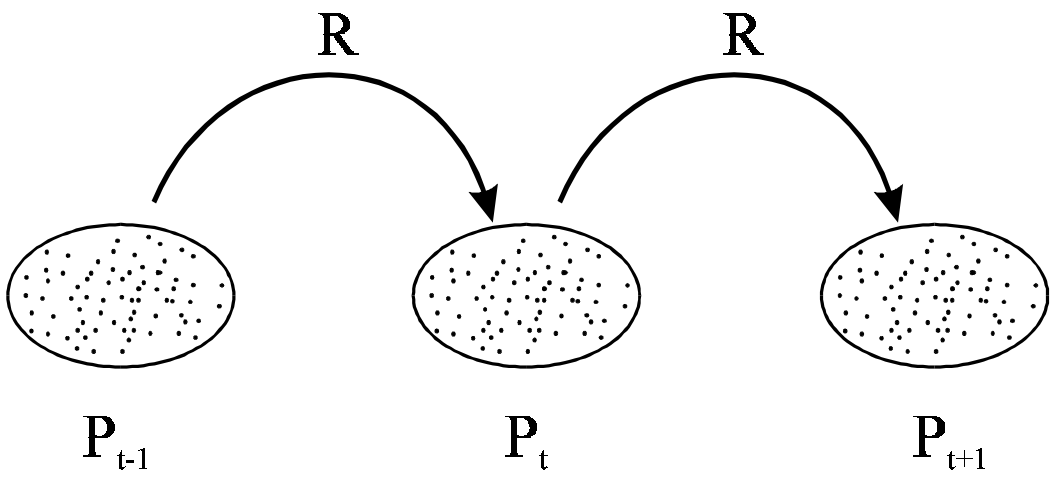
Reprodukcia je proces v ktorom z populácie sú kvázináhodne vybrané dva rodičovské chromozómy α_1 a α_2 v závislosti na ich sile (chromozómy s väčšou silou majú väčšiu pravdepodobnosť byť vybrané), pôvodné chromozómy sa reprodukujú na dva nové chromozómy – potomkov α_1' a α_2'

$$(\alpha_1', \alpha_2') = O_{repro}(\alpha_1, \alpha_2)$$

Reprodukčný operátor obsahuje dve časti, a to *kríženie* a *mutáciu*. Naviac, tento operátor má stochastický charakter, operácie kríženia a mutácie sa vykonávajú len s určitou pravdepodobnosťou.

Evolúcia je rekurentný proces, v ktorom populácia chromozómov P_{t+1} v čase $t+1$ je určená len pomocou predchádzajúcej populácie P_t v čase t

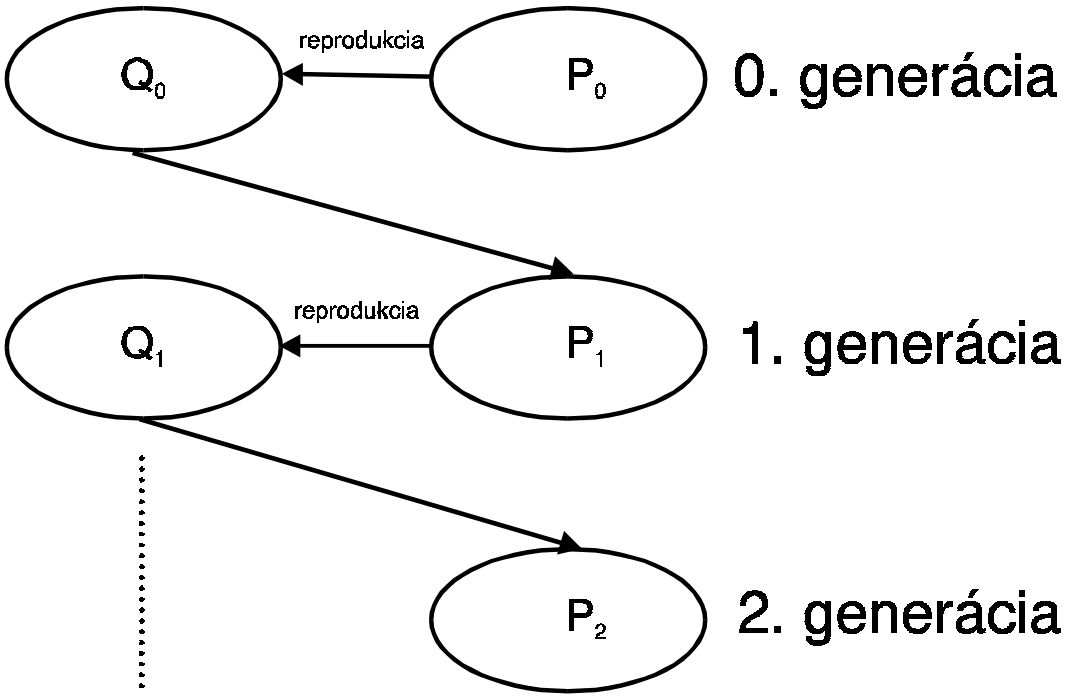
$$P_{t+1} = R(P_t)$$



Pseudopascalovská verzia algoritmu evolúcie v živej prírode, založenej na Darwinovskom prirodzenom výbere.

```
procedure Evolution;
begin t:=0;
      P:=randomly generated population of
         chromosomes;
      while t<t_max do
      begin t:=t+1; Q:=∅;
            while |Q|<|P| do
            begin α1:=O_select(P);
                  α2:=O_select(P);
                  if random<P_repro then
                  (α1',α2')=O_repro(α1,α2)
                  else
                  (α1',α2')=(α1,α2);
                  Q:=Q∪{α1',α2'};
            end;
            P:=Q;
       end;
       P_fin:=P;
end;
```

Rekurentnosť evolúcie



Aký je vzťah medzi horolezeckým algoritmom a evolučným algoritmom?

- Evolučný algoritmus sa zakladá na simultánnej optimalizácii celej populácie chromozómov, zatiaľ čo horolezecký algoritmus optimalizuje náhodne len jeden objekt - chromozóm.
- Ak v procese reprodukcie chromozómov uvažujeme len mutácie, tak evolučný algoritmus je veľmi podobný horolezeckému algoritmu.
- Možnosť úniku z lokálneho minima pomocou (spočiatku málo výhodnej) mutácie v evolučnom algoritme je spojená s potenciálnou nevýhodou, vygenerovaný chromozóm môže mať menšiu silu ako pôvodný chromozóm.
- V evolučnom algoritme je lokálne hľadanie optimálneho riešenia horolezeckého algoritmu nahradené efektívnejším spôsobom simultánnej optimalizácie celej populácie chromozómov.

Prečo v evolučnom algoritme do reprodukčného procesu vyberáme chromozómy kvázináhodne?

