

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

EVOLUČNÍ ALGORITMY

Autor textu:
Ing. Petr Honzík, Ph.D.

Květen 2014

Komplexní inovace studijních programů a zvyšování kvality výuky na FEKT VUT v Brně
OPVK CZ.1.07/2.2.00/28.0193



evropský
sociální
fond v ČR



EVROPSKÁ UNIE



MINISTERSTVO ŠKOLSTVÍ,
MLÁDEŽE A TĚLOVÝCHOVY



OP Vzdělávání
pro konkurenceschopnost

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Obsah přednášky

1. Optimalizační algoritmy
2. Genetické algoritmy
3. Genetické programování
4. Diferenciální evoluce

Optimalizace

- Snaha o nalezení minimální nebo maximální hodnoty tzv. **účelové nebo cílové funkce**
- **Geometrický problém** v prostoru o $D+1$ dimenzích, kde D je počet parametrů účelové funkce a zbývající rozměr je výstupní hodnota účelové funkce, jejíž extrémní hodnota je hledána
- Dva základní **typy kombinatorických problémů**:
 - variační (s opakováním, min/max funkce, ...)
 - permutační (variace bez opakování, obchodní cestující)

Úvod – co dělají GA

- Genetické algoritmy slouží k prohledávání prostoru hypotéz (např. nastavení modelu) napodobováním přirozeného vývoje v přírodě (evoluce) – přežívá přizpůsobenější jedinec
- Zmražená evoluce – nová biologická evoluční teorie
- **Přednosti:**
 - lze řešit libovolně složitý problém bez apriorní znalosti
- **Omezení:**
 - pokaždé nalezeno jiné řešení
 - nemožnost rozpoznat optimum
 - ne vždy dostatečně přesné
 - řada parametrů algoritmu – stupňů volnosti

Historie

- 19. stol. Charles Darwin „O původu druhů“
- 1960 I. Rechenberg – idea GA
- 1975 John Holland „Adaptation in Natural and Artificial Intelligence“ – uvedení GA
- 1989 David Goldberg – obecné využití GA
- 1992 J. R. Koza – genetické programování
- 1996 Zbigniew Michalewicz – vylepšení

Terminologie

- **Operandy**

- gen
- chromozom
- *genom, genotyp, fenotyp*
- *jedinec (rodič, potomek)*
- generace, populace

- **Ostatní**

- adaptace
- evoluční počítání
- genetický algoritmus

- **Operátory**

- kódování
- fitness (kvalita)
- selekce
- křížení
- mutace
- elitismus
- ukončovací podmínka

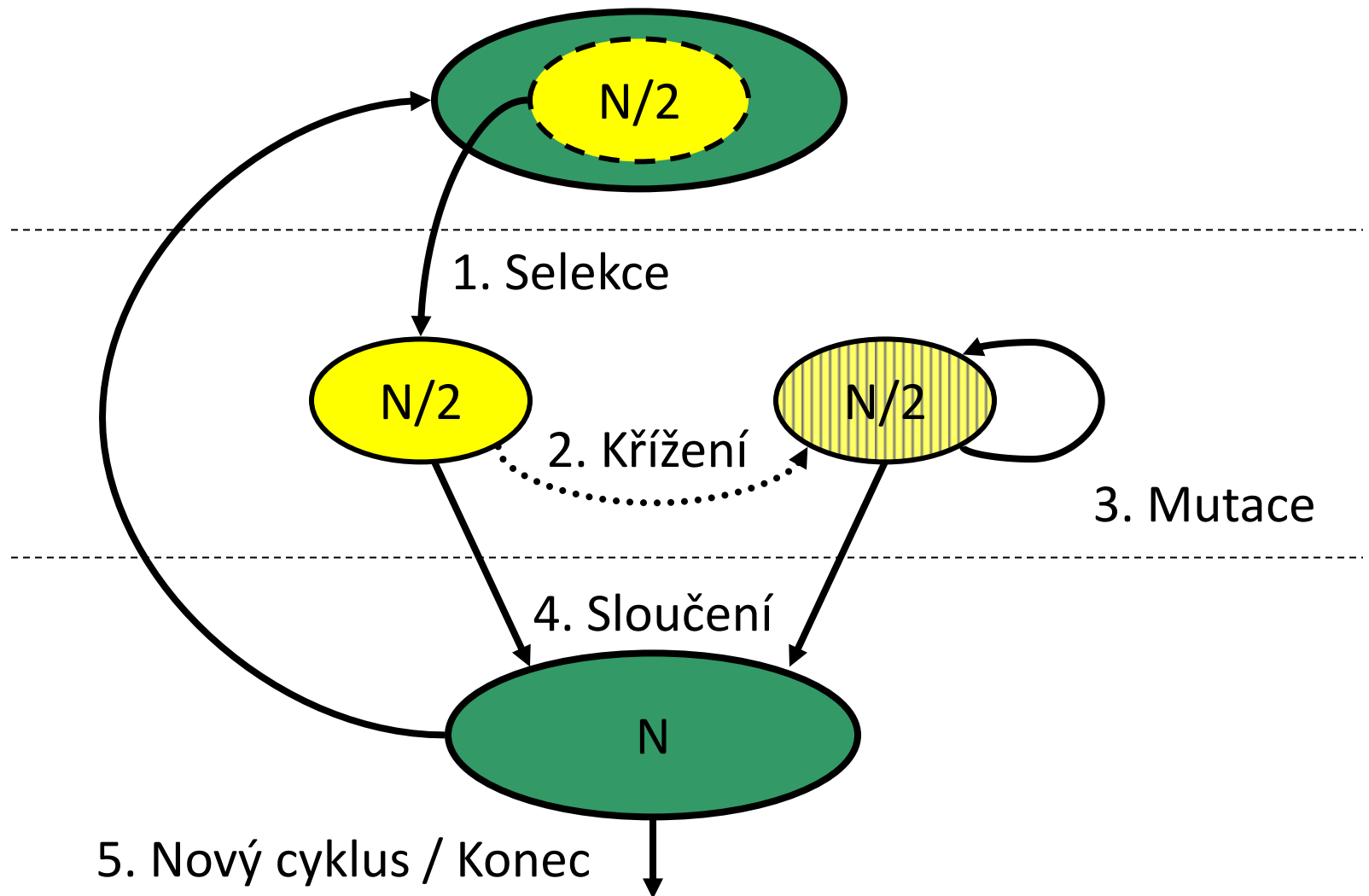
VIS: ? Co je ...? (kromě termínů označených kurzívou)

