

# OPTIMALIZACE – GENETICKÉ ALGORITMY



**Kurz:**     **Datové struktury a algoritmy**

---

**Lektor:**   Doc. Ing. Radim Burget, Ph.D.

**Autor:**    Doc. Ing. Radim Burget, Ph.D.

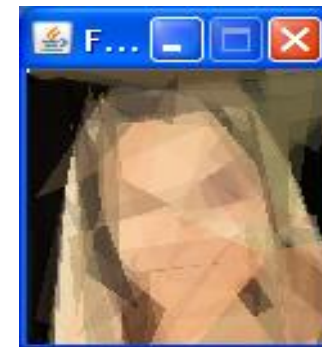
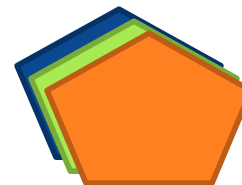


INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Vytvoření této videopřednášky bylo podpořeno projektem č. CZ.1.07/2.2.00/28.0098  
Evropského sociálního fondu a státním rozpočtem České republiky.

# Motivace

- Poskládejte 50 polygonů tak, aby se co nejvíce obraz podobal předloze
- Lze měnit:
  - Tvar
  - Barvu
  - Pozici
  - Průhlednost



# Cíl

- I. **Úvod do optimalizace**
- II. Genetické algoritmy
- III. Paralelizace GA
- IV. Typické příklady GA

# Obecná definice optimalizačního problému

- Hledáme hodnoty pro vybranou množinu proměnných:  $X = (x_1, \dots, x_n)$ , které maximalizují (nebo minimalizují) hodnotu účelové funkce:  $x_0 = f(x_1, \dots, x_n)$ , přičemž mohou být zadány omezující podmínky, které musí optimalizační úloha respektovat.
- Omezující podmínky mohou být zadány např. ve tvaru (ne)rovníc:

$$\begin{cases} g_1(x_1, \dots, x_n) & (> = <) & b_1 \\ g_2(x_1, \dots, x_n) & (> = <) & b_2 \\ & \vdots & \\ g_m(x_1, \dots, x_n) & (> = <) & b_m \end{cases}$$

# Příklad optimalizačního problému

## • Úkol:

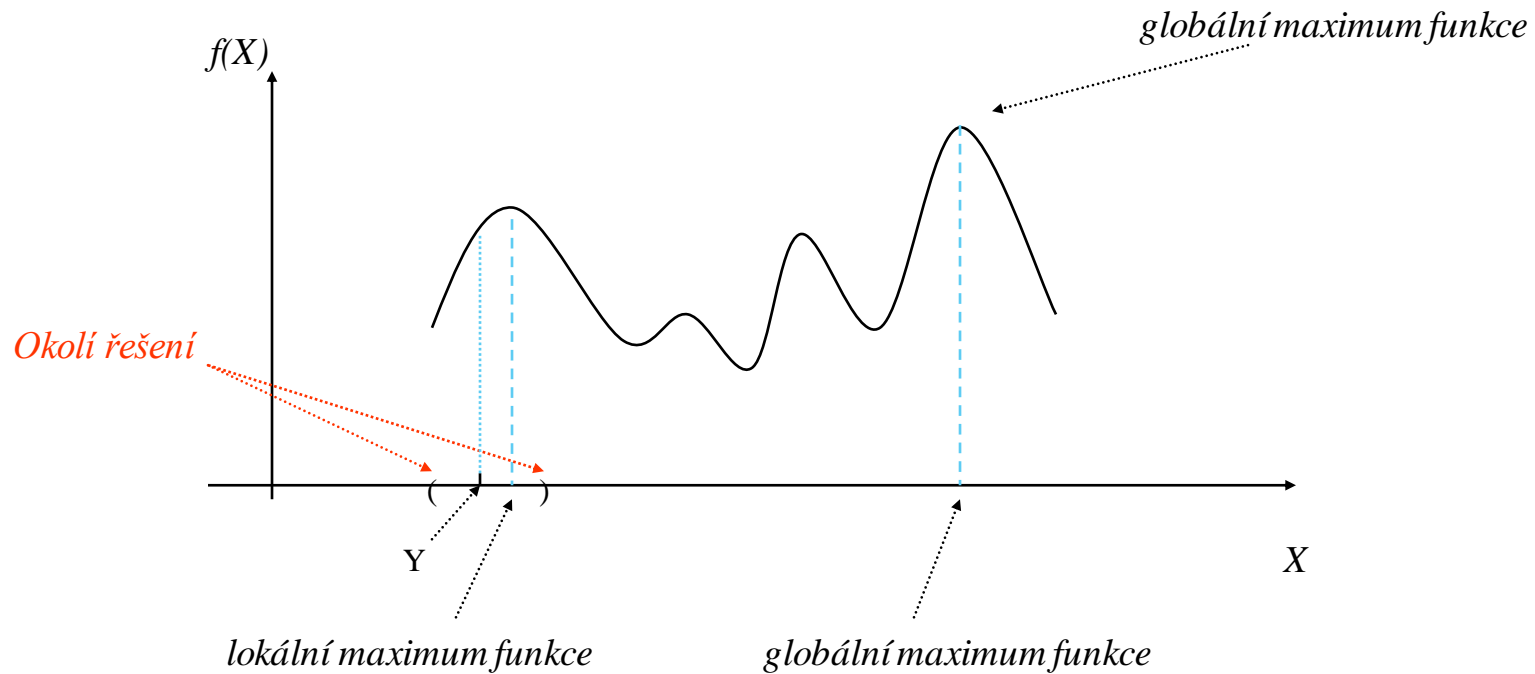
- Najít nejvyšší bod v krajině

## • Řešení:

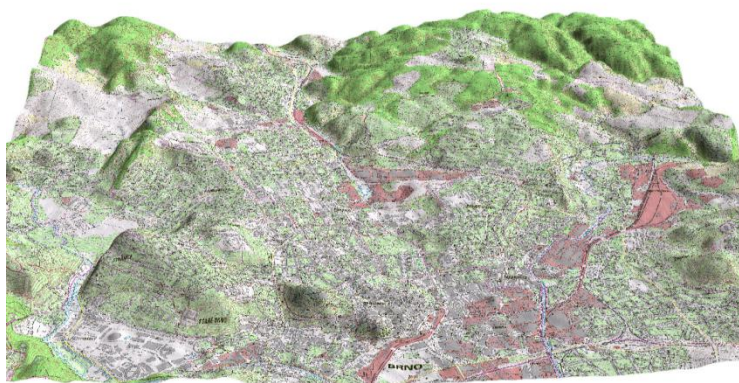
- $X = (x_1, \dots, x_n)$ , (souřadnice na mapě)
- $x_0 = f(x_1, \dots, x_n)$ , (nadmořská výška)
- Maximalizační úloha (najdi nejvyšší bod)
- Prohledávacím prostorem je „krajina“ (kopce, údolí, roviny, plošiny...)

Vždy je dobré si pro názornost problém vizualizovat, což lze jednoduše provést, když se  $n$  rovná nízkým číslům, avšak v praxi se  $n$  často rovná číslům v řádu deseti tisíců a vizualizaci je třeba zjednodušovat

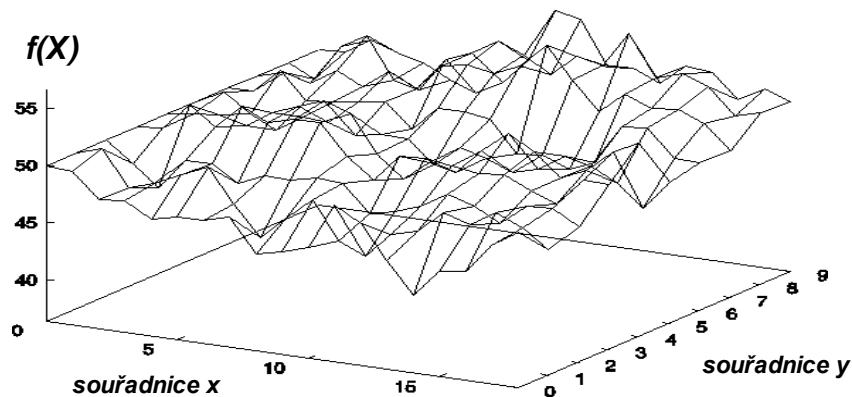
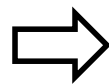
# Vizualizace obecného problému



# Vizualizace konkrétního problému



***Plastická mapa krajiny***



***3D graf krajiny***

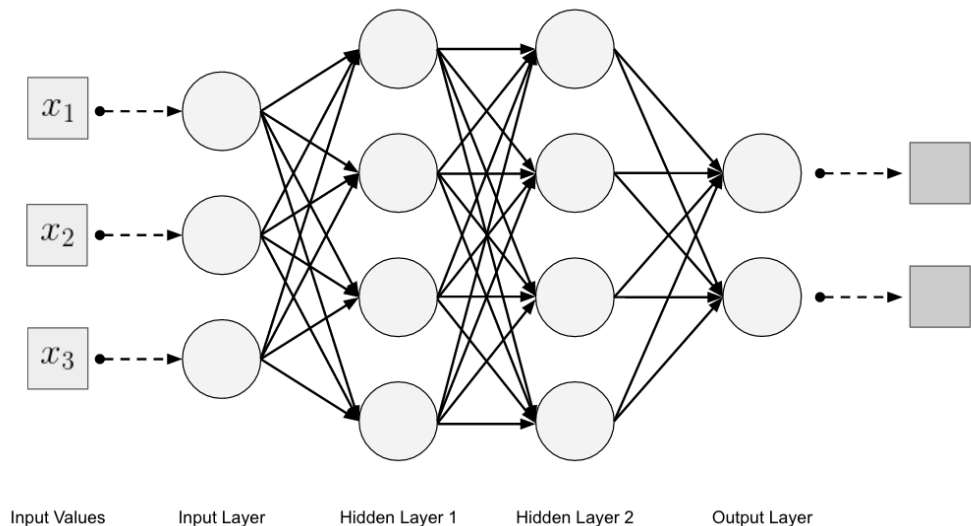
# Další definice a základní pojmy

- Každý vektor  $X$  splňující omezující podmínky se nazývá *možné řešení*.
- Možných řešení může být velmi mnoho, ale jedno řešení maximalizující nebo minimalizující zadanou funkci, se nazývá *optimální řešení*.
- Pozn.:
  - Častým problémem je, že nelze jednoduše vytvořit model problému
  - Počet možných řešení exponenciálně roste s velikostí řešeného problému
  - Nakonec, i optimálních řešení může být více, rozdíl může být pouze v konfiguraci každého jedince – závisí na podstatě řešeného problému



# Příklad: Optimalizační problém

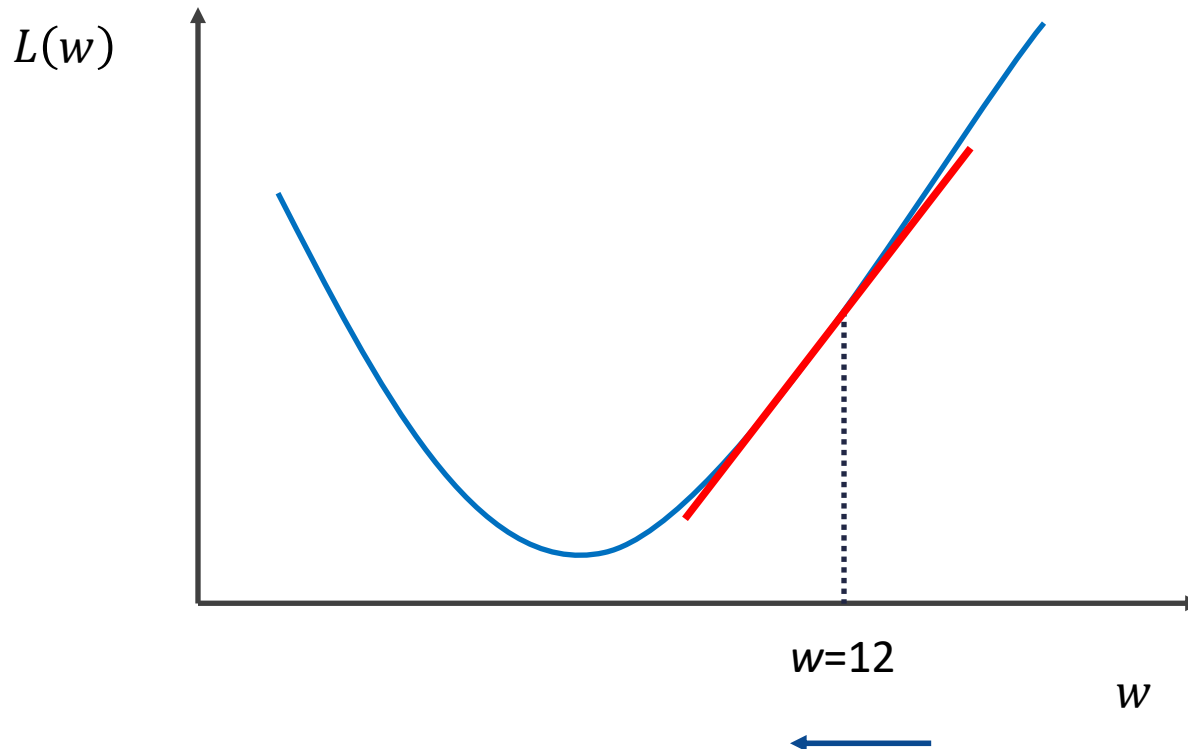
- Kolik vrstev?
- Kolik neuronů v každé vrstvě?
- Jaké aktivační funkce?