- Ak by sme do reprodukčného procesu vyberali len tie chromozómy, ktoré majú najväčšiu silu, potom by sme s určitou pravdepodobnosťou podstatne ohraničili oblasť *D*, na ktorej hľadáme optimálne riešenie.
- Evolučný algoritmus vyberá chromozómy do procesu reprodukcie *kvázináhodne*.
- Obrazne môžeme povedať, že v rámci evolučného algoritmu nevystupuje *deux ex machina*, ktorý vopred pozná výsledok evolúcie populácie chromozómov a môže zasahovať výberom chromozómov do procesu reprodukcie.

Základné pojmy genetického algoritmu - populácia a sila chromozómov

- Genetický algoritmus (GA) bol vynájdený v polovici 70-tých rokov Hollandom. V súčasnosti patrí medzi najpopulárnejšie evolučné optimalizačné algoritmy.
- Základné princípy genetického algoritmu sú určitou špecifikáciou všeobecných princípov evolúcie a jej algoritmizácie.

Chromozóm α je binárny vektor fixnej dĺžky k

$$\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k) \in \{0,1\}^k$$

Populácia P je multimnožina obsahujúca chromozómy α

$$P = \{\alpha_1, \alpha_2, \ldots, \alpha_p\}$$

Účelová funkcia f je definovaná nad množinou binárnych vektorov dĺžky k

$$f : \{0,1\}^k \rightarrow R$$

Evolučná úloha je nájsť globálne minimum tejto funkcie nad množinou $\{0,1\}^k$

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

Funkcia f "reprezentuje prostredie" v ktorom existujú chromozómy populácie.

Biologická terminológia: chromozóm α reprezentuje *genotyp* organizmu, zatiaľ čo funkčná hodnota $f(\alpha)$ reprezentuje jeho *fenotyp*. Mierou *úspešnosti* chromozómu je jeho funkčná hodnota.

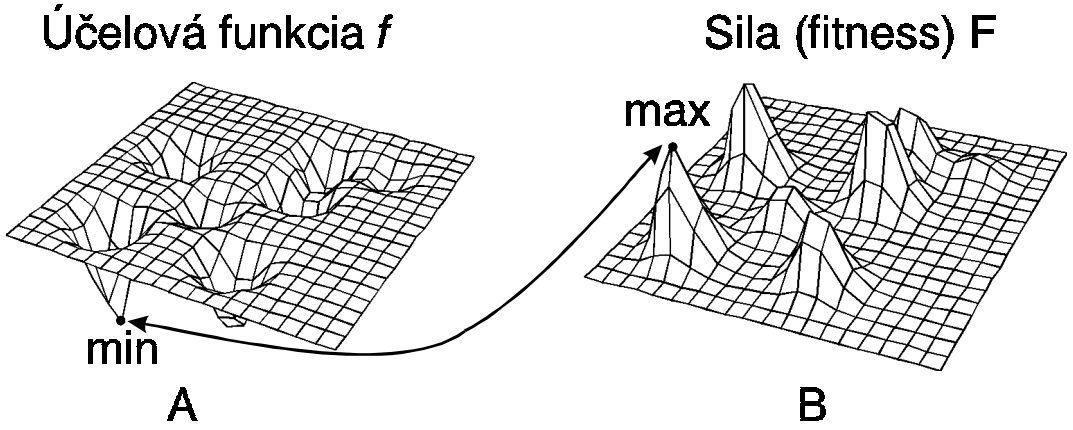
Chromozóm je tým **úspešnejší** (má **väčšiu silu**), čím je jeho funkčná hodnota (genotyp) menšia

$$\forall \alpha_1, \alpha_2 \in P : f(\alpha_1) \leq f(\alpha_2) \Rightarrow F(\alpha_1) \geq F(\alpha_2) \geq 0$$

Renormalizovaná sila

$$F'(\alpha) = \frac{F(\alpha)}{\sum_{a' \in P} F(\alpha')}$$

$$\sum_{\alpha \in P} F'(\alpha) = 1$$



Akým konkrétnym spôsobom realizovať zobrazenie hodnôt účelovej funkcie na silu?

Prvá metóda je jednoduché lineárne zobrazenie funkčných hodnôt na silu tak, že maximálnej (minimálnej) funkčnej hodnote je priradená minimálna (maximálna) sila

$$F(\alpha) = \frac{F_{max} - F_{min}}{f_{min} - f_{max}} f(\alpha) + \frac{f_{min} F_{min} - f_{max} F_{max}}{f_{min} - f_{max}}$$

$$f_{min} \leq f(\alpha) \leq f_{max}$$

$$F_{min} = \varepsilon \leq F(\alpha) \leq F_{max} = 1$$

Finálna formula pre lineárnu transformáciu funkčných hodnôt na silu

$$F(\alpha) = \frac{1}{f_{min} - f_{max}} \left[(1 - \varepsilon) f(\alpha) + f_{min} \varepsilon - f_{max} \right]$$

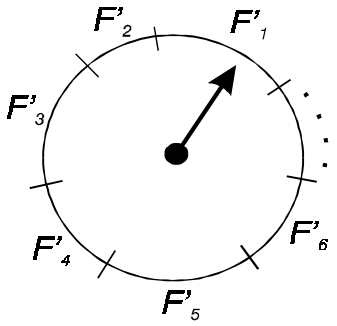
Druhá metóda je založená na usporiadaní funkčných hodnôt do rastúcej postupnosti. Nech $Q=(q_1,q_2,...,q_{|P|})$ je taká permutácia chromozómov z populácie P, ktorá usporiada hodnoty účelovej funkcie do neklesajúcej postupnosti

$$f\left(\alpha_{q_1}\right) \leq f\left(\alpha_{q_2}\right) \leq ... \leq f\left(\alpha_{q_{|P|}}\right)$$

Potom chromozómu α_{q_1} ($\alpha_{q_{|P|}}$)priradíme maximálnu (minimálnu) silu $F_{\max}=1$ ($F_{\min}=\varepsilon$)

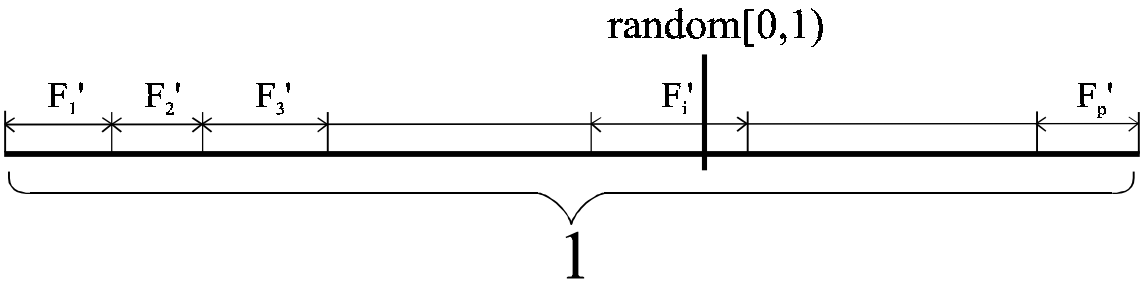
$$F\left(\alpha_{q_i}\right) = \frac{1}{1-|P|}\left[(1-\varepsilon)i + \varepsilon - |P|\right] \left(\text{pre } i = 1,2,...,|P|\right)$$

