Základy umělé inteligence

Evoluční výpočetní techniky, Genetický algoritmus, Genetické programování

Ing. Tomáš Řehořek

Computational Intelligence Group (CIG),
Katedra teoretické informatiky (KTI),
Fakulta informačních technologií (FIT),
České vysoké učení technické v Praze (ČVUT)

BI-ZUM, LS 2016/17, 4. přednáška

https://edux.fit.cvut.cz/courses/BI-ZUM/

Minulá přednáška

- Algoritmy iterativní optimalizace
  - Hill climbing,
  - Simulované žíhání,
  - Tabu prohledávání,
  - úvod do **populačních metod**,
    - ★ metod pracujících s více kandidujícími řešeními, která jsou v interakci

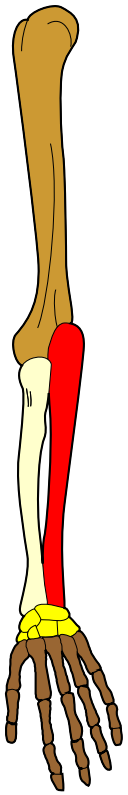
- Dnešní přednáška: **Evoluční výpočetní techniky**
  - **Genetický algoritmus**,
  - **Genetické programování**

Evoluční výpočetní techniky

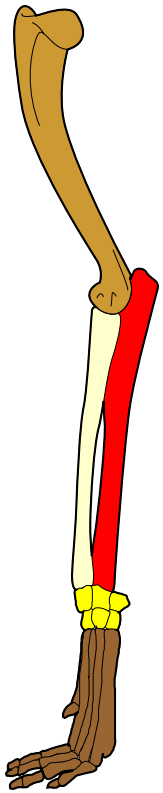
- rodina metod stochastické populační iterativní optimalizace
- inspirovány evoluční biologií, zejm. pak:
  - genetickou dědičností (J.G.Mendel),
  - přirozeným výběrem, přežití silnějšího (Ch.Darwin),
- fungují na principu „šlechtění“ populace kandidujících řešení,
- vysoce kvalitní řešení jsou **selekcí** vybrána k **reprodukci**, během níž může nastat:
  - **křížení** – vzájemná výměna genetické informace mezi řešeními,
  - **mutace** – náhodné nebo řízené odchylky od rodičů

Evoluční výpočetní techniky

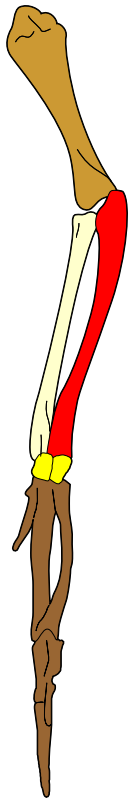
Příklad efektu mutací z evoluční biologie kmenu strunatců:



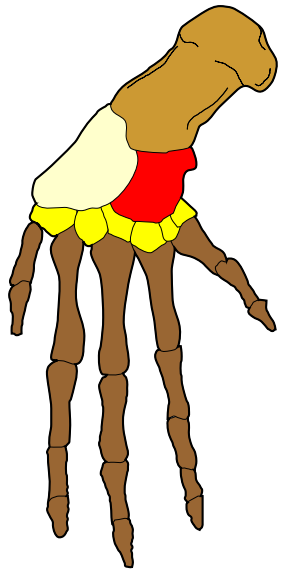
Člověk



Pes



Pták



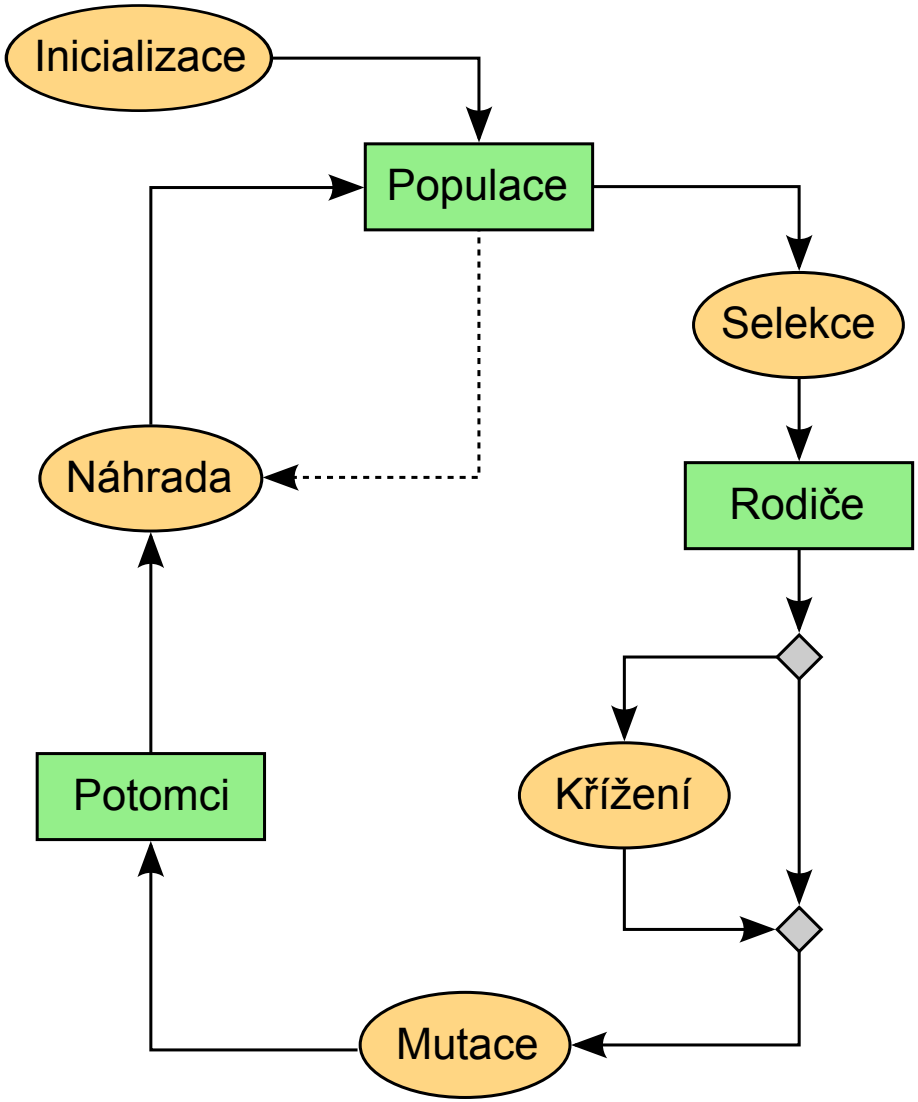
Velryba

Evoluční výpočetní techniky

Co všechno můžeme šlechtit?