# Vybrané metody optimalizace

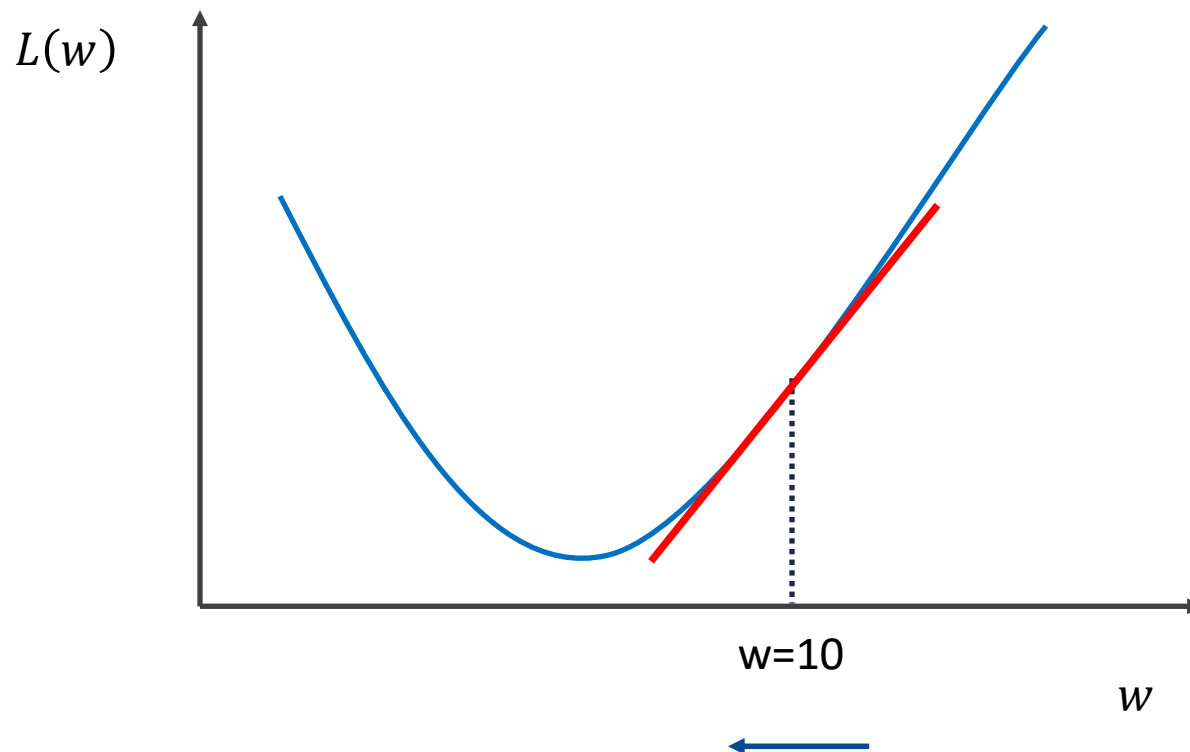
- Hrubá síla – (všechny možnosti – vždy nejlepší výsledky)
- **Náhodné hledání (Random Search)** - neefektivní
- **Horolezecká metoda (Hill climbing)**
  - V okolí nejlepšího bodu se hledá další lepší řešení
- **Mřížkové hledání (Grid Search)**
  - Kombinace s každý s každým (kolik neuronů ve vrstvách)
- **Metoda gradientního sestupu** (např. moderní neuronové sítě)
- **Evoluční algoritmy**
  - **Genetické algoritmy**
  - **Genetické programování**

# Gradientní sestup (Gradient Descent, GD) (myšlenka)



1. Začneme náhodnou hodnotou  $w$  (např.  $w = 12$ )
2. Spočteme sestup (derivace)  $L(w)$  v bodě  $w = 12$ . (např.  $dL/dw = 6$ )
3. Update  $w$  :  
$$w = w - \text{lambda} * (dL / dw)$$

# Gradientní sestup (Gradient Descent, GD) (myšlenka)



2. Spočteme sestup  
(derivace)  $L(w)$  v bodě  
 $w = 12$ . (např.  $dL/dw = 6$ )

3. Update  $w$  :

$$w = w - \text{lambda} * (dL / dw)$$

# Gradientní sestup (Gradient Descent, GD)

$\lambda = 0.01$  ... rychlost učení

Náhodně inicializujeme  $w$  a  $b$

**for**  $e = 0$ , počet epoch **do**


Spočteme:  $\underbrace{dL(w, b)/dw}$  and  $\underbrace{dL(w, b)/db}$

Aktualizujeme:  $w = w - \lambda dL(w, b)/dw$

Aktualizujeme:  $b = b - \lambda dL(w, b)/db$

**end**

$$L(w, b) = \sum_{i=1}^n -\log f_{i, label}(w, b)$$

 **náročné**

**Rychlost učení**

# (mini-dávkový) Stochastický Gradientní Sestup (SGD)

$$\lambda = 0.01$$

Inicializujeme  $w$  a  $b$  náhodně

$$l(w, b) = \sum_{i \in B} -\log f_{i, \text{label}}(w, b)$$

**for**  $e = 0$ , počet epoch **do**

**for**  $b = 0$ , počet dávek **do**

        Spočteme:  $dl(w, b)/dw$  and  $dl(w, b)/db$

        Aktualizujeme:  $w = w - \lambda dl(w, b)/dw$

        Aktualizujeme:  $b = b - \lambda dl(w, b)/db$

**end**

**end**

# Aktualizace hybnosti

Namísto

$$w = w - \lambda \frac{dl(w, b)}{dw}$$

Použijeme:

$$v = \rho v + \lambda \frac{dl(w, b)}{dw}$$

$\rho$  (ró) typicky 0.8-0.9

$$w = w - v$$

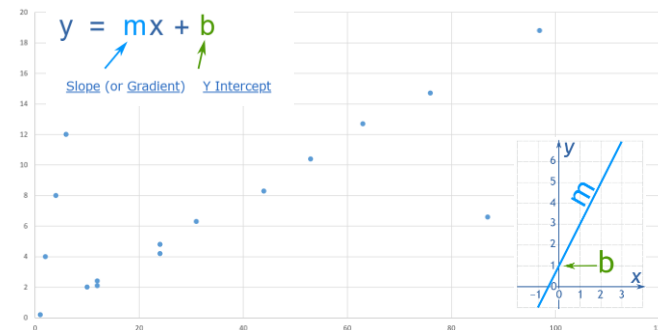
# Ukázka implementace

```

/* # of iterations and alpha = learning rate */
int iterations = 140000;
float alpha = 0.07f;

for (int i = 0; i < iterations; i++) {
    float h0 = 0.0f; // sum of errors for theta0
    float h1 = 0.0f; // sum of errors for theta1
    for (int j = 0; j < numOfSamples; j++) {
        h0 = h0 + ((theta0 + x[j] * theta1) - y[j]); // error1
        h1 = h1 + ((theta0 + x[j] * theta1) - y[j]) * x[j]; // error2
    }
    // update theta (using temp values)
    float tempTheta0 = theta0 - (alpha * h0) / (float) numOfSamples;
    float tempTheta1 = theta1 - (alpha * h1) / (float) numOfSamples;
    theta0 = tempTheta0;
    theta1 = tempTheta1;
}

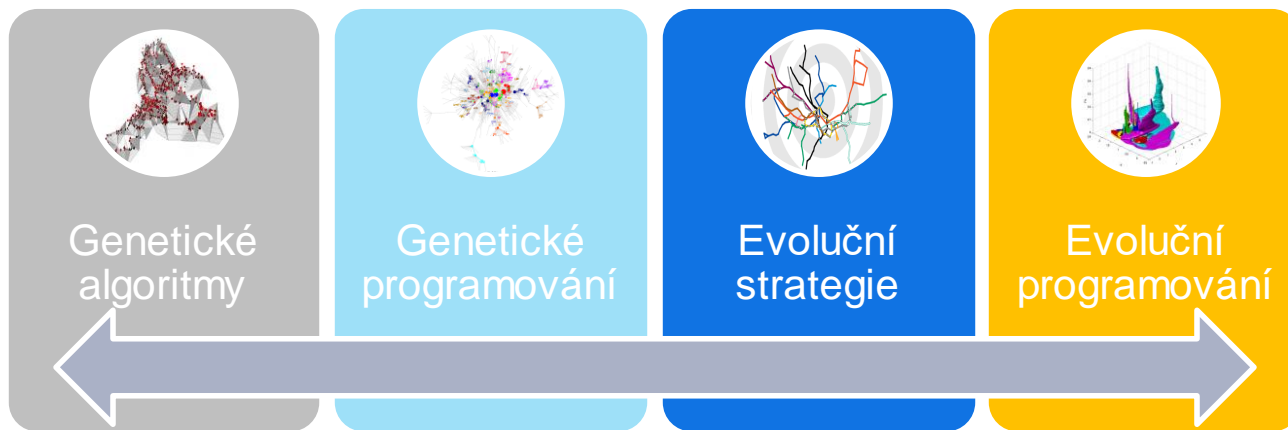
```





# Evoluční algoritmy (EA)

- zastřešují řadu přístupů využívajících tyto *modely biologické evoluce*:
  - přirozený výběr (silnější jedinci, podle hodnoty fitness funkce)
  - náhodný genetický drift (mutace, decimace jedince s vysokou fitness)
  - reprodukční proces (křížení jedinců – výměna genetické informace)



## Motivace – „závody ve zbrojení“

- netopýři mají sonar, kterým hledají můry, sonar je sám o sobě komplikovaný
- můry vyvinuly měkké pokrytí těla, které absorbuje netopýří vysílání
- netopýři přešli na nové frekvence
- můry přišly s novým pokrytím a s “rušičkou” (vlastní signál interferuje s netopýřím)
- netopýři přišli s novými leteckými manévry a naučily se vypínat sonar (čímž dělají rušení méně efektivním)



# V čem je problém?

- Téměř každý problém lze popsat parametry
- Záležitostí optimalizace je potom nalezení takových parametrů, pro které dává celý systém ve výsledku nejlepší výsledky