Pomocou renormalizovaných síl môžeme jednoducho realizovať kvázináhodnosť výberu chromozómov (čo patrí medzi základné imperatívy evolučných algoritmov) pomocou pojmu **ruleta**



Chromozóm s indexom *i* je vybraný (kvázináhodne), ak pre náhodne generované číslo *z* intervalu [0,1) s rovnomernou distribúciou pravdepodobnosti platí

$$\sum_{k=1}^{i-1} F'_k \le z < \sum_{k=1}^{i} F'_k$$



Jednoduchá implementácia rulety

```
function Roulette_Wheel:integer;
begin aux:=random;
      bound_lower:=0; bound_upper:=F'1;
      i:=0; criterion:=false;
      while (i<|P|) and (not criterion) do
      begin i:=i+1;
            criterion:=(bound_lower≤aux)
            and (aux<bound_upper);
            bound_lower:=bound_upper;
            bound_upper:=bound_upper+F'i+1;
      end;
      Roulette_wheel:=i;
end;
```

Poznámka: Korektná implementácia algoritmu je založená na binárnom hľadaní.

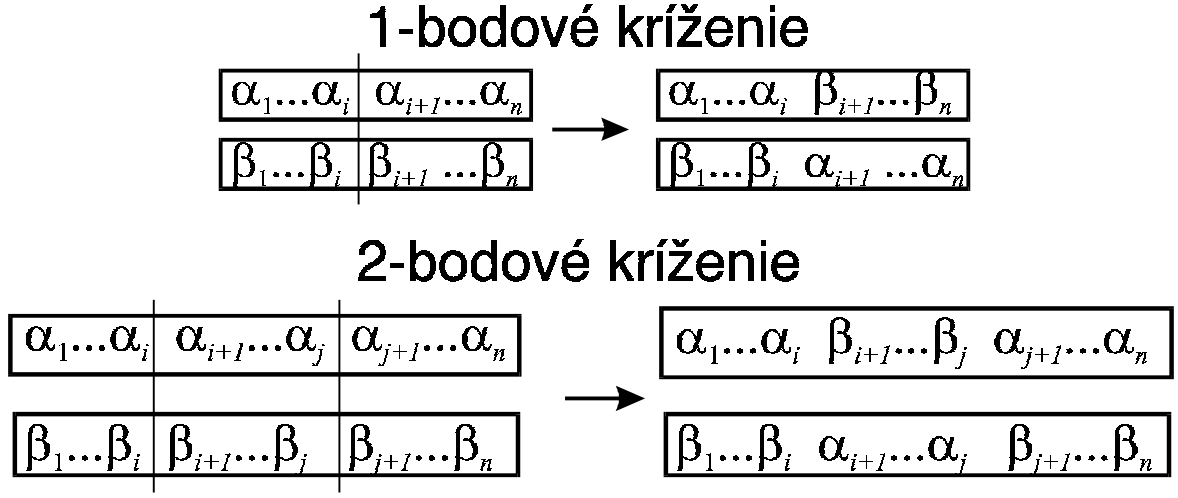
Kríženie chromozómov

Pre chromozómy z populácie P sú definované dve základné operácie:

1. operácia mutácie a
2. operácia kríženia.

Majme dva chromozómy $\alpha,\beta \in \{0,1\}^k$, operácia kríženia bude formálne interpretovaná ako operátor - zobrazenie, ktorý tejto dvojici chromozómov priradí dva nové chromozómy o rovnakej dĺžke ako pôvodné

$$(\alpha',\beta') = O_{cross}(\alpha,\beta)$$



Pseudopascalovská implementácia kríženia

```
procedure Crossover;
begin cross_point:=1+random(k-1);
      for i:=1 to cross_point do
      begin α'i:=αi; β'i:=βi end;
      for i:=cross_point+1 to k do
      begin α'i:=βi; β'i:=αi end;
end;
```

Operátor reprodukcie obsahuje kríženie a mutáciu

$$(\alpha',\beta') = O_{repro}(\alpha,\beta)$$

$$(\tilde{\alpha},\tilde{\beta}) = O_{cross}(\alpha,\beta)$$

$$\alpha' = O_{mut}(\tilde{\alpha}) \quad \text{a} \quad \beta' = O_{mut}(\tilde{\beta})$$

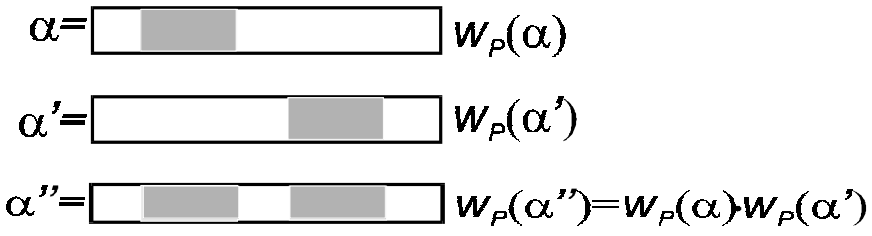
Význam kríženia

- Kríženie patrí medzi základnú črtu genetického algoritmu, ak ho odstránime, potom sa nám genetický algoritmus zredukuje na niečo blízke horolezeckému algoritmu.
- Používanie kríženia odlišuje genetický algoritmus od ostatných stochastických evolučných algoritmov, ktoré v rámci populácie objektov - riešení - chromozómov tiež používajú reprodukciu založenú na ich sile a aplikujú mutácie na jednotlivé objekty.
- Medzi pracovníkmi zaoberajúcimi sa aplikáciami genetického algoritmu panuje presvedčenie, že efektívnosť týchto algoritmov je vo všeobecnosti menšia ako efektívnosť genetického algoritmu hlavne z toho dôvodu, že nepoužívajú kríženie.