Princip GA – obdoba evoluce

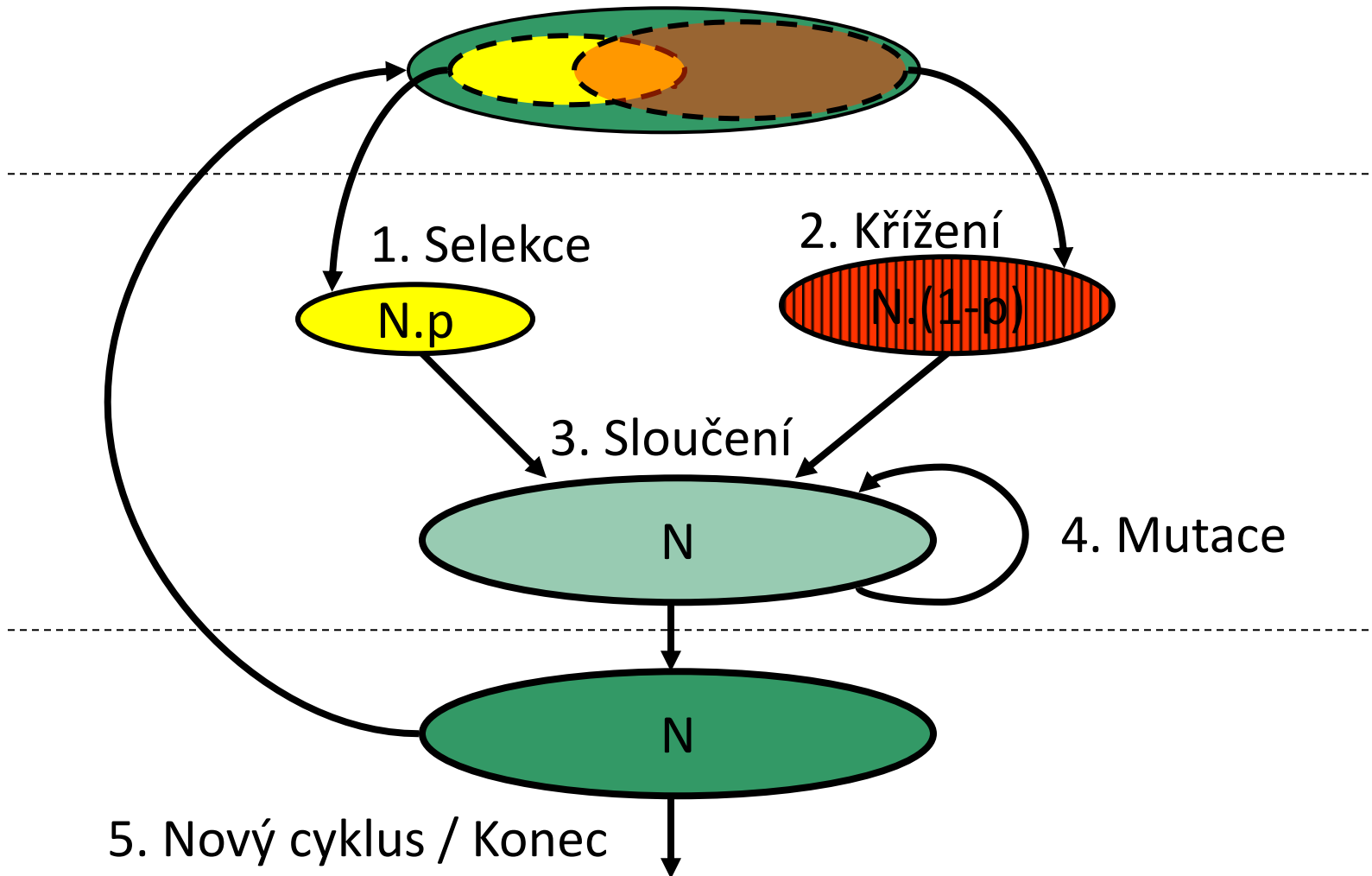
1. vytvoření počáteční generace
2. vyhodnocení počáteční generace
3. selekce
4. křížení, mutace
5. vyhodnocení nové generace
6. ukončovací podmínka
 - nesplněna – návrat do 3. bodu
 - splněna - konec

VIS: ? Napište ideové schéma genetického algoritmu.

Princip GA – varianta I.



Princip GA – varianta II.



Kódování

chromozom = ***kódování*** (skutečné vlastnosti);

- binární kódování

chromozom: | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |

- kódování do reálných čísel

chromozom: | 2,36 | 0,057 | -3600 |

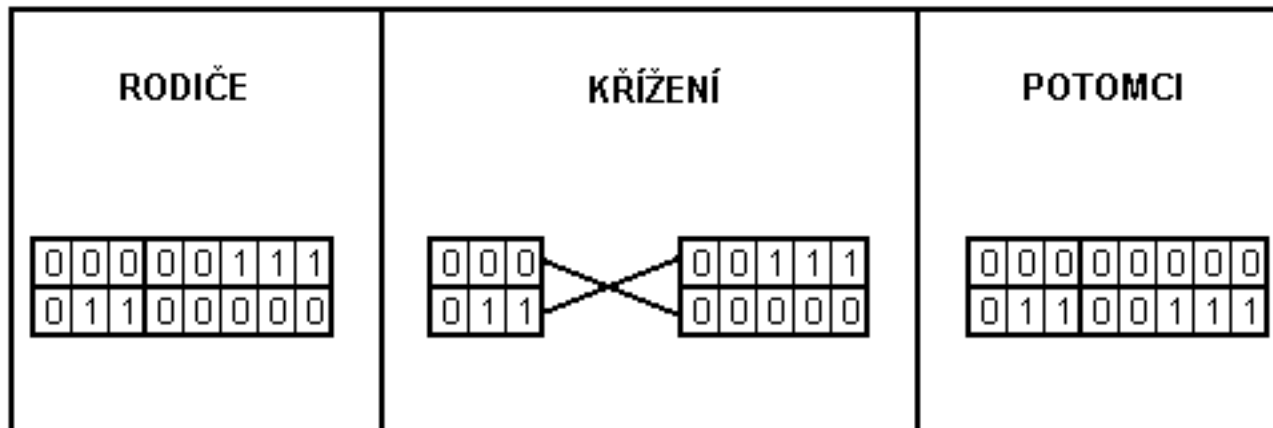
Křížení

jedinec = *křížení* (rodič1,rodič2);

- Základní „evoluční“ operátor GA
- Na základě křížení může ze dvou slabých rodičů vzniknout silný potomek
- Různé typy křížení podle typu kódování chromozomu (bin., real.)

Křížení – binární kódování

8 genů, 7 pozic pro křížení, 3. pozice křížení



Křížení – reálné kódování

- **Simple crossover** [Goldberg 1989]
- Two point crossover [Eshelman, 1989]
- Uniform crossover [Syswerda 1989]
- Flat crossover [Radcliffe 1991]
- Arithmetic crossover [Michalewicz 1992]
- **Convex crossover**
- Fuzzy crossover
- **Blend crossover- α (BLX- α)** [Eshelman, 1993]
- Linear BGA crossover [Schlierkamp-Voosen, 1994]
- Simulated binary crossover (SBX) [Voigt, 1995]
- CIXL1 [Hervás-Martínez, 2003]
- CIXL2 [Ortiz-Boyeret, 2005]
- ...