Zdroj: Amnesia-The Dark Descent

# Evoluční algoritmy (EA) – obecný algoritmus

begin

$t := 0;$

inicializujPopulaci  $P(t);$

ohodnot'  $P(t);$

while neníKonec do

$t := t + 1;$

$P' := \text{vyberJedince } P(t);$

křížJedince  $P'(t);$

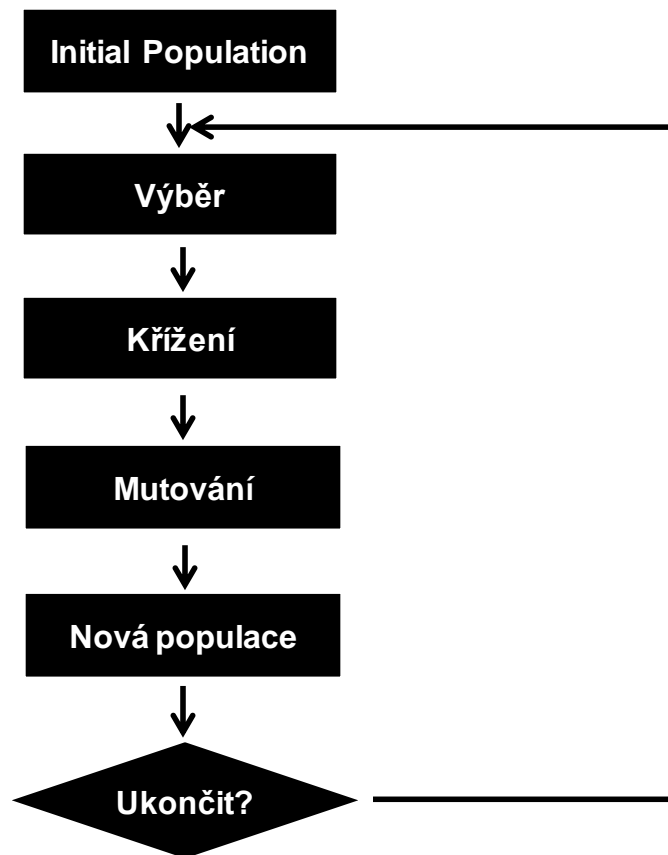
mutujJedince  $P'(t);$

ohodnot'  $P'(t);$

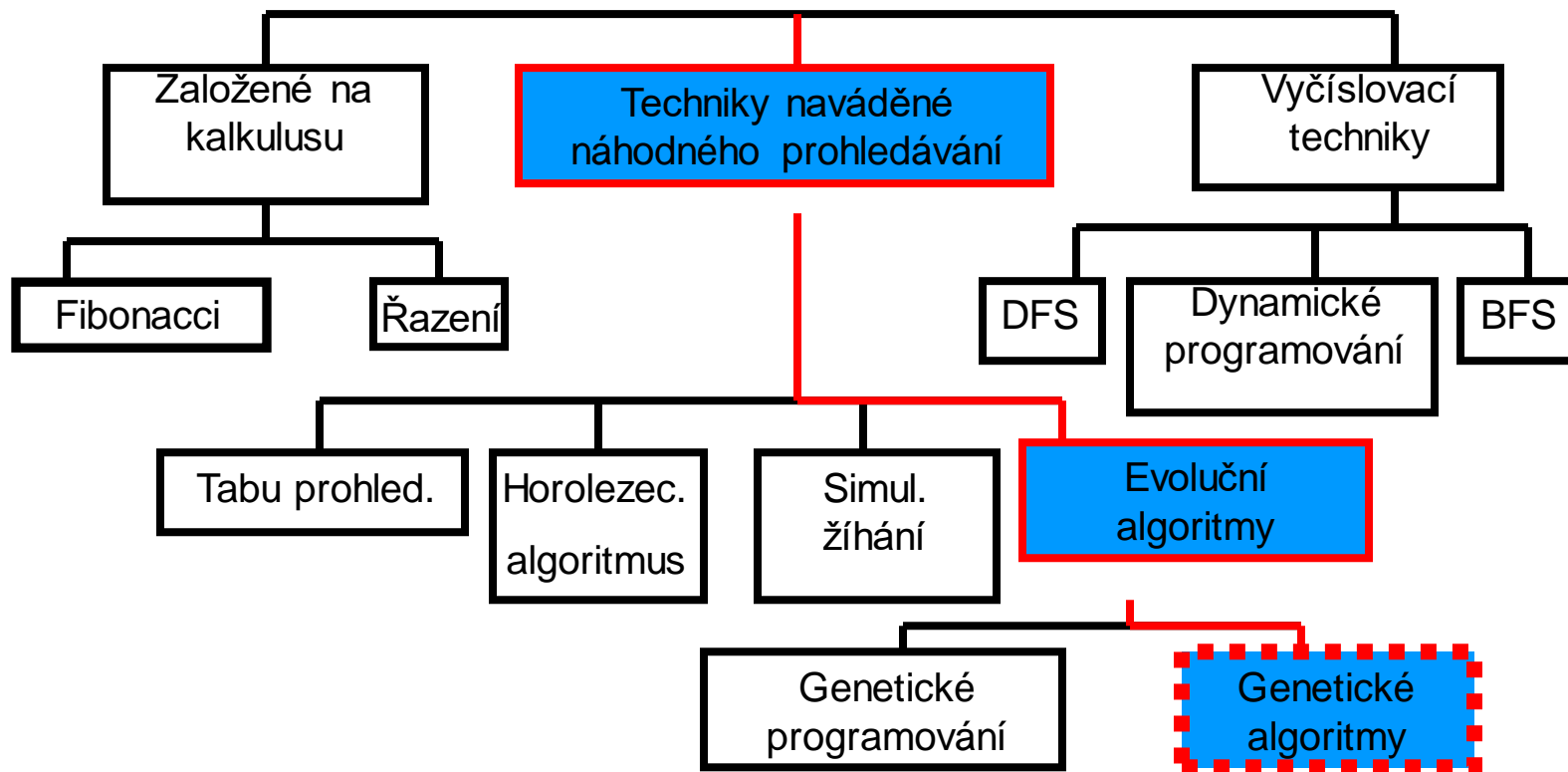
$P := \text{vytvořNovouPopulaci } P, P'(t);$

od

end



# Techniky prohledávání



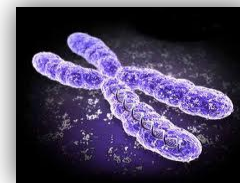
# Cíl

- I. Úvod do optimalizace
- II. **Genetické algoritmy**
- III. Paralelizace GA
- IV. Typické příklady GA

# GA – Pohled do historie

- GA založeny na Darwinově myšlence „přežití nejsilnějších“
  - GA v 70tých letech 20tého století je formuloval R. Holland
  - GA v základu používaly dva genetické operátory: křížení a inverzi
  - Poprvé použity jako prohledávací mechanismus pro adaptivní systémy UI
- Kniha „*Genetic Algorithms in Search, Optimization, and Machine Learning*“ napsána D. Goldbergem v roce 1989 (studentem R. Hollanda) se zasloužila o největší rozmach problematiky genetických algoritmů

# GA – Základní pojmy



- Rozdíl mezi individuem (**fenotyp**) a jeho reprezentací (**genotyp**)
  - V biologii, každá buňka nese určitý počet informací (chromozomů)
  - V informatice je individuum (jedinec) charakterizován jedním chromozomem
  - Lze tedy zavést zjednodušení, Genotyp = Chromozom
- **Chromozom** se dále dělí na geny (uspořádány lineárně)
- **Gen** na i-té pozici reprezentuje stejnou charakteristiku v každém jedinci
- **Alela** je hodnota, které může nabývat gen (např. 0 nebo 1)



# Příklad

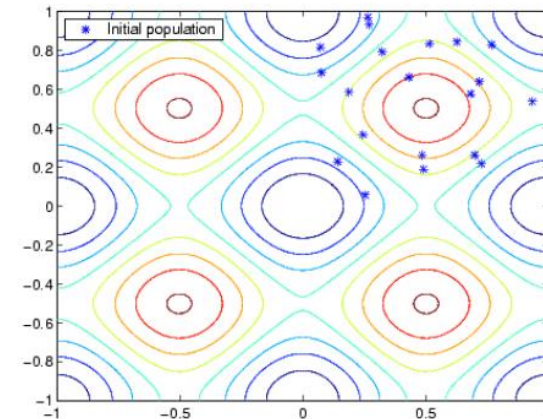
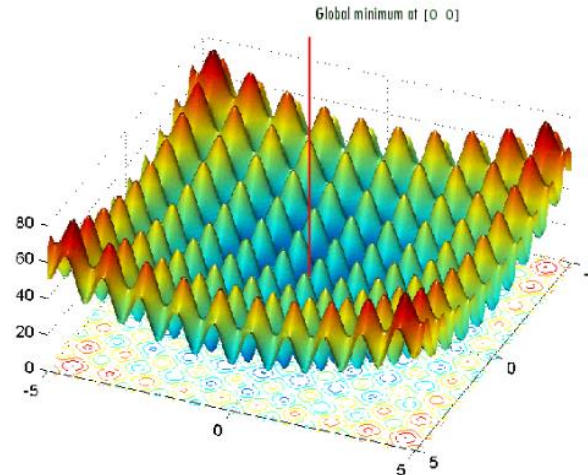
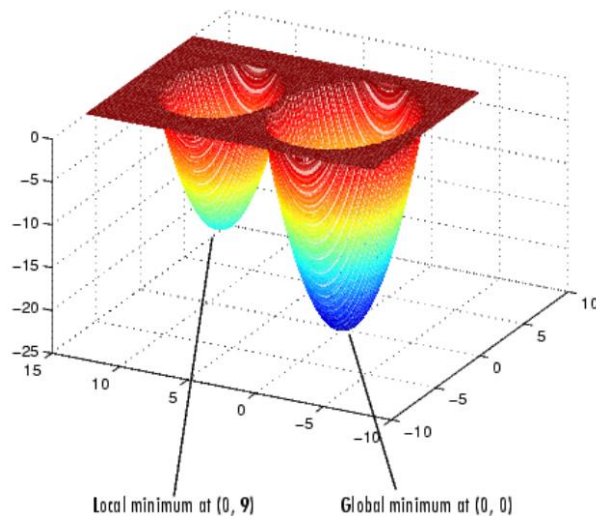
- Biologie – výrazně složitější
- EA, GA – matematická metoda, která se inspičuje biologií, pomocí parametrů kóduje podobu

