



MATEMATICKO-FYZIKÁLNÍ  
FAKULTA  
Univerzita Karlova

# Přírodou inspirované prohledávací algoritmy

*Roman Neruda, Martin Pilát*

19. ZÁŘÍ 2017



# *Před úvodem*

*Jak to je?*

OSNOVA PREDNASKY TAK JAK JE TED

- Evoluční algoritmy
- Jednoduchý genetický algoritmus
- Teorie schémat
- Reprezentace a operátory v GA
- Evoluce kooperace
- Evoluční strategie
- Diferenciální evoluce
- Particle swarm optimization
- Evoluční strojové učení
- Vícekriteriální optimalizace
- Evoluční kombinatorická optimalizace
- Ladění, řízení, metaevoluce
- Teorie EVA podruhé
- Evoluční programování
- Genetické programování
- Neuroevoluce
- Memetické algoritmy
- Dynamické krajiny fitness
- Tabu search, scatter search
- Biologicky věrnější evoluce

*Jak to bude?*

*Pok pok pokusy*

SKORO TO TU MUZEME SMAZAT NE?

---

**procedure** EVOLUČNÍ ALGORITMUS $t \leftarrow 0$ *Inicializuj* populaci  $P_t$  náhodně vygenerovanými jedinci*Ohodnoť* jedince v populaci  $P_t$ **while** neplatí *kritérium ukončení* **do**vyber z  $P_t$  rodiče  $P'_{t+1}$  *Rodičovskou selekcí**Rekombinací* rodičů vzniknou potomci  $P'_{t+1}$ *Mutuj* potomky  $P'_{t+1}$ *Ohodnoť* potomky  $P'_{t+1}$ *Enviromentální selekcí* vyber  $P_{t+1}$  z  $P_t$  a  $P'_{t+1}$  $t \leftarrow t + 1$ **end while****end procedure**

---

Algoritmus 1: Schéma evolučního algoritmu

# Evoluční algoritmy

## Evoluce, geny a DNA

### Obecné schéma evolučního algoritmu

Oblast evolučních výpočtů či algoritmů (v angličtině evolutionary computing) zastřešuje několik proudů, které se zpočátku vyvíjely samostatně. Za prehistorii této disciplíny lze považovat Turingovy návrhy na využití evolučního prohledávání z roku 1948 <sup>1</sup> a Bremermannovy první pokusy o implementaci optimalizace pomocí evoluce a rekombinace z roku 1962 <sup>2</sup>. Během šedesátých let se objevily tři skupiny výzkumníků, kteří nezávisle na sobě vyvíjely a navrhly své varianty použití evolučních principů v informatice. Holland publikoval v roce 1975 svůj návrh genetických algoritmů <sup>3</sup>, zatímco skupina Fogela a spolupracovníků vyvinula metodu nazvanou evoluční programování <sup>4</sup>. Nezávisle na nich přišli Rechenberg a Schwefel v Německu na metodu nazvanou evoluční strategie <sup>5</sup>. Až do přelomu osmdesátých a devadesátých let existovaly tyto směry bez výraznější interakce, ale poté se spojily do obecnější oblasti evolučních algoritmů. V té době Koza vytváří metodu genetického programování, Dorigo publikuje disertaci s návrhem mravenčích optimalizačních algoritmů a vznikají první pokusy o aplikaci evoluce na vývoj umělých neuronových sítí.

U zrodu různých variant evolučních algoritmů stála inspirace přírodními jevy, konkrétně jde o Darwinovu teorii přírodního výběru a zjednodušené principy genetiky, které poprvé načrtl Mendel. Z genetiky se evoluční algoritmy inspirovaly diskretní reprezentací genotypu, z biologické evoluční teorie používají Darwinovu myšlenku o výběru jedinců v prostředí s omezenými zdroji, který závisí na míře přizpůsobení se jedinců danému prostředí.

Základní obecnou myšlenku evolučních algoritmů lze vyjádřit následujícím způsobem. Mějme populaci jedinců v prostředí, které určuje jejich úspěšnost – fitness. Tito jedinci navzájem soupeří o možnost reprodukce a přežití, která závisí právě na hodnotě fitness. Jde tedy o množinu kandidátů na řešení problému definovaného prostředím. Způsoby reprezentace jedinců, jejich výběru a rekombinace závisí na konkrétním dialektu evolučních algoritmů, které probereme vzápětí.

Základní princip fungování evolučních algoritmů je tedy následující <sup>6</sup>. Na začátku algoritmu vygenerujeme (nejčastěji náhodně) první iniciální populaci jedinců. Všechny jedince v populaci ohodnotíme ohodnocovací funkcí. Hodnota této funkce určuje šanci výběru jedinců během rodičovské selekce. Vybraní jedinci jsou potom rekombinováni pomocí rekombinačního operátoru, který typicky ze dvou jedinců vytváří jednoho či dva potomky, a

<sup>1</sup>

Missing ref.

<sup>2</sup>

Missing ref.