Křížení – reálné kódování

- **Simple crossover**

- obdoba binárního křížení
- není příliš efektivní

R1:	7,8	41	12,3	13	17
R2:	12	-53	123	4	1

P1:	7,8	41	12,3	4	1
P2:	12	-53	123	13	17

- **Convex crossover**

$$P = p \cdot R_1 + (1 - p) \cdot R_2$$

- kde p je náhodné číslo z intervalu $\langle 0;1 \rangle$, R_1 a R_2 jsou rodiče, P je potomek

Blend Crossover – BLX- α

- Mějme rodiče R_1 a R_2 , pak gen g potomka P na stejné pozici jako u obou rodičů g_1 a g_2 je vygenerován jako náhodné číslo z intervalu v závorce výrazu:

$$g = rand(\min(g_1, g_2) - \alpha \cdot |g_1 - g_2|, \max(g_1, g_2) + \alpha \cdot |g_1 - g_2|)$$

- Empiricky zjištěno, že stabilně dobré výsledky dává algoritmus pro $\alpha=0,5$
- Variantou je metoda definující 3 potomky dle výše uvedeného algoritmu pro extrémny intervalu a průměr. Ze 3 potomků jsou vybráni 2 s největší zdatností

$$P_1 = \frac{R_1 + R_2}{2} \quad P_2 = \frac{3 \cdot R_1 - R_2}{2} \quad P_3 = \frac{-R_1 + 3 \cdot R_2}{2}$$

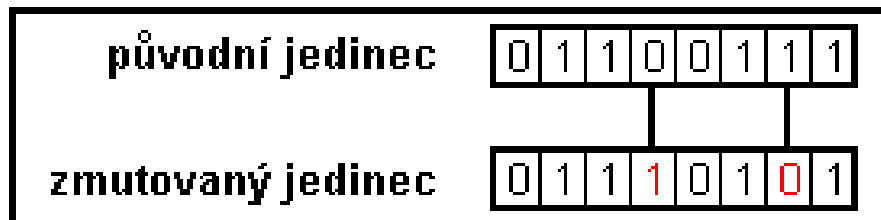
Mutace

jedinec = *mutace* (jedinec);

- dochází k ní zřídka
- změna genu „ladící“ konečnou kvalitu
- dělení na 2 typy podle typu kódování chromozomu

Mutace – binární kódování

- binární kódování



Mutace – reálné kódování

- změna o předem danou hodnotu, o náhodnou hodnotu
- nahrazení náhodným číslem z daného intervalu
- dynamická mutace (Michalewicz), mění svou velikost v čase

- x_i je původní hodnota genu před mutací
- x_i^* je nová hodnota zmutovaného genu
- X_{MAX} je maximální možná hodnota x_i
- X_{MIN} je minimální možná hodnota x_i
- T je maximální počet generací
- r je náhodné číslo z intervalu (0;1)
- B (=5) je parametr nelinearity mutace
- t index generace (maximálně T)

$$x_i^* = \begin{cases} x_i + \Delta(t, X_{MAX} - x_i) \\ \textit{náhodně vyber} \\ x_i - \Delta(t, x_i - X_{MIN}) \end{cases}$$

$$\Delta(t, y) = y \left(1 - r \left(1 - \frac{t}{T} \right)^B \right)$$

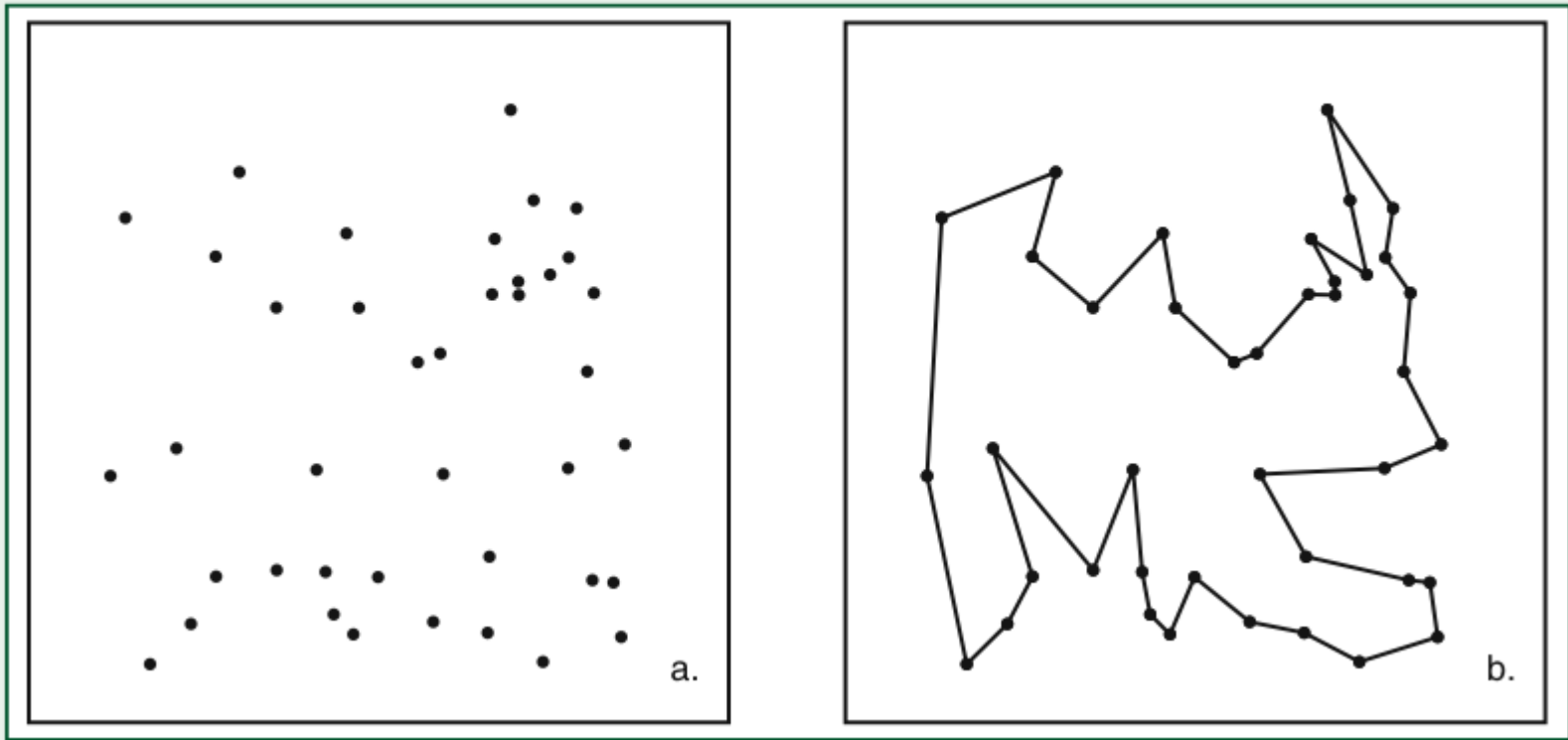
Permutační úlohy

- Existují úlohy, jejichž cílem je dosáhnout optimálního **uspořádání N prvků** na základě nějaké metriky definující vzdálenost mezi dvěma libovolnými prvky.
- Typickou úlohou je **problém obchodního cestujícího** (TSP – traveling salesman problem), cílem je najít v mapě měst co nejkratší trasu začínající a končící ve stejném městě a procházejícím každým dalším městem právě jednou.
- Jedná se o **NP-těžkou** úlohu (not polynomial).
- Počet možných způsobů jak města projít určuje počet **variací bez opakování ze všech N prvků = permutace**, při N městech existuje $(N-1)!$ různých cest.
- Pro tento typ **permutačních úloh** je třeba upravit implementaci křížení i mutace oproti variačním úlohám.

Permutační úlohy - TSP

Figure 1.

Example 40-node TSP (a) and path-following (b) arrays.