Dokrytí			
Dokrytí			
Velikost	Rychlost	Pokrytí těla	Barva
1	4	2	4

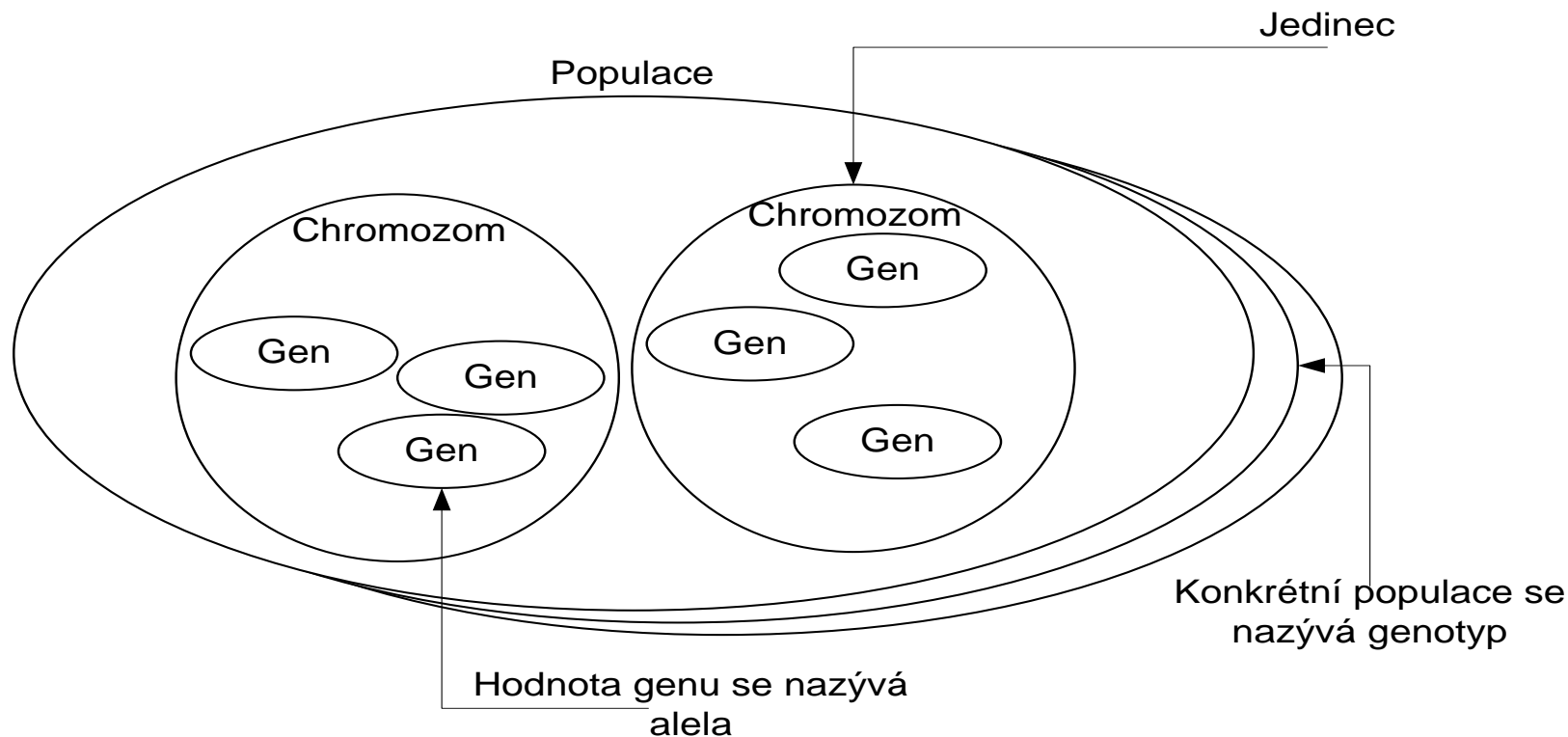


# Lokální vs. globální; velká výzva

- This an important challenge !



# Genetické algoritmy



# GA – Etapy návrhu algoritmu

1	• Reprezentace problému
2	• Vytvoření počáteční populace
3	• Vytvoření fitness funkce
4	• Operátory selekce
5	• Genetické operátory
6	• Obnova populace
7	• Ukončení algoritmu
8	• Kontrola běhu algoritmu

# Reprezentace problému (kódování chromozomu)

- Zvolíme relevantní geny, které bude chromozom obsahovat
  - **Příklad:** Máme problém, kdy je třeba nalézt obličej pachatele, potřebujeme tedy zakódovat jeho podobu, geny tedy např. budou: délka vlasů, barva vlasů, barva očí, tvar obličeje, tvar nosu, atd.
- Zvolíme či bude délka chromozomu statická či variabilní
  - **Příklad:** V případě zmiňovaného příkladu bude délka statická, každý obličej budeme kódovat stejnými parametry
- Zvolíme jakých hodnot budou moci geny nabývat
  - **Příklad:** Je třeba si předem definovat množiny (číselníky), ze kterých budeme doplňovat hodnoty do jednotlivých genů, např. pro barvu očí to bude množina: hnědá, černá, modrá, zelená...

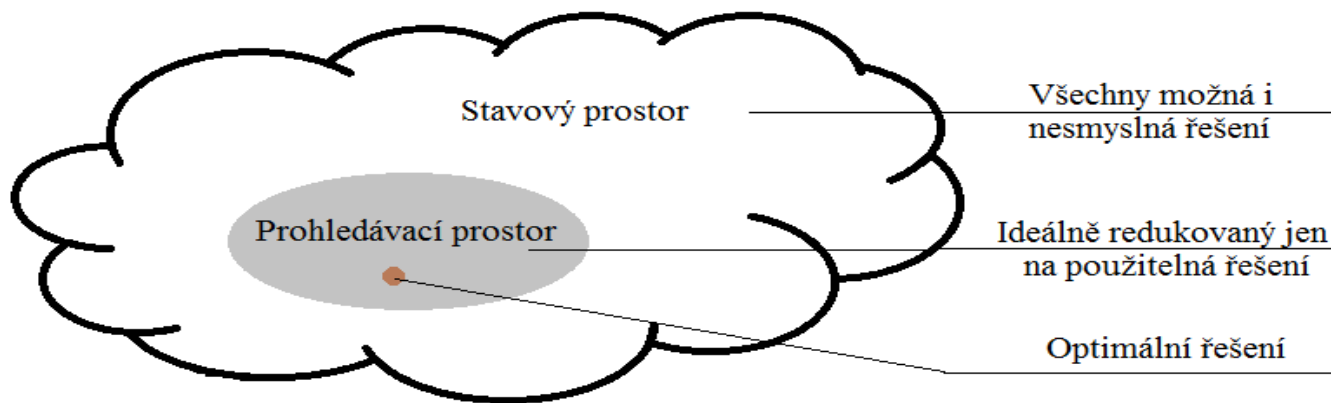
# Reprezentace problému (kódování chromozomu)

- **Možné typy kódování**

- Binární kódování, nejčastěji se používá Grayův kód (změna jednoho bitu)
  - **Příklad:** 0000 | 0001 | 0011 | ...
- Reálné kódování
  - **Příklad:** 1,3 | 2,54 | 3,14 | ...
- Znakové kódování
  - **Příklad:** A | M | K | ...
- Objektové kódování
  - **Příklad:** ObjA | ObjB | ObjC | ...
- Atd.

# Vytvoření počáteční populace

- Zpravidla náhodný proces
- Musí být pokryta co největší část stavového prostoru
- Pomocí teorie schémat lze generovat jedince „blíže“ optimálnímu řešení



# Vytvoření počáteční populace

- Uniformně náhodné
- Založené na heuristice
- Založené na znalosti
- Genotyp z předchozí evoluce (pokračování)



# Vytvoření počáteční populace

- Lze použít i informovanou metodu
  - Musí být postavena na znalosti stavového prostoru
  - Může vést k nalezení lepších řešení a zkrátit čas výpočtu
  - Může způsobit nevratné nasměrování algoritmu k sub-optimálnímu řešení

# Vytvoření fitness funkce

- Kvantitativně vyjadřuje kvalitu každého řešení
- Obvykle reálné číslo, vyšší značí kvalitnější řešení
- Nejjednodušeji může být fitness funkce vyjádřena jako:
  - dosažení požadované přesnosti algoritmu
  - množství času potřebné pro výpočet algoritmu
  - množství chyb mezi skutečným a požadovaným výstupem algoritmu

# Vytvoření fitness funkce

## Hrubá fitness (raw fitness)

- Odchylka mezi dosaženou a předpokládanou hodnotou
- Často transformována na referenční hodnotu (další typy fitness, viz dále)

$$r(i, t) = \sum_{j=1}^{N_e} |S(i, j) - C(j)|$$

$t$  je číslo generace

$S(i, j)$  je výsledek vyhodnocení jedince  $i$  pro  $j$ -tý prvek trénovací množiny

$N_e$  celkový počet prvků trénovací množiny

$C(j)$  je očekávaná hodnota pro vstup odpovídající  $j$ -tému prvku trén. množ.

# Vytvoření fitness funkce

## Standardizovaná fitness (standardized fitness)

- Přepoččet hrubé fitness na určitou referenční hodnotu
- Optimální řešení je rovno nejvyšší hodnotě

$$s(i, t) = r_{max} - r(i, t)$$

$r(i, t)$  je hodnota hrubé fitness (předchozí slajd)

$r_{max}$  je referenční hodnota fitness

# Vytvoření fitness funkce

## Přizpůsobená fitness (adjusted fitness)

- Hodnota leží vždy v intervalu  $<0; 1>$
- Optimální řešení je rovno 1

$$a(i, t) = \frac{1}{1 + s(i, t)}$$

$s(i, t)$  je hodnota standardizované fitness (předchozí slajd)

# Vytvoření fitness funkce

## Normalizovaná fitness (normalized/proportional fitness)

- Hodnota leží vždy v intervalu  $<0; 1>$
- Optimální řešení je rovno 1
- Suma normalizovaných fitness v populaci je rovna 1

$$n(i, t) = \frac{a(i, t)}{\sum_{k=1}^M a(k, t)}$$

$a(i, t)$  je hodnota přizpůsobené fitness (předchozí slajd)

$M$  je velikost populace

# Vytvoření fitness funkce

- **Metody na úpravu fitness funkce, podpora/potlačení elitářství**

- **Ranking**

- Přiděluje fitness nikoliv na základě velikosti objektivní funkce, ale podle pořadí daného řešení vzhledem k populaci
- **Příklad:** nejlepší řešení bude mít očekávaný počet kopií 3, druhé nejlepší 2 a ostatní do výběru po 1 a zbytek (nejhorší řešení) 0.

- **Linear scaling**

- Určuje fitness tak, aby fitness celé populace měla lineární průběh, byl zachován průměr hodnot objektivních funkcí a aby maximální fitness byl v intervalu 1,2-2,0 násobku průměru (voleno uživatelem)

# Multi-Objektivní EA (MOEA)

- Rozšiřuje EA, které sledují více než jediný cíl v jediné fitness hodnotě
- Tyto cíle jdou zpravidla proti sobě (např. přesnost vs. složitost)
- Standardní EA, jedinec **A** se říká, že je lepší než **B** pokud **A** má lepší fitness hodnotu nežli **B**
- Pro MOEA: jedinec **A** je lepší než jedinec **B** pokud **A** **dominuje nad B**



# Definice: Dominance v MOEA

- Jedinec **A** dominuje nad jeincem **B** pokud:
  - **A** je horší než **B** ve všech parametrech popřípadě
  - **A** je lepší alespoň v jednom parametru oproti **B**

# Operátory selekce

- Selektce většinou koresponduje s fitness hodnotou
- Avšak, je třeba zachovat dostatečnou různorodost populace

- **Selekční tlak/intenzita**

- S vyšší hodnotou, algoritmus rychle konverguje k řešení, ale současně se zvyšuje možnost předčasné konvergence (sub-optimální řešení)
- Problém je řešen zavedením určitého počtu generací, které jsou zapotřebí, aby selekce zaplnila celou populaci jedinců nejlepším chromozomem při neúčasti rekombinačního a mutačního operátoru

$$I = \frac{\overline{M^*} - \overline{M}}{\overline{\sigma}}$$

$\overline{M}$  průměrná hodnota fitness funkce v populaci před selekcí

$\overline{M^*}$  průměrná hodnota fitness funkce v populaci po selekci

$\overline{\sigma}$  rozptyl fitness funkce před selekcí

# Operátory selekce

## • Ruletový výběr (roulette wheel selection)

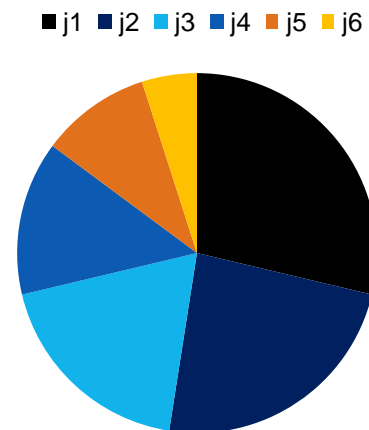
- Pravděpodobnost výběru závisí na poměrné kvalitě jedince
- Neboli, velikost výseče na ruletě závisí na velikosti fitness funkce

$$p_i = \frac{f_i}{\sum_{j=1}^N f_j}, i \in \{1, \dots, N\}$$

$p_i$  představuje pravděpodobnost výběru

$f_i$  představuje hodnotu fitness funkce  $i$ -tého jedince

$\sum_{j=1}^N f_j$  představuje součet hodnot fitness všech jedinců



# Operátory selekce

## • Ruletový výběr – vlastnosti

- Pokud bude v populaci jeden jedinec značně převyšovat ostatní, budou další populace tvořeny z převážné většiny jeho geny
- Když se GA blíží nalezení optimální hodnoty, jsou všechny hodnoty jedinců podobné a lepší jedinci nejsou dostatečně zvýhodněni před jinými



