



Teorie kognitivních systémů

12 Genetické algoritmy

- Biologická evoluce, genetika
- Evoluční výpočty (EC)
- Kódování genotypu
- Základní algoritmus EC
- SOEA vs MOEA
- Genetické algoritmy (GA)
- Operátory GA
- Parametry GA
- Aplikace





Evoluční algoritmy (*Evolutionary Algorithms*)

Úvodní informace

Definice: Evoluční algoritmus (EA) – metaheuristický optimalizační postup založený na tvorbě a práci s **populacemi řešení**, který využívá biologické mechanismy jako např. **dědičnost, křížení, mutace, kompetitivní výběr** – přežití nejsilnějšího).



Výhody:

- EA se chovají velmi dobře v extrémně široké škále úloh;
- charakter „černé skřínky“ vyžaduje splnění jen několika málo předpokladů o řešené úloze (to je podobné jako u ANN);
- stanovení cíle optimalizace obvykle nevyžaduje příliš hluboké porozumění řešenému problému (narozdíl od jiných postupů, kde je nutné zkonstruovat nějakou heuristiku ručně) – není nezbytné mít k dispozici analytický předpis optimalizace.



Evoluční algoritmy

Trocha historie ...

- První experimenty s počítačovou simulací biologické evoluce prováděli kolem roku 1980 **Stephen F. Smith** – profesor robotiky na Carnegie Mellon University, a později (1985) **Nichael L. Cramer** – tehdy výzkumník u Texas Instruments.
- Za duchovního otce této oblasti umělé inteligence je považován **John R. Koza** – profesor na Stanford University; v roce 1992 vydal knihu ***Genetic Programming: On the Programming of Computers by Means of Natural Selection.***

Ke stažení online:

<http://www.ru.lv/~peter/zinatne/ebooks/MIT - Genetic Programming.pdf>



Evoluční algoritmy

Výchozí úvahy

Charles Darwin: *O původu druhů (On the Origin of Species)*

— geniální vědecké dílo, vyd. 1859, identifikuje principy přirozeného výběru jako hnací sílu přírodního vývoje.

Darwinovy závěry lze shrnout do 10 postřehů/závěrů:

- Jedinci daného druhu jsou plodní více, než je nezbytné, aby mohli počít více potomků, než se může dožít dospělého věku.
- Bez přítomnosti vnějších vlivů (katastrofy, člověk) zůstává velikost populace daného druhu zhruba konstantní.
- Bez přítomnosti vnějších vlivů jsou zásoby potravy omezené, ale vývoj jejich množství je v čase stabilní.
- V případě pohlavně se rozmnožujících druhů jsou každí dva jedinci **odlišní**.



Evoluční algoritmy

Výchozí úvahy

- Tato odlišnost ovlivňuje jejich schopnost přežít do dospělého věku a mít potomky.
- Značná část specifické výbavy jedince (tj. té odlišnosti od jiných jedinců, která mu dovolila přežít) **se dědí**.
- Jedinci s horší dědičnou výbavou (odlišní „špatným“ způsobem) mají nižší pravděpodobnost přežití a rozmnožení.
- Přeživší jedinci se rozmnoží a pravděpodobně předají svou specifickou výbavu svým potomkům.
- Druh se tak bude pomalu měnit a stále více se přizpůsobovat konkrétnímu prostředí, což může dokonce vyústit ve **vznik zcela nového druhu**.

EA = abstrakce popsáného biologického procesu: změny v sémantice popisu řešení problému řízené orientací na cíl (*goal-driven*).



Evoluční algoritmy

Trocha moderní genetiky

- Organismus (jedinec určitého druhu) se skládá z **buněk**.
- V jádru každé buňky je stejná sada **chromozomů**.
- Chromozom = řetězec DNA, sloužící jako generativní model organismu (popisuje, jak bude organismum vypadat).
- Chromozom je tvořen **geny**, bloky DNA.
- Každý gen kóduje určitý **znak** organismu (např. barvu očí).
- Konkrétní realizace (tj. tvar, hodnota) genu se nazývá **alela**, tj. alely genu pro barvu očí jsou „modré“, „hnědé“, „zelené“, ap.
- Každý gen má v chromozomu svojí pozici, tzv. **lokus**.
- Veškerý genetický materiál jedince (kompletní sekvence DNA jedné sady chromozomů, všechny geny i nekódující sekvence) se nazývá **genom** (lidský genom má 20 488 genů).
- Konkrétní množina alel, tj. specifická instance genomu, se nazývá **genotyp**.