<sup>3</sup> John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992

<sup>4</sup> David B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ, USA, 1995

<sup>5</sup> Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies – a comprehensive introduction. 1(1):3–52, May 2002

<sup>6</sup> Agoston E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, 2003

---

**procedure** EVOLUČNÍ ALGORITMUS

---

 $t \leftarrow 0$ *Inicializuj* populaci  $P_t$  náhodně vygenerovanými jedinci*Ohodnoť* jedince v populaci  $P_t$ **while** neplatí *kritérium ukončení* **do**vyber z  $P_t$  rodiče *Rodičovskou selekcí**Rekombinací* rodičů vzniknou potomci*Mutuj* potomky*Ohodnoť* potomky*Enviromentální selekcí* vyber  $P_{t+1}$  z  $P_t$  a potomků $t \leftarrow t + 1$ **end while****end procedure**

---

Algoritmus 2: Schéma evolučního algoritmu

pomoci operátoru mutace, který typicky provádí drobné změny jednoho jedince. Tímto postupem si vytvoříme množinu nových kandidátů řešení, a tito noví jedinci potom soutěží s původními jedinci o místo v nové populaci. Výběr jedinců do nové populace (tedy jakési slití rodičů a potomků) má na starosti enviromentální selekce beroucí v úvahu fitness jedinců a případně další ukazatele jako je například stáří jedinců. Tím je vytvořena nová generace a tento cyklus pokračuje do splnění určitého kritéria ukončení, což je nejčastěji dostatečně dobrý nejlepší jedinec nebo předem určený počet generací.

# Genetické algoritmy

## Genetické algoritmy

Genetické algoritmy jsou asi nejznámější součástí evolučních výpočtů a v různých obměnách se používají hlavně při řešení optimalizačních úloh. Je zajímavé, že původní Hollandovou motivací při návrhu genetického algoritmu bylo studovat vlastnosti přírodou inspirované adaptace<sup>7</sup>. Velká část původní literatury byla věnována popisu principů, jak genetický algoritmus pracuje při hledání řešení úlohy. Zajímavé jsou paralely s matematickým problémem dvourukého bandity, který je příkladem na udržování optimální rovnováhy mezi explorací a exploatací.

Nejjednodušší varianta genetického algoritmu pracuje s binárními jedinci, to znamená, že parametry řešené úlohy je nutno vždy zakódovat jako binární řetězce. Tento přístup je výhodný z hlediska jednoduchosti použitých operátorů, ale binární zakódování na druhou stranu nemusí být nejvhodnější reprezentací problému. Způsob fungování jednoduchého genetického algoritmu je také poměrně jednoduchý. Algoritmus přechází mezi populacemi řešení tak, že nová populace zcela nahradí předchozí. Výběr rodičů je často realizován tzv. ruletovou selekcí, která vybírá jedince náhodně s pravděpodobností výběru úměrné jejich fitness. Rekombinačním operátorem je jednobodové křížení, které náhodně zvolí stejnou pozici v rodičích a vymění jejich části. Pravděpodobnost uskutečnění operace křížení je jedním z parametrů programu a obvykle je poměrně vysoká (0,5 i více). Mutace provádí drobné lokální změny tak, že prochází jednotlivé bity řetězce a každý bit s velmi malou pravděpodobností změní. Pravděpodobnost mutace je typicky nastavena, tak aby došlo průměrně ke změně jednoho bitu v populaci (oblíbená dolní mez) nebo v jedinci (horní mez).

Ruletovou selekci si dle metafory můžeme představit tak, že kolo rulety rozdělíme na výseče odpovídající velikostí hodnotám fitness jedinců a při výběru pak  $n$  krát vhodíme kuličku. Často používaným vylepšením ruletové selekce je tzv. stochastický univerzální výběr, který hodí kuličku do rulety jen jednou a další jedince vybírá deterministicky posunem pozice kuličky o  $1/n$ . Tento výběr pro malá  $n$  lépe aproximuje ideální počty zastoupení jedinců v další generaci. Dalšími varianty rodičovské selekce nepracují s absolutními hodnotami fitness, ale vybírají náhodně v závislosti na pořadí jedince v populaci setříděné podle fitness, což zanedbává absolutní rozdíly mezi hodnotami. Další variantou rodičovské selekce je tzv.  $k$ -turnaj, kdy nejprve vybereme  $k$  jedinců náhodně a z nich pak vybereme nejlepšího.

V současnosti se oblast genetických algoritmů neomezuje jen na binární kódování jedinců, časté je celočíselné, permutační nebo reálné kódování,

<sup>7</sup> John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992

---

**procedure** JEDNODUCHÝ GENETICKÝ ALGORITMUS

---

 $t \leftarrow 0$ *Inicializuj* populaci  $P_t$   $N$  náhodně vygenerovanými binárními jedinci  
délky  $n$ *Ohodnoť* jedince v populaci  $P_t$ **while** neplatí *kritérium ukončení* **do****for**  $i \leftarrow 1, \dots, N/2$  **do**vyber z  $P_t$  2 rodiče *Ruletovou selekcí*S pravděpodobností  $p_C$  *Zkříž* rodičeS pravděpodobností  $p_M$  *Mutuj* potomky*Ohodnoť* potomkyPřidej potomky do  $P_{t+1}$ **end for**Zahoď  $P_t$  $t \leftarrow t + 1$ **end while****end procedure**

---

Algoritmus 3: Schéma Hollandova genetikého algoritmu

která ale vyžadují specifické operace křížení a mutace<sup>8</sup>. O některých se zmíníme dále.