## Ruletový výběr – příklad

Jedinec	j1	j2	j3	j4	j5	j6
Fitness	30	14	10	4	3	2
$p_i$	0,48	0,22	0,16	0,06	0,05	0,03

# Operátory selekce

## • Ruletový výběr – varianta 2 (rank selection)

- Jedinci jsou seřazeni vzestupně podle jejich hodnoty fitness funkce
- Velikost místa na ruletě se vypočítá podle rovnice:

$$p_i = \frac{i_i}{\sum_{j=1}^N j} = \frac{2 \cdot i}{N \cdot (N + 1)}, i \in \{1, \dots, N\}$$

$i$  ... pořadí jedince v populaci

$N$  ...představuje velikost populace

$\sum_{j=1}^N j$  ...je součet pořadí všech jedinců

# Operátory selekce

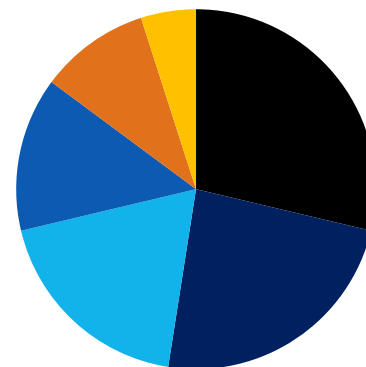
## • Ruletový výběr – varianta 2 – vlastnosti

- Tento druh selekce potlačuje roli nadprůměrně dobrých jedinců, kteří by byli schopni negativně ovlivnit vytváření dalších generací
- Zajišťuje také selekční tlak, když se algoritmus blíží k optimálnímu řešení, když už mezi jedinci nejsou velké rozdíly

## Ruletový výběr – varianta 2 – příklad

Jedinec	j1	j2	j3	j4	j5	j6
Fitness	20	14	10	4	3	2
Pořadí	1	2	3	4	5	6
$p_i$	0,29	0,24	0,19	0,14	0,1	0,05

■ j1 ■ j2 ■ j3 ■ j4 ■ j5 ■ j6



# Operátory selekce

## • Turnajový výběr (tournament selection)

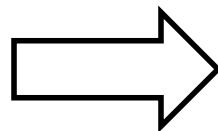
- Náhodně je vybráno  $n$  jedinců, postupným porovnáváním je vybrán nejlepší
- Jde v podstatě o analogii turnaje mezi rivaly před biologickou reprodukcí
- Většinou mezi sebou soupeří dva jedinci ( $n = 2$ )
- Experimenty dokazují, že je v převážné většině případů nejvhodnější



# Genetické operátory – Elitářství

- Zaručuje monotónní (neklesající) hodnotu fitness nejlepšího jedince
- Vybere se  $n$  ( $n=1$ ) jedinců, kteří jsou zkopírováni do nové populace
- Předchází ztrátě nejlepšího řešení, které by mohlo být znehodnoceno

Rodič



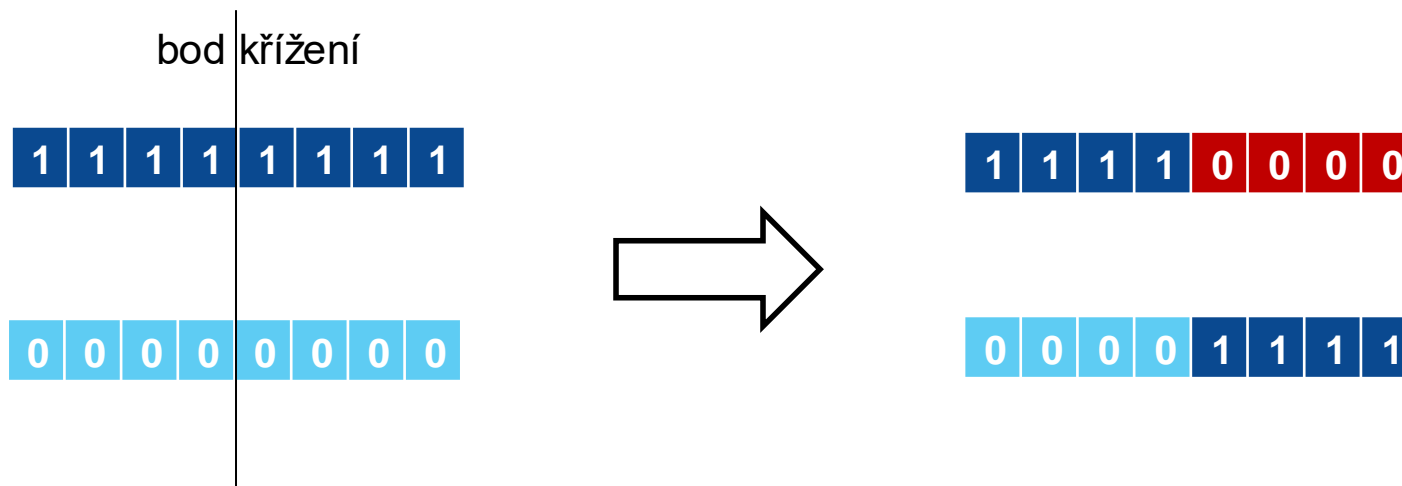
Potomek





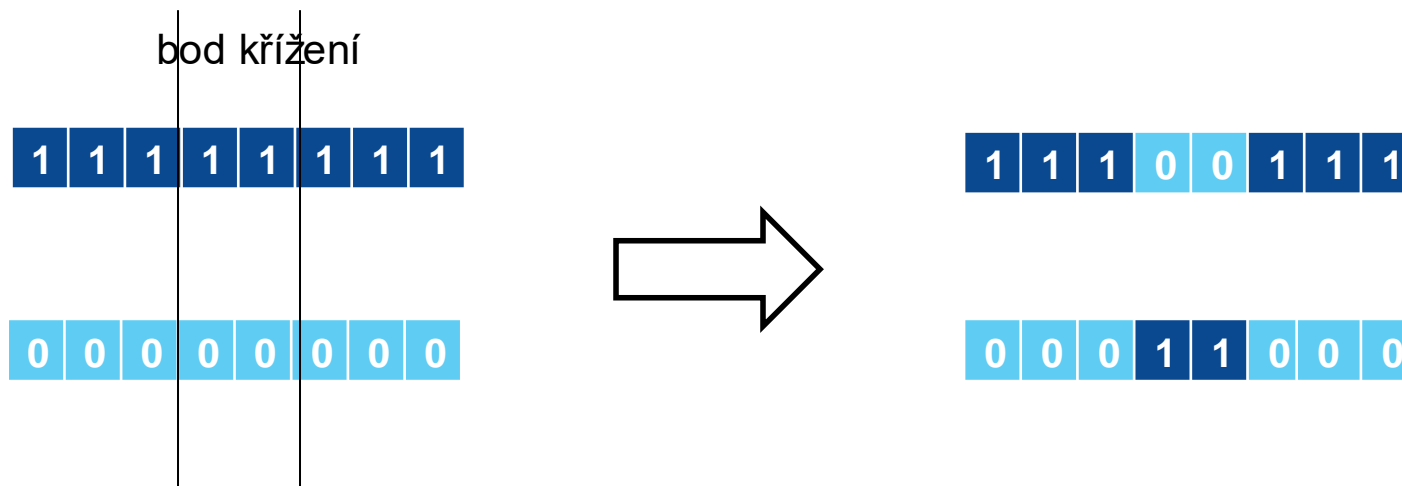
# Genetické operátory – Křížení

- Základní operátor, nejobecnějším typem je n-bodové křížení
  - Na příkladu ukažme jednobodové křížení



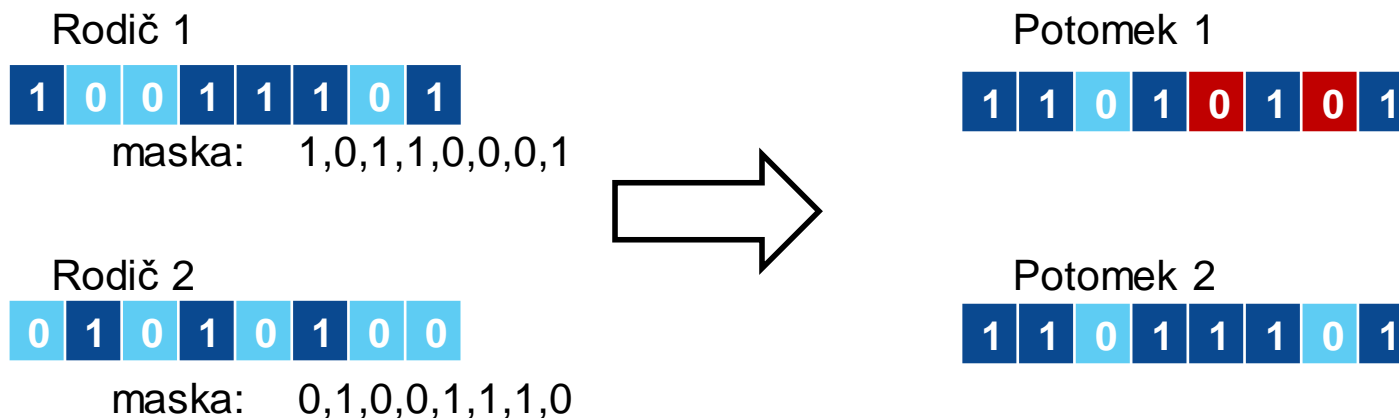
# Genetické operátory – Křížení

- Základní operátor, nejobecnějším typem je n-bodové křížení
  - Na příkladu ukažme dvoubodové křížení



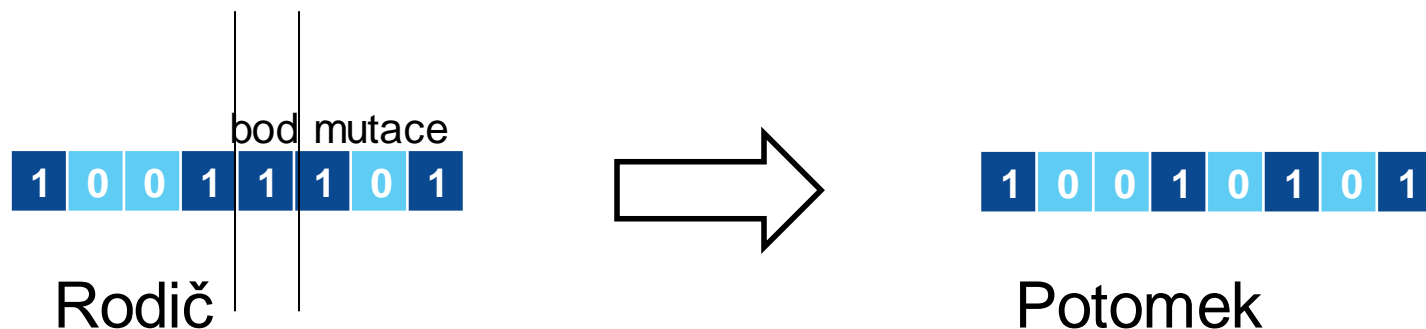
# Genetické operátory – Křížení

- Dalším typem je uniformní křížení
  - Příliš rozvrací kód chromozomu
  - Ale lze je dobře použít pro vnesení diverzity do populace
  - Používá křížící masku, která udává, které geny se kříží, maska pro druhého rodiče je inverzní k masce prvního rodiče