Permutační křížení - vybrané algoritmy

- Partially Mapped Crossover (PMX) [Goldberg, 1985]
- **Order Crossover (OX) [Davis, 1985]**
- Edge Crossover (EX) [Whitley, 1989]
- Subtour Exchange Crossover (SXX) [Yamamura, 1992]
- Edge Exchange Crossover (EXX) [Maekawa, 1996]
- Edge Assembly Crossover (EAX) [Nagata, 1997]
- Partition Crossover (PX) [Whitley, 2009]
- Order Based Crossover (OBX)
- **Edge Recombination Crossover (ERX)**
- **Cycle Crossover (CX)**
- Maximal Preservative Crossover (MPX)
- Alternating-Position Crossover (APX)
- Position Based Crossover (PBX)
- Adaptive Goal Guided Recombination (AGGX) [Gog, 2006]
- Best-Worst Recombination (BWX) [Gog, 2007]
- Best Order Crossover (BOX)

Křížení s rekombinací hran (ERX)

- mějme sekvence měst:
1-2-3-4-5-6 a 5-6-3-1-4-2
- to tabulky vypíšu každému městu sousedy v rodičích
- potomek vytvořen postupným výběrem prvků, které mají nejméně sousedů
- po výběru každého prvku jej smažu v tabulce měst i sousedů
- potomek např. 5-6-1-2-3-4

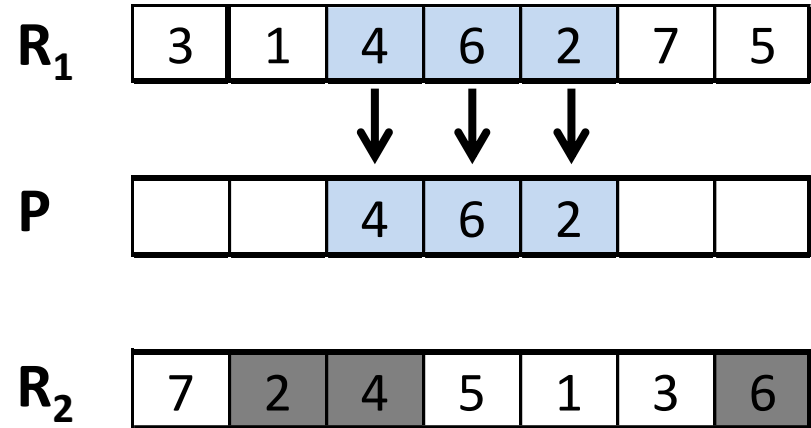
Město	Sousedé	Město	Sousedé
1	2,3,4,6	4	1,2,3,5
2	1,3,4,5	5	2,4,6
3	1,2,4,6	6	1,3,5

Město	Sousedé	Město	Sousedé
1	2,3,4,6	4	1,2,3
2	1,3,4	6	1,3
3	1,2,4,6		

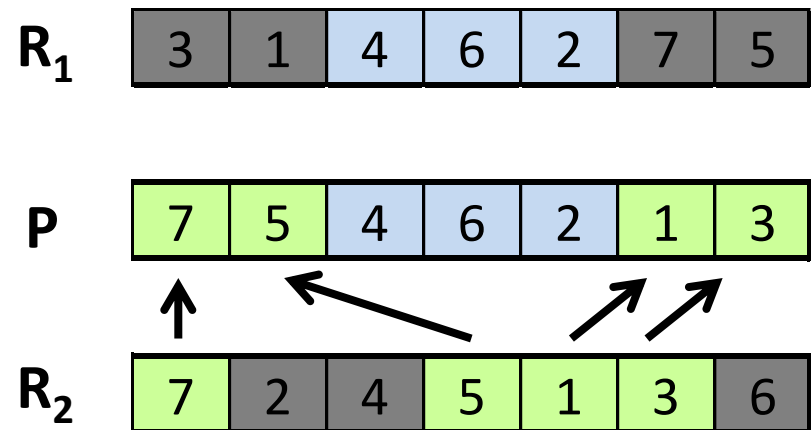
Město	Sousedé	Město	Sousedé
1	2,3,4	3	1,2,4
2	1,3,4	4	1,2,3

Order Crossover (OX)

- Vyber v prvním rodiči R_1 libovolný podřetězec a zkopíruj jej do potomka P .
- Ve rodiči R_2 tyto uzly odstraň (již jsou použity).

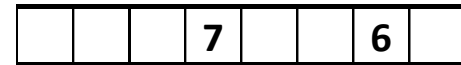
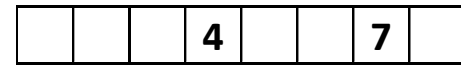
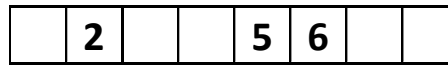
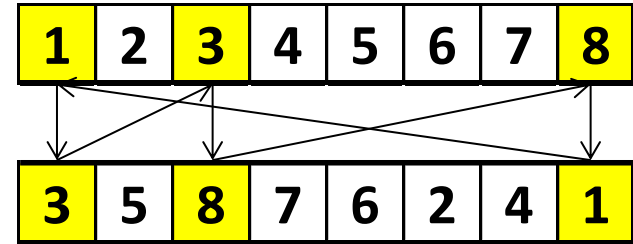


- Ze druhého rodiče R_2 kopíruj do volných míst potomka P zbývající uzly v pořadí, v jakém se ve druhém rodiči vyskytují.

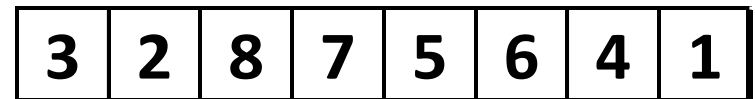
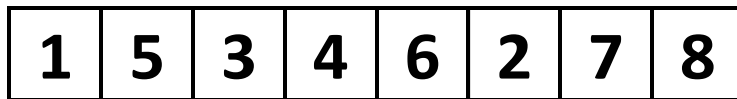


Cycle Crossover (CX)

- Najdi v rodičích vzájemné cykly (žlutě označen 1. cyklus)
- Níže cykly 1., 2. a 3.

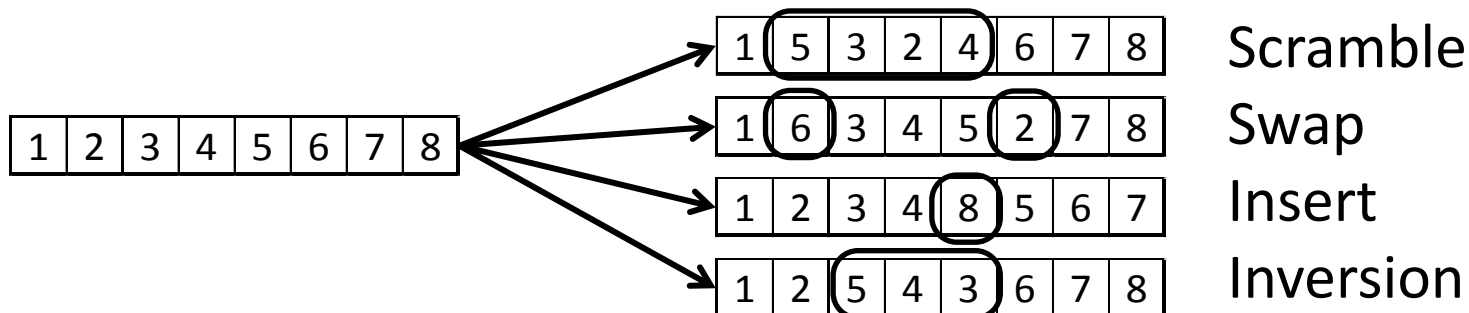


- Do prvního potomka zahrň liché cykly (1., 3., ...) z prvního a sudé z druhého rodiče, u druhého potomka postupuj obráceně.