- Počítačové programy, subrutiny,
- Tvary závodních aut, letadlových trysek,
- Rozvrh výuky vzhledem k časovým omezením místností/vyučujících,
- Analogové nebo digitální elektrické obvody, trasy plošných spojů,
- Herní inteligenci,
- Plány pro rozvoz zboží,
- Umělé neuronové sítě,
- Chování robotů, umělý život,
- Obrázky, „umělecká“ díla,
- . . . a mnoho dalších

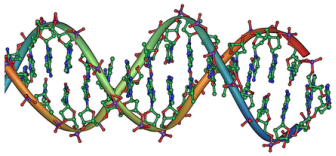
Evoluční výpočetní techniky: Obecné schéma

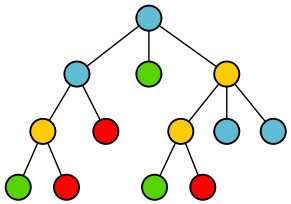


Evoluční výpočetní techniky: Pojmosloví

Genotyp (angl. *genotype*)

- reprezentace řešení používaná evolučním algoritmem,
- např. binární řetězce, vektory z $\mathbb{R}^n$, stromy,
- „gramatika“ řešení

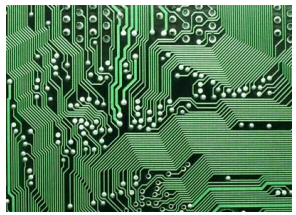




Fenotyp (angl. *phenotype*)

- „sémantika“ řešení

```
...
14: P1=Q1^(1/2.31);
15: P2=Q2^(3.25);
16: goto 31
17: A1=0.23478
18: A2=9.2155
19: if(P2-A1>A2) goto 15;
20: A1*=A2;
...
```



Evoluční výpočetní techniky: Pojmosloví

Fitness

- označení pro **kriteriální funkci** optimalizačního problému,
- vyjadřuje míru adaptace kandidujícího řešení na dané prostředí

Jedinec (individuál)

- označení pro kandidující řešení,
- někdy chápán jako dvojice $(x, f(x))$, kde x je genotyp a $f(x)$ je fitness tohoto genotypu

Populace

- množina šlechtěných jedinců

Generace

- čítač hlavních cyklů evolučního algoritmu,
  - 0. generace – populace byla vygenerována operátorem inicializace,
  - n. generace – populace prošla n cykly selekce, reprodukce a náhrady

Genetické operátory

Inicializace

- Vytvoření počáteční populace

Selekce

- výběr jedinců z populace na základě jejich fitness,
- výslednou množinu vybraných jedinců nazýváme **rodiče**

Reprodukce

- proces tvorby potomků z rodičů,
- typicky probíhá **křížením**, resp. **mutací** rodičů,
- výslednou množinu jedinců nazýváme **potomstvo** (angl. *offspring*)

Genetické operátory

Mutace

- drobná změna genotypu jedince

Křížení (rekombinace, angl. *crossover, recombination*)

- vzájemná výměna informace mezi jedinci,
- konstrukce nového genotypu ze dvou a více jiných

Náhrada

- náhrada (některých nebo všech) jedinců v původní populaci potomky

Evoluční výpočetní techniky: Historie

- Evoluční algoritmy vynalezeny nezávisle několikrát na sobě
  - mnoho rysů mají společných – typický příklad konvergující evoluce :-)
- 1930 – S. Wright: vztah genotypu a fenotypu
  - geny živočichů se shlukují do druhů majících vysokou fitness
- 60.–70. léta
  - **Evoluční programování** – I. Richenberg, H.P.Schwefel, Berlín, Německo,
  - **Evoluční strategie** – L.J. Fogel, California, USA,
  - **Genetický algoritmus** – J. Holland, Michigan, USA,
- přelom 80. a 90. let
  - výzkumníci se setkávají a shodují se na ***Evolutionary computation*** coby názvu oboru
    - ★ v přednášce budeme dále používat pojem „evoluční algoritmus“
- 90. léta
  - **Genetické programování** – J.R.Koza,
  - **Diferenciální evoluce** – R.Storn, K.Price

Genetický algoritmus (GA)

- Jeden z nejúspěšnějších a nejznámějších evolučních algoritmů,
- J.Holland, University of Michigan, 70. léta,
- od základu navržen jako univerzální „black-box solver“ optimalizující binární řetězce (vektory z $\{0,1\}^n$)
  - přirozené kódování v informatice,
  - tyto řetězce nazýváme **chromozomy**,
- chromozomy mají pevnou délku n a algoritmus řeší optimalizační problém

$$\underset{\mathbf{x}\in\{0,1\}^n}{\text{maximize}} \quad f(\mathbf{x})$$

Algorithm 1 Genetic Algorithm (GA)

$\mathcal{P} \leftarrow \{\}$
for $i \leftarrow 1$ **to** μ **do**
  $\mathbf{x} \leftarrow$ random_individual()
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{(\mathbf{x}, f(\mathbf{x}))\}$
end for
repeat
  $\mathcal{O} \leftarrow \{\}$
  for $i \leftarrow 1$ **to** $\frac{\mu}{2}$ **do**
    $\mathbf{p_1} \leftarrow \text{selection}(\mathcal{P})$
    $\mathbf{p_2} \leftarrow \text{selection}(\mathcal{P})$
    $(\mathbf{o_1}, \mathbf{o_2}) \leftarrow \text{crossover}(\mathbf{p_1}, \mathbf{p_2})$
    $\tilde{\mathbf{o}}_1 \leftarrow \text{mutate}(\mathbf{o_1})$
    $\tilde{\mathbf{o}}_2 \leftarrow \text{mutate}(\mathbf{o_2})$
    $\mathcal{O} \leftarrow \mathcal{O} \cup \{(\tilde{\mathbf{o}}_1, f(\tilde{\mathbf{o}}_1)), (\tilde{\mathbf{o}}_2, f(\tilde{\mathbf{o}}_2))\}$
  end for
  $\mathcal{P} \leftarrow \mathcal{O}$
until termination condition is met