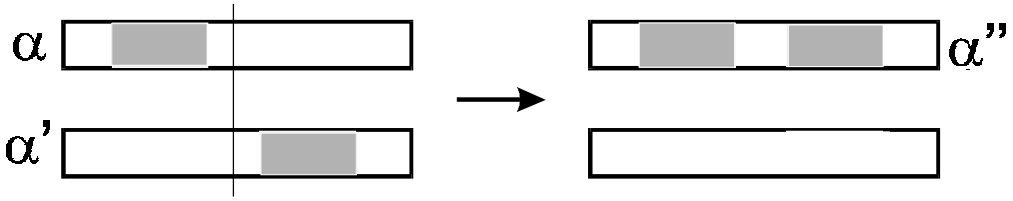
Význam kríženia (pohlavnej reprodukcie)

- Pohlavná reprodukcia je v prírode všeobecne využívaná hlavne medzi zložitejšími druhmi našej planéty. Pohlavná reprodukcia je neobyčajne zložitý proces produkcie potomkov.
- Druh využívajúci pohlavnú reprodukciu musí byť podstatne zložitejší ako druh, ktorý ju nevyužíva.
- Pretože pohlavná reprodukcia jednoznačne zvíťazila v aréne prirodzeného výberu, musia existovať vážne dôvody, prečo tento komplikovaný spôsob reprodukcie patrí medzi základné črty, ktoré odlišujú nižšie druhy od vyšších druhov.

Jeden z hlavných dôvodov (podporený dôkladnou matematickou analýzou) výhodnosti kríženia je fakt, že pohlavná reprodukcia umožňuje rýchlu výmenu výhodných vlastností, ktoré podstatne zvyšujú silu jedincov - potomkov, čo je len veľmi obtiažne dosiahnuteľné len pomocou mutácií.

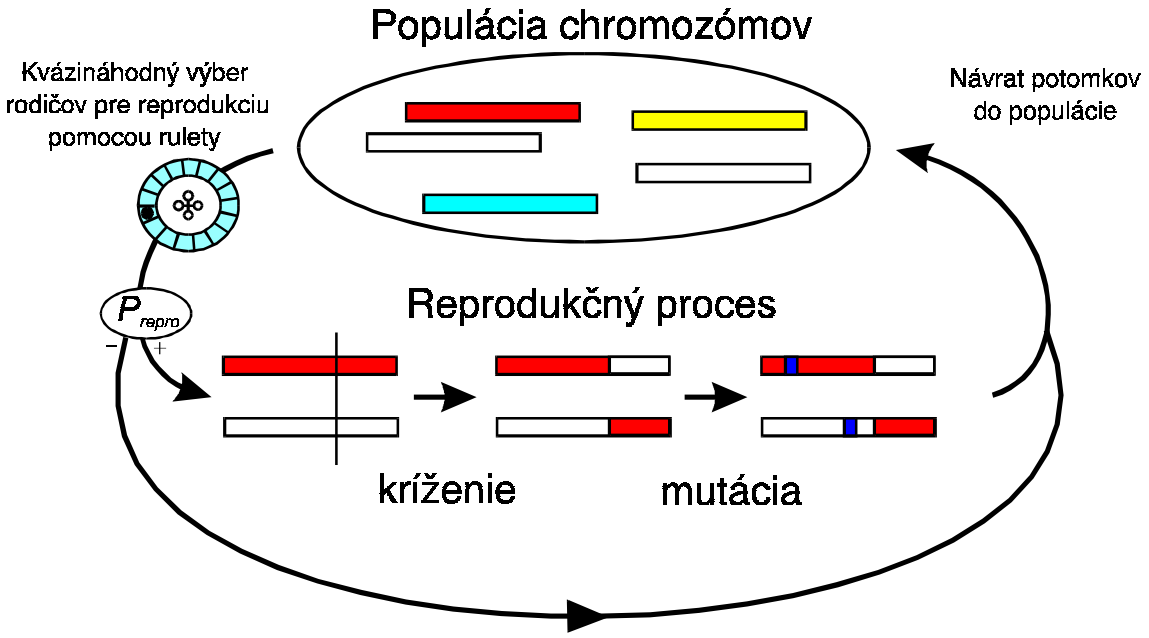


A



B

Schematické znázornenie genetického algoritmu



Implementácia genetického algoritmu

```
procedure Genetic_Algorithm;
begin t:=0; stop_criterion:=false;
      P:=randomly generated population;
      while (t<tmax) and
            (not stop_criterion) do
      begin t:=t+1;
            Q:=∅;
            each chromosome is evaluated
            by fitness;
            while |Q|<|P| do
            begin α1:=Oselect(P);
                  α2:=Oselect(P);
                  if random<Prepro then
                     R(α1,α2,α1',α2') else
                  begin α1' := α1;
                        α2' := α2
                  end;
                  Q:= Q ∪ {α1',α2'};
            end;
            P:=Q;
            if convergence criteria are
            fulfilled then
            stop_criterion:=true;
       end;
       αopt:=best chromosome of P;
end;
```

Problém zastavenia genetického algoritmu

1. Prepísaný počet evolučných epoch (premenná *t*).
2. Výpočet analógu pravdepodobnostného vektora $\boldsymbol{w}=(w_1,w_2,...,w_k)$ z horolezeckého algoritmu s učením. Nech populácia P v i-tom kroku (generácii) obsahuje chromozómy $P=\left(\alpha_1,\alpha_2,...,\alpha_p\right)$, kde $\alpha_i=\left(\alpha_1^{(i)},\alpha_2^{(i)},...,\alpha_k^{(i)}\right)$. Komponenty pravdepodobnostného vektora sú určené takto

$$w_j=\frac{1}{p}\sum_{i=1}^{p}\alpha_j^{(i)}\ \left(\text{pre } j=1,2,...,k\right)$$

Parameter usporiadania definovaný pre takto určený pravdepodobnostný vektor má tvar

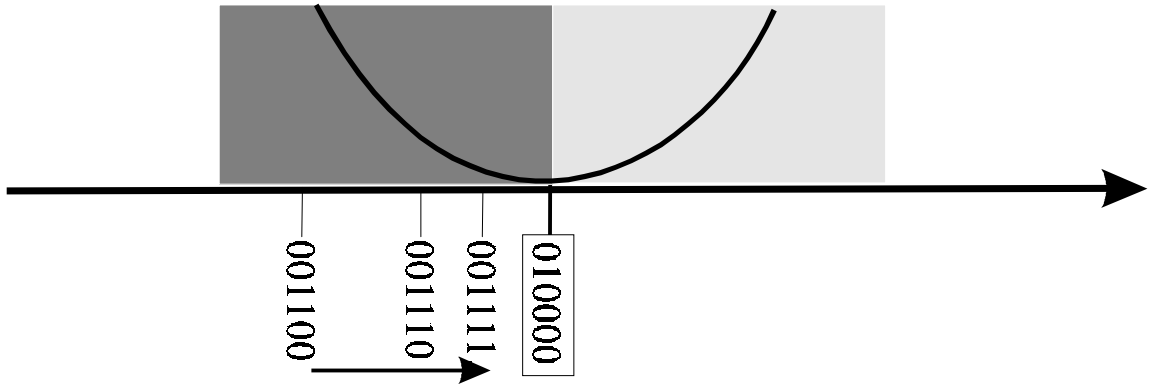
$$\chi(\boldsymbol{w}) = \frac{4}{k}\sum_{j=1}^{k}\left(w_j - 0.5\right)^2$$

Tento parameter s funkčnými hodnotami z intervalu [0,1] má tú vlastnosť, že pre náhodne generovanú populáciu P (prvé iteračné kroky genetického algoritmu) sa blíži nule (je malé kladné číslo). Tak ako evolúcia populácie pokračuje, hodnota parametru rastie a asymptoticky sa blíži jednotke.