Evoluční algoritmy

Trocha moderní genetiky

genotyp + vliv prostředí → **fenotyp**

- **Fenotyp** je soubor vlastností/znaků, jenž je vytvářen i prostředím, ve kterém se jedinec pohybuje.

Jedinci se stejným genotypem nemusí být stejným fenotypem, a naopak, jedinci se stejným fenotypem nemusí být stejným genotypem.

- Genetická informace kódovaná určitým genem se nemusí nutně projevit jako znak.
- Informace nesená genomem specifikuje vlastnosti daného organismu (vzhled, inteligence, temperament) a také o jaký organismus je jedná (člověk, prase, atp.).





Evoluční algoritmy

Od přírodní podoby k formalismu

Řešení problému (př. hledání kořene rovnice) = výběr jedinců (kandidátů řešení, tj. např. čísel, která mohou být kořenem) ze **stavového prostoru** (*Search/State Space*) takových, že jsou ve smyslu zvoleného optimalizačního kritéria vhodní jako řešení daného problému, tj. z evolučního pohledu jsou **životaschopní**.

Na takovou úlohu lze také nazírat tak, že se snažíme křížením méně vhodných kandidátů (např. čísel, která nejsou kořenem zadáné rovnice) „vypěstovat“ nějakého vhodnějšího.

Základní problémy evolučních algoritmů:

- reprezentace genotypu
- vyjádření životaschopnosti (zda bude jedinec připuštěn ke křížení s dalšími jedinci a tak vytvoří novou–lepší generaci)





Evoluční algoritmy

Reprezentace genotypu – kódování

Zakódování genotypu (tedy instancí chromozomů) jedince je **silně závislé na typu řešeného problému**. Neexistuje obecné řešení...

Běžné způsoby kódování:

- (i) **binární kódování (*Binary Encoding*)**
 - binární kódování genů je typické pro **genetické algoritmy** (což je podmnožina EA).
- (ii) **permutační/numerické kódování (*Permutation Encoding*)**
- (iii) **kódování hodnotou (*Value Encoding*)**
- (iv) **kódování stromem (*Tree Encoding*)**

U některých typů kódování je pak velmi nesnadné definovat evoluční operátory, tj. způsoby, jakými vzniká nová generace ze současné...



Evoluční algoritmy

Binární kódování

Binární kódování — každý chromozom je reprezentován řetězcem bitů (rychlé, jednoduché, představitelné, snadná implem.)

chromozom 1	0	0	1	0	1	1	1	0	1	0
chromozom 2	1	0	1	1	0	0	1	1	0	1

Binární kódování genů je typické a nejběžnější pro **genetické algoritmy** (podmnožina EA), mj. proto, že bylo používáno při prvních experimentech s GA.

Příklad problému: **Balení evakuačního zavazadla (krosny)**

Problém: Mějme předměty určitého objemu a hodnoty. Krosna má omezenou kapacitu. Vybíráme předměty tak, aby obsah krosny měl co největší hodnotu, ale nepřekročil její kapacitu.

Kódování: Každý bit odpovídá jednomu předmětu.



Evoluční algoritmy

Permutační kódování

Permutační kódování — chromozom je tvořen posloupností čísel (přirozených), obvykle s významem indexu.

chromozom 1	5	2	8	3	1	7	9	4	6	0
chromozom 2	8	4	2	1	9	5	7	6	0	3

Používá se **výhradně** pro problémy hledání optimálního uspořádání (Travelling Salesman). Často je třeba po křížení a mutaci provést korekci (když vznikne z hlediska úlohy nesmyslný stav).

Příklad problému: **Travelling Salesman Problem** (TSP)

Problém: Jsou dány města a vzdálenosti mezi nimi. Obchodní cestující musí navštívit všechny, ale chce cestovat co nejméně. Hledá se tedy optimální sekvence návštěvy měst.

Kódování: Chromozom = posloupnost návštěvy měst.