Genetický algoritmus: Operátory inicializace

Neinformovaná inicializace

- nejčastěji používáná, univerzální,
- vygeneruje populaci zcela náhodných birních vektorů
  - pro každý bit hod mincí,

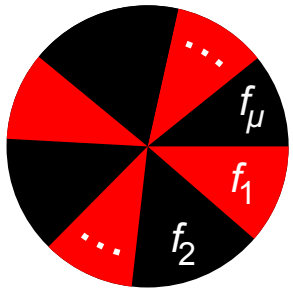
Informovaná inicializace

- vychýlení počáteční populace směrem ke „slibným“ oblastem stavového prostoru,
  - pomocí nějaké jednoduché heuristiky,
  - recyklace jedinců z předchozích běhů algoritmu,
- hrozí nebezpečí, že nenávratně umístí celou populaci do lokálně optimální oblasti, z níž nepůjde vyváznout mutací ani křížením

Operátory selekce

Mnoho možností, nejčastěji:

- **Ruletová selekce** (angl. *Roulette Wheel selection*)
  - Pravděpodobnost výběru jedince je přímo úměrná jeho fitness

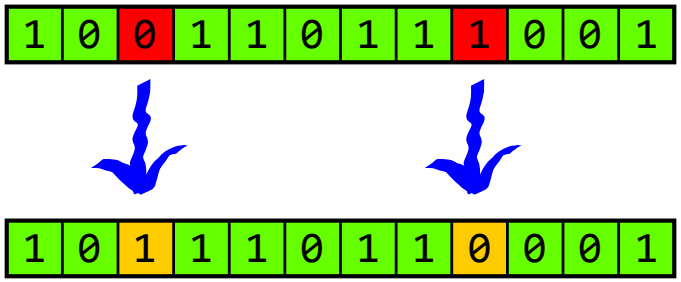


$$P_i = \frac{f_i}{\sum_{j=1}^{\mu} f_j}$$

- **Turnajové selekce** (angl. *Tournament selection*)
  - Náhodně vylosujeme k jedinců a vybereme toho s největší fitness
  - Není závislá na konkrétních hodnotách, kterých fitness nabývá

Genetický algoritmus: Operátor mutace

- Nejčastěji používáme tzv. **bit-flip** mutaci – inverzi několika náhodných bitů v chromozomu



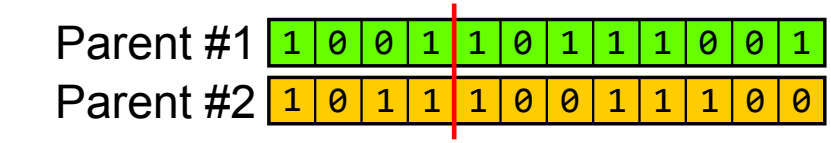
Algorithm 2 bitflip_mutation($\mathbf{x}$)

for $i \leftarrow 1$ **to** n **do**
  if random($\langle 0, 1 \rangle$) $< p_m$ **then**
    $\mathbf{x}[i] \leftarrow \neg \mathbf{x}[i]$
  end if
end for

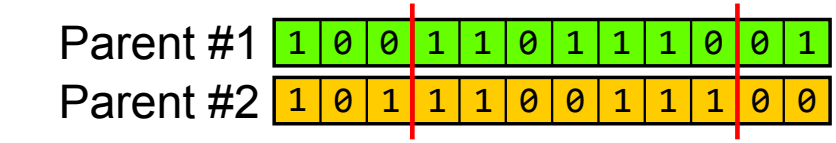
- pravděpodobnost mutace bitu p_m je typicky malé číslo, řádově 10^{-2}

Genetický algoritmus: Operátory křížení

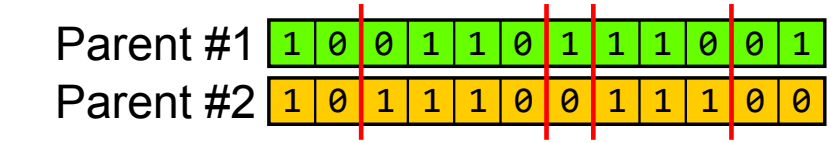
- V úvahu připadá několik různých operátorů, nejčastěji však:



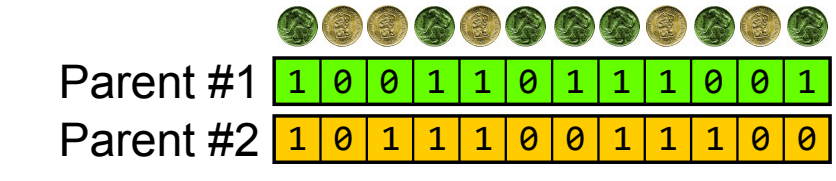
Jednobodové křížení
(One-Point Crossover)



Dvoubodové křížení
(Two-Point Crossover)



***n*-bodové křížení**
(*n*-Point Crossover)



Uniformní křížení
(Uniform Crossover)

- někdy se křížení zcela vynechává (zejm. není-li vhodné pro dané kódování problému), šlechtění pak probíhá pouze mutací,
- připadají v úvahu i křížící operátory specializované pro daný typ problému

Genetický algoritmus: Příklad pro Knapsack problem

Problém batohu je dán následovně:

- máme k dispozici batoh o kapacitě W,
- je dána množina předmětů $T = \{t_1, t_2, \ldots, t_n\}$, kde každý předmět má svou cenu $v(t_i)$ a hmotnost $w(t_i)$.

Naším úkolem je nalézt podmnožinu $Z \subseteq T$ o maximální možné celkové ceně takovou, že se vejde do batohu, neboli

$$\arg\max_{Z \subseteq T} \sum_{z \in Z} v(z) \quad \text{t.ž.} \quad \sum_{z \in Z} w(z) \leq W.$$

Tento problém je tzv. **NP-úplný**, tj. není znám žádný algoritmus, který by nalezl jeho optimální řešení v polynomiálním čase.
Alespoň přibližné řešení nám může pomoci nalézt genetický algoritmus.

Genetický algoritmus: Příklad pro Knapsack problem

Řešení problému **zakódujeme** pomocí **binárního vektoru**, jehož i-tá složka určuje, zda do batohu máme vložit předmět x_i.

Sestavíme **fitness funkci**, která genetickému algoritmu sdělí, jak je dané řešení dobré.