Hammingova bariéra

- Dva chromozómy α=(00100...0) a β=(000111...1) reprezentujú dve susedné celé čísla
- Tieto dva chromozómy aj keď reprezentujú dve "susedné" čísla, ich Hammingovská vzdialenosť je veľká.
- Predpokladajme, že optimálne riešenie odpovedá binárnemu reťazcu α=(00100...0) a že populácia obsahuje chromozómy, ktoré sú blízke alebo totožné s binárnym reťazcom β=(000111...1).
- Chromozómy populácie (00011**...*) sa vyvíjajú smerom k optimálnemu riešeniu tak, že po určitom počte iteračných krokov populácie bude obsahovať v prevažnej miere chromozómy β=(000111...1).
- Pravdepodobnosť zmeny (00011**...*) mutáciou na (00100**...*) je veľmi malá.

- Evolúcia populácie chromozómov, ktorá bola inicializovaná chromozómami typu (00011**...*), má len veľmi malú šancu byť smerovaná do výslednej populácie, ktorá by obsahovala optimálne riešenie α=(00100...0), a s najväčšou pravdepodobnosťou skončí u riešenia β=(000111...1).
- Táto skutočnosť sa v genetickom algoritme nazýva Hammingova bariéra a predstavuje vážne ohraničenie aplikovateľnosti genetického algoritmu za predpokladu, že sa používa štandardné kódovanie binárnych reťazcov.



Ako sa vyhnúť Hammingovej bariére v genetickom algoritme?

Prvý prístup je založený na použití **operátora inverzie**, ktorý zmení binárny reťazec $\alpha = (\alpha_1, \alpha_2, ..., \alpha_k)$ na iný binárny reťazec $\beta = (\beta_1, \beta_2, ..., \beta_k)$

$$\beta = O_{inv}(\alpha)$$

Pre náhodne vygenerované celé číslo $1 < a < k$ (bod inverzie), komponenty binárneho vektora pre indexy $i>a$ sú zamenené za ich komplementy

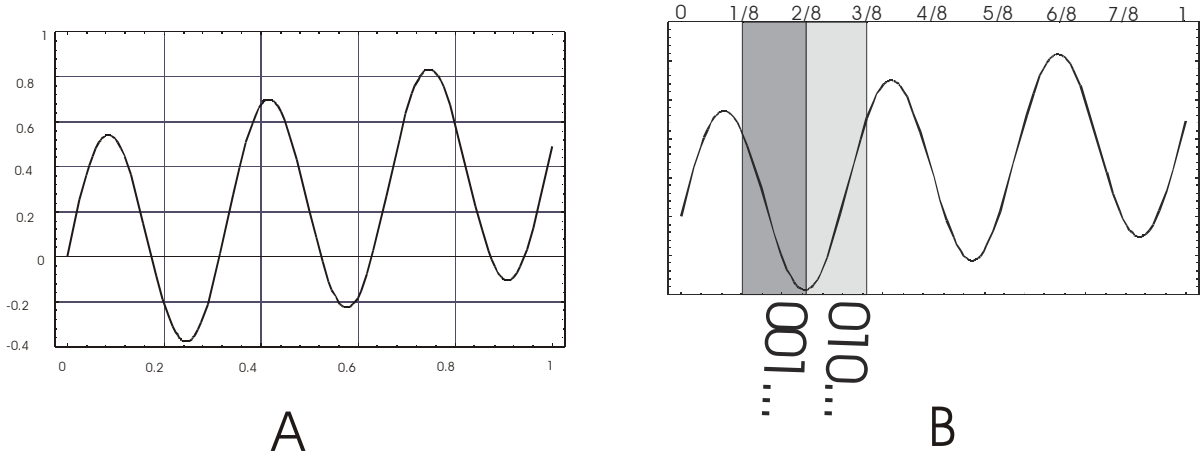
$$\beta_i = \begin{cases} \alpha_i \ (\text{pre } i \in [1, a]) \\ 1\text{-}\alpha_i \ (\text{pre } i \in [a+1, k]) \end{cases}$$

Stochastický operátor inverzie odstraňuje vznikajúce Hammingove bariéry.

Ilustračný príklad

Základné pojmy z predchádzajúcej kapitoly budú ilustrované jednoduchou implementáciou genetického algoritmu s nasledujúcou účelovou funkciou

$$f(x) = e^{-x^2/100} \times sin(10x) \times cos(9x) \quad \forall x \in [0,1]$$



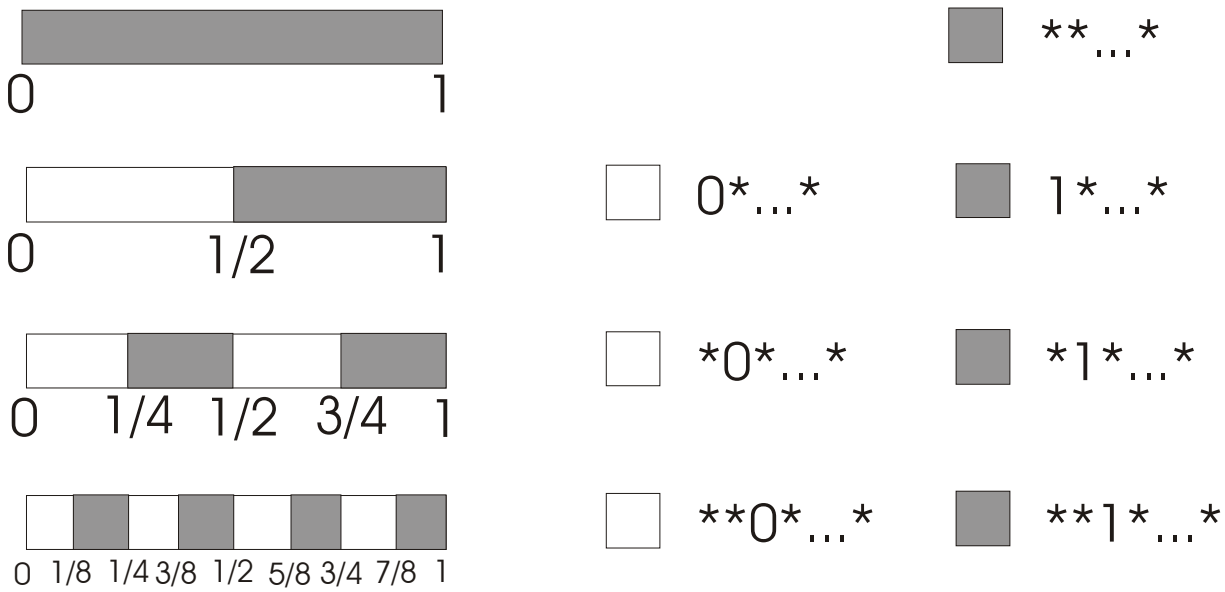
(A) Funkcia má tri minimá: (1) x_1=0.245322, f(x_1)=-0.377681, globálne minimum, (2) x_2=0.576377, f(x_3)=-0.226259, (3) x_3=0.907692, f(x_3)=-0.104824.