# Genetické operátory – Mutace

- Aplikuje se s velmi malou pravděpodobností
- Významná zvláště v populacích s málo jedinci
- Pravděpodobnost použití zvyšujeme pro vyšší diverzitu populace



# Obnova populace

- Generativní s úplnou obnovou
  - Rodiče jsou kompletně nahrazení potomky
- Částečná obnova populace
  - Pouze jeden potomek nahradí nejslabšího jedince
- V praxi se používá kombinace těchto způsobů
- Techniky obnovy populace:
  - Podle kvality jedince určené fitness funkcí (selekční metoda)
  - Elitářství (nejlepší jedinec je přímo překopírován do nové populace)
  - Faktor přemnožení (potomek nahrazuje rodiče s podobným genotypem)

# Ukončení algoritmu

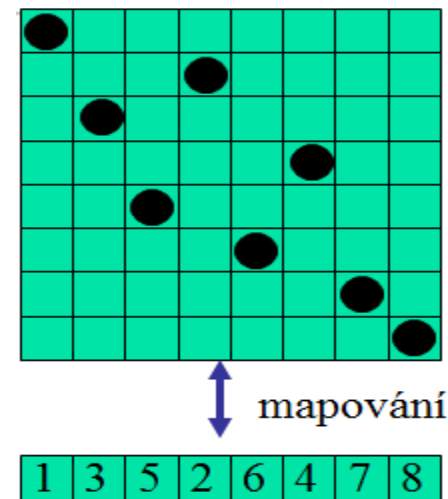
- Počet dosažených generací
- Počet fitness ohodnocení
- Vyčerpaný čas pro výpočet algoritmu
- Nezlepšující se fitness
- Neexistuje odlišnost v populaci
- Dosažená přesnost hledaného řešení / nalezeno řešení
- Kombinace předchozích

# Kontrola běhu algoritmu

- Velikost populace
- Poměr křížení
- Poměr mutace
- Atd.

# GA – Základní pojmy – Příklad

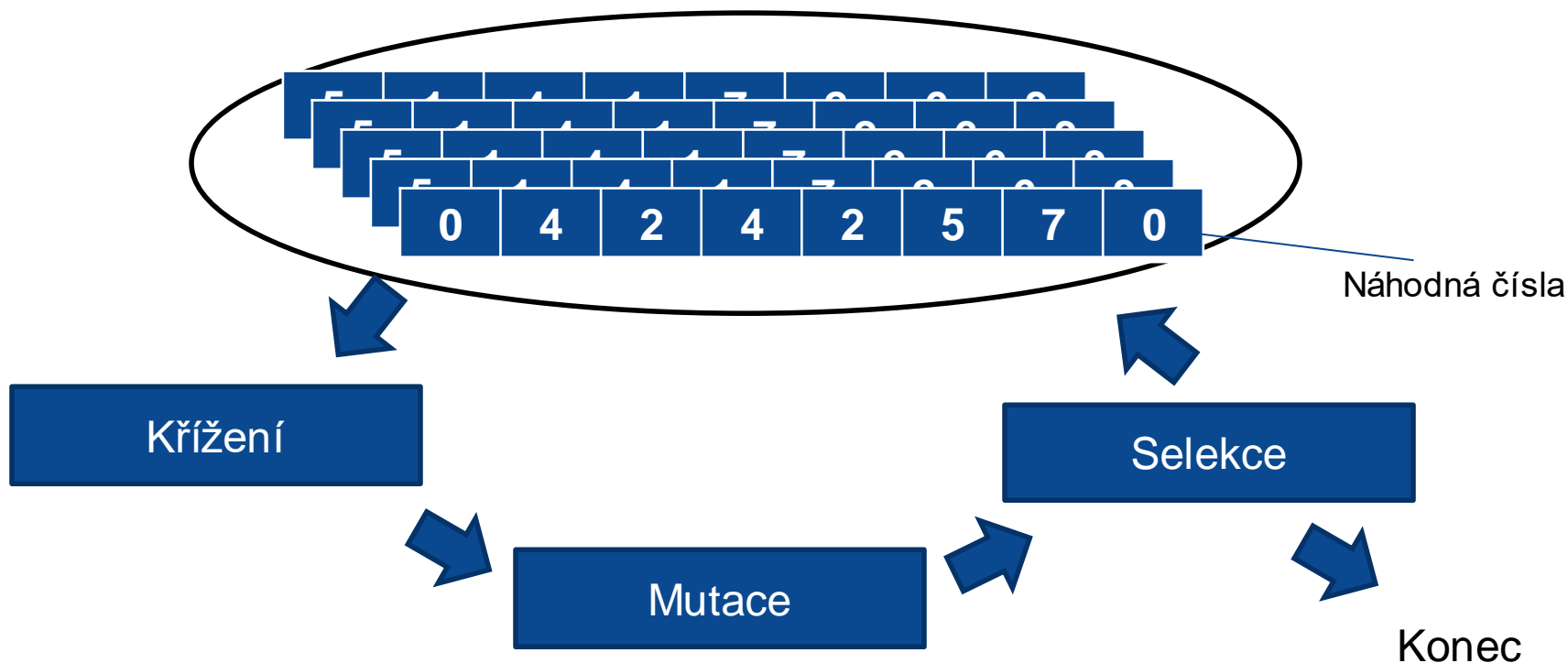
- Problém 8mi dam
- Na šachovnici 8x8 rozmístěte 8 dam tak, aby žádná neohrožovala jinou
- Konfigurace dam na šachovnici – **Fenotyp**
- Možná konfigurace chromozomu (čísel 1-8) – **- Genotyp**
- Genotyp = Chromozom
- Chromozom obsahuje celkem 8 genů (8 dam).
- Gen může nabývat hodnot (alel) 1-8.





# Příklad: řešení problému 8 dam pomocí GA

- Vytvoření náhodné populace např. velikosti 5:



# Rozložení polygonů (ne až tak typické 😊)

- Rozmístěte 50 polygonů (pětiúhelníků) tak, aby co nejlépe odpovídaly přeloze obrázku
  - Jak bude vypadat podoba možného chromozomu?
  - Jak bude vypadat fitness funkce?



POLYGON 1														POLYGON 2													
X1	Y1	X2	Y2	X3	Y3	X4	Y4	X5	Y5	Color RED	Color GREEN	Color BLUE	Transparency	X1	Y1	X2	Y2	X3	Y3	X4	Y4	X5	Y5	Color RED	Color GREEN	Color BLUE	Transparency
88	12	88	12	139	30	88	12	88	12	139	30	1	40	88	12	88	12	139	30	88	12	88	12	139	30	1	40

...

POLYGON 50													
X1	Y1	X2	Y2	X3	Y3	X4	Y4	X5	Y5	Color RED	Color GREEN	Color BLUE	Transparency
88	12	88	12	139	30	88	12	88	12	139	30	1	40

# Cíl

- I. Úvod do optimalizace
- II. Genetické algoritmy
- III. **Paralelizace GA**
- IV. Typické příklady GA

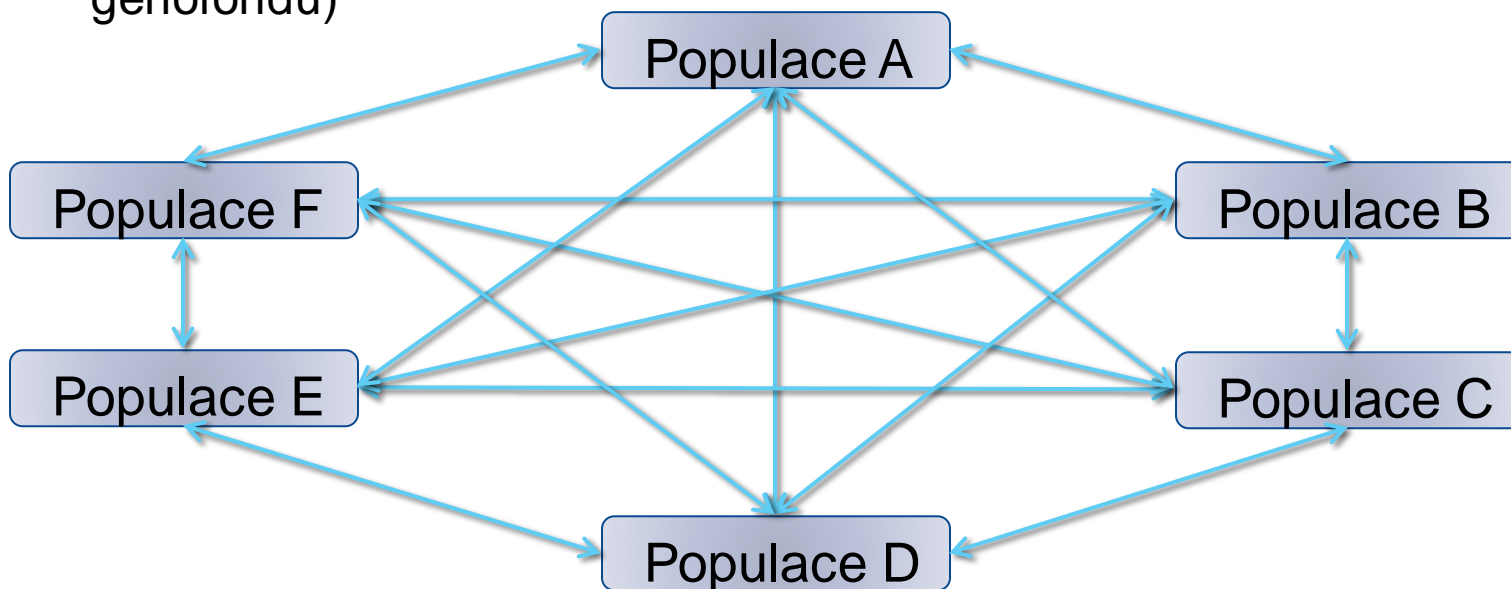
# Možnosti paralelizace

- Paralelizace GA je relativně snadná, existuje několik způsobů
  - Paralelizace operací nad jednou populací (výpočet fitness, selekce)
  - Paralelizace sub-populací
  - Kombinace obou
- Paralelní GA využívají menších populací (demy)
- Sub-populace rychle konvergují k lokálnímu optimu
- Nižší rozmanitost v sub-populacích – řeší se migrací jedinců
- Základní 3 modely paralelizace
  - Migrační model
  - Globální model
  - Difuzní model

# Možnosti paralelizace

## • Migrační model

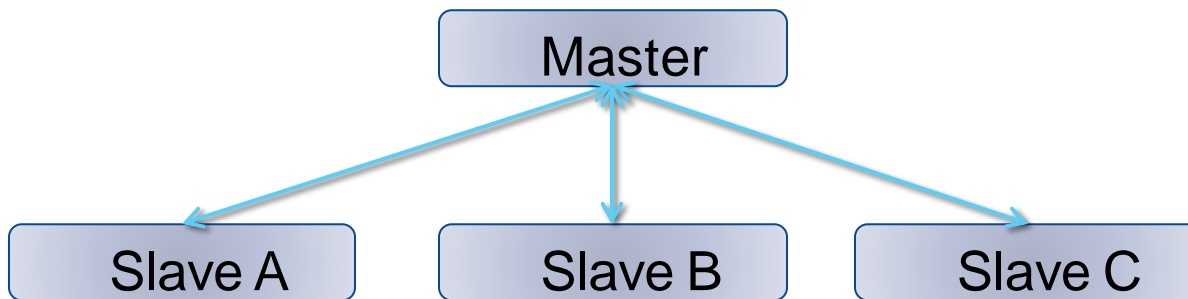
- Populace rozdělena do více sub-populací vyvíjejících se nezávisle
- Po určitém počtu generací dochází k migraci jedinců (výměna genofondu)