<sup>8</sup> Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.)*. Springer-Verlag, London, UK, UK, 1996; and Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1996



# Spojité optimalizace

Mnoho optimalizačních problémů z běžného života se dá definovat jako optimalizace funkce

$$f : D \rightarrow \mathcal{R},$$

kde  $D \subseteq \mathcal{R}^d$ . Je proto přirozené, že mnoho výzkumníků se zabývá právě evolučními algoritmy, kterou jsou schopné optimalizovat takové funkce. Problém optimalizace takových funkcí se v literatuře objevuje pod pojmem *spojitá optimalizace*<sup>9</sup>. Je důležité si uvědomit, že název vyjadřuje pouze to, že prostor, ve kterém se hledají řešení je spojitý ( $\mathcal{R}^n$ ), samotná optimalizovaná funkce  $f$  být spojitá nemusí.

<sup>9</sup> anglicky *continuous optimization*

Velmi často je definiční obor funkce  $D$   $d$ -rozměrný interval

$$D = [l_1, u_1] \times \cdots \times [l_d, u_d],$$

kde  $l_i$  a  $u_i$  jsou dolní a horní meze pro  $i$ -tou proměnnou. Objevují se ale i obecnější problémy, kde je  $D$  dána pomocí množiny podmínek tvaru  $g(\vec{x}) \leq 0$  a  $h(\vec{x}) = 0$ , pro  $g, h : \mathcal{R}^d \rightarrow \mathcal{R}$ .

V této kapitole napřed budeme uvažovat optimalizační problém tvaru

$$\begin{array}{ll} \min_{\vec{x}} & f(\vec{x}) \\ \text{za podmíněk} & \vec{x} \in [l_1, u_1] \times \cdots \times [l_d, u_d]. \end{array}$$

## Vlastnosti funkcí

Je zřejmé, že některé typy funkcí budou pro evoluční algoritmy lehčí, než jiné. Velký vliv na efektivitu evolučního algoritmu mají především vlastnosti jako multi-modalita, separabilita a podmíněnost.

Funkce je *multi-modální*, pokud má velké množství lokálních optim. Je zřejmé, že v takovém případě může mít algoritmus problém s uváznutím v lokálním optimu a je potřeba tomu přizpůsobit operátory. Existuje i oblast multi-modální optimalizace, kde je cílem najít co nejvíce různorodých lokálních optim.

Separabilní funkce jsou naopak pro optimalizaci jednodušší. Jsou to takové funkce, které se dají zapsat pomocí funkcí jedné proměnné. Formálně, funkce  $f(x_1, \dots, x_n) : \mathcal{R}^n \rightarrow \mathcal{R}$  je *aditivně separabilní*, pokud se dá zapsat jako součet funkcí  $f_1(x_1), \dots, f_n(x_n)$ , tj.  $f(x_1, \dots, x_n) = \sum_{i=1}^n f_i(x_i)$ . Obdobně můžeme zadefinovat i funkci multiplikativně separabilní. Z hlediska optimalizace je velkou výhodou separabilních funkcí, že se dají optimalizovat po jednotlivých složkách vektoru, tj. optimum můžeme najít tak, že vždy zafixujeme hodnoty  $n - 1$  parametrů a optimalizujeme jen podle jednoho.

Další vlastností, které výrazně ovlivňuje evoluci je podmíněnost funkce. Ta vyjadřuje, jak moc se liší vliv jednotlivých proměnných na hodnotu funkce. Pro kvadratické funkce (u kterých vrstevnice vypadají jako elipsoidy), je jejich podmíněnost druhá odmocnina poměru mezi délkou nejdelší a nejkratší osy elipsoidu. Pokud je podmíněnost funkce velká, říká se, že je funkce špatně podmíněná. Z hlediska evolučního algoritmu je důležité, že by s různým vlivem proměnných na hodnotu funkce měly počítat operátory.

Příklady posledních dvou vlastností vidíme na obrázku 1, který ukazuje vrstevnice funkcí dvou proměnných. Horní funkce je jednoduchý dvoudimenzionální paraboloid, který je separabilní a dobře podmíněný. Na prostředním obrázku je paraboloid, který má jednu osu delší než druhou a znázorňuje tedy špatnou podmíněnost<sup>10</sup>. Poslední funkce potom kombinuje špatnou podmíněnost s neseparabilitou.

Výše uvedené vlastnosti jsou ty, které nejvíce ovlivňují efektivitu evolučních algoritmů při spojitě optimalizaci, nejsou to ale všechny. Některé algoritmy například mohou využívat různé symetrie dané funkce, naopak funkce, které mají své globální optimum jen ve velmi malé oblasti prohledávaného prostoru a jinak jsou konstantní jsou pro evoluci velmi těžké obecně.