(B) Rozdelenie intervalu [0,1] na osem podintervalov rovnakej dĺžky. Každému tomuto podintervalu môže byť priradený binárny kód.

No.	Schéma	Rozdelenie intervalu [0,1]							
		1/8	2/8	3/8	4/8	5/8	6/8	7/8	
1	∗∗∗...	■	■	■	■	■	■	■	■
2	∗∗0...	■		■		■		■	
3	∗∗1...		■		■		■		■
4	∗0∗...	■	■			■	■		
5	∗00...	■				■			
6	∗01...		■				■		
7	∗1∗...			■	■			■	■
8	∗10...			■				■	
9	∗11...				■				■
10	0∗∗...	■	■	■	■				
11	0∗0...	■		■					
12	0∗1...		■		■				
13	00∗...	■	■						
14	000...	■							
15	001...		■						
16	01∗...			■	■				
17	010...			■					
18	011...				■				
19	1∗∗...					■	■	■	■
20	1∗0...					■		■	
21	1∗1...						■		■
22	10∗...					■	■		
23	100...					■			
24	101...						■		
25	11∗...							■	■
26	110...							■	
27	111...								■

Obrázok 4.14. Delenie intervalu [0,1] podľa schém, ktorých prvé tri znaky sú špecifikované. Napríklad, schéma č. 20 tvaru 1∗0...určuje reálnu premennú z prieniku intervalov [1/2,5/8]∪[6/8,7/8].

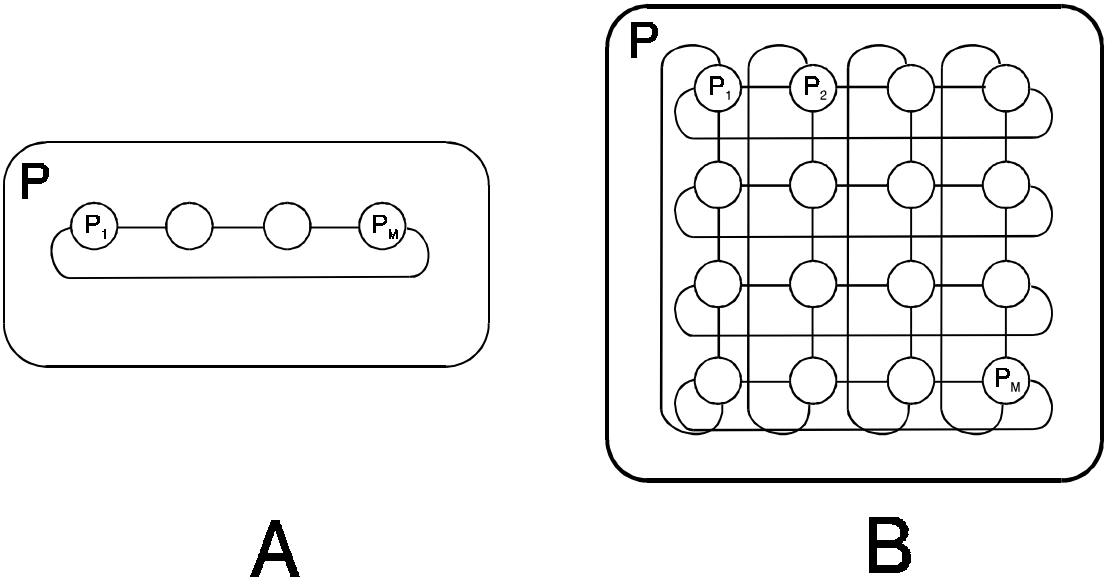
Globálne minimum funkcie je približne vyjadrené v binárnej reprezentácii chromozómami, ktoré obsahujú schémy 001∗...∗ (tieto schémy odpovedajú reálnym číslam z intervalu [1/8,1/4]) alebo 010∗...∗ (tieto schémy odpovedajú reálnym číslam z intervalu intervalu [1/4,3/8]).
Pretože globálne minimum je v bode x=0.245..<1/4, toto minimum sa vyskytuje v blízkosti hornej hranici prvého intervalu, čiže binárny reťazec odpovedajúci riešeniu je približne vyjadrený ako 0011111... .



Druhý prístup je založený na použití použitie Grayovho kódu pre binárnu reprezentáciu reálnych premenných.

Paralelný genetický algoritmus

Genetický algoritmus je sekvenčný algoritmus. V literatúre existuje mnoho verzií tzv. **paralelného genetického algoritmu** (PGA) v ktorých je populácia rozdelená na podpopulácie, pričom evolúcia prebieha "nezávisle" nad týmito podpopuláciami, t.j. medzi nimi dochádza k občasnej stochastickej interakcii v rámci ktorej si vymenia chromozómy. Obvykle, tieto paralelné verzie sú prezentované (a tiež aj aplikované) ako spôsob diverzifikácie a intenzifikácie genetického algoritmu.



A B

$P=P_1 \cup P_2 \cup ... \cup P_M$ a $P_i \cap P_j = \varnothing$, pre i≠j.

7. prednáška

Kombinatoriálne optimalizačné problémy

Formulácia základných kombinatoriálnych optimalizačných problémov

Jeden z prvých veľkých úspechov evolučných algoritmov bolo ich použitie pre riešenie notoricky obtiažnych kombinatoriálnych problémov, ktorých CPU čas rastie buď faktoriálovo alebo exponenciálne s rastom dimenzie problému (NP = "**n**on**p**olynomialy" obtiažne).

Dva typy kombinatoriálnych problémov

Typ 1. Funkcia f je definovaná nad symetrickou grupou S_N zloženou zo všetkých permutácií N objektov

$$f : S_N \to R$$

kde permutácie z S_N sú definované ako N-tica celých rôznych čísel

$$P = \left(p_1, p_2, \ldots, p_N \right)$$

Optimalizačný problém má tvar

$$P_{opt} = \arg \min_{P \in S_N} f\left(P\right)$$

Riešenie tohto optimalizačného problému je obvykle veľmi obtiažne v dôsledku skutočnosti, že symetrická grupa S_N obsahuje $N!$ permutácií. Z týchto dôvodov môže nastať situácia, že pre veľké N CPU čas zhruba rastie ako $N!$.