V nejtriviálnějším případě:

$$f(\mathbf{x}) = \begin{cases} \sum_{z \in \{t_j | x_j = 1\}} v(z), & \text{pokud } \sum_{z \in \{t_j | x_j = 1\}} w(z) \leq W, \\ 0, & \text{pokud } \sum_{z \in \{t_j | x_j = 1\}} w(z) > W \end{cases}$$

Jak to dělat lépe? $\rightarrow$ Cvičení :-)

Úlohu nyní můžeme předat k vyřešení genetickému algoritmu

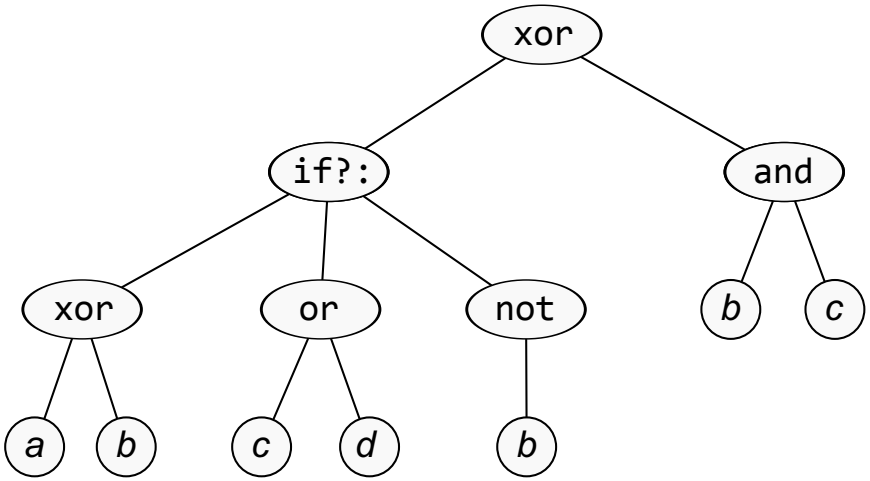
Genetické programování (GP)

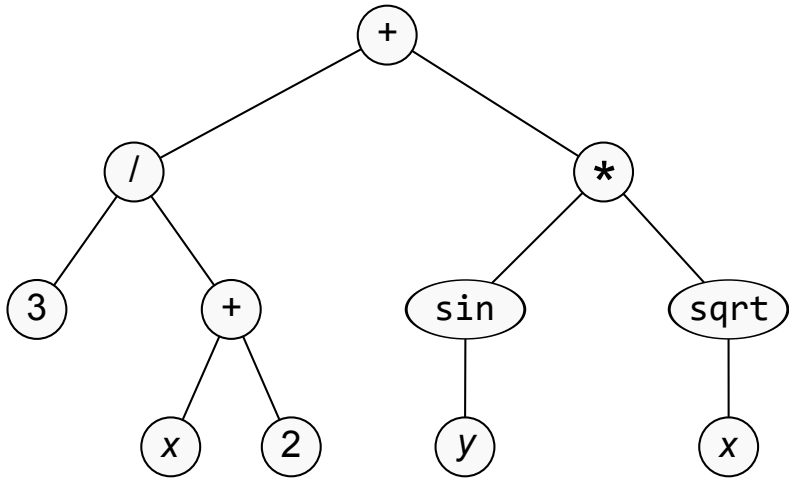
- Další velmi prominentní paradigma v evolučních algoritmech,
- J.R.Koza, Standford University, počátek 90. let
  - rozšíření a propagace původní práce N.L.Cramera
- genotypem jsou orientované kořenové **stromy** (nikoli lineární chromozomy fixní jako u GA),
- je flexibilnější – struktura a velikost řešení jsou rovněž předmětem evoluce,
- v původní variantě byly šlechtěny tzv. S-expressions jazyka LISP, díky dobré propagaci dnes existuje ohromné množství aplikačních domén

Genetické programování: Stromový genotyp

Stromy v GP sestávají z:

- **terminálů** (listů) – *T*
  - vstupy vyvíjených programů, nezávislé proměnné
- **funkcí** (vnitřních uzlů) – *F*
  - např. aritmetické operace, algebraické funkce, logické funkce atd.