# Možnosti paralelizace

## • Globální model

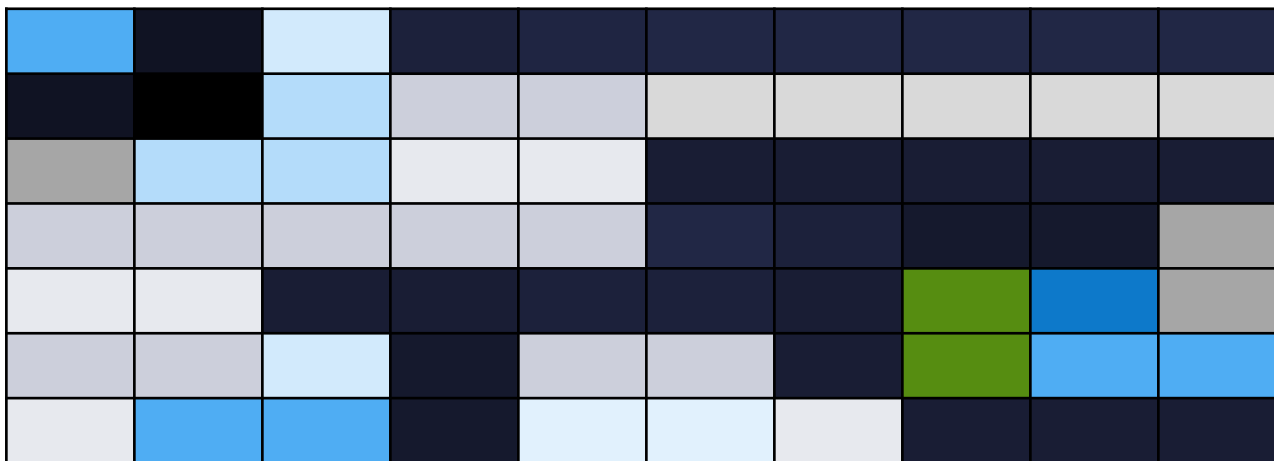
- Nerozděluje populace do sub-populací
- Rozděluje průběh výpočtu genetického algoritmu do paralelních procesů
- Řídící proces (master) provádí selekci a přiřazuje ohodnocení všem jedincům, ostatní procesy jako je křížení, mutace a samotný výpočet fitness funkce jsou distribuovány na další procesy (slave)



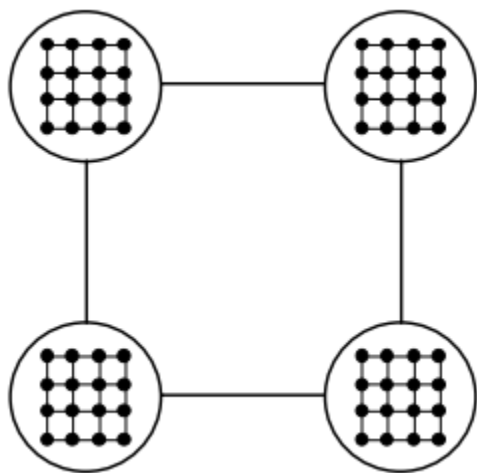
# Možnosti paralelizace

## • Difuzní model

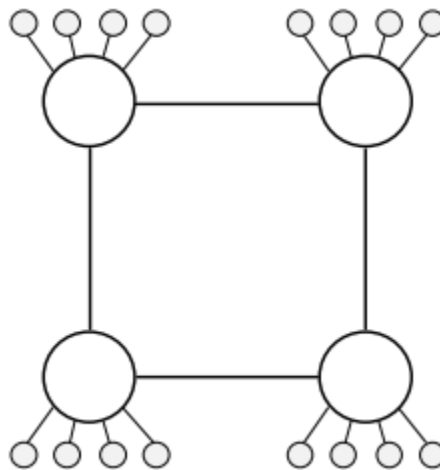
- Každý jedinec je řízen zvlášť
- Jedinec vstupuje do křížení jen s jedinci v okolí (4okolí, 8okolí)
- Tmavší místa značí vyšší fitness hodnotu, světlejší nižší



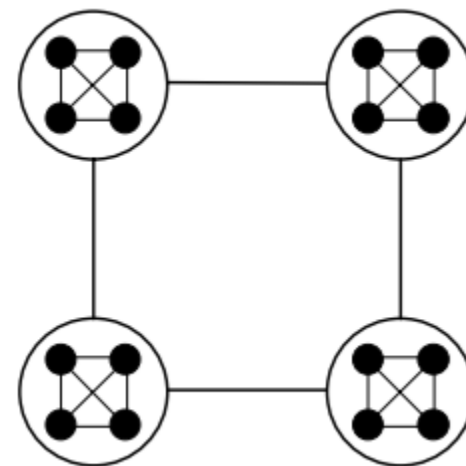
# Hybridní GA Paralelizace



Řídká a detailní



Řídká a master slave



Řídká a řídká

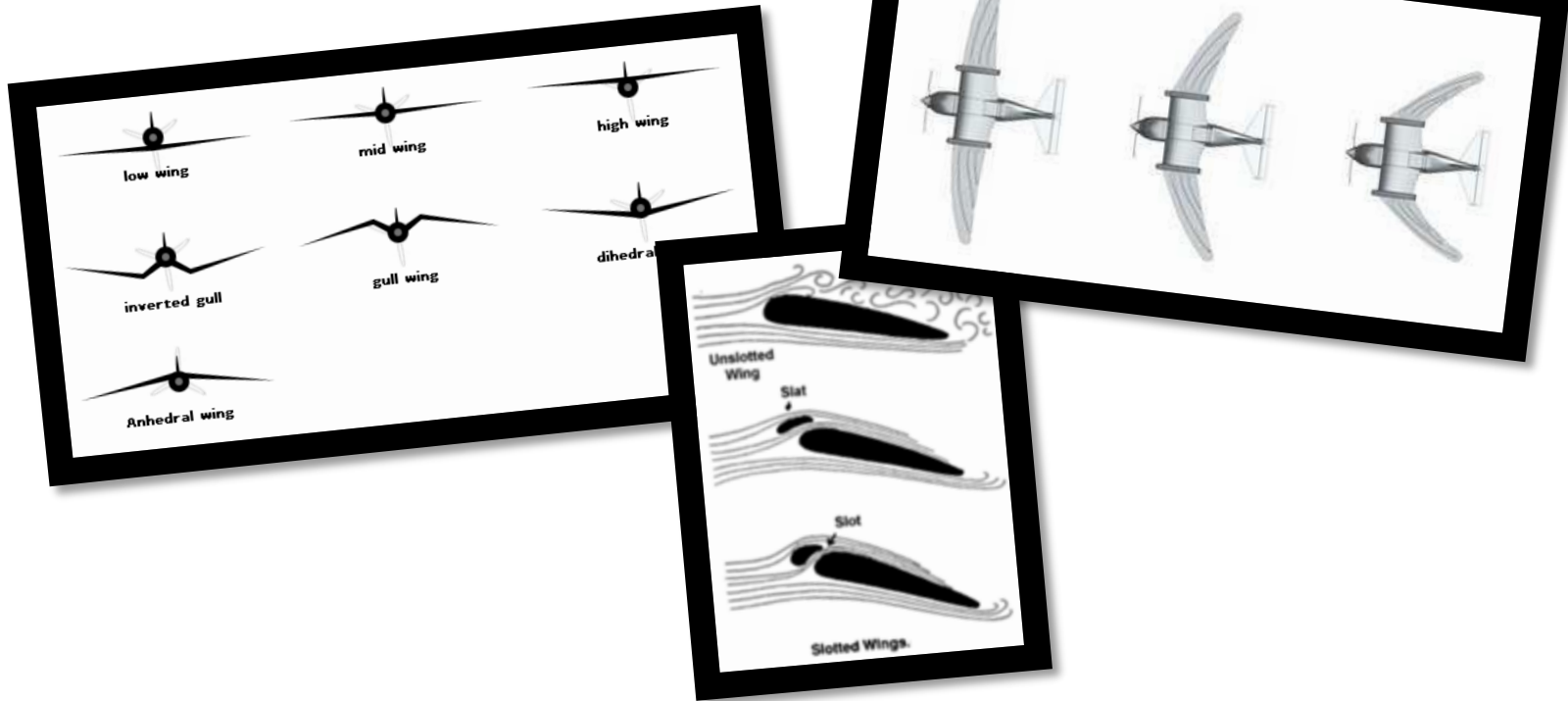


# Cíl

- I. Úvod do optimalizace
- II. Genetické algoritmy
- III. Paralelizace GA
- IV. **Typické příklady GA**

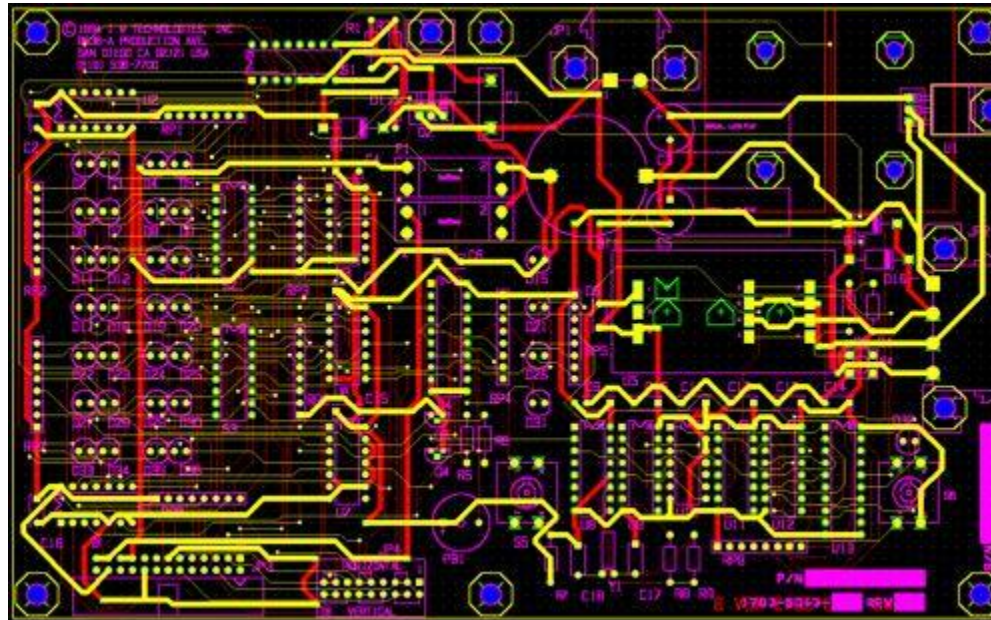
# Příklady použití

- Návrh křídel letadla [8][9]



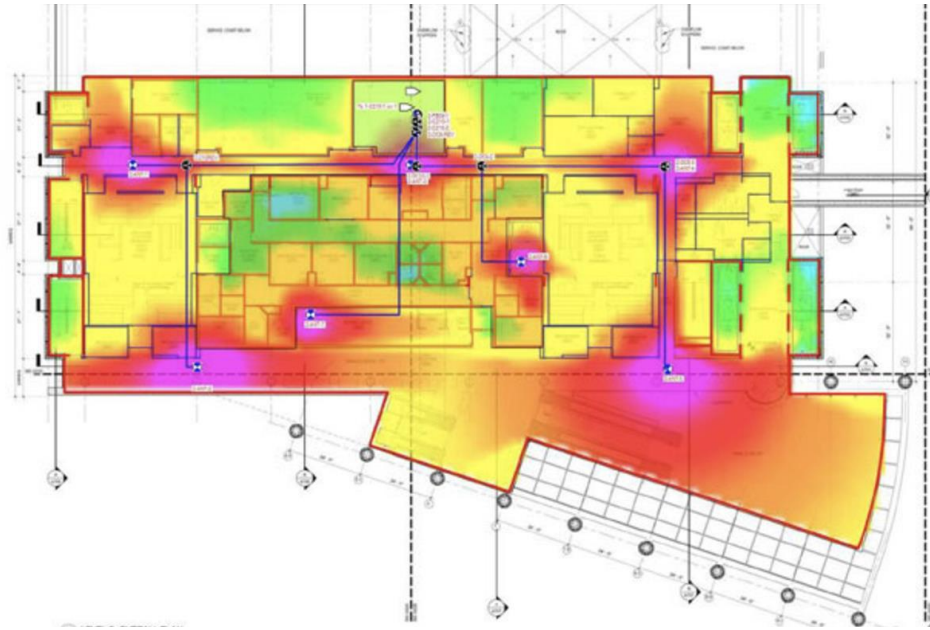
# Příklady použití

- Návrh elektronických obvodů, známé jako „evolvable hardware“[2]