Evoluční algoritmy

Kódování hodnotou

Kódování hodnotou — chromozom je tvořen množinou hodnot (reál. čísla, znaky, výčtové konstanty).

chromozom 1	3.5	2.9	1.8	0.2	9.4	2.1	7.6	3.2	4.7	3.1
chromozom 2	G	C	D	A	E	H ^m	G	D	G	A

Vhodné pro nestandardní, komplikované úlohy. Nevýhodou je, že se obvykle musí nadefinovat specifické provedení křížení a mutace.

Příklad problému: **Trénování neuronové sítě**

Problém: Je dána neuronová síť s určitou architekturou. Chceme najít váhy synapsí (natreňovat síť) tak, aby dávala požadované výsledky.

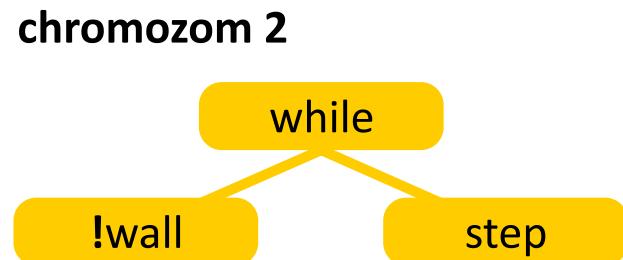
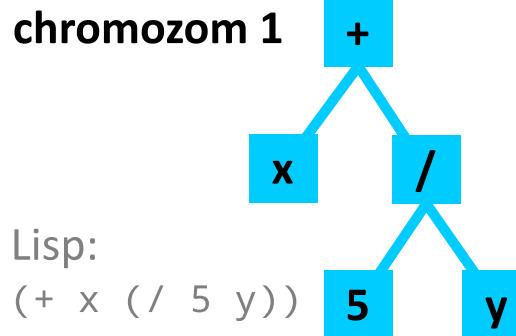
Kódování: Chromozom = hodnoty synaptických vah.



Evoluční algoritmy

Kódování stromem

Kódování stromem — chromozom je strom nějakých objektů. Používá se především pro automatický vývoj výrazů či programů, tzv. **genetické programování**.



Příklad problému: Hledání funkce podle tabulky hodnot

Problém: Je dána tabulka hodnot nezávislé (x) a závislé (y) proměnné. Hledáme funkci, která „vyprodukuje“ hodnoty v tabulce.

Kódování: Chromozom = rozkladový strom předpisu funkce.



Evoluční algoritmy

Funkce zdatnosti (*Fitness Function*)

Specifická podoba **cenové funkce** (*Cost/Objective Function*), používaná v EA. Vyjadřuje kvalitu nalezeného kandidáta řešení, tj. např. jak **blízko je získaný kandidát k cíli optimalizace**.

Neexistuje jeden konkrétní předpis, liší se podle řešené úlohy a zvoleného kódování genotypu...

- Je to **metrika**, tj. měla by mít vlastnosti metriky.
- Mnohy není analyticky vyjádřitelná, pak jí nahrazuje nějaký její odhad.

Příklad: Při řešení hry „15“ to může být počet nesprávně umístěných kamenů. Další příklady viz demonstrační programy.



Evoluční algoritmy

Stavový prostor a řešení úlohy

Prostor řešení dané úlohy \mathbb{G} je abstrakcí množiny všech možných řetězců DNA v přírodě a jeho prvky $g \in \mathbb{G}$ jsou abstrakcí přírodních **genotypů**.

Každý jedinec je instancí svého genotypu.

Kandidáti řešení úlohy (čili **fenotypy**) $x \in \mathbb{X}$ ve stavovém prostoru řešené úlohy \mathbb{X} jsou instancemi genotypů vytvořenými tzv. genotypově-fenotypovým mapováním: $x = gpm(g)$.

Jejich schopnost přežít (**Fitness**) se vyhodnocuje podle cenové funkce (*Objective Function*), která se při řešení úlohy optimalizuje (což směruje evoluci řešení určitým směrem).