Typ 2. Nech R_{set}={1,2,...,r} je množina obsahujúca prvých r celých čísel, jej priamy súčin R_{set}^N obsahuje N-tice

$$\pi = (\pi_1, \pi_2, ...\pi_N)$$

Jednotlivé komponenty tohto výrazu sú celé čísla z uzavretého intervalu [1,r]. Funkcia

$$f : R_{set}^N \rightarrow R$$

zobrazuje N-tice π na reálne čísla, optimalizačný problém má tvar

$$\pi_{opt} = arg \min_{\pi \in R_{set}^N} f(\pi)$$

V dôsledku toho, že kardinalita množiny R_{set}^N je r^N, optimalizačný problém môže patriť medzi obtiažne s exponenciálnym rastom CPU času pre veľké N. Poznamenajme, že pre r=2 sa tento optimalizačný problém redukuje na obyčajný binárny problém.

Úloha obchodného cestujúceho

Typ 1 kombinatoriálneho problému bude ilustrovaný známou úlohou obchodného cestujúceho (TSP, Traveling Salesman Problem).

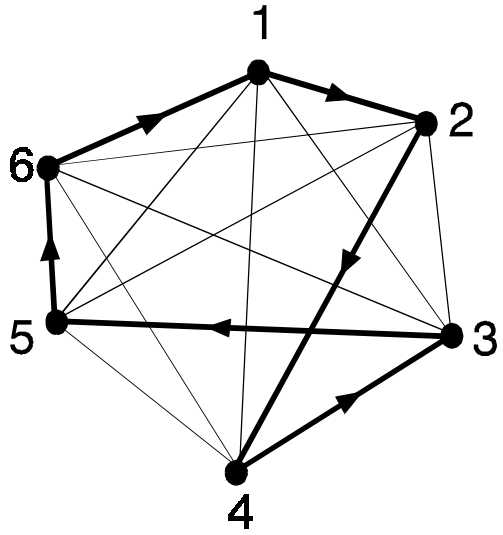
Grafovo-teoretická formulácia je nasledovná

Nech G je úplný graf obsahujúci N vrcholov a $N(N-1)/2$ hrán (spojov).

Každá hrana $[i,j]$, spájajúca i-ty s j-tym vrcholom (mestom) je ohodnotená kladným číslom $d(i,j)$, ktoré sa interpretuje ako vzdialenosť medzi danými vrcholmi - mestami.

Služobná cesta obchodného cestujúceho (Hamiltónov cyklus) navštívi každé mesto len raz, pričom sa vracia do východzieho mesta.

Táto cesta je určená pomocou permutácie N objektov - miest, odpovedá ceste, ktorá je zahájená v meste p_1, potom pokračuje postupne v mestách p_2, p_3, ..., p_N, a na záver sa vráti do východzieho mesta p_1.



Každej ceste je priradená jej dĺžka

$$f(P) = d(p_1, p_N) + \sum_{i=2}^{N} d(p_{i-1}, p_i)$$

Cieľ úlohy TSP je nájsť takú cestu - permutáciu P_{opt} , ktorá poskytuje minimálnu dĺžku cesty $f_{opt}=f(P_{opt})$.

Mutácia permutácie - cesty

Cesta P môže byť zmenená na novú cestu P' pomocou stochastického operátora mutácie O_{mut} špecifikovaného pravdepodobnosťou P_{mut}.

$$P \in S_N \Rightarrow P' = O_{mut}(P) \in S_N$$

$$\lim_{P_{mut} \to 0} O_{mut}(P) = P$$

```
procedure Permutation_Mutation;
begin for i:=1 to N do p'i:=pi;
      for i:=1 to N do
      if random<Pmut then
      begin j:=1+random(N);
            aux:=p'i;
            p'i:=p'j;
            p'j:=aux
      end;
end;
```

$O_{\text{mut}}(2,1,4,3,5) \rightarrow (2,5,4,3,1)$

Kríženie pernutácií - ciest

K tomu, aby sa tento typ kombinatotiálnych úloh mohol študovať genetickým algoritmom, musíme zaviesť ešte operáciu kríženia medzi dvoma permutáciami

$$(P',Q') = O_{cross}(P,Q)$$

ktoré zachováva ich charakter permutácie, t.j. P' a Q' sú taktiež permutácie. Študujme dve permutácie P a Q n objektov, vyberme bod kríženia a tak, že $1<a<n$

$$P = (p_1,...p_a, p_{a+1},..., p_n),\ Q = (q_1,...q_a, q_{a+1},..., q_n)$$

Keď vymeníme segmenty, ktoré nasledujú za bodom kríženia, dostaneme dva nové objekty

$$\hat{P} = (p_1,...p_a, q_{a+1},..., q_n),\ \hat{Q} = (q_1,...q_a, p_{a+1},..., p_n)$$

Žiaľ, takto vytvorené objekty nemusia byt už permutácie.

Môže nastať situácia, že nejaké celé číslo sa v objektoch $\hat{P}$ alebo $\hat{Q}$ vyskytuje dvakrát. Z tohto dôvodu je potrebné aplikovať "opravný proces" (nazývaný čiastočné priradenie - partial matching), ktorý z objektov $\hat{P}$ a $\hat{Q}$ vytvorí normálne permutácie (poznamenajme, že v literatúre je popísaných mnoho rôznych druhov kríženia dvoch permutácií a spôsobov ich opráv).

Pre ilustráciu študujme kríženie medzi dvoma permutáciami

$$P = (3,4,5,1,2,6,10,8,9,7)$$
$$Q = (1,2,6,10,8,3,4,5,7,9)$$

predpokladajme, že bod kríženia je a=4. Ak vymeníme medzi dvoma permutáciami segmenty po bode kríženia dostaneme

$$\hat{P} = (3,4,5,1,8,3,4,5,7,9),$$
$$\hat{Q} = (1,2,6,10,2,6,10,8,9,7)$$

Vidíme, že objekty $\hat{P}$ a $\hat{Q}$ nie sú permutácie, pretože obsahujú niektoré indexy dvakrát.