Mutace – permutační úloha

- **Swap mutation**
 - prohození dvou náhodně vybraných prvků
- **Insert mutation**
 - Jsou náhodně vybrány 2 prvky, druhý prvek umístěn za první
- **Inversion mutation**
 - V náhodné sekvenci a je změněno pořadí prvků
- **Scramble mutation**
 - V náhodné sekvenci je náhodně změněno pořadí prvků



Selekce

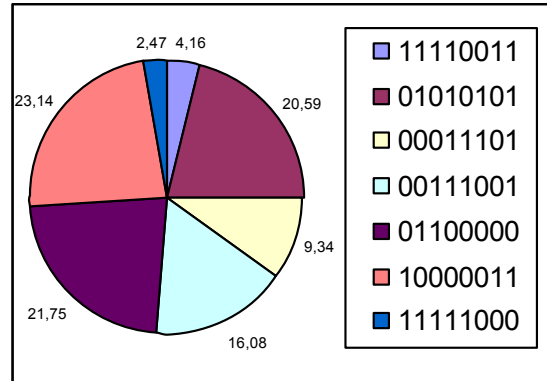
rodiče_generace_N = *selekce* (generace[N-1]);

- na základě selekce jsou vybráni jedinci, kteří se stanou rodiči
- 3 základní typy selekce
 - vážená ruleta
 - poziční selekce
 - metoda TURNAJ

? Uvedte tři základní typy selekce.

Selekce – 3 typy

- vážená ruleta
 - lokální extrém



JEDINEC								KVALITA
0	0	1	1	1	0	0	1	6,943
0	1	0	1	0	1	0	1	8,889
0	0	0	1	1	1	0	1	4,031
0	1	1	0	0	0	0	0	9,389
1	0	0	0	0	0	1	1	9,992
1	1	1	1	1	0	0	0	1,067
0	1	1	0	0	0	0	0	9,389

- poziční selekce
 - modifikace rulety, práce s pořadím (ordinalita)

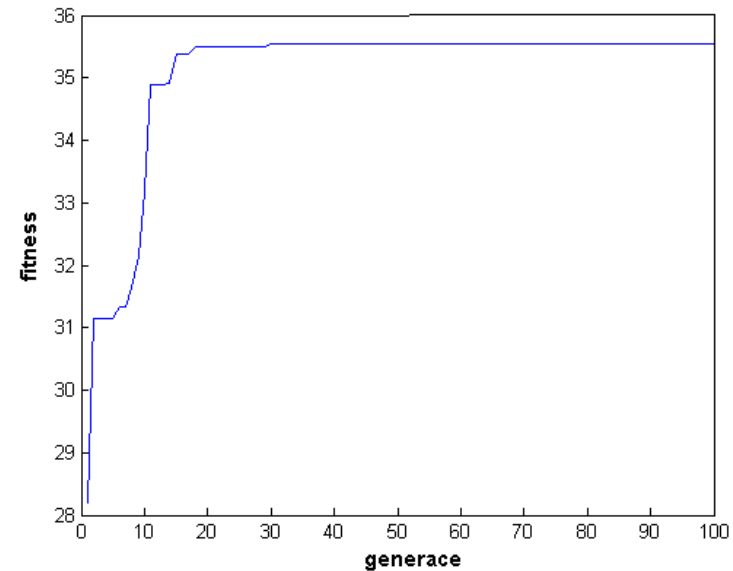
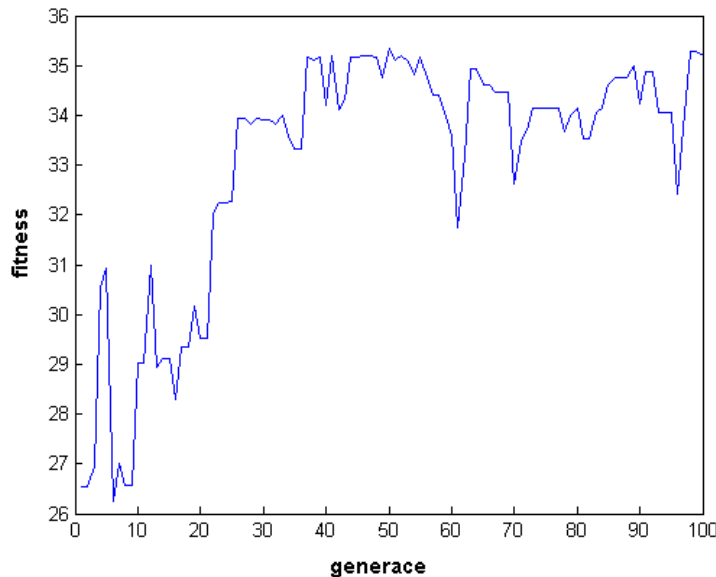
- metoda TURNAJ

- Vyber náhodně k jedinců (např. 3) s pravděpod. p (např. 0,8) vyber nejlepšího, s pravděpod. $p(1-p)$ druhého nejlepšího až po $k-1$ (poslední z k -tice má p rovnu doplňku součtu předešlých pravděpodobností do 1).

Elitismus

jedinci[] = *elitismus* (generace);

- nejlepší jedinec/jedinci nepodstupují selekci



VIS: ? Vysvětli PRICNIP a SMYSL elitismu.

Fitness

kvalita = *fitness* (jedinec)

- hodnoticí funkce, funkce vhodnosti, kritériální funkce, účelová funkce, cílová funkce
- hledání maxima – fitness je samotná funkce
- může být i záporná
- hledání minima (přímo MNČ, ML)

Parametry GA I.

- volba kódování (*reál.č. binárně, přesnost*)
- velikost počáteční populace (*lok.extr., rychlost*)
- volba hodnoticí funkce (*LF, finance/čas*)
- typ selekce (*v.r.-rychlá, lok.extr., různé*)
- typ křížení (*1-2 body, reálné 2. a 3. metoda*)
- pravděpodobnost křížení (*uváděno 0,6 – 1, spíše větší hodnota*)

Parametry GA II.

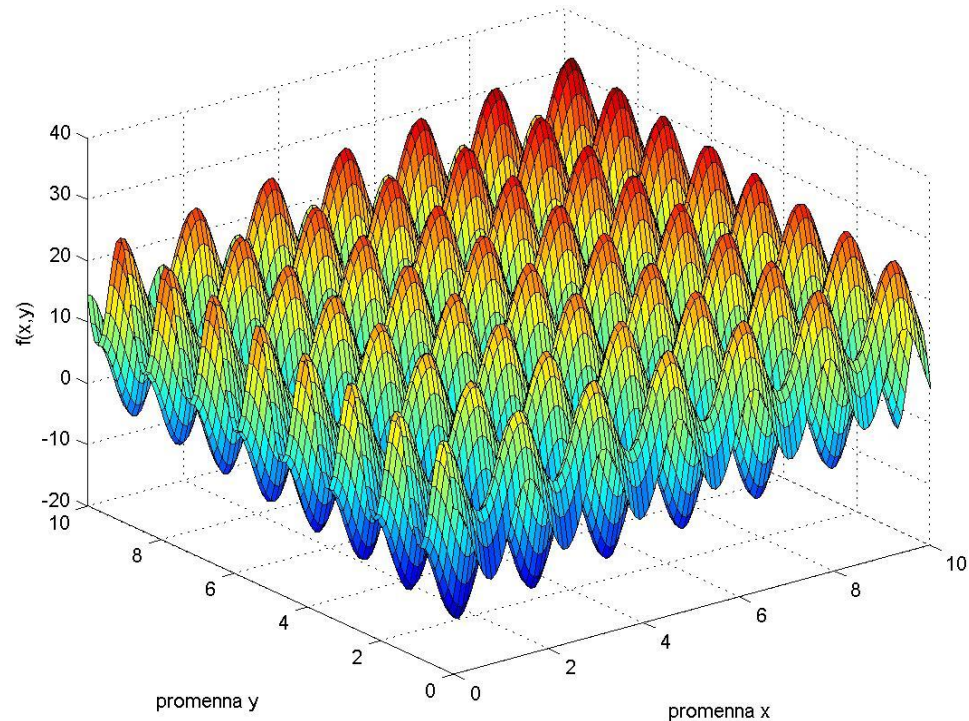
- typ mutace (*u reálného kódování dynamická*)
- pravděpodobnost mutace (*bin. 0,001–0,15; reálné kódování 0,01 – 0,5*)
- ukončovací podmínka (*počet generací, nejlepší jedinec stejný po K generací*)