Evoluční algoritmy

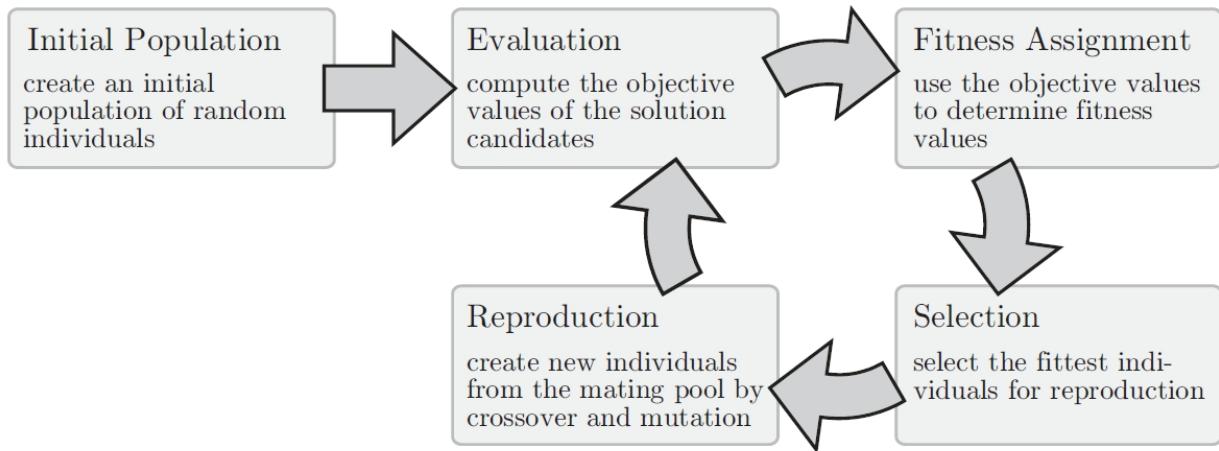
Kanonické paradigma

EA lze na 2 velké skupiny:

SOEA = Single Objective Evolutionary Algorithm

MOEA = Multi-Objective Evolutionary Algorithm

Základní cyklus je pro obě varianty stejný:



(obrázek převzat z knihy Thomas Weise: Global Optimization Algorithms)



Evoluční algoritmy

SOEA vs MOEA

SOEA – jediná cenová funkce, protože algoritmus hledá optimální řešení s ohledem na jediné kritérium (např. „vyšlechtění“ antény, která nejlépe vyzařuje v daném frekv. spektru)

MOEA – fitness funkce (schopnost přežít) každého jedince se vypočítává podle více cenových funkcí, protože algoritmus hledá optimální řešení podle více kritérií, tzv. výsledné řešení bude suboptimální, ale splní (aspoň částečně) více kritérií (např. např. „vyšlechtění“ antény, která nejlépe vyzařuje v daném frekv. spektru a je co nejmenší)

(obrázek: anténa mikrosatelu NASA New Millenium ST5, tvar byl navržen evolučním algoritmem tak, aby měl optimální směrovou charakteristiku vyzařování, zdroj: wikipedia.org)





Genetické algoritmy

Úvodní informace

Genetický algoritmus (GA) – heuristický postup hledání řešení aplikací principů **evoluční biologie** (křížení, mutace, kompetitivní výběr):

- vhodný pro řešení složitě modelovatelných problémů, pro které není znám deterministický algoritmus řešení;
- zejména se nasazuje na nelineární optimalizační úlohy a prohledávání rozsáhlých stavových prostorů;
- patří mezi **evoluční algoritmy** či **evoluční výpočty** (*Evolutionary Computation*) – techniky napodobující přírodní výběr.





Genetické algoritmy

Obecný předpis algoritmu

1. **[start]** – Generujeme náhodnou populaci Pop_{curr} o n chromozomech (možná řešení úlohy, tj. kandidáti ze stavového prostoru).
2. **[fitness]** – Vyjádříme fitness $f(\mathbf{x})$, \forall chromozomy $\mathbf{x} \in \text{Pop}_{\text{curr}}$.
3. **[nová populace]** – Vytvoříme novou populaci Pop_{new} :
 - (i) **[výběr]** – Vybereme 2 rodiče z populace podle jejich fitness (čím vyšší fitness, tím větší šance, že bude vybrán)
 - (ii) **[křížení]** – Zkřížíme (s danou p-stí křížení) vybrané rodiče a tak vytvoříme potomky.
 - (iii) **[mutace]** – Potomky zmutujeme (s danou p-stí mutace).
 - (iv) **[přijetí]** – Jsou-li „v pořádku“, zařadíme je do Pop_{new} .
4. **[nahrazení]** – Generaci Pop_{curr} nahradíme vytvořenou generací Pop_{new} pro další iteraci, $\text{Pop}_{\text{curr}} \leftarrow \text{Pop}_{\text{new}}$.
5. **[test]** – Bylo-li dosaženo cíle, **stop**, **return** nejlepšího z Pop_{curr} .
6. **[iterace]** – Pokračujeme krokem 2.