K oprave týchto objektov na permutácie zostrojíme zobrazenia f_{QP} a f_{PQ}

$$f_{PQ} = \begin{pmatrix} 2\,6\,10\,8\,7\,9 \\ 8\,3\,4\,5\,9\,7 \end{pmatrix}$$

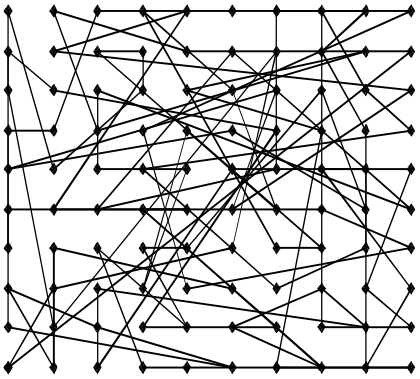
$$f_{QP} = \begin{pmatrix} 8\,3\,4\,5\,9\,7 \\ 2\,6\,10\,8\,7\,9 \end{pmatrix}$$

Objekt $\hat{P}$ opravíme pomocou zobrazenia f_{QP} a objekt $\hat{Q}$ pomocou zobrazenia f_{PQ}. Prvé tri indexy v P' zameníme za $3\rightarrow6$, $4\rightarrow10$ a $5\rightarrow8\rightarrow2$. Podobne, prvé indexy 2, 6 a 10 v Q' zameníme za $2\rightarrow8\rightarrow5$, $6\rightarrow3$ a $10\rightarrow4$, dostaneme opravené objekty, ktoré už sú permutácie a sú považované za výsledok výmeny medzi permutáciami P a Q

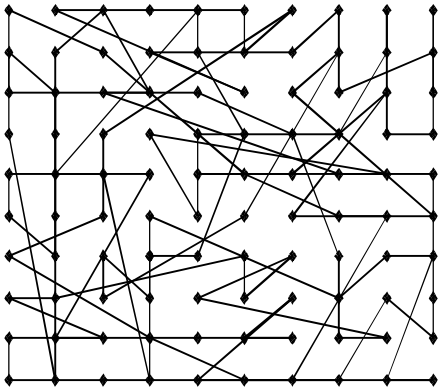
$$P' = (6,10,2,1,8,3,4,5,7,9)$$

$$Q' = (1,5,3,4,2,6,10,8,9,7)$$

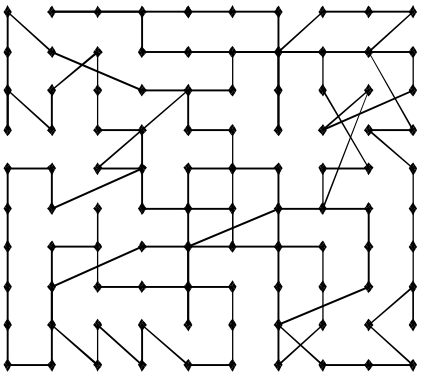
Úloha obchodného cestujúceho na mriežke 10×10



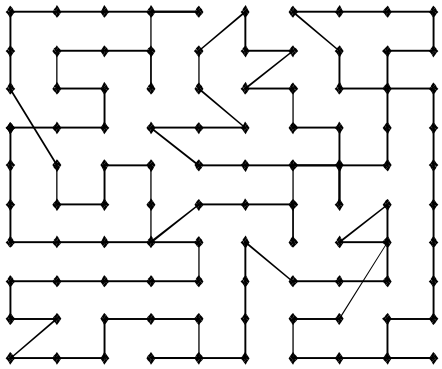
f_{opt}=462, Epoch=0



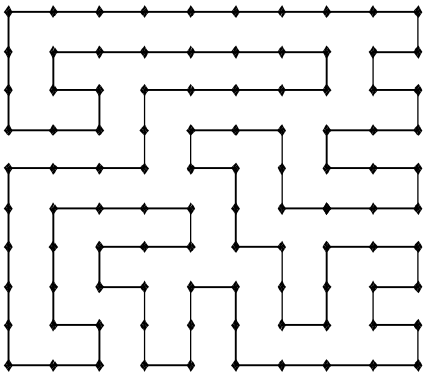
f_{opt}=298, Epoch 500



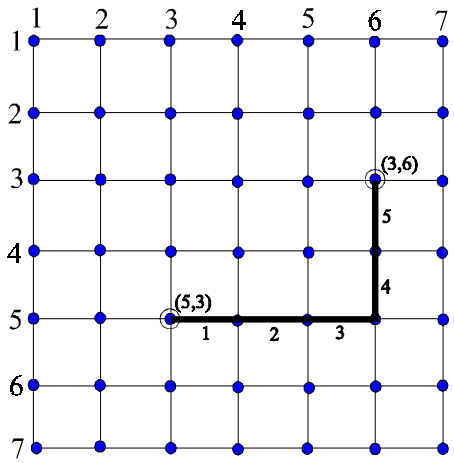
f_{opt}=164, Epoch=1000



f_{opt}=124, Epoch=2000



f_{opt}=100, Epoch=5000

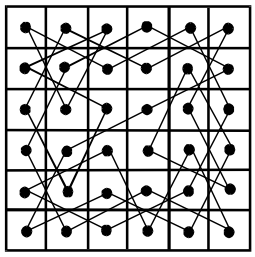


Ortogonálna mriežka bodov typu N×N, pričom vzdialenosť medzi jednotlivými bodmi sa počíta pomocou Hammingovej (L_1) vzdialenosti. Pre tento špeciálny prípad optimálna vzdialenosť je

$$f_{opt} = \begin{cases} N^2 \,(\text{pre párne } N) \\ \\ N^2 + 1 \,(\text{pre nepárne } N) \end{cases}$$

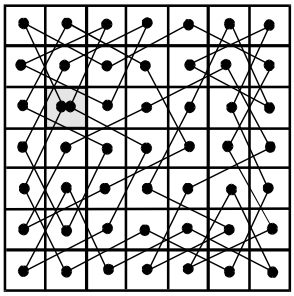
Pohyb koňom po šachovnici

Chessboard 6x6

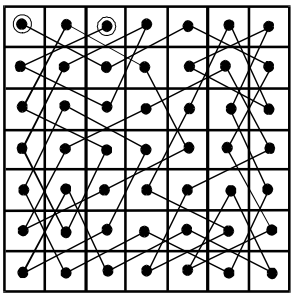


A

Chessboard 7x7

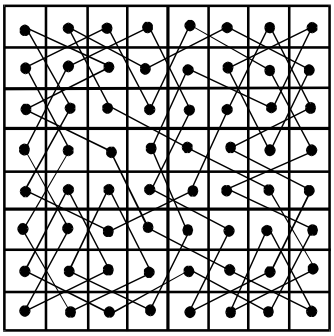


B

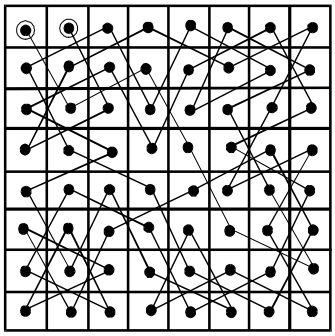


C

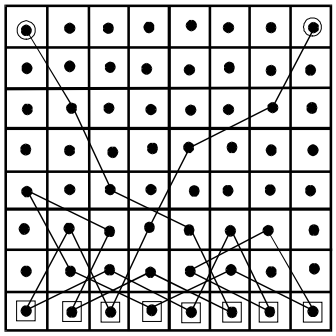
Chessboard 8x8



D



E



F