### Kódování pro spojitou optimalizaci

Jedno z prvních rozhodnutí, které je potřeba udělat při návrhu evolučního algoritmu je výběr kódování jedince. Vzhledem k tomu, že ve spojitě optimalizaci pracujeme s vektory reálných čísel, je otázka výběru kódování relativně snadná a jedinci jsou v naprosté většině případů kódování jako vektor typu float nebo double.

??? Existuje pro tohle ↓ nějaká reference?

Dalo by se uvažovat i o kódování jedince přímo po vzoru Hollandova genetického algoritmu, tj. binárním vektorem, a používat jednoduchá  $n$ -bodová křížení a bit-flip mutace, ale taková reprezentace trpí tím, že změna různých bitů v číslech vede k výrazně různým změnám hodnoty (např. změna bitu na konci mantisy vs. změna bitu na začátku exponentu).

### Operátory pro spojitou optimalizaci

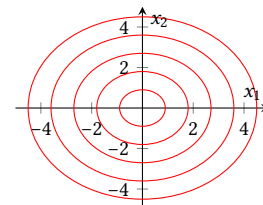
Pro spojitou optimalizaci můžeme samozřejmě použít stejné operátory jako pro každé jiné vektorové kódování jedince, tedy například  $n$ -bodové, případně uniformní křížení. Nicméně častěji se používají operátory specializované, které přímo využívají toho, že jedinci jsou vektory čísel.

Tzv. aritmetické křížení vektorů počítá vážený průměr dvou rodičů tak, aby vytvořilo potomka, potomci se tedy spočítají jako

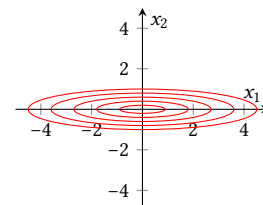
$$o_1 = w * p_1 + (1 - w) * p_2,$$

$$o_2 = (1 - w) * p_1 + w * p_2,$$

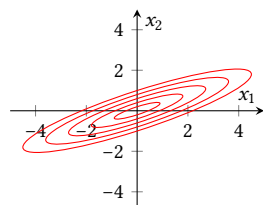
kde  $p_1$  a  $p_2$  jsou rodiče a  $w \in (0, 1)$  je (případně náhodně) zvolená váha. Ačkoliv se takové křížení relativně často používá hlavně v jednodušších aplikacích, jeho velkou nevýhodou je, že výslední potomci se nikdy nemohou dostat z konvexního obalu počáteční populace.



(a) Dobře podmíněná separabilní funkce



(b) Špatně podmíněná funkce



(c) Špatně podmíněná a neseparabilní funkce

Obrázek 1: Příklady různých vlastností funkcí

<sup>10</sup> podmíněnost zobrazené funkce jen cca 4.8, nedá se tedy považovat za špatně podmíněnou, objevují se i funkce s podmíněností  $10^6$

Mutace pro spojitou optimalizaci se občas rozdělují na dva typy – ovlivněné a neovlivněné. Neovlivněná mutace generuje novou hodnotu pro složku vektoru nezávisle na aktuální hodnotě, ovlivněná mutace naopak aktuální hodnotu používá. Typickým příkladem neovlivněné mutace je vygenerování nového čísla z daného rozsahu. Ovlivněná mutace typicky přičítá k dané složce vektoru číslo z normálního rozdělení  $\mathcal{N}(\mu, \sigma^2)$ .

Jednou z nevýhod výše uvedeného aritmetického křížení je, že potomci jsou velmi často relativně daleko od svých rodičů a nejsou jim tedy moc podobní. Právě z toho důvodu Deb a Agrawal navrhli simulované binární křížení (SBX)<sup>11</sup>. Hlavní inspirací pro vytvoření SBX bylo jednobodové křížení binárních řetězců, které reprezentují čísla ve dvojkové soustavě. Při náhodné volbě bodu pro křížení velká část nových potomků je relativně blízko k jednomu z rodičů. Jednobodové křížení má navíc další pěknou vlastnost – oba potomci jsou stejně daleko od průměru rodičů.

Cílem SBX je právě simulovat velikosti změn, které se dějí při jednobodovém křížení binárních řetězců. Noví potomci se v SBX křížení spočítají jako

$$o_{1,2} = \frac{1}{2}(p_1 + p_2) \pm \frac{1}{2}\beta(p_2 - p_1),$$

kde  $p_1$  a  $p_2$  jsou rodiče a  $\beta$  je náhodné číslo s pravděpodobnostním rozdělením

$$P(\beta) = \begin{cases} \frac{1}{2}(n+1)\beta^n & \text{pro } \beta \leq 1 \\ \frac{1}{2}(n+1)\frac{1}{\beta^{n+2}} & \text{pro } \beta > 1, \end{cases}$$

kde  $n$  je parametr rozdělení, který určuje, jak často budou hodnoty  $\beta$  blízko 1. Pro vyšší hodnoty  $n$  je tato pravděpodobnost vyšší (viz Obrázek 2).