Genetické algoritmy

Operátory GA – Křížení (Crossover)

Náhodně vybereme bod překřížení
(může jich být i více) ...

bod překřížení
(Crossover Point)

chromozom 1 – otec

1	0	1	1	0	0	1	0	1	0
---	---	---	---	---	---	---	---	---	---

chromozom 2 – matka

0	0	1	0	1	1	1	1	0	1
---	---	---	---	---	---	---	---	---	---

... a do chromozomu potomka kopírujeme obsah chromozomu 1. rodiče až po bod překřížení, pak pokračujeme obsahem chromozomu 2. rodiče:

potomci {

0	0	1	0	1	1	1	0	1	0
1	0	1	1	0	0	1	1	0	1



Genetické algoritmy

Operátory GA – Křížení (*Crossover*)

- Způsobů provedení **křížení** může být celá řada:
 - (i) 1 či více bodů překřížení,
 - (ii) způsob náhodného výběru polohy bodu překřížení,
 - (iii) optimalizace vzdálenosti bodů překřížení (je-li jich více),
 - (iv) volba počtu rodičů.
- **Křížení** může být i velice komplikované a zásadním způsobem závisí na zvoleném způsobu kódování chromozomů.
- Specifické řešení překřížení vytvořené na míru pro konkrétní problém může dramaticky ovlivnit výkon genetického algoritmu.

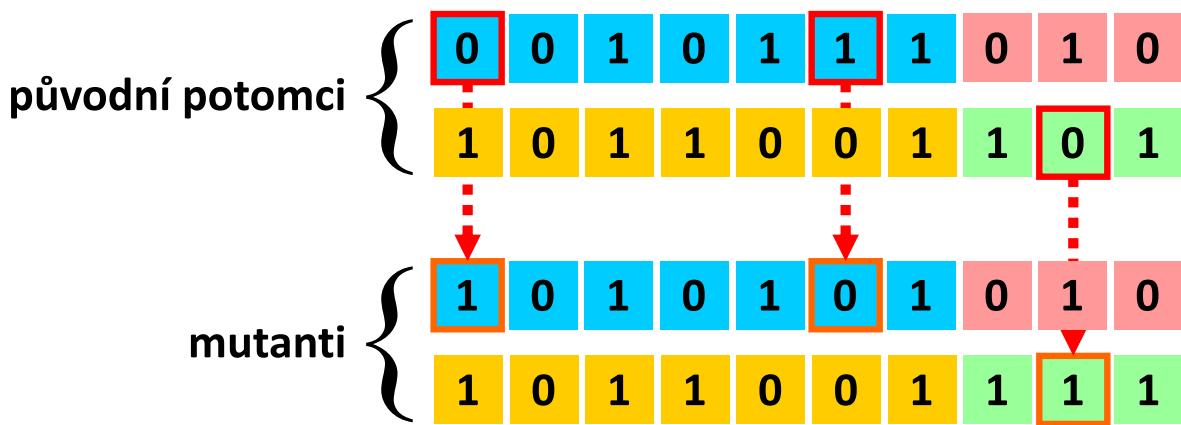
Je třeba za každou cenu zabránit stavu, kdy budou křížením opakovaně vznikat stejné populace!



Genetické algoritmy

Operátory GA – Mutace (*Mutation*)

Mutace se provádí **vždy až po křížení** a jejím hlavním smyslem je zabránit uváznutí algoritmu v lokálním optimu řešeného problému.



Binární kódování → inverze několika náhodně vybraných bitů.
Numerické kódování (permutace) → mutaci lze realizovat prohozením hodnot např. 2 sousedních genů.