Příklad - zadání

$$f(x, y) = x + (10 \cdot \sin(5 \cdot x)) + (7 \cdot \cos(4 \cdot y)) + y$$

$$x \in \langle 0, 10 \rangle$$

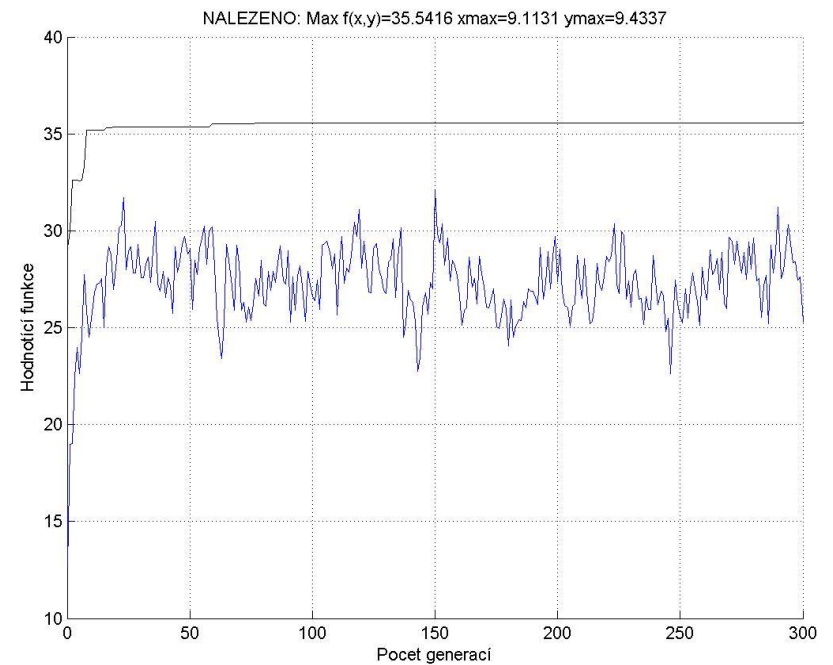
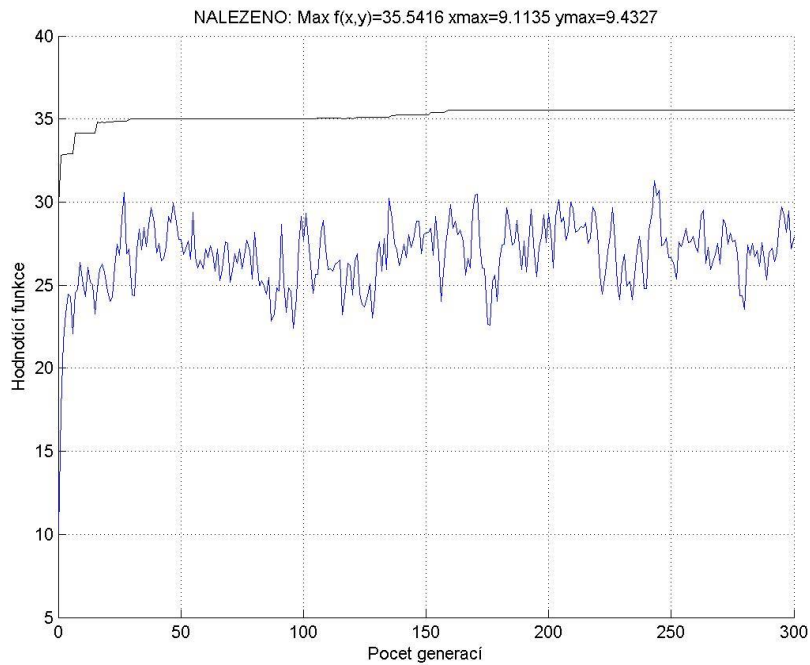
$$y \in \langle 0, 10 \rangle$$



Příklad – zvolené parametry

	Binární kódování	Reálné kódování
Velikost počáteční populace	25	25
Elitismus	Ano	Ano
Typ selekce	Vážená ruleta	Vážená ruleta
Typ křížení	Jednobodové	Vnitřní soutěž
Pravděpodobnost křížení	0,95	0,95
Typ mutace	-	Z intervalu (-0,005;0,005)
Pravděpodobnost mutace	0,05	0,56
Ukončovací podmínka	Počet generací: 300	Počet generací: 300

Příklad – výsledek



Shrnutí GA

- GA vycházejí z **evolučního počítání**, které je součástí umělé inteligence. Hlavní myšlenka má kořeny v Darwinově teorii o přirozeném vývoji na základě schopnosti **adaptace**. Podobně jako u biologický předloh jsou vlastnosti jedinců **kódovány** do **genů**, jejichž soubor vytváří **chromozom** jedince. Kódování rozlišujeme **binární** a **kódování do reálných čísel**.
- Na počátku aplikace řešící problém pomocí GA je vytvořena nultá **generace**. Následuje cyklus, ve kterém je spočtena všem jedincům velikost jejich **hodnoticí funkce (fitness)**; následuje **selekce**, na jejímž základě jsou vybráni **rodiče**; operandy **křížení** a **mutace** pak dávají vzniknout novým **potomkům**; pokud nová generace splňuje **ukončovací podmínky**, je cyklus ukončen, jinak se opět vrací na svůj začátek.
- Existuje řada typů genetických operátorů. Jejich správná volba a nastavení zásadně ovlivňují kvalitu konečného výsledku. Velký počet možných nastavení, z nichž jen některá vedou k řešení, představuje hlavní problém při použití GA. Přesto v praxi existuje řada jejich úspěšných aplikací.

Genetické programování (GP)

- GP – kódování nejen parametrů, ale i struktury modelu (typicky do stromových struktur)
 - Uzly – funkce (*operátory*)
 - Listy – proměnné, konstanty (*operandy*)
- Vhodná volba kódování klíčová (vhodná volba operátorů, operandů)
- Existuje řada kódování pro různé typy problémů (algebraické výrazy, algoritmy,...)

Jaký je rozdíl mezi genetickým programováním a genetickým algoritmem?

Ideový příklad kódování

- Operátory (unární, binární, ...)