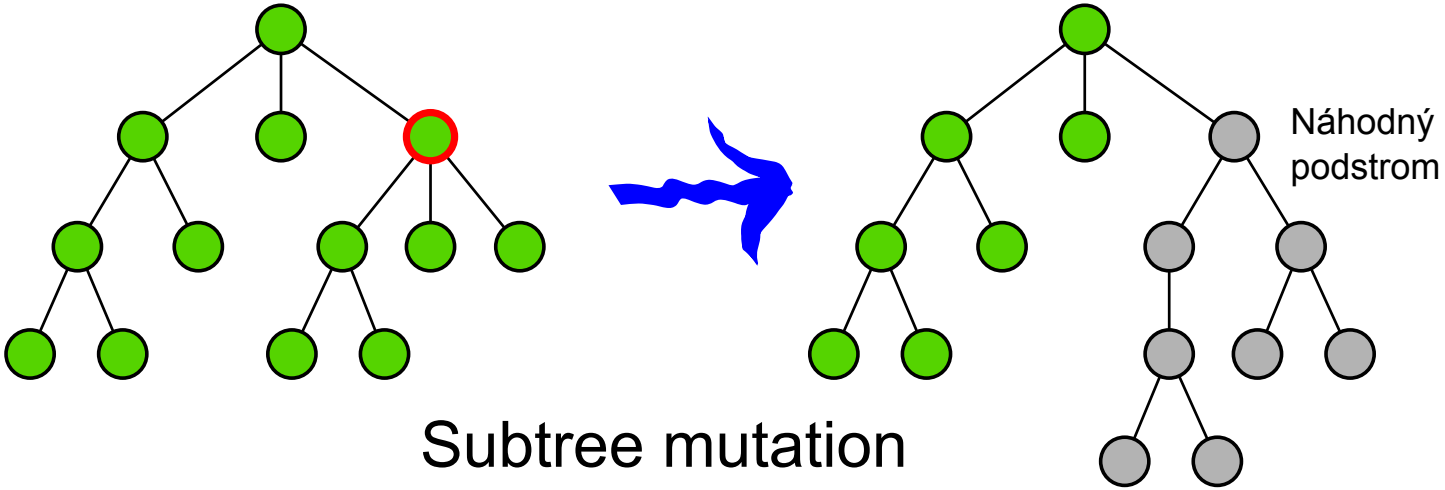




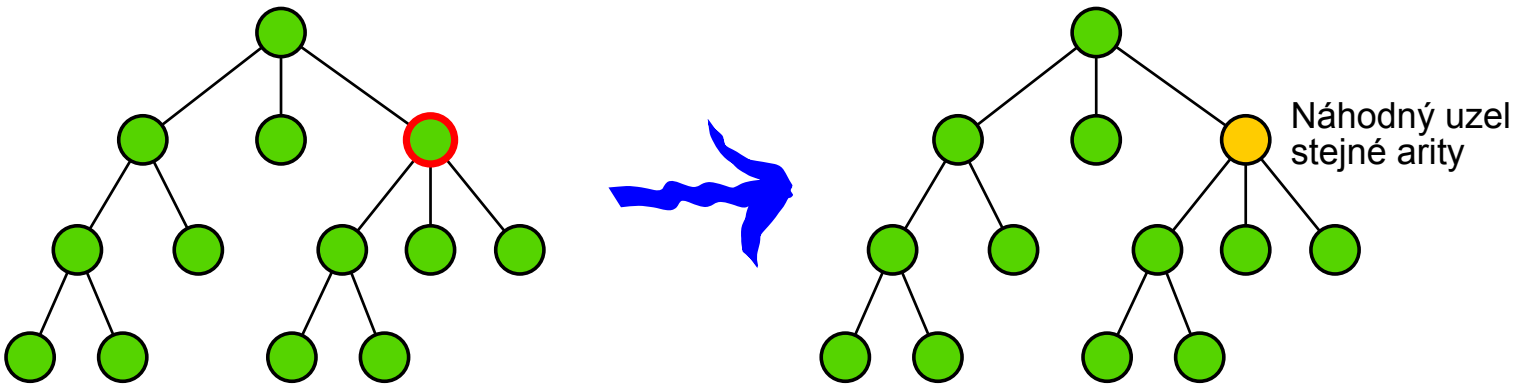
Genetické programování: Operátory inicializace

- Inicializace spočívá ve vygenerování náhodného stromu
  - poněkud složitější, než generování náhodných binárních vektorů
- Existuje více metod:
  - **GROW**
    - ★ větve stromu mají hloubku $\leq D_{max}$,
    - ★ uzly v hloubce $d < D_{max}$ náhodně vybrány z $F \cup T$,
    - ★ uzly v hloubce $d = D_{max}$ náhodně vybrány z T,
  - **FULL**
    - ★ větve stromu mají hloubku přesně D_{max},
    - ★ uzly v hloubce $d < D_{max}$ náhodně vybrány z F,
    - ★ uzly v hloubce $d = D_{max}$ náhodně vybrány z T,
  - **PCT1** (Probabilistic Tree-Creation)
    - ★ náhodně generuje strom se střední hodnotou počtu uzlů E_{tree},
    - ★ pravděpodobnost vybrání neterminálu je spočítána z arit neterminálů