Genetické algoritmy

Parametry GA

Genetické algoritmy mají 2 základní parametry:

(i) **pravděpodobnost křížení** – může být 0%, tj. potomek je dokonalou kopíí rodiče; může být 100%, tj. všichni potomci vznikají křížením; a může (obvykle) být někde mezi...

Je-li 0%, celá nová generace je vytvořena z dokonalých kopíí chromozomů původní populace, ale **to neznamená, že je stejná!** (mutace)

Křížení se provádí proto, aby nové chromozomy měly ty „dobré“ části původních a snad vznikly i nějaké „lepší“...

Rozumné je nechat alespoň část populace přežít do nové generace (nezničit všechny původní chromozomy).





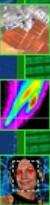
Genetické algoritmy

Parametry GA

Genetické algoritmy mají 2 základní parametry:

(ii) **pravděpodobnost mutace** – může být 0%, tj. potomek je po případném křížení už ponechán v dosaženém stavu; může být 100%, tj. celý chromozom se změní.

Mutace by neměla nastávat moc často (stejně jako v přírodě), protože tím by vlastně genetický algoritmus degradoval na prosté **náhodné prohledávání** stavového prostoru, které jednak nemusí vůbec vést k nalezení řešení a jednak je velmi pomalé.





Genetické algoritmy

Parametry GA

Další parametry:

(iii) **velikost populace** – množství chromozomů v populaci
(v jedné generaci).

Pokud je chromozomů **málo**, GA má jen omezené možnosti jejich křížení, a tak se prohledá jen malá část stavového prostoru.

Pokud je chromozomů **mnoho**, GA se značně zpomaluje.

Výzkumem bylo zjištěno, že po dosažení určitého limitu (který závisí především na kódování chromozomů a charakteru úlohy) už nevede zvětšování populace k urychlení řešení problému.



Genetické algoritmy

Výběr chromozomů pro křížení

Podle Darwinovy evoluční teorie **přežijí jen nejsilnější a ti pak počnou potomky...**

Metod výběru „nejsilnějších“ chromozomů pro potřeby GA existuje mnoho:

- **výběr ruletovým kolem** (*Roulette Wheel Selection*)
- **Boltzmanův výběr** (*Boltzman Selection*)
- **výběr turnajem** (*Tournament Selection*)
- **žebříčkový výběr** (*Rank Selection*)
- **výběr z ustáleného stavu** (*Steady-State Selection*)
- **elitářství** (*Elitism*)





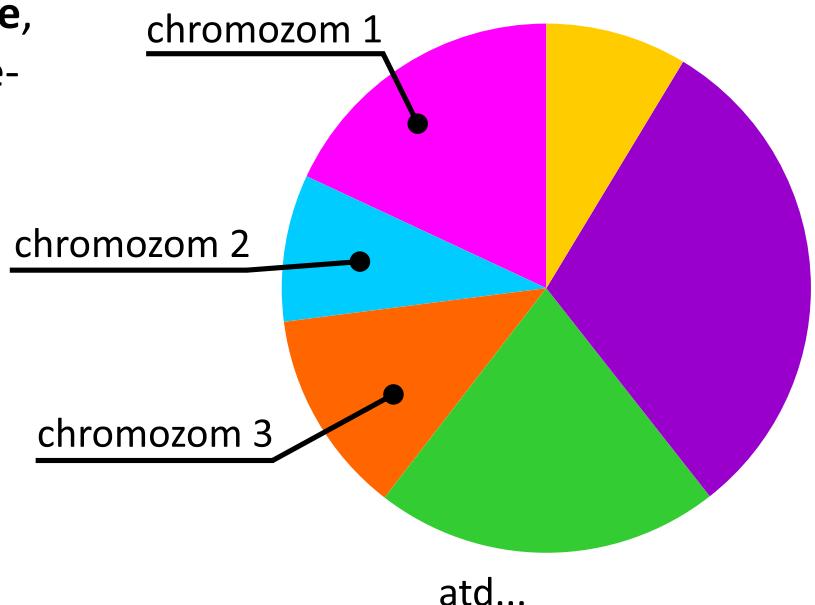
Genetické algoritmy

Výběr ruletovým kolem

Rodiče jsou vybráni podle své životaschopnosti: Čím lepší daný chromozom je, tím pravděpodobněji bude vybrán ke křížení.