- typ I: $+ - * / ^$, ... (01234)

- typ II: $\sin, \sqrt{}, \dots$ (56)

- Operandy

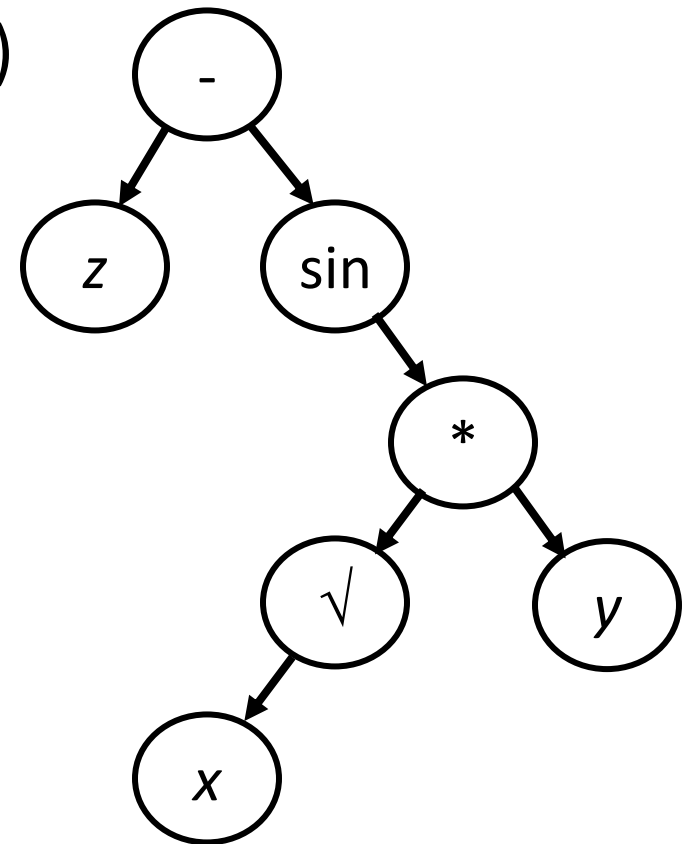
- x, y, z, \dots (789)

- Kódování (prefixový zápis)

- 1 9 5 2 6 7 8

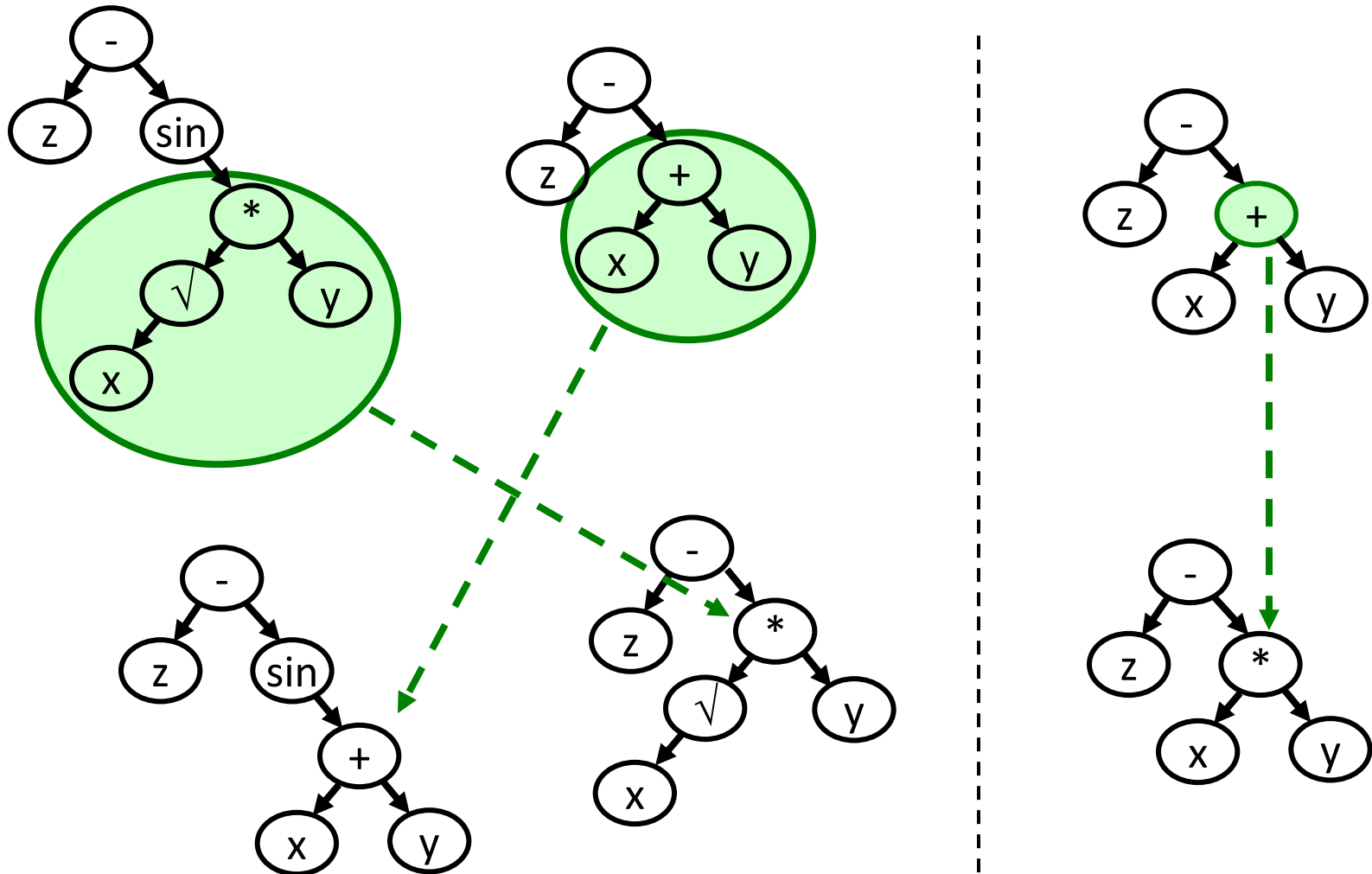
- $- z \sin * \sqrt{x} y$

- $-(z, \sin(*(\sqrt{x}, y)))$



Př.: převed' posloupnost 30574986278 do algebraického výrazu

Křížení, mutace



Diferenciální evoluce

- **Historie**
 - 1997 odvození první verze diferenciální evoluce (DE)
 - od 1999 – úspěšné implementace
- **Proč *evoluční* optimalizační algoritmus**
 - vytváří v cyklech celé generace možných řešení (připomíná tradiční evoluční techniky – genetické algoritmy)
 - princip je umělý, nekopíruje žádný přírodní proces (genetické algoritmy, mravenčích kolonie, simulované žíhání, rojové algoritmy, ...)