Genetické programování: Operátory mutace

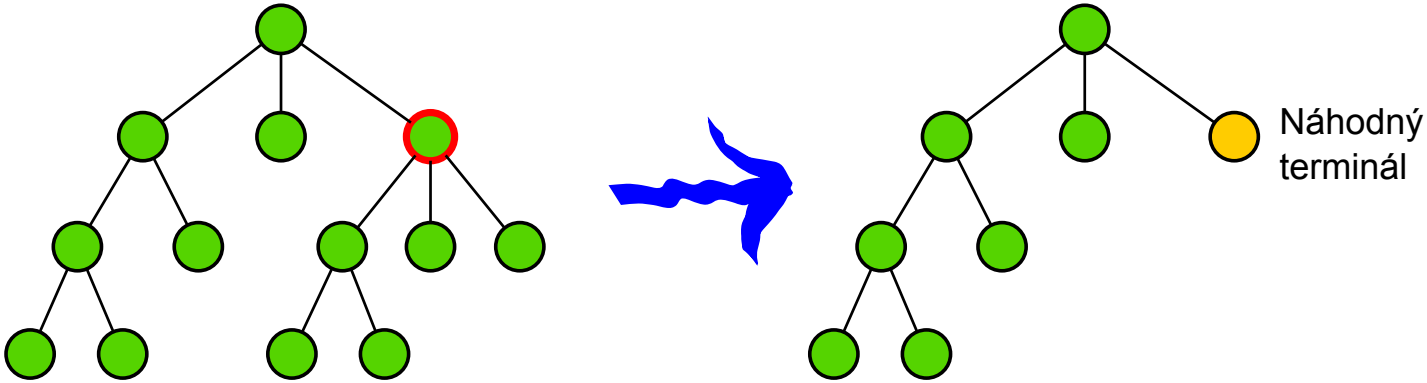


Subtree mutation

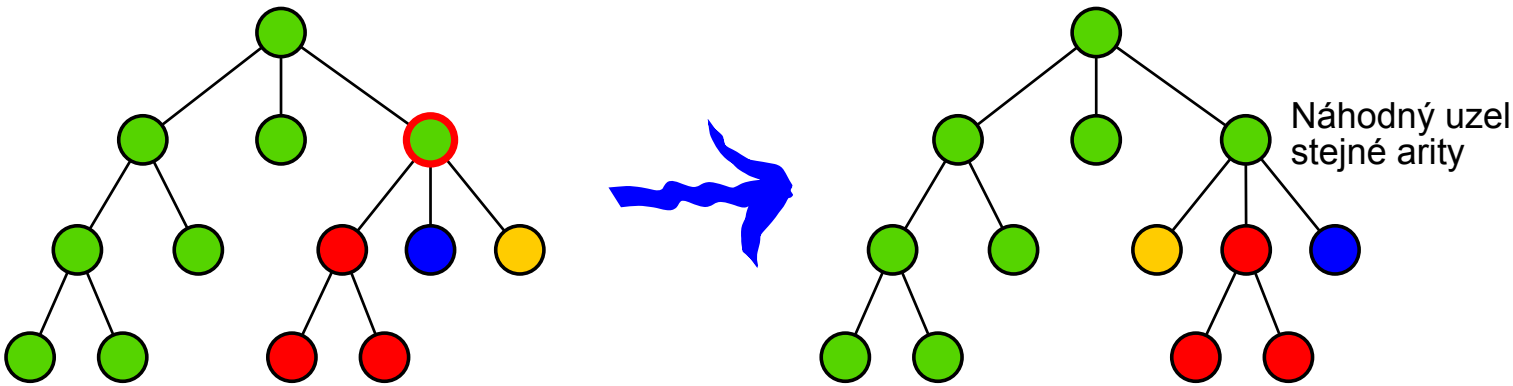


Point mutation

Genetické programování: Operátory mutace



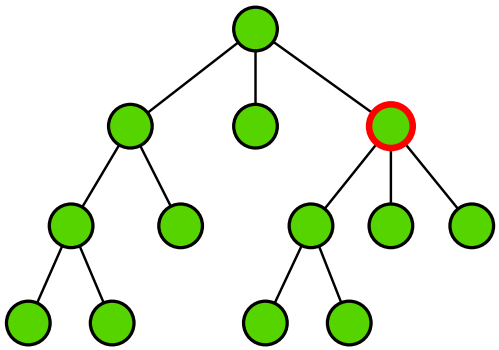
Shrink mutation



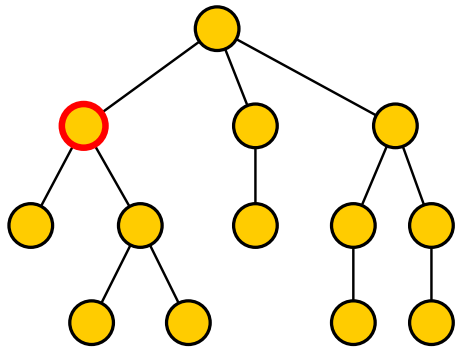
Permutation mutation

Genetické programování: Operátor křížení

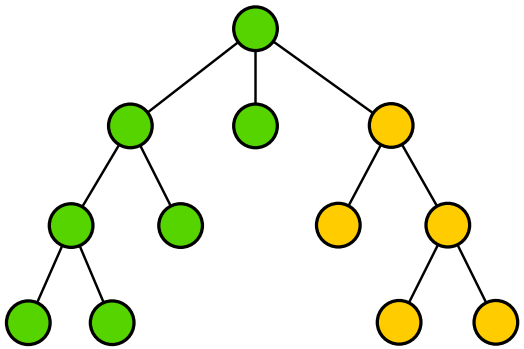
- Nejčastěji se používá výměna dvou náhodně zvolených podstromů:



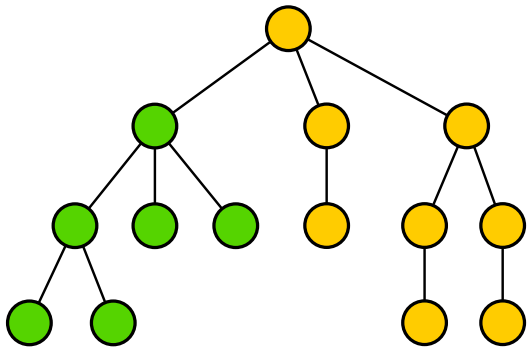
Parent #1



Parent #2



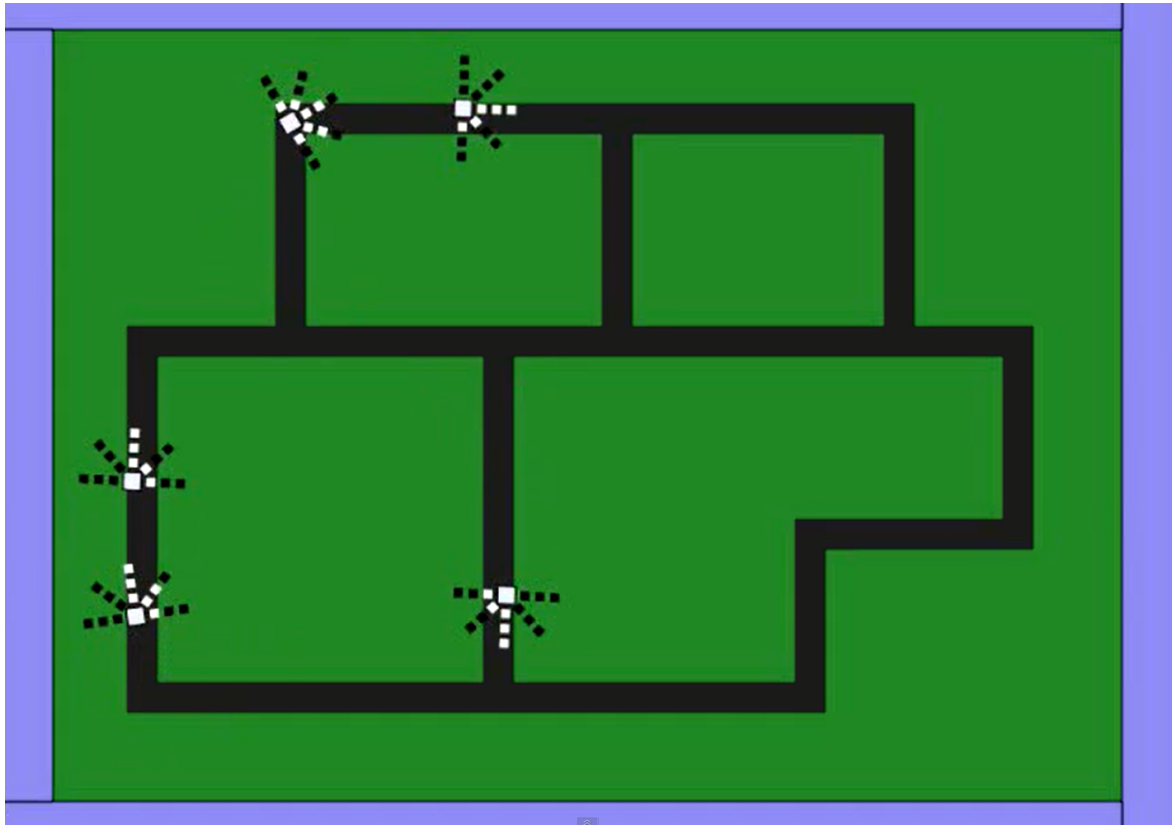
Offspring #1



Offspring #2

Genetické programování: Příklad

Jak pomocí genetického programování vyšlechtit chování robotů, kteří jezdí po silnicích a nesráží se?