Podobnou motivaci jako SBX má i tzv. *polynomiální mutace* (PM)<sup>12</sup>. V tomto případě se simuluje velikost změn v bit-flip mutaci binárně kódovaných čísel. To znamená, že nově vytvořený jedinec je s velkou pravděpodobností blízko svému rodiči. Gaussovská mutace se chová podobně, ale u PM je pravděpodobnost malých změn mnohem větší. Nový jedinec se v polynomiální mutaci vytvoří jako

$$o = p + \delta\Delta_{max}$$

kde  $\Delta_{max}$  je maximální velikost mutace a  $\delta$  je náhodné číslo z rozdělení

$$P(\delta) = \frac{1}{2}(n+1)(1-|\delta|)^n, \quad \delta \in (-1, 1).$$

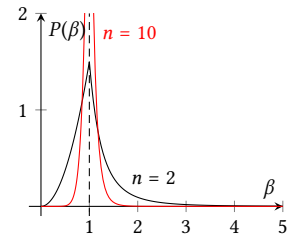
Proměnná  $n$  má v tomto případě podobný význam jako u SBX a určuje tvar distribuce. Opět větší hodnota  $n$  vede k větší pravděpodobnosti menších změn.

### Diferenciální evoluce

Všechny výše popsané operátory mají jednu zásadní nevýhodu a tou je, že operují s jedinci po složkách, jsou tedy vhodné především pro separabilní funkce. *Diferenciální evoluce*<sup>13</sup> přináší jiný přístup, který tímto problémem netrpí.

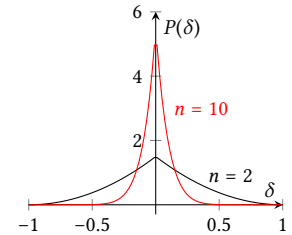
V diferenciální evoluci se jedinec vytváří pomocí jediného operátoru, který v sobě kombinuje jak mutaci, tak křížení. Jeho vstupem jsou hned čtyři jedinci z populace rodičů. Jako rodič  $p_4$  se většinou volí postupně všichni

<sup>11</sup> Kalyanmoy Deb and Ram Bhushan Agrawal. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 1995



Obrázek 2: Distribuce  $\beta$  v SBX křížení pro  $n = 2$  a  $n = 10$ .

<sup>12</sup> Kalyanmoy Deb and Mayank Goyal. A combined genetic adaptive search (GeneAS) for engineering design. *Computer Science and Informatics*, 26:30–45, 1996



Obrázek 3: Pravděpodobnostní rozdělení  $\delta$  v polynomiální mutaci pro  $n = 2$  a  $n = 10$ .

<sup>13</sup> Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, Dec 1997

jedinci z populace (není tedy volen náhodně), další tři rodiče jsou zvoleny náhodně. Nový jedinec vzniká tak, že se k rodiči  $p_3$  přičte rozdíl rodičů  $p_2$  a  $p_3$  a potom se provede křížení s rodičem  $p_4$ . Postup se dá zapsat jako

$$o^i = \begin{cases} p_3^i + F(p_1^i - p_2^i) & \text{pro } r^i < CR \text{ nebo } i = R \\ p_4^i & \text{jinak,} \end{cases}$$

kde  $F$  a  $CR$  jsou parametry mutace a křížení,  $r_i$  je náhodné číslo z rovnoměrného rozdělení  $\mathcal{U}(0, 1)$ ,  $R$  je náhodné číslo z množiny  $\{1, \dots, d\}$ , které zajišťuje, že v každém jedinci alespoň jedna pozice vznikne pomocí první řádky, a  $o^i$  a  $p_j^i$  značí  $i$ -tou složku potomka resp.  $j$ -tého rodiče.

Právě první část definice zajišťuje, že diferenciální evoluce je invariantní vůči rotacím, posunům a škálování prohledávaného prostoru a tedy funguje lépe pro neseparabilní a špatně podmíněné funkce. Pro toto chování je velmi důležitá volba parametru  $CR$ . Pro vysoké hodnoty  $CR$  větší část jedince vzniká právě pomocí rozdílu dvou rodičů, pro nízké hodnoty  $CR$  naopak je většina jedince tvořena rodičem  $p_4$ . Parametr  $CR$  se doporučuje nastavovat na nižší hodnoty ( $CR = 0.2$ ) pro separabilní funkce a na vyšší hodnoty pro neseparabilní funkce ( $CR = 0.9$ ).

Parametr  $F$  určuje, jak velká část rozdílu dvou jedinců se použije a tedy i to, jak velké se při mutaci dělají kroky. Nejčastěji se doporučuje hodnota okolo  $F = 0.8$ , ale vyskytují se nastavení mezi 0.5 a 2.0. Někdy se dokonce  $F$  bere jako náhodná hodnota z intervalu  $[0.5, 1.0]$ .

Kromě výše uvedeného operátoru se v diferenciální evoluci liší i selekce. Nový potomek se porovnává s rodičem  $p_4$  a v populaci se nechá jen lepší z obou. To je i důvod, proč se často rodič  $p_4$  nevolí náhodně.

V průběhu let vznikl systém pro klasifikaci různých variant diferenciální evoluce, popsaná verze se dnes nazývá DE/rand/1/bin. Označení vyjadřuje, že se jedná o diferenciální evoluci, kde jsou rodič  $p_3$  pro mutaci je volen náhodně, vybírá se jedna dvojice a používá se binomiální křížení. Existuje ale mnoho dalších variant<sup>14</sup>. Například se místo náhodně zvoleného jedince  $p_3$  bere vždy nejlepší jedinec z populace (DE/best/. /.) nebo se bere více dvojic při mutaci. Pro  $k$  dvojic potom vypadá operátor v diferenciální evoluci typu DE/. /k/. jako

$$o^i = \begin{cases} p_3^i + F \sum_{k=1}^k (p_{1,k}^i - p_{2,k}^i) & \text{pro } r^i < CR \text{ nebo } i = R \\ p_4^i & \text{jinak,} \end{cases}$$

kde značení je stejné jako výše.

Místo binomiálního křížení se občas používá tzv. *exponenciální křížení*. To používá stejný parametr  $CR$ , ale jiným způsobem. Jedná se vlastně o obdobu jedno- nebo dvou-bodového křížení. Začne se na náhodné pozici v jedinci a kopírují se parametry z druhého jedince, dokud je náhodně zvolené číslo z  $\mathcal{U}(0, 1)$  menší než  $CR$ . Pokud se narazí dříve na konec vektoru, pokračuje se znovu od začátku. Varianta s tímto křížením se nazývá DE/. /./exp a ačkoliv je velmi oblíbená, vysloužila si v poslední době kritiku<sup>15</sup> kvůli tomu, že v křížení je větší pravděpodobnost, že se překopírují hodnoty, které jsou na sousedních pozicích, a kvůli tomu je algoritmus závislý na pořadí proměnných.

<sup>14</sup> Efrén Mezura-Montes, Jesús Velázquez-Reyes, and Carlos A. Coello Coello. A comparative study of differential evolution variants for global optimization. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO '06*, pages 485–492, New York, NY, USA, 2006. ACM

<sup>15</sup> Ryoji Tanabe and Alex Fukunaga. Reevaluating exponential crossover in differential evolution. In Thomas Bartz-Beielstein, Jürgen Branke, Bogdan Filipič, and Jim Smith, editors, *Parallel Problem Solving from Nature – PPSN XIII: 13th International Conference, Ljubljana, Slovenia, September 13–17, 2014. Proceedings*, pages 201–210, Cham, 2014. Springer International Publishing

# Evoluční strategie

*Evoluční strategie*<sup>16</sup> jsou z historického hlediska jistou alternativou Hollandova genetického algoritmu. Jsou o něco starší a je na nich zajímavé, že jsou o něco komplikovanější. V současnosti se používají především pro řešení problémů spojitě optimalizace, ale myšlenky, které se v této oblasti vyskytují, se dají použít i jinde.

Evoluční strategie se dělí do dvou skupin podle způsobu, jakým pracují s populacemi rodičů a potomků. V obou případech jsou důležité parametry  $\mu$  a  $\lambda$ , které označují počet rodičů a počet potomků, kteří z nich vznikají. Pro druhy evolučních strategií potom existuje ustálené značení  $(\mu, \lambda)$ -ES a  $(\mu + \lambda)$ -ES. V prvním případě ("čárková selekce") máme populaci  $\mu$  rodičů, ze kterých vytvoříme  $\lambda$  potomků ( $\lambda > \mu$ ), z těch potom vybereme nejlepších  $\mu$  jako rodiče do další generace. Ve druhém případě ("plus selekce") z  $\mu$  rodičů vytvoříme opět  $\lambda$  potomků, ale před selekcí napřed sloučíme rodiče a potomky do jedné populace velikosti  $\mu + \lambda$ . Z té se potom opět vybere  $\mu$  nejlepších jedinců jako rodiče do další generace.

Jednou ze základních vlastností evolučních strategií, které je odlišují od jiných typů evolučních algoritmů je to, že obsahují nějakou formu samoadaptace parametrů. V případě spojitě optimalizace se tedy kromě samotných hodnot vektoru vyvíjí například i parametry pro mutaci. Technicky se tedy potom jedinec skládá ze dvou částí – samotného zakódovaného vektoru čísel  $\vec{x}$  a vektoru tzv. *endogenních parametrů*  $\vec{s}$ , které právě obsahují všechny parametry, které ovlivňují chování operátorů<sup>17</sup>. Je důležité si uvědomit, že endogenní parametry nijak přímo neovlivňují fitness jedince, jejich hodnoty se vyvíjí jen díky tomu, že jedinci s lepší hodnotou endogenních parametrů mají po aplikaci genetických operátorů častěji lepší fitness. Pro samotnou evoluci endogenních parametrů se mohou používat stejné operátory jako pro samotného jedince s fixně nastavenými parametry, nicméně moderní evoluční strategie častěji používají deterministický způsob nastavení těchto parametrů.

Důležitou součástí evolučních strategií je rekombinace. Ta v zásadě odpovídá křížení v genetických algoritmech a jejím cílem je vytvořit nového jedince kombinací jedinců z populace. V evolučních strategiích se ale velmi často používají rekombinace, které kombinují všechny jedince. Častá je tzv. *interpolační rekombinace*, která jednoduše spočítá průměr všech jedinců v populaci – nový jedinec je vlastně centroid celé populace. Používá se i její varianta, která používá vážený součet, kde lepší jedinci v populaci mají větší váhu při výpočtu průměru (nejhorší jedinci se dokonce mohou úplně ignorovat). Kromě těchto dvou nejčastěji používaných rekombinací se dají používat i křížení známá z genetických algoritmů, není to ale moc obvyklé. Exis-

<sup>16</sup> I. Rechenberg. *Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. PhD thesis, Stuttgart, GER, 1973; and Hans-Paul Schwefel. *Numerische optimierung von computer-modellen mittels der evolutionsstrategie*. PhD thesis, 1977

<sup>17</sup> Vedle pojmu "endogenní parametry" se ještě občas objevuje pojem "exogenní parametry", který označuje parametry algoritmu, které se nemění, jako např. velikost populace.

tuje i uniformní rekombinace, která každou složku vektoru jedince vybírá z náhodného jedince v populaci.

Použitá rekombinace se občas objevuje i v notaci evolučních strategií, v takovém případě se píše jako  $(\lambda/\rho + \mu)$ , kde právě  $\rho$  označuje použitou rekombinaci.

Základním operátorem v evolučních strategiích ale je mutace. Typicky jde o gaussovskou mutaci s určitým rozptylem. Zde se objevuje několik možností, kde nejjednodušší je sférická mutace, která přičítá k jedinci vektor z normálního rozdělení  $\sigma \mathcal{N}(0, I)$ , kde  $I$  je jednotková matice. O něco složitější varianta potom používá jiný rozptyl pro každou souřadnici vektoru, tj. k jedinci se přičítá hodnota z normálního rozdělení  $\mathcal{N}(0, \text{diag}(\vec{\sigma}))$ , kde  $\text{diag}(\sigma)$  značí diagonální matici s vektorem  $\vec{\sigma}$  na hlavní diagonále. Konečně v nejkomplicovanějším případě se jedinci generují s normálního rozdělení s plnou kovarianční maticí  $\mathcal{N}(0, \Sigma)$ . Ačkoliv první dva způsoby vyžadují mnohem nižší množství endogenních parametrů (1 respektive  $d$ ), jejich nevýhoda spočívá v tom, že nejsou schopny zachytit závislosti mezi parametry a hodí se tak především pro separabilní problémy. Na druhou stranu způsob s plnou kovarianční maticí je invariantní vůči rotacím a škálování prohledávaného prostoru (a díky dalším vlastnostem evolučních strategií i k monotónnímu škálování fitness funkce).

### *Pravidlo jedné pětiny*

Vzhledem k relativně velkému množství endogenních parametrů v evolučních strategiích a k tomu, že vhodné nastavení parametrů je různé pro různé optimalizační problémy a dokonce i v různých fázích evoluce, vznikla potřeba parametry nastavovat adaptivně. Při pevném nastavení velikosti mutace typicky pozorujeme tři fáze evoluce. V první fázi evoluce je konvergence pomalá, protože změny dělané mutací jsou příliš malé. Následuje fáze rychlé konvergence, kde jsou velikosti změn optimální pro danou fázi výpočtu, a v poslední fázi se konvergence zase zpomalí, protože jsou změny moc velké (algoritmus “skáče” kolem optima).

První metoda adaptivního nastavování vzešla z Rechenbergových experimentů,<sup>18</sup> které zkoumaly vliv rozptylu Gaussovské mutace v  $(1 + 1)$ -ES při optimalizaci dvou funkcí. Ukázalo se, že pro obě funkce algoritmus nejrychleji konverguje, pokud se velikost mutace zmenší, když je pravděpodobnost, že je potomek lepší než rodič menší než cca  $\frac{1}{5}$ , a zvětší, když je tato pravděpodobnost větší než cca  $\frac{1}{5}$ . Z tohoto pozorování vzniklo pravidlo  $\frac{1}{5}$ .<sup>19</sup>

Proč takové pravidlo funguje? Představme si optimalizaci lineární funkce, v takovém případě je pravděpodobnost, že potomek je lepší než rodič, přesně 50%. Obecnou funkci můžeme v každém bodě podle Taylorova pravidla aproximovat pomocí lineární funkce, tato aproximace je tím přesnější, čím blíže jsme bodu, ve kterém jsme funkci aproximovali, tedy, při malé velikosti mutace bude většina potomků blízko a pravděpodobnost, že jsou lepší než rodič, je kolem 50%. Naopak, při nekonečném rozptylu mutace pravděpodobnost zlepšení odpovídá části prostoru, kde má funkce lepší hodnotu, než v daném bodě. Ve chvíli, kdy se algoritmu přiblíží optimu je tato pravděpodobnost většinou blízko 0. Ve skutečnosti se dokonce ukazuje, že pro většinu funkcí pravděpodobnost zlepšení monotónně roste s klesajícím rozptylem

<sup>18</sup> I. Rechenberg. *Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. PhD thesis, Stuttgart, GER, 1973

<sup>19</sup> V angličtině “one fifth rule”

mutace. Přesná hodnota pro optimální pravděpodobnost zlepšení závisí na optimalizované funkci, nicméně právě  $\frac{1}{2}$  je často doporučovaná.





# Obsah

<i>Před úvodem</i>	3
<i>Jak to je?</i>	3
<i>Jak to bude?</i>	3
<i>Pok pok pokusy</i>	3
 <i>Evoluční algoritmy</i>	5
<i>Evoluce, geny a DNA</i>	5
<i>Obecné schéma evolučního algoritmu</i>	5
 <i>Genetické algoritmy</i>	7
<i>Genetické algoritmy</i>	7
 <i>Spojité optimalizace</i>	9
<i>Vlastnosti funkcí</i>	9
<i>Kódování pro spojitou optimalizaci</i>	10
<i>Operátory pro spojitou optimalizaci</i>	10
<i>Diferenciální evoluce</i>	11
 <i>Evoluční strategie</i>	13
<i>Pravidlo jedné pětiny</i>	14
 <i>Rejstřík</i>	25
 <i>Literatura</i>	29



## *Seznam obrázků*

- 1 Příklady různých vlastností funkcí 10
- 2 Distribuce  $\beta$  v SBX křížení pro  $n = 2$  a  $n = 10$ . 11
- 3 Pravděpodobnostní rozdělení  $\delta$  v polynomiální mutaci pro  $n = 2$  a  $n = 10$ . 11



## *Seznam tabulek*



## *Seznam algoritmů*

1	Schéma evolučního algoritmu . . . . .	4
2	Schéma evolučního algoritmu . . . . .	6
3	Schéma Hollandova gentického algoritmu . . . . .	8





# *Rejstřík*






Evoluční algoritmus, 5  
Evoluční programování, 5

Evoluční strategie, 5

Genetický algoritmus, 3, 5, 7



## *Todo list*

	OSNOVA PREDNASKY TAK JAK JE TED . . . . .	3
	SKORO TO TU MUZEME SMAZAT NE? . . . . .	3
	<b>Missing ref.</b> . . . . .	5
	<b>Missing ref.</b> . . . . .	5
	??? Existuje pro tohle ↓ nějaká reference? . . . . .	10



## *Literatura*

- [1] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies – a comprehensive introduction. 1(1):3–52, May 2002.
- [2] Kalyanmoy Deb and Ram Bhushan Agrawal. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 1995.
- [3] Kalyanmoy Deb and Mayank Goyal. A combined genetic adaptive search (GeneAS) for engineering design. *Computer Science and Informatics*, 26:30–45, 1996.
- [4] Agoston E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, 2003.
- [5] David B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ, USA, 1995.
- [6] Frédéric Gruau. *Neural Network Synthesis Using Cellular Encoding And The Genetic Algorithm*. PhD thesis, L’universite Claude Bernard-lyon I, 1994.
- [7] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.
- [8] James Kennedy, James F Kennedy, Russell C Eberhart, and Yuhui Shi. *Swarm intelligence*. Morgan Kaufmann, 2001.
- [9] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.
- [10] John R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, MA, USA, 1994.
- [11] John R. Koza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, Norwell, MA, USA, 2003.
- [12] John R. Koza, David Andre, Forrest H. Bennett, and Martin A. Keane. *Genetic Programming III: Darwinian Invention & Problem Solving*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 1999.

- [13] Efrén Mezura-Montes, Jesús Velázquez-Reyes, and Carlos A. Coello Coello. A comparative study of differential evolution variants for global optimization. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO '06*, pages 485–492, New York, NY, USA, 2006. ACM.
- [14] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.)*. Springer-Verlag, London, UK, UK, 1996.
- [15] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1996.
- [16] I. Rechenberg. *Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. PhD thesis, Stuttgart, GER, 1973.
- [17] Hans-Paul Schwefel. *Numerische optimierung von computer-modellen mittels der evolutionsstrategie*. PhD thesis, 1977.
- [18] Gene I. Sher. *Handbook of Neuroevolution Through Erlang*. Springer Publishing Company, Incorporated, 2012.
- [19] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.
- [20] Kenneth O Stanley and Risto Miikkulainen. A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130, 2003.
- [21] Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, Dec 1997.
- [22] Ryoji Tanabe and Alex Fukunaga. Reevaluating exponential crossover in differential evolution. In Thomas Bartz-Beielstein, Jürgen Branke, Bogdan Filipič, and Jim Smith, editors, *Parallel Problem Solving from Nature – PPSN XIII: 13th International Conference, Ljubljana, Slovenia, September 13–17, 2014. Proceedings*, pages 201–210, Cham, 2014. Springer International Publishing.