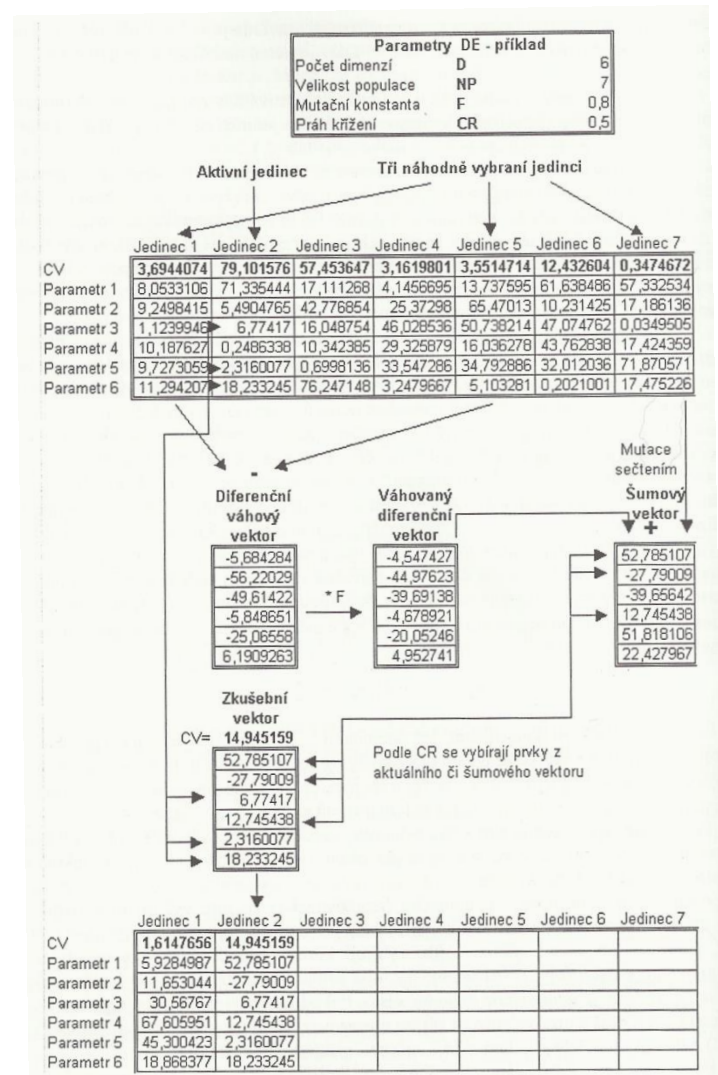
Základní terminologie

- Aktivní jedinec
 - během tvorby nové generace se právě jednou stane aktivním jedincem každý z jedinců stávající generace
- Rodič
 - náhodně vybraný jedinec ze stávající generace
- Šumový vektor
 - jedinec vzniklý rekombinací rodičů
- Zkušební jedinec
 - Jedinec vzniklý rekombinací rodičů a aktivního jedince

Princip DE

- Vytvoření nového jedince:
 - Vyber ze stávající generace **aktivního jedince** (hlavní rodič)
 - Vyber 3 další vzájemně různé **rodiče**
 - Ze všech 4 rodičů vytvoř **zkušební vektor**
 - Porovnej nově vytvořený **zkušební vektor** s **aktivním jedincem** a do nové generace zkopíruj toho lepšího

- Schéma převzato z: Mařík V. a kol.: Umělá inteligence 4, Akademie věd České republiky, Academia Praha. (4) 2003, ISBN 80-200-1044-0



Parametry DE 1/2

- **Účelová funkce $U(X)$** – *je dána*
 - funkce, jejíž min/max hodnotu hledáme (též cílová funkce, fitness, ...)
- **Dimenze problému D** – *vyplývá z U*
 - počet argumentů účelové funkce
- **Velikost populace P** – *1. parametr DE*
 - počet jedinců v každé z generací
 - minimální velikost $P=4$
 - doporučený počet P z intervalu $\langle 2D; 100D \rangle$
- **Mutační konstanta F** – *2. parametr DE*
 - míra vlivu mutace
 - z intervalu $\langle 0; 2 \rangle$
 - doporučovaná hodnota z intervalu $\langle 0,2; 0,9 \rangle$

Parametry DE 2/2

- **Pravděpodobnost křížení C** – 3. parametr DE
 - v intervalu $\langle 0;1 \rangle$
 - doporučované hodnoty pro neseparovatelnou účelovou funkci (nejčastější) v intervalu $\langle 0,8;0,9 \rangle$
 - $U(X) = x_1^2 + x_1x_2 + x_2^2$
 - pro separovatelnou účelovou funkci v intervalu $\langle 0;0,1 \rangle$
 - $U(X) = x_1^2 + x_2^2$
 - $C=0$ – křížení neproběhne, nový jedinec bude kopií aktivního jedince
 - $C=1$ – nový jedinec bez vlivu aktivního jedince (téměř náhodné prohledávání)

Kódování jedinců

- reálná čísla – přímá implementace do algoritmu
- diskrétní – operace jako u reálných, zaokrouhlení se provádí až **před výpočtem U** (zvyšuje možný počet vzniklých jedinců, zabraňuje uvíznutí v lokálním extrému)
- kvalitativní – převedou se na vektor indexů, dále se pracuje jako s diskrétními

Algoritmus DE

```
i=1
vygeneruj první generaci G(1)
while not (ukončovací podmínka)
  for each J from G(i)
    náhodně vyber různé R1,R2,R3
    SV=(R1+R2).F + R3
    for each A from J
      if (rand < CR)
        ZV(A)=SV(A)
      else
        ZV(A)=J(A)
      end if
    end for
    if ( CV(ZV)>CV(J) )
      add(ZV) to G(i+1)
    else
      add(J) to G(i+1)
    end if
  end for
  i=i+1
end while
```

Množiny jedinců

G(i) – i-tá generace (množina jedinců)

Jedinci (vektory)

J – tzv. aktivní jedinec

R – rodič, tedy jedinec

SV – šumový vektor (jedinec)

ZV – zkušební vektor (jedinec)

Atributy

A – atribut

J(A) – atribut jedince J

Parametry algoritmu DE

CR – pravděpodobnost křížení

F – mutační konstanta

Funkce

CV – účelová funkce

Základní vlastnosti DE

- má pouze 3 parametry (velká výhoda oproti GA)
- podporuje hybridní vstupní prostor (kvalitativní a kvantitativní veličiny dohromady)
- nevyužívá účelovou funkci při křížení – jednodušší rekombinační algoritmus => snižuje riziko uvíznutí v lokálním extrému
- jednoduchost – lze naprogramovat např. v tabulkovém editoru

Problém stagnace DE

- **Stagnace** – uvíznutí v lokálním extrému
- **Vznik** – typickou příčinou je skutečnost, že z jedinců stávající generace nelze rekombinací vytvořit nového lepšího jedince
- **Rozpoznání**
 - nezlepšuje se U např. 20 generací, přičemž jedinci stále různorodí
 - průběh nejlepšího a nejhoršího jedince „příliš brzy“ ustálen na jedné hodnotě
- **Řešení** – změna parametrů F , C během výpočtu
- **Prevence** – zajistit dostatečně velký počet možných nových jedinců
 - dostatečně velké P (např. $10D$)
 - $C=(0;1)$ – tedy otevřený interval
 - $F \neq$ celé číslo ($F=1$ může způsobit, že rozdíly různých jedinců vytvoří stejné šumové vektory, což se např. u $F=0,99$ nestane)

Problém opuštění definičního oboru

- nový jedinec je mimo definiční obor
 - vygenerování nové náhodné hodnoty (doporučovaný postup)
 - zaokrouhlení na mezní hodnotu (riziko stagnace)
 - odečtení o velikost přetečení (nutno ošetřit riziko opětovného překročení meze z druhé strany)

Doporučená literatura

- [1] Honzík, P.: *Strojové učení*, elektronická skripta VUT.
- [2] Mitchell, T.: *Machine Learning*. MacGraw Hill Science, 1997.
- [3] Hastie T.et.al.: *The Elements of Statistical Learning*. Springer, 2001.
- [4] Mařík, V. a kol.: *Umělá inteligence 4*, 2003.
- [5] ...nepřeberné množství materiálů na internetu...