http://www.youtube.com/watch?v=lmPJeKRs8gE

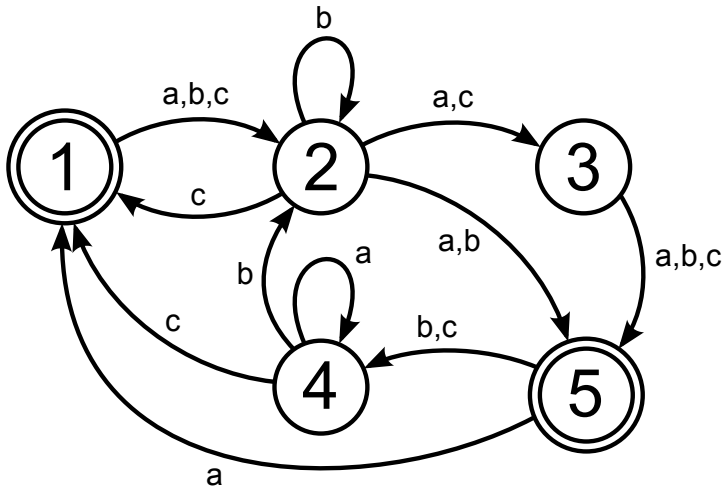
Evoluční programování (EP)

- L.J.Fogel, University of California, 1960,
- vynalezeno ještě před Genetickým algoritmem J.Hollanda,
- snaha badatelů z počátků informatiky přidat adaptaci do rigidních systémů,
- Fogel se zabýval multiagentními systémy, kde chování agentů bylo vyjádřeno **stavovými automaty**,
- cílem bylo „vyšlechtění umělé inteligence“ – odvození patřičné akce vzhledem k danému stavu prostředí,
- používá pouze operátory mutace, chybí operátor křížení

Evoluční programování (EP)

Stavový automat je popsán:

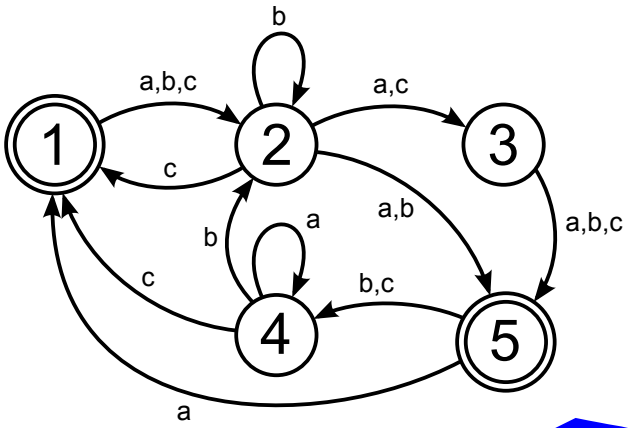
- počátečním a cílovým stavem,
- množinou stavů,
- tabulkou přechodů



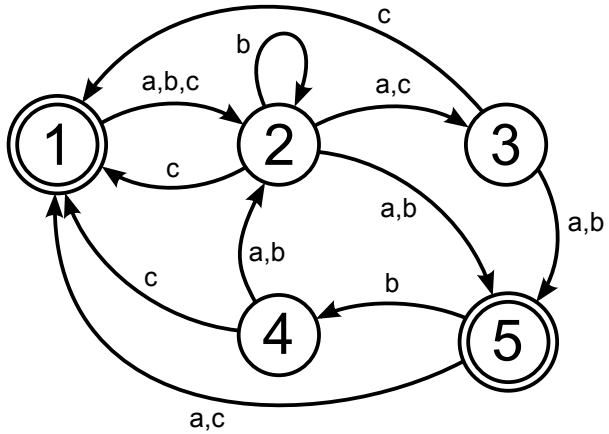
	a	b	c
1	2	2	2
2	3	2	3
3	5	5	5
4	4	2	1
5	1	4	4

Toto vše může podléhat mutaci.

Evoluční programování: Příklad mutace



	a	b	c
1	2	2	2
2	3	2	3
3	5	5	5
4	4	2	1
5	1	4	4



	a	b	c
1	2	2	2
2	3	2	3
3	5	5	1
4	2	2	1
5	1	4	1

Evoluční strategie (ES)

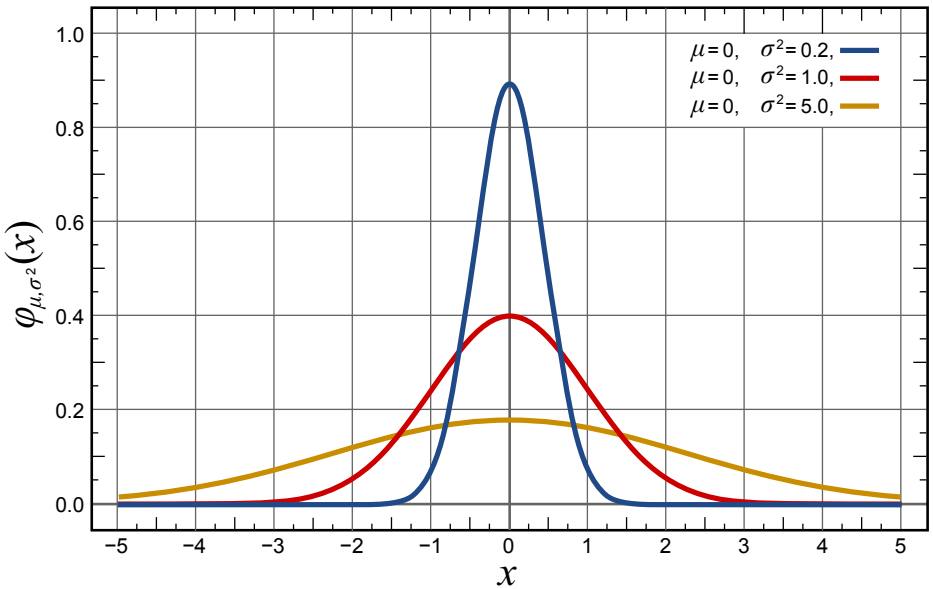
- I. Rechenberg, H.P. Schwefel, Technische Universität Berlin, 60. léta,
- další z nezávislých výzkumů směřujícím k evolučním algoritmů,
- genotypem byly **vektory reálných čísel** z $\mathbb{R}^n$
  - důležitost této domény jsme ukázali v minulé přednášce,