Životaschopnost rodiče určuje **velikost výseče**, kterou obsadí na ruletovém kole.

Pak se „hodí kulička“ a vybere nějaký chromozom (non-uniform random)

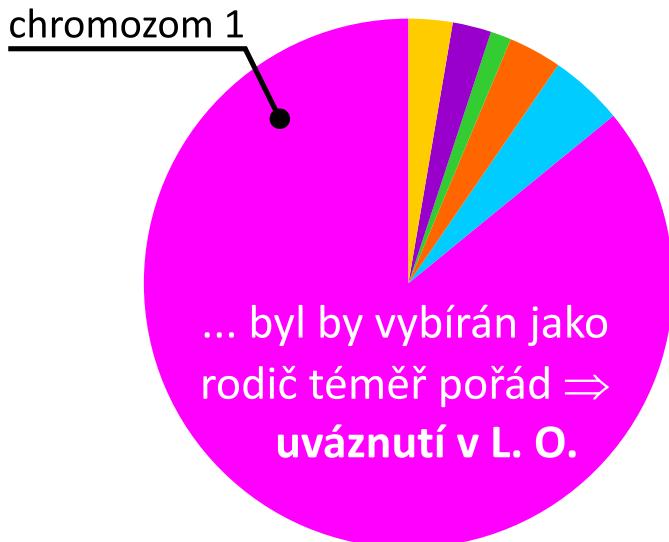




Genetické algoritmy

Žebříčkový výběr

Ruletový výběr nebude fungovat dobře tehdy, když se životaschopnost jednotlivých chromozomů výrazně liší...



Chromozomy se seřadí podle životaschopnosti a nejhoršímu se dá váha 1, druhému

nejhoršímu 2, atd... Nejlepší bude mít váhu N. Součet vah pak představuje 100% (celé ruletové kolo) a každý chromozom tam zabírá plochu odpovídající poměru své váhy k součtu všech vah.



Genetické algoritmy

Výběr z ustáleného stavu

Klíčová myšlenka: Značná část chomozomů přežije do další generace (v přírodě).

Realizace výběru z ustáleného stavu v GA: Z každé generace se vybere několik nejlepších (s nejvyšší životaschopností) chromozomů k vytvoření nové generace (počet N vybraných je parametrem algoritmu).

Naopak N nejhorších (s nejnižší životaschopností) se odstraní a nahradí potomky těch vybraných N nejlepších.

Zbytek populace přežije do další generace.



Genetické algoritmy

Elitářství

Klíčová myšlenka: Když se křížením a mutacemi tvoří nová generace, je značná pravděpodobnost, že přijdeme o nejlepší chromozomy...

Realizace elitářství v GA: N nejlepších chromozomů se nejprve beze změny zkopíruje do nové generace. Zbytek nové generace se vytvoří klasickým způsobem.

Elitářství velmi zásadně zvyšuje výkon GA, protože zabraňuje ztrátě nejlepších dosud nalezených kandidátů řešení.





Aplikace genetických algoritmů

Typické problémové oblasti

GA se nasazují na **velmi obtížně řešitelné problémy** (NP-hard), **strojové učení** a automatické **generování programů** (tzv. **genetické programování** – nejčastěji v LISPu).

Typické oblasti aplikace GA:

- nelineární dynamické systémy – predikce, analýza dat
- návrh topologie a trénování umělých neuronových sítí
- výpočty trajektorií robotů, plánování strategií
- logistika, plánování a optimalizace sekvencí
- hledání tvarů složitých organických molekul (proteinů)
- generování funkcí pro práci s obrázky (denoising, ap.)

V oblasti mimovědeckých aplikací pak zejména v umění: generování **obrazů**, **video** a **hudby**.





Aplikace genetických algoritmů

Výhody a nevýhody nasazení



Výhody:

- přirozeně paralelizovatelné
- odolnost vůči uváznutí v lokálních extrémech
- snadná implementace



Nevýhody:

- výpočetní náročnost
- pomalá konvergence
- citlivost na volbu vhodné funkce životaschopnosti
- citlivost na volbu mapování stavu úlohy na obsah chromozomu

Kam dál?

<http://aitopics.org/topic/evolutionary-algorithms>