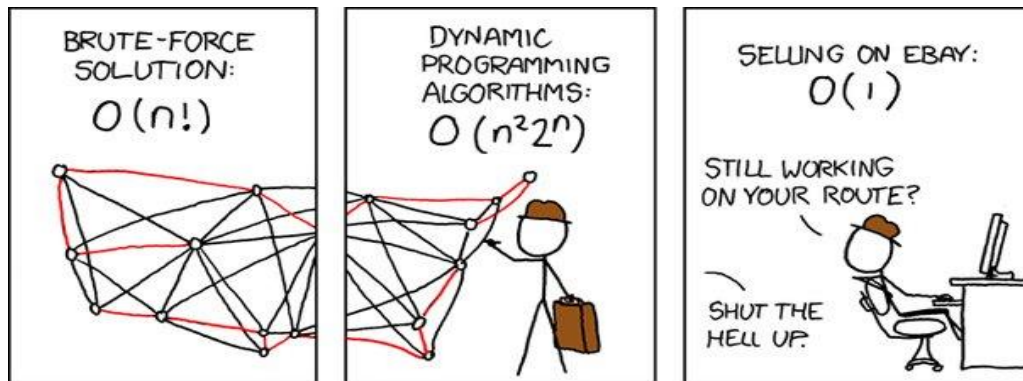
# Příklady použití

- Wireless sensor/ad-hoc networks [4].



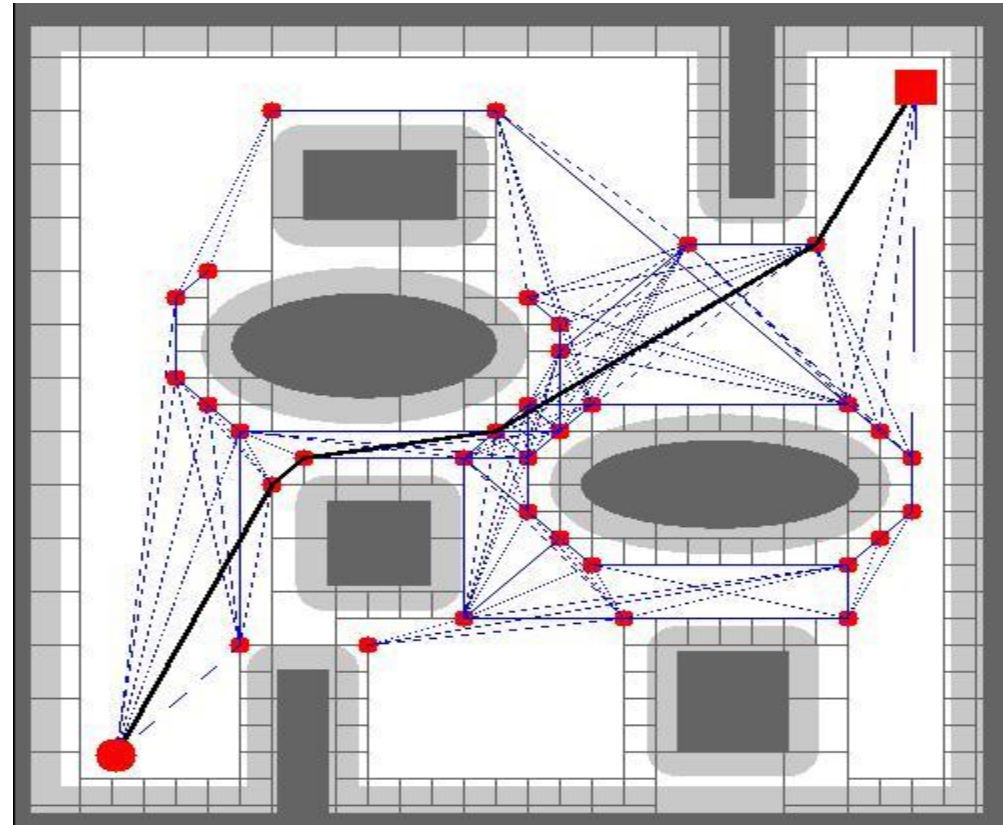
# Problém obchodního cestujícího

- Nalezení nejkratší cestovní trasy pro obchodního cestujícího mezi  $N$  městy
- Struktura chromozomu:  
Brno | Ostrava | Praha | Plzeň ...



# Nalezení cesty pro robota

- Nalezení cesty pro robota v terénu z bodu A do bodu B tak, aby se úspěšně vyhnul všem překážkám
- Struktura chromozomu:  
Left | Right | Down | Up...





# Plánování pracovních úloh

	machine sequence					
job1	3	1	2	4	6	5
job2	2	3	5	6	1	4
job3	3	4	6	1	2	5
job4	2	1	3	4	5	6
job5	3	2	5	6	1	4
job6	2	4	6	1	5	3

	processing times					
job1	3	6	1	7	6	3
job2	10	8	5	4	10	10
job3	9	1	5	4	7	8
job4	5	5	5	3	8	9
job5	3	3	9	1	5	4
job6	10	3	1	3	4	9

job1<job2: 1 1 0 1 0 0  
 job1<job3: 0 1 1 0 0 0  
 job1<job4: 1 1 0 0 1 0  
 job1<job5: 1 1 1 1 0 0  
 job1<job6: 1 1 0 0 0 0  
 job2<job3: 1 0 1 0 0 0  
 job2<job4: 1 1 1 1 0 0  
 job2<job5: 1 1 1 1 1 1  
 job2<job6: 1 1 1 0 0 0  
 job3<job4: 1 1 1 0 0 1  
 job3<job5: 1 1 1 1 0 0  
 job3<job6: 1 1 1 1 0 1  
 job4<job5: 1 1 0 1 0 0  
 job4<job6: 1 1 1 0 1 0  
 job5<job6: 1 0 1 0 0 0

	job sequence					
machine1	1	4	3	6	2	5
machine2	2	4	6	1	5	3
machine3	3	1	2	5	4	6
machine4	3	6	4	1	2	5
machine5	2	5	3	4	6	1
machine6	3	6	2	5	1	4

Machines: M=6, Jobs: N=6

chromosome

110100|011000| ... |111010|101000

$M*N*(N-1)/2 = 90$  bits

	time-slots																													
machine1																														
machine2																														
machine3																														
machine4																														
machine5																														
machine6																														

# Hledání podezřelých osob

- Hledání podezřelé osoby v obsáhlé databázi obličejů, svědek popisuje pachatele, aby došlo k jeho přesné identifikaci



- Struktura chromozomu:

skull_shape				hair_cut				spectacles				beard			
0	0	1	0	1	0	1	1	0	0	1	0	0	1	1	0
hair_colour								eye_colour							



# Výběr příznaků pro umělou inteligenci



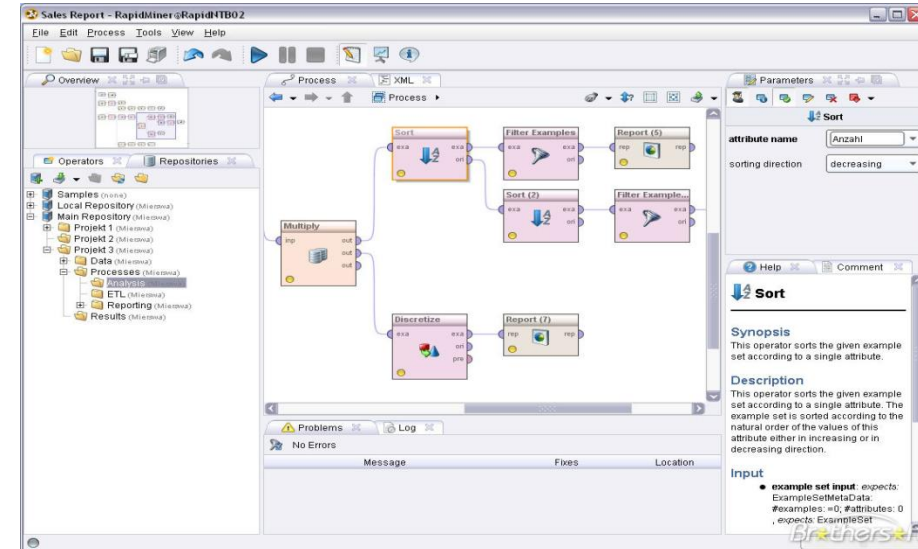
- *Které příznaky vybrat, pokud je atributů velký počet?*
- *10 000 000 ... kombinace bez opakování*

# Výběr příznaků pro umělou inteligenci

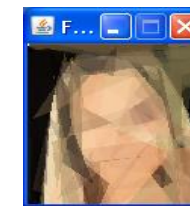
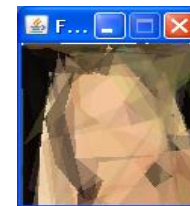
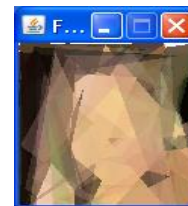
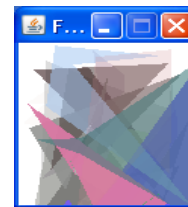
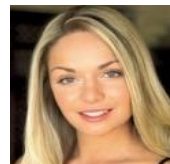
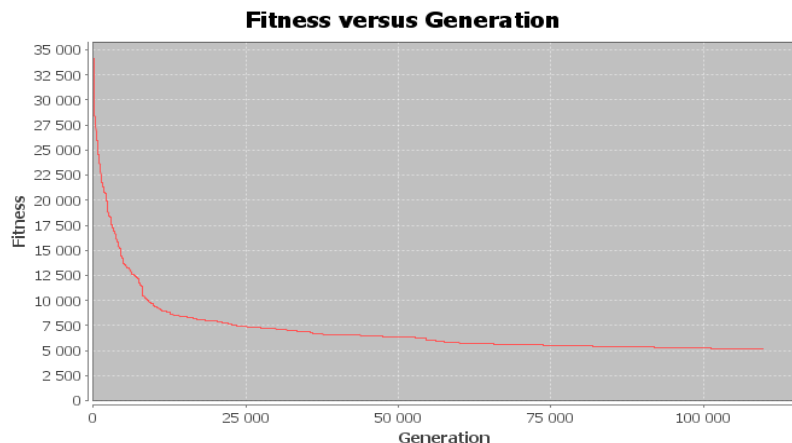
Umělá  
inteligence



- ***Jaké jsou optimální nastavení parametrů jednotlivých operátorů?***



# Rozložení polygonů (ne až tak typické 😊)



POLYGON 1														POLYGON 2													
X1	Y1	X2	Y2	X3	Y3	X4	Y4	X5	Y5	Color RED	Color GREEN	Color BLUE	Transparency	X1	Y1	X2	Y2	X3	Y3	X4	Y4	X5	Y5	Color RED	Color GREEN	Color BLUE	Transparency
88	12	88	12	139	30	88	12	88	12	139	30	1	40	88	12	88	12	139	30	88	12	88	12	139	30	1	40

POLYGON 50																											
X1	Y1	X2	Y2	X3	Y3	X4	Y4	X5	Y5	Color RED	Color GREEN	Color BLUE	Transparency														
88	12	88	12	139	30	88	12	88	12	139	30	1	40														

# Děkuji za pozornost