Evoluční strategie – Operátory selekce

- Původní verze: $(1 + \lambda)$-ES,
  - 1 rodič vyprodukuje λ potomků,
  - nejlepší z těchto potomků je pak rodičem celé následující generace,
  - de-facto odpovídá klasickému Hill climbigu!
- Pokročilá verze: $(\mu + \lambda)$-ES a (μ, λ)-ES
  - μ rodičů vyprodukuje λ potomků,
  - $(\mu + \lambda)$-ES
    - ★ z těchto $\mu + \lambda$ potomků je do další generace vybráno μ nejlepších,
  - (μ, λ)-ES
    - ★ do další generace je vybráno μ nejlepších pouze z potomků

Evoluční strategie – Operátory mutace

- „Malá změna“ jednotlivých složek *n*-rozměrného vektoru,
- nejčastěji se používá tzv. **gaussovská** mutace
  - přičtení náhodného čísla z normálního rozdělení,
  - nejčastěji je změna malá, občas větší, ve výjimečných případech velká



Evoluční strategie – Operátory mutace

Pokročilé verze zavádí tzv. **endogenní strategické parametry**

- forma „metaevoluce“,
- jedinec kromě samotného vektoru ukládá ještě vektor rozptylů gaussovských rozdělení pro jednotlivé dimenze,
  - jedinec $C_i = (G_i, S_i)$,
  - G_i jsou genetické parametry – informace o řešení,
  - S_i jsou standardní odchylky – informace, jak má být řešení mutováno v dané dimenzi do dalších generací,
- standardní odchylky se mění v závislosti na úspěšnosti mutace
  - pravidlo „$1/5$“: úspěšnost mutace by měla být přibližně 20 %,
  - pokud je úspěšnost větší než $1/5$, zmenši rozptyl,
  - pokud je úspěšnost menší než $1/5$, zvětši rozptyl

Evoluční strategie – Operátory křížení

- Původní verze používala pouze mutaci, žádné křížení,
- moderní odvozeniny ES operátory křížení zavádí,
- křížení je zásadně uniformní (po složkách),
- dvě nejdůležitější varianty:
  - **diskrétní** – daná složka vektoru se překopíruje vždy od jednoho z rodičů,
  - **aritmetická** – průměr z hodnot rodičů,
- přibývá další parametr, ρ, který určuje počet rodičů, kteří se podílí na tvorbě potomka,
- označujeme jako $(\mu/\rho \overset{+}{,} \lambda)$-ES:
  - $\rho = 1 \ldots$ standardní $(\mu \overset{+}{,} \lambda)$-ES,
  - $\rho = 2 \ldots$ křížení podobné GA – 2 rodiče,
  - $\rho = \mu \ldots$ extrémní případ, kdy se na tvorbě potomka podílí celá populace

Exploatace vs. explorace

- Evoluční algoritmy jsou kompromisem mezi dvěma principy:
  - **Exploatace** – „šplhání do kopce“, zkoumání bezprostředního okolí
    - ★ selekce, (křížení),
    - ★ „lokální složka“ evolučního algoritmu,
  - **Explorace** – „náhodné procházky prostorem“ bránící uváznutí v lokálním optimu
    - ★ mutace, (křížení),
    - ★ „globální složka“ evolučního algoritmu,
- Poměr mezi explorací a exploatací lze řídit pomocí různých parametrů:
  - **selekční tlak** – jak moc preferovat dobrá řešení před špatnými?
  - **míra mutace** – jak moc velké mají být náhodné změny prováděné při reprodukci?

Exploatace vs. explorace

Míra mutace?

- **příliš velká**
  - poškozuje užitečnou informaci v genotypech,
- **příliš malá**
  - brání náhodnému zlepšování řešení,
- **dynamická**
  - řízená úspěšností (ES),
  - řízená časem, např. v kombinaci s principem simulovaného žíhání
    - ★ v počátečních generacích velké mutace,
    - ★ v pozdějších generacích malé mutace

Vše souvisí s problémem **předčasné konvergence** populace.

Udržení diverzity

Největší hrozbou pro evoluční algoritmus je tzv. **předčasná konvergence**

- celá populace začne být stejná, ovládnutá několika typy jedinců,
  - mutace již není schopna tato řešení zlepšit, změna by musela být příliš velká,
  - křížení nefunguje, neboť jedinci jsou příliš rozdílní
- přesně odpovídá uvíznutí v lokálním optimu

Biologická evoluce?

- masivně paralelní, konvergence v ní neexistuje,
- dělí populaci na subpopulace – různé živočišné druhy,
- každý živočišný druh představuje populaci kandidujících řešení, která se všechna nachází poblíž společného lokálního optima
- inspirace pro evoluční algoritmy – tzv. **niching**

Niching

- rodina metod, které přímo nebo nepřímo berou v potaz podobnost genotypů,
- příliš nepodobní jedinci se nekříží anebo mezi sebou nesoupeří,
- nejdůležitější formy nichingu:
  - **ostrovní model**,
  - **fitness sharing**,
  - **deterministic crowding**,
  - **novelty search**.

Metody nichingu

Ostrovní model

- evoluce probíhá na různých ostrovech,
- každý ostrov sleduje svou vlastní evoluční trajektorii ve stavovém prostoru,
- k výměně informace mezi ostrovy dochází pouze zřídka (přelet mezi ostrovy)

Fitness sharing

- příliš podobní jedinci se dělí o společnou fitness hodnotu,
- hlavní myšlenka: na jednotlivých vrcholcích ve stavovém prostoru je jen omezený počet zdrojů

Metody nichingu

Deterministic crowding

- potomci soutěží o přežití s podobnějším z rodičů
- má-li potomek vyšší fitness než rodič, nahradí jej,
- má-li potomek nižší fitness než rodič, přežije,
- tento mechanismus brání tomu, aby se příliš dobrý typ jedince přemnožil a obsadil celou populaci

Novelty search

- niching mezi fenotypy,
- bonifikace jedinců přinášejících novelty („něco nového“),
- inspirace biologickou fitness
  - charakteristiky jedinců s vysokou fitness nejsou předem známy,
  - evoluční výhodu mají jedinci nacházející nečekaná, nová a předem nezmámá využití pro již existující fenotyp

Zajímavé odkazy

Flexible Muscle-Based Locomotion for Bipedal Creatures
`http://vimeo.com/79098420`

Mona Lisa from 1500 characters
`http://www.youtube.com/watch?v=TManzvC9pi8`

15 Real-World Applications of Genetic Algoritms
`http://brainz.org/15-real-world-applications-genetic-algorithms/`