EVA II

NAIL 086 - 2013/14

Roman Neruda

ÚVOD

Témata, literatura, osnova.

Literatura

- Mitchell, M.: *Introduction to Genetic Algorithms*. MIT Press, 1996.
- Eiben, A.E and Smith, J.E.: *Introduction to Evolutionary Computing*, Springer, 2007.
- Michalewicz Z.: *Genetic Algorithms + Data Structures = Evolution Programs* (3ed), Springer, 1996
- Holland, J.: *Adaptation in Natural and Artificial Systems*, MIT Press, 1992 (2nd ed).
- Goldberg, D.: *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

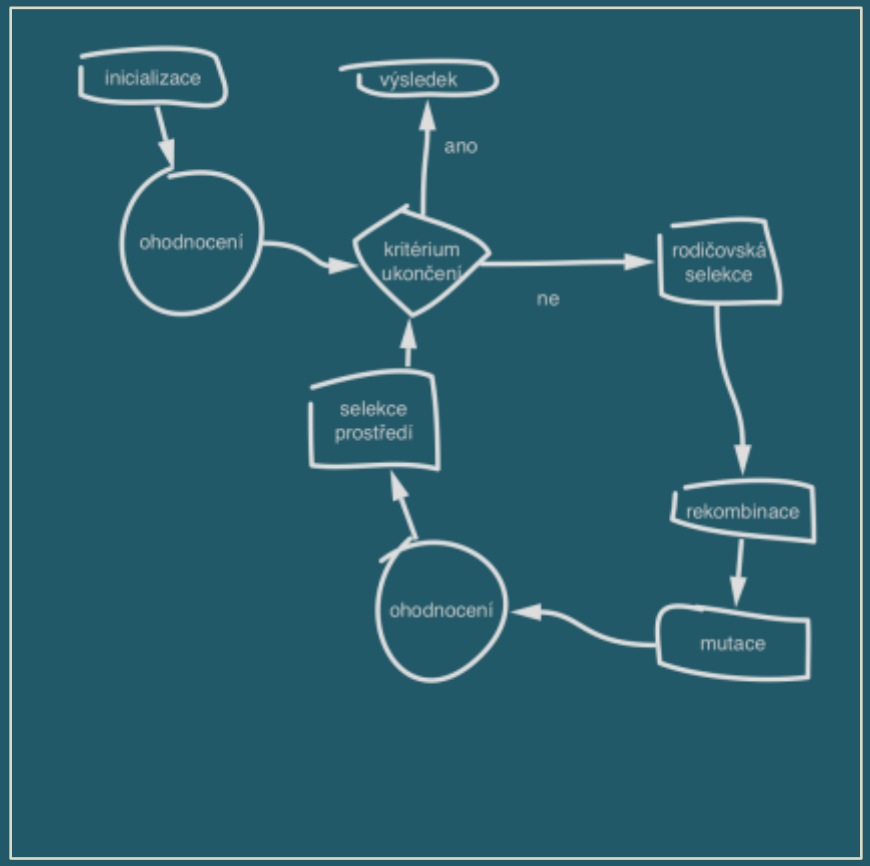
Témata

- Teoretické modely GA – konečné/nekonečné populace, Markovovské řetězce
- Black box optimization
- Evoluční programování
- Genetické programování
- Scatter search, Tabu search
- Neuroevoluce – gramatické kódování, celulární kódování, NEAT, HyperNEAT
- Artificial life a podobné algoritmy
- Memetické algoritmy
- Ladění, řízení, meta-evoluce, ko-evoluce

Skoro vše, co bylo dosud považováno za pevné, lze měnit

LADĚNÍ, ŘÍZENÍ, META EVOLUCE

Obecný EA



- Náhodně vytvoř iniciální populaci P(0)
- V cyklu vytvářej P(t) z P(t+1):
  - Výběr rodičů
  - Rekombinace a mutace
  - Tím vznikají noví jedinci
  - Environmentální selekce vybere P(t+1) na základě P(t) a nových jedinců

Ladění vs. řízení

- EA má různé parametry (velikost populace, pravděpodobnosti operací, velikost turnaje, velikost elitismu, ...)
- Běžná praxe je *vyladit* hodnoty parametrů pro danou úlohu (většinou pokusně)
- Ony jsou ale často na sobě závislé a všechny kombinace nezvládneme projít
- Což takhle zkusit parametry nechápat staticky, ale jako funkce času nebo nějakých ukazatelů práce EA a měnit je - *řídit*.

Jak?

- Pevně stanovená strategie
  - Funkce změny parametru závislá na čase
  - Někdy deterministická
    - Zmenši pravděpodobnost mutace o 0.1% každou generaci
  - Někdy stochastická
    - Simulované žíhání
- Adaptivně
  - Na základě informací o kvalitě řešení, statistických vlastnostech populace, ...
- Sebe-adaptivně
  - Evoluce evoluce
    - Například rozptyly mutace v genomu

Příklad 1: Reprezentace

- Aneb adaptivní změna reprezentace na základě absolutního ukazatele diverzity populace:
  - Spouštím GA iterativně, každý běh hledá řešení dokud není diverzita populace menší než 1
    - (měří se pomocí Hammingovy vzdálenosti mezi nejlepším a nejhorším jedincem).
  - Nejlepší řešení z daného běhu je referenční, jedinci jsou jen vzdálenosti od něj.
  - Pokud najdu podruhé stejné referenční řešení, zvětším počet bitů na kódování čísel

Příklad 2: Mutace

- Typická gaussovská mutace reálných vektorů:
  - $x(t+1) = x(t) + N(0,s)$, kde s je rozptyl
- Deterministická změna:
  - $s(t) = 1 – 0.9\ (t/T)$, kde $t=0...T$
- Evoluční strategie prvního druhu:
  - optimální úspěšnost mutace je 1/5,
    - když je to víc, zvětš s o cca 10%,
    - když je to míň, zmenši s o cca 10%
- Evoluční strategie druhého druhu (sebe-adaptace):
  - Dej s (klidně i jiná pro každé x) ke genomu a evolvuj

Příklad 3: Selekce

- Boltzmanovský turnaj:
  - Pravděpodobnost, že vyhraje horší závisí exponenciálně na rozdílu fitness lomeno "teplotou"
  - Ta s časem klesá
  - Pokud se uvízne, je možné přitopit
- Simulované žíháni jako lokální prohledávání:
  - Generuju řešení v okolí, namísto klasického posunutí do lepšího řešení opět boltzmanovská pravděpodobnost, že se posunu i do horšího

Příklad 4: Populace

- Nepřímé ovládání velikosti populace v závislosti na fitness jedinců:
  - Při vzniku nového jedince se mu určí délka života závislá na jeho fitness
  - Po uplynutí daného počtu generací jedinec umírá
  - Jelikož lepší žijí déle a počet potomků je úměrný délce života, systém podporuje šíření úspěšných genů

Více než schémata, lépe než schémata

TEORIE EVA PODRUHÉ

Přesné modely GA

- 90.léta: Vose, Lepins, Nix, Whitley, ...
- Snaha zachytit:
  - jak přesně vypadají populace
  - zobrazení přechodu k další populaci
  - vlastnosti tohoto zobrazení
  - asymptotické chování jednoduchého GA
- Nekonečné populace
- Konečné populace

Ještě jednodušší jednoduchý GA

- Náhodná počáteční populace l bin. řetězců x
  - Fitness f(x)
  - Opakuj dokud nenaplníš novou:
    - Vybrat selekcí 2 jedince, zkřížit s p_c, 1 zahodit
    - Mutovat každý bit s p_M
    - Vložit do nové
- To celé opakuj dokud nenajdeš dost dobré x

Formalizujeme JJJGA

- Každý řetězec je reprezentován číslem $0..2^l$
  - 00000111 je jako 7
- Populace t je reprezentována:
  - dvěma vektory: p(t) a s(t) délky 2^l
  - p_i (t) část populace t, kterou zabírá řetězec i
  - s_i (t) pravděpodobnost selekce řetězce i
- p(t) definuje složení populace
- s(t) shrnuje pravděpodobnosti výběru

Operátor velké Gé

- Máme fitnes f, definujme matici F(i,j):
  - F(i,i) = f(i); F(i,j) = 0 pro i<>j
- Pak s(t) = F p(t)/(Σ F(j,j) p_j(t))
  - (to je vlastně definice proporcionální selekce)
- Chceme definovat G realizující JJGA:
  - p(t+1) = G p(t), anebo obdobně s(t+1) = G s(t)
- Pak iterujeme G ... G s(0) a všechno o JJGA víme
- G = F°M (F fitness, M křížení a mutace)

Nechť G=F

- E(x) očekáváná (střední hodnota)
- E(p(t+1)) = s(t)
- s(t+1) ~ F p(t+1) (liší se jen o multiplikativní c)
- Takže: E(s(t+1)) ~ F s(t)
  - V případě konečné populace nám výběrové chyby mohou způsobit odchylku od E(.)
  - Čím větší populace, tím menší odchylka
  - U nekonečné populace to je přesné ale nepraktické, když člověk není bůh

Nechť G=M

- M rekombinační op. (zahrnuje křížení i mutaci)
- Odvození je těžší:
  - Využijeme pomocnou veličinu r(i,j,k)
  - r(i,j,k) ... pravděpodobnost, že k vznikne z i a j
  - Když známe r(i,j,k), můžeme spočítat p(t+1)
  - Tedy po složkách a v očekávané hodnotě, což nám u nekonečných populací nevadí, že :
- $E(p_k(t+1)) = \Sigma_i \Sigma_j \, s_i(t) \, s_j(t) \, r(i,j,k)$

R(i,j,0)

$$r_{i,j}(0) = \frac{(1-p_m)^l}{2}\left[\eta^{|i|}\left(1-p_c+\frac{p_c}{l-1}\sum_{c=1}^{l-1}\eta^{-\Delta_{i,j,c}}\right)\right.$$

$$\left.+\eta^{|j|}\left(1-p_c+\frac{p_c}{l-1}\sum_{c=1}^{l-1}\eta^{\Delta_{i,j,c}}\right)\right].$$

A zbytek analogicky

Výsledky

- JJGA pracující prostřednictvím G je dynamický systém, p(t) či s(t) jsou body (trajektorie).
- Jaké jsou pevné body? (ehm, to zrovna NEVÍME)
  - Pevné body F jsou populace se stejnou fitness
  - Stabilní pevný bod F: maximální stejná fitness
  - (Jediný) pevný bod M: stejné pravděpodobnosti s (resp. stejné zastoupení jedinců p)
  - Kombinace míchání M a ostření F - *punctuated equilibria (známe z biologie)*

Konečné populace

- Markovovské řetězce:
  - Stochastický proces v diskrétním čase
  - Systém má stavy, pravděpodobnost přechodu z jednoho stavu do druhého závisí jen na tom 1 stavu
  - Tabulka stavů
  - Tabulka přechodových pravděpodobností ze stavu i do stavu j nám zcela popisuje takový systém
- Budeme modelovat GA nad konečnou populací jako markovovský řetězec

Stavy

- Stav Markovovského řetězce je konkrétní populace.
- Populací o n jedincích délky l je kombinační číslo, označme ho N:
  - $N=(n+2^l-1)$ nad (2^l-1) rozumíme si?
- Matice Z
  - Sloupce jsou populace
  - Z(y,i) = počet jedinců y v populaci i
- (takže sloupce Z jsou stavy M.ř.)

Přechodová matice

- Q matice pravd.přechodu mezi stavy:
- NxN
- (např. pro n=l=8 má 10^{29} prvků!)
- Odvození komplikované, takže zase pro ilustraci:

$$Q_{i,j} = n! \prod_{y=0}^{2^l-1} \frac{\left[\mathcal{M}\left(F\vec{\phi}_i/|F\vec{\phi}_i|\right)_y\right]^{Z_{y,j}}}{Z_{y,j}!}.$$

K čemu to je

- Přesná analýza GA pro danou f
- Ač prakticky těžko proveditelná
- Asymptotické výsledky – vhled do konvergenčních vlastností a chování GA
- **Dokázala se korespondence ideálního JJGA s nekonečnou populací a modelu s konečnou populací (když n jde k nekonečnu, jak už to bývá)**

EVOLUČNÍ PROGRAMOVÁNÍ

Evoluční programování

- L. Fogel, 60.léta (starší než Hollandovy GA)
- Snaha vyvinout „umělou inteligenci“:
- předpověď stavu prostředí, ve kterém se agent nachází
- odvození patřičné akce s ohledem na stav prostředí
- Agent: konečný automat
- Prostředí: posloupnosti symbolů konečné abecedy

Konečný automat

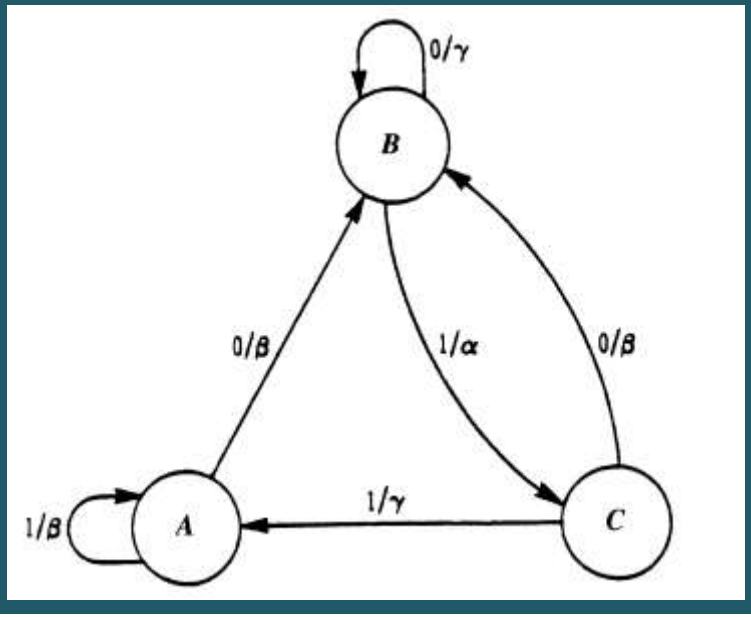


Table 3-1 The Response of the Finite-State Machine Shown in Figure 3-2 to a String of Symbols. In This Example, the Machine Starts in State C.

Present State	*C*	*B*	*C*	*A*	*A*	*B*
Input Symbol	0	1	1	1	0	1
Next State	*B*	*C*	*A*	*A*	*B*	*C*
Output Symbol	β	α	γ	β	β	α

EP nad KA

- Populace konečných automatů
- Předkláadají se jim vstupy,
- Výstup automatu je porovnán s následujícím vstupem v posloupnosti
- Fitness: úspěšnost predikce (různě měřená)
  - Vše nebo nic
  - Absolutní chyba
  - Mean square error

EP nad KA pokr.

- Vznik nových automatů: mutace
  - Změna výstupního symbolu
  - Změna následného stavu
  - Přidání stavu
  - Odebrání stavu
  - Změna počátečního stavu
- Křížení neexistuje (!)
- Polovina populace nahrazena novými automaty
- Často do fitness zahrnuta i velikost automatu

Příklad: predikce prvočísel

- Učí se řetězce 0 a 1,
  - 0 znamená číslo na dané pozici není prvočíslo.
- Iterativně:
  - učí se pevnou délku,
  - pak zvětšit vstup o 1 symbol
- Fitness:
  - bod za každý správný symbol
  - Záporný člen fitness: - 0.01 * N (=počet stavů)
- Automaty se 'zjednodušovaly' až k jednomu stavu a 0.

EP dnes

- Velmi benevolentní k zakódování jedinců
  - Kódování má být přirozené pro problém
  - Často Genotyp = Fenotyp
- Více “chytrých”mutací, těsné přizpůsobení problému
- Žádné křížení
- Rodičovská selekce: každý je vybrán jednou
- Environmentální selekce: z rodičů a potomků se turnajem vybere nová populace

EP nad reálnými čísly

- Kódování
  - Vektor reálných čísel
  - Obsahuje i rozptyly mutací
- Mutace
  - Malá náhodná změna s příslušným rozptylem
- Meta-evoluce
  - Upravujeme parametry, které používáme v EP
- Selekce: rodiče všichni, pak turnaj

EP nad R

procedure Meta-EP
t←0
Inicializuj P_t N náhodně vygenerovanými reálnými vektory $\vec{x}_t = (x_{1t},\ldots,x_{nt},\sigma_{1t},\ldots,\sigma_{nt})$
Ohodnoť jedince v populaci P_t
while (neplatí kritérium ukončení) do
    for i ← 1, . . . , N do
        k rodiči $\vec{x}_t$ vygenruj potomka $\vec{y}_t$ mutací:
        for j ← 1,...,n do
            $\sigma_j' \leftarrow \sigma_j \cdot (1+\alpha \cdot N(0,1))$ $x_j' \leftarrow x_j + \sigma_j' \cdot N(0,1)$
        end for
        Vlož $\vec{y}_t$ do kandidátské populace potomků P'_t
    end for
    Turnajovou selekcí vyber P_{t+1} z rodičů P_t a potomků P_t'
    Zahoď P_t a P_t'
    t←t+1
end while
end procedure

Příklad pokr.

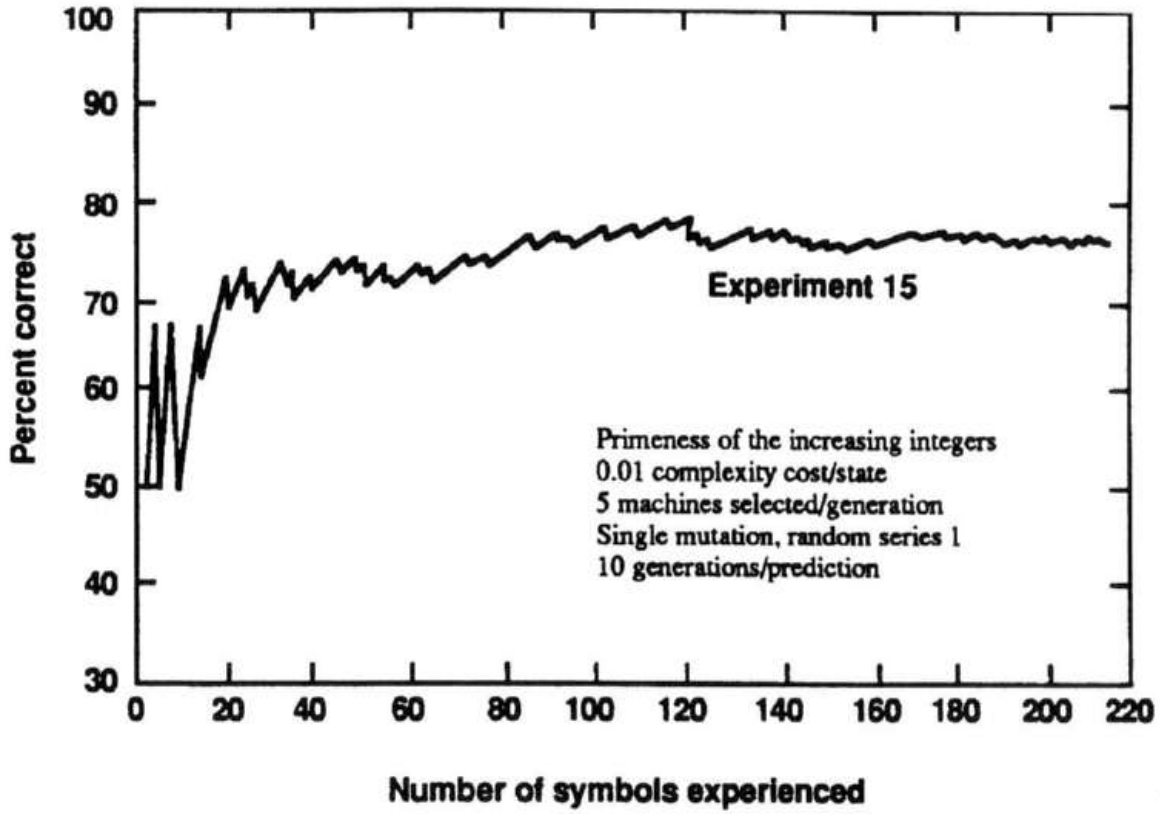


Figure 3-3. The cumulative percentage of correct predictions in the first 200 prime numbers (from Fogel et al., 1966, p. 37).

EP vs GA

- Křížení:
  - Výměna bloků?
  - Divná mutace?
- Jones 1995: headless chicken experiment:
  - Tradiční křížení
  - Náhodné křížení (s náhodným řetězcem)
- There is no crossover without the idea of crossover, do not call headless chicken a chicken, although it has many chicken features.

Bezhlavé kuře

- V experimentech s jasnými bloky překoná křížení náhodné křížení
- Když je ale náhodné křížení lepší, jde vlastně o makromutaci
- V takovém případě nemáme jasné bloky
  - Buď špatné zakódování
  - Nebo těžký problém pro GA s křížením
- Poučení: má naše kuře hlavu?

GA vs EP: experiment

Table 4.1 The Number of Parameters, The Binary Coding Length, and The Functions Studied in Schraudolph and Belew (1992). These Follow Previous Efforts By De Jong (1975). The Operation $[x_i]$ in $f3$ Returns The Greatest Integer Less Than or Equal To x_i. The $N(0, 1)$ in $f4$ Represents A Standard Gaussian Random Variable.

Function	Dimension	Bit Length
$f1$	3	10
$f2$	2	12
$f3$	5	10
$f4$	30	8
$f5$	2	17

Function	Parameter Range
$f1: F(x) = \sum_{i=1}^{3} x_i^2$	$[-5.12, 5.12]$
$f2: F(x) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$	$[-2.048, 2.048]$
$f3: F(x) = \sum_{i=1}^{5} [x_i],$	$[-5.12, 5.12]$
$f4: F(x) = \sum_{i=1}^{30} ix_i^4 + N(0, 1)$	$[-1.28, 1.28]$
$f5: F(x)^{-1} = 1/K + \sum_{j=1}^{25} f_j(x)^{-1}.$	$[-65.536, 65.536]$

$$f_j(x) = c_j + \sum_{i=1}^{2} (x_i - a_{ij})^6.$$

where $K = 500$, $c_j = j$, and

$$[a_{ij}] = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & \dots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & -16 & \dots & 32 & 32 & 32 \end{bmatrix}$$

$f6$: $F(x, y) = x^2 + 2y^2 - 0.3\cos(3\pi x) - 0.4\cos(4\pi y) + 0.7,$

$f7$: $F(x, y) = x^2 + 2y^2 - 0.3(\cos(3\pi x)\cos(4\pi y)) + 0.3,$

$f8$: $F(x, y) = x^2 + 2y^2 - 0.3(\cos(3\pi x) + \cos(4\pi y)) + 0.3.$

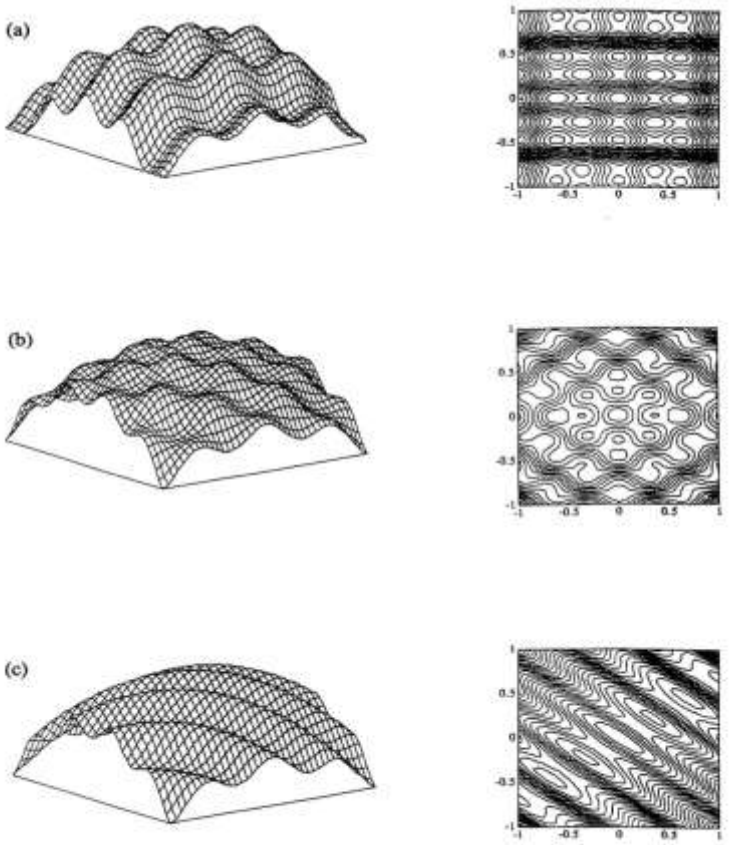


Figure 4-17 Inverted and contour plots of the three Bohachevsky functions studied in Fogel and Stayton (1994). (a) $F(x, y) = x^2 + 2y^2 - 0.3\cos(3\pi x) - 0.4\cos(4\pi y) + 0.7$. (b) $F(x, y) = x^2 + 2y^2 - 0.3(\cos(3\pi x)\cos(4\pi y)) + 0.3$. (c) $F(x, y) = x^2 + 2y^2 - 0.3(\cos(3\pi x) + \cos(4\pi y)) + 0.3$.

GA vs EP exp pokr

Table 4-3 Results for the Best Score in the Population and the Mean of All Parents' Scores After 10,080 Function Evaluations, Averaged Over 10 Trials With the Genetic Algorithm Techniques (Both With And Without Dynamic Parameter Encoding) and 500 Trials With "Evolutionary Programming" (after Fogel and Stayton, 1994). Evolutionary Programming Outperforms Both Genetic Methods On Functions $f1, f2$, and $f6$–$f8$, and Yields Comparable Performance on $f3$–$f5$. The Values in Parentheses Indicate the Standard Deviations

		Average Best	Average Mean
$f1$:	EP	3.149×10^{-66} (2.344×10^{-129})	1.087×10^{-65} (1.794×10^{-128})
	DPE	1.056×10^{-11} (1.072×10^{-21})	3.618×10^{-10} (1.060×10^{-18})
	GA	2.836×10^{-4} (4.587×10^{-8})	6.135×10^{-1} (1.627×10^{-1})
$f2$:	EP	1.215×10^{-14} (3.357×10^{-26})	8.880×10^{-14} (2.399×10^{-24})
	DPE	2.035×10^{-2} (3.315×10^{-3})	8.785×10^{-2} (8.090×10^{-3})
	GA	2.914×10^{-2} (6.280×10^{-4})	1.722×10^{0} (2.576×10^{0})
$f3$:	EP	0.0 (0.0)	0.0 (0.0)
	DPE	0.0 (0.0)	0.0 (0.0)
	GA	0.0 (0.0)	1.307×10^{0} (1.172×10^{-1})
$f4$:	EP	-2.575×10^{0} (7.880×10^{-1})	-4.274×10^{-1} (3.206×10^{-2})
	DPE	-2.980×10^{0} (1.009×10^{-1})	4.704×10^{-1} (1.283×10^{-1})
	GA	-4.599×10^{-1} (4.265×10^{-1})	1.312×10^{1} (2.941×10^{0})
$f5$:	EP	4.168×10^{0} (9.928×10^{0})	4.194×10^{0} (1.022×10^{1})
	DPE	3.502×10^{09} (1.265×10^{1})	1.642×10^{1} (7.101×10^{2})
	GA	9.980×10^{-1} (3.553×10^{-15})	1.021×10^{1} (7.165×10^{1})
$f6$:	EP	5.193×10^{-96} $(1.348 \times 1010^{-188})$	9.392×10^{-94} (4.410×10^{-184})
	DPE	1.479×10^{-9} (1.460×10^{-18})	8.340×10^{-7} $(4.649 \times 1010^{-14})$
	GA	2.629×10^{-3} (1.103×10^{-5})	4.022×10^{1} (6.467×10^{3})
$f7$:	EP	8.332×10^{-101} (3.449×10^{-198})	2.495×10^{-99} (3.095×10^{-195})
	DPE	2.084×10^{-9} (6.831×10^{-18})	6.520×10^{-7} $(7.868 \times 1010^{-14})$
	GA	4.781×10^{-3} (2.146×1010^{5})	3.541×10^{1} (2.922×10^{3})
$f8$:	EP	1.366×10^{-105} (4.479×10^{208})	3.031×10^{103} $(2.122 \times 1010^{203})$
	DPE	1.215×10^{5} (5.176×10^{10})	3.764×10^{1} (9.145×1010^{1})
	GA	2.444×10^{3} (4.511×10^{5})	2.788×1010^{1} (1.368×10^{5})

GENETICKÉ PROGRAMOVÁNÍ

Vyvíjet programy, odvěký sen lidstva

- Obecná struktura genetického programování:
- Vygeneruj náhodné programy
- Ohodnoť programy
  - tak, že je spustíš a otestuješ na datech
- Vygeneruj novou generaci programů:
  - Výběr dle fitness
  - Křížení dvou programů
  - Mutace programu
- Pokračuj jako obvykle, dokud je třeba

Genetické programování stromové

- John Koza:
- Programy jsou reprezentovány jako syntaktické stromy
- Terminály jsou proměnné a konstanty
- Neterminály jsou operace
- Křížení je výměna podstromu
- Mutace je generování podstromu
- Fitness se určí během programu

Varianty GP

- Mutace:
- Ukazuje se výhodné používat více typů mutace:
  - Náhodná či systematická **mutace konstant**
  - Náhodná výměna uzlu stejné arity
  - Permutace
  - Záměna neterminálu za terminál
  - Zmenšující mutace (menší podstrom, nový jedinec z podstromu)
- Křížení:
  - Uniformní křížení na podstromu

Varianty GP

- Inicializace:
  - Generování náhodných stromů do určitého počtu uzlů
  - Generování náhodných stromů do určité hloubky
  - Ramped half-and-half: kombinace předchozích dvou postupů v populaci

ADF

- Automaticky definované funkce
  - Připojí se k programu
  - Jsou charakterizovány svou aritou
  - Mají omezené množiny terminálů i neterminálů
- Program sestává z hlavního programu a několika podstromů ADF
- Volání ADF ja pak novým terminálem hlavního programu
- Operace GP pracují buď jen v rámci ADF či v rámci hlavního programu
- Kromě ADF byly navrženy automaticky definované další struktury kódu: loops, iterations, recursions

Bloat / bobtnání v GP

- Programy mají tendenci narůstat, existují různé teorie proč. V přírodě existuje většinou nějaký fyzikální limit, který přílišnému růstu zabrání parohy jsou těžké
- Restrikce velikosti stromu, restrikce hloubky stromu:
  - buď jako penalizující člen ve fitness
  - nebo jako kontrola a eliminace nových jedinců
- Anti-bloat operátory:
  - mutace a křížení, které nezvětšují jedince,
  - případně mutace, které ho přímo zmenšují

Cesta dolů: lineární GP

- Lineární GP:
  - Program je lineárně reprezentován, typicky v nějakém idealizovaném strojovém/byte kódu
  - Jednodušší, prý přirozenější reprezentace
  - Jednodušší operátory (křížení mutace)
  - Často rychlá emulace běhu
  - Ale velké riziko toho, že vznikne nesmyslný program
  - Oblíbené u artificial life, evoluce kódu agentů v hrách apod

Cesta nahoru: grafové GP

- Grafové GP:
  - Program není strom, ale graf, často acyklický
  - Vznikly nejdříve jako rozšíření na paralelní programy
  - Později obecnější tvar, evoluce obvodů, neuronových sítí, ...
  - Komplikované genetické operátory (křížení obecných grafů)

NEUROEVOLUCE

Učení NS pomocí EVA

- První pokusy od 80.let
- Učení parametrů (vah)
- Učení struktury (spoje, architektura)
- Učení vah i architektury najednou
- Použití v úlohách reinforcement learning – není možné učit metodami učení s učitelem (robotika)
- Hybridní metody – kombinace EA s lokálním prohledáváním apod

Učení vah

- Přímočaré
  - Zakódování vah do vektoru,
  - floating point GA,
  - standardní operátory
- Většinou je pomalejší než specializované gradientní algoritmy (často řádově)
- + Lze ho paralelizovat
- + Lze ho použít i pro úlohy, kde mám fitness, ale ne chybu v každém kroku, takže gradientní algoritmy nemohu použít

Učení struktury

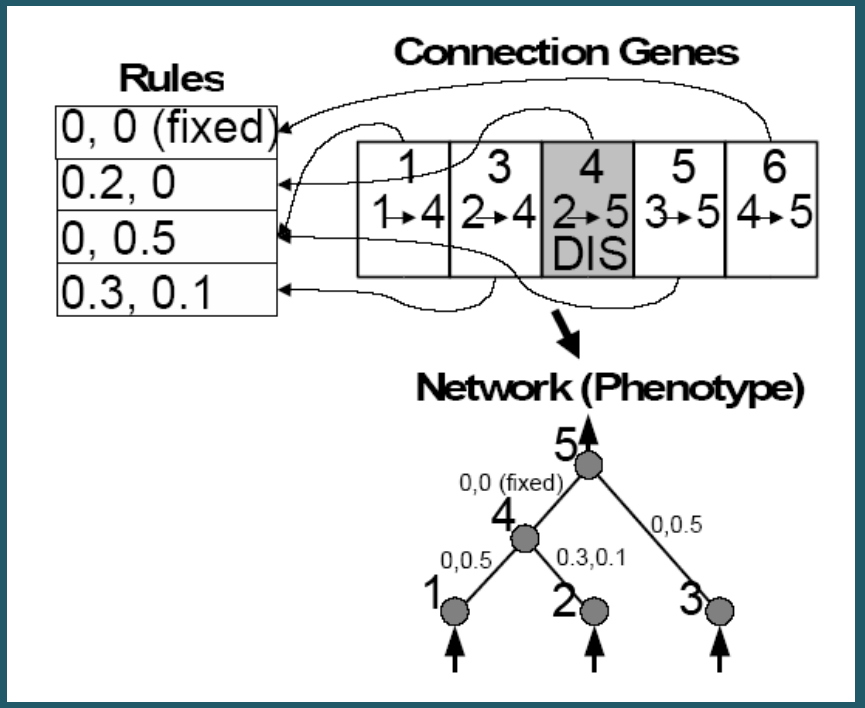
- Fitness = postavit síť, inicializovat, zkusit učit, nejlépe víckrát
- Přímé kódování
  - Realizuji strukturu sítě např. jako binární matici,
  - pracuji s evolucí linearizovaných dlouhých binárních vektorů
- Gramatické kódování
  - Kitano navrhl vyvíjet 2D formální gramatiky, které jsou „programem“ pro vytvoření matice

Učení struktury II

- Růst sítě ve 2D
  - Rané pokusy v evoluční robotice, velmi neefektivní
- Celulární kódování
  - Gruau navrhl použití Genetického programování
  - Program v GP je vlastně programem, jak nakreslit síť operacemi:
    - přidej neuron, rozděl neuron sériově, rozděl neuron paralelně, přepoj synapsi, apod.

NEAT

- K. Stanley – Neuroevolution of augmenting topologies
- Síť je seznam hran, každá hrana má:
  - informace o svých vrcholech,
  - vahách, a
  - *rodné číslo.*



NEAT pokr.

- Kříží se jen hrany se stejnými rodnými čísly, zbytek se přenáší do jedince beze změn
  - Tím se umožňuje křížení jen mezi hranami, které mají stejný evoluční původ
- Na vektorech hran s rodnými čísly se definuje podobnost (jak moc se dvě sítě od sebe liší hranami)
  - Při evoluci jsou podobné sítě zahrnuty do stejného druhu, fitness je relativní vzhledem k druhu
  - To umožní ochranu nových topologií než se jim dovyvinou váhy
- Pozdější aplikace stejného principu v algoritmu HyperNEAT.

MEMETICKÉ ALGORITMY

R. Dawkins

- Mém
  - evoluce idejí/myšlenek v lidské společnosti
  - něco jako biologická evoluce ale bez DNA
- Dva typy memetických algoritmů
  - Simulace mémů ve společnosti
    - Na hraně s psychologií, literaturou (urban legends), religionistikou
  - Využití ideje memetického/kulturního prostoru pro metaheuristiku v rámci EVA
    - Na to se podíváme

Memetické algoritmy

```
t = 0;
initialize(P(t=0));
P(t=0).localSearch();
evaluate(P(t=0));
while isNotTerminated() do
    P(t) = selectIndividuals();
    mutate(P(t));
    P(t).localSearch();
    evaluate(P(t));
    P(t+1) = buildNextGenerationFrom(P(t));
    t = t +1;
end
```

Memetické algoritmy

- Kombinace lokálního prohledávání uvnitř klasického cyklu EVA
- Lokální prohledávání:
  - Hill climbing
  - Simulované žíhání
  - Gradientní prohledávání (např. back propagation při učení neuronových sítí)
  - …

Jak si poradit s výsledkem

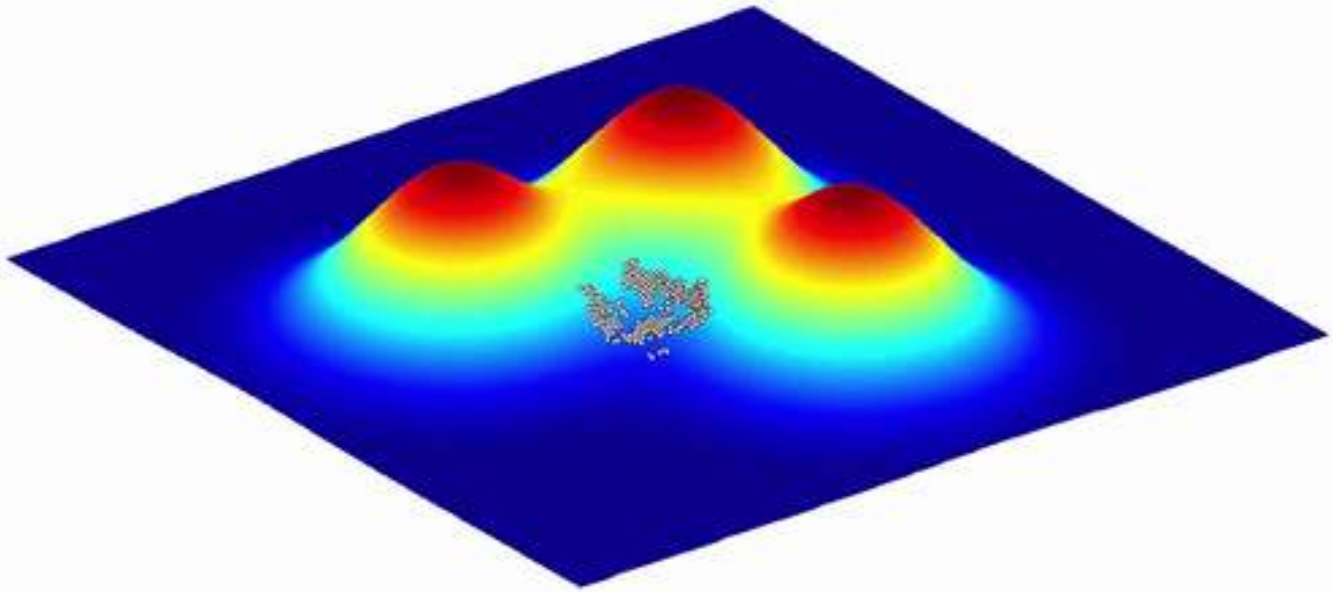
- Lamarckismus
  - Když lokálním prohledáváním najdu lepšího jedince, vezmu ho
  - To je darwinisticky nekorektní, změnil jsem genotyp na základě fenotypu
- Baldwinismus
  - Když lokálním prohledáváním najdu lepšího jedince, vezmu jen jeho fitness
  - Ale neměním genotyp

DYNAMICKÉ KRAJINY FITNESS

Krajina fitness

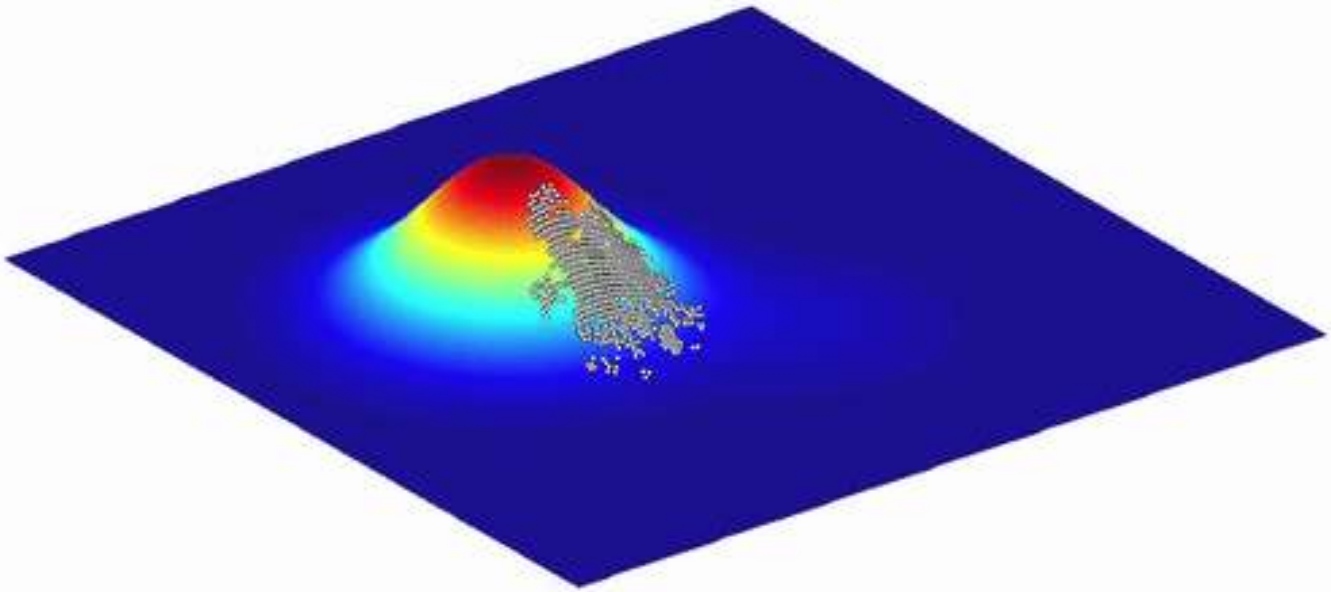
- Sewall Wright, 1932
- Grafická reprezentace fitness na:
  - Genotypu
  - Frekvenci alel
  - (nějakém rysu z) fenotypu
- Oblíbené v EVA
  - Teoretický nástroj zkoumání
  - Grafická ilustrace genotyp-fitness
    - Ale pozor na neintuitivnost vyšších dimenzí

Static fitness landscape



Population size, N = 2,304
Mutation rate, μ = 0.05 per trait

Dynamic fitness landscape



Population size, N = 2,304
Mutation rate, μ = 0.5 per trait

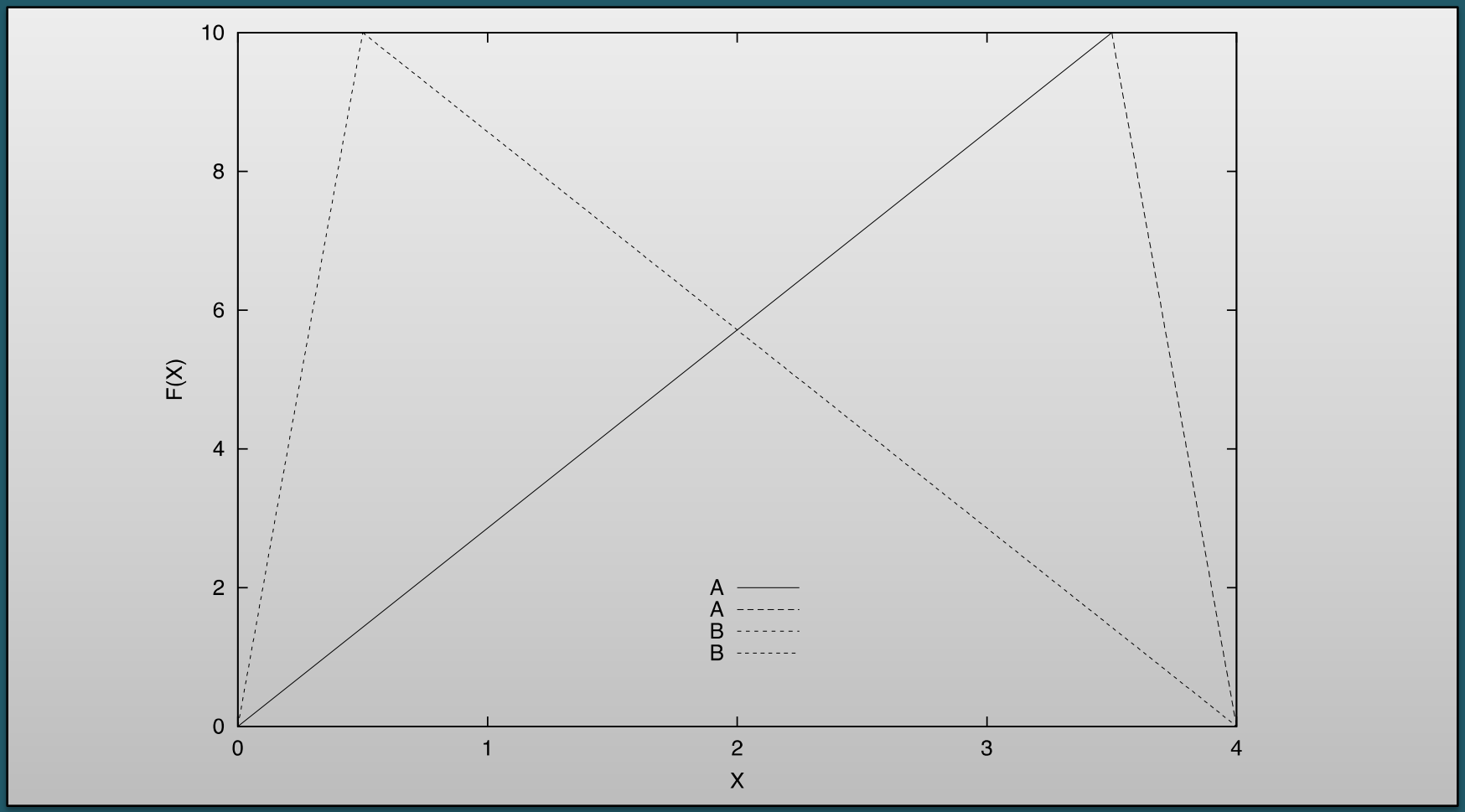
Měnící se fitness

- Proč?
  - Robotovi se rozsvítí
  - Na burze nastane krize
  - Doučování
  - Turnaje agentů mezi sebou
- Řešení:
  - Člověk změní, restart
- Lze pokračovat v evolučním algoritmu

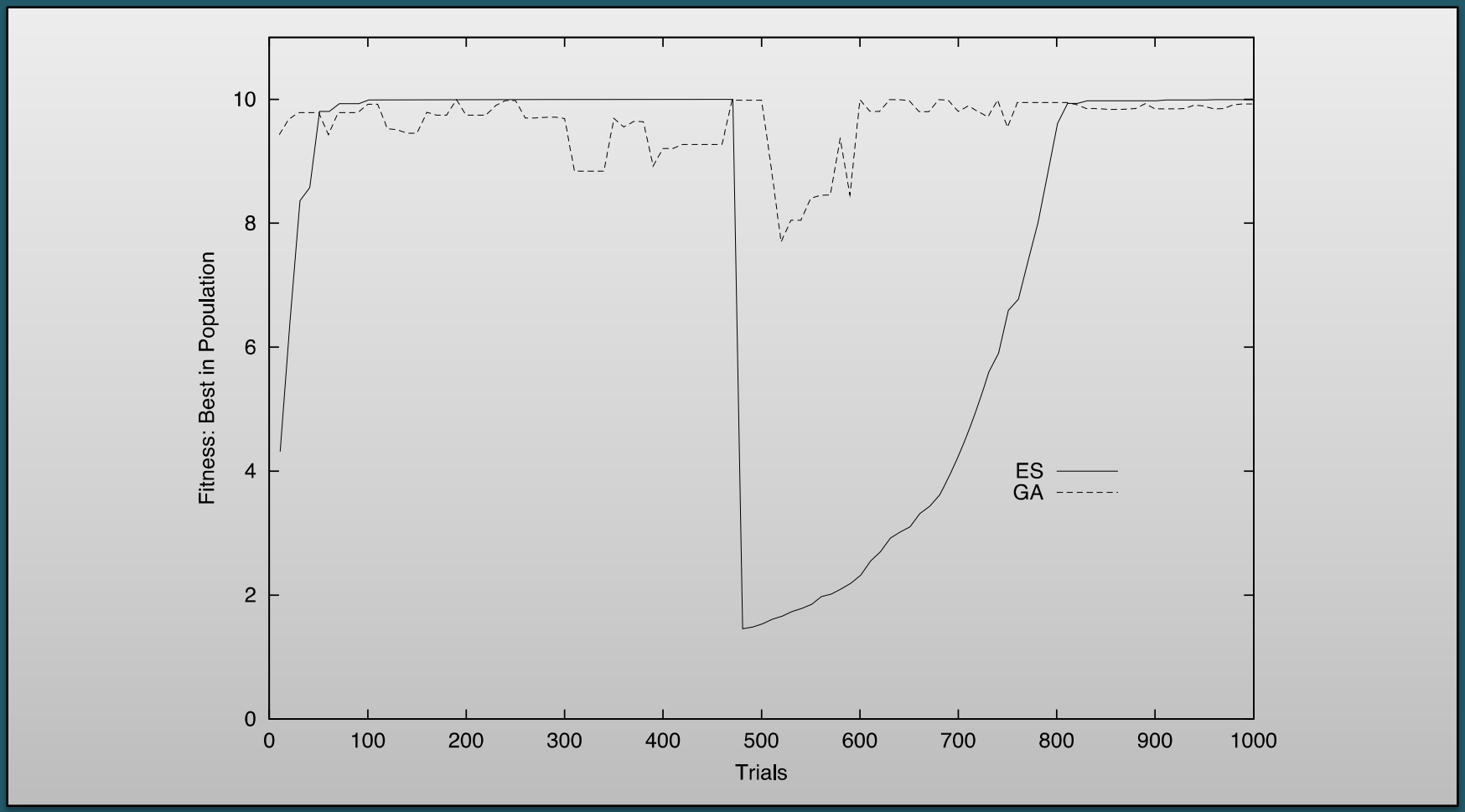
Klasické EVA na dynamické f.l.

- Většinou se nechovají moc dobře
- Typicky chceme optimalizovat EVA na úlohu
  - Rychle zkonvergovat k lokálními extrému
  - Ztratíme tím diverzitu
- Zde chceme spíš obecnost a schopnost změny
- Kenneth de Jong:
  - Porovnávání (1+10) ES a GA s 10 jedinci

Jednoduchá krajina o dvou stavech



Reakce na změnu fitness



Modifikace EVA pro dynamické f.l.

- Goldberg, Smith
  - diploidní reprezentace funguje jako dlouhodobá paměť při oscilaci
- Cobb, Grefenstette
  - Vyvolaná hypermutace
    - Při poklesu fitness se výrazně zvýší mutation rate
  - Náhodná migrace
    - Při poklesu fitness se generují noví náhodní jedinci
- Čárka v ES, odolnější pro změny fitness landscape

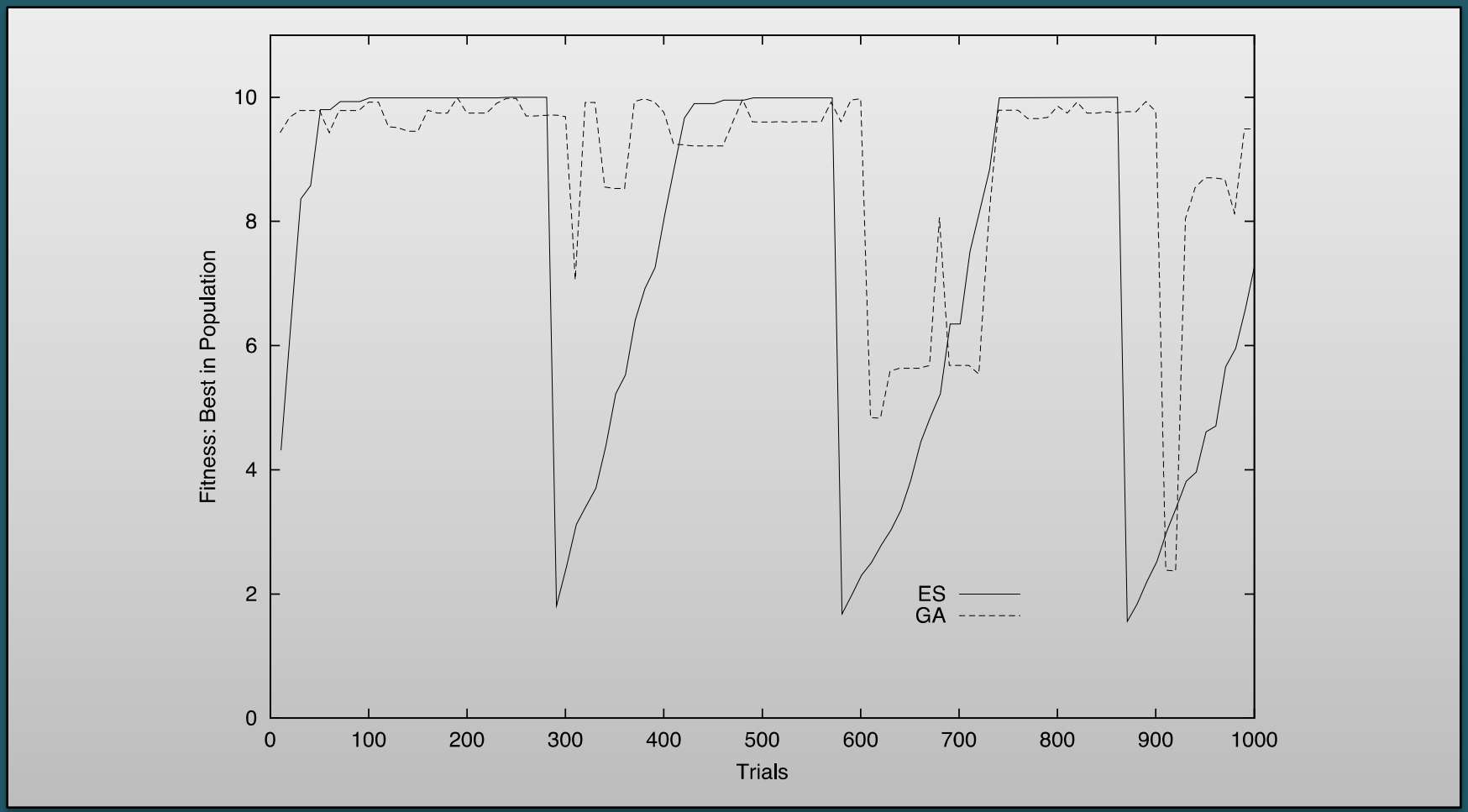
Další modifikace

- Udržet diverzitu populace
  - Zmenšit selekční tlak
  - Crowding, niching
    - Zabrání jedincům převážit v populaci
  - Subpopulace
    - Ostrovní model
  - Dynamické druhy
    - Pomocí tag-bitů
    - Množení jen mezi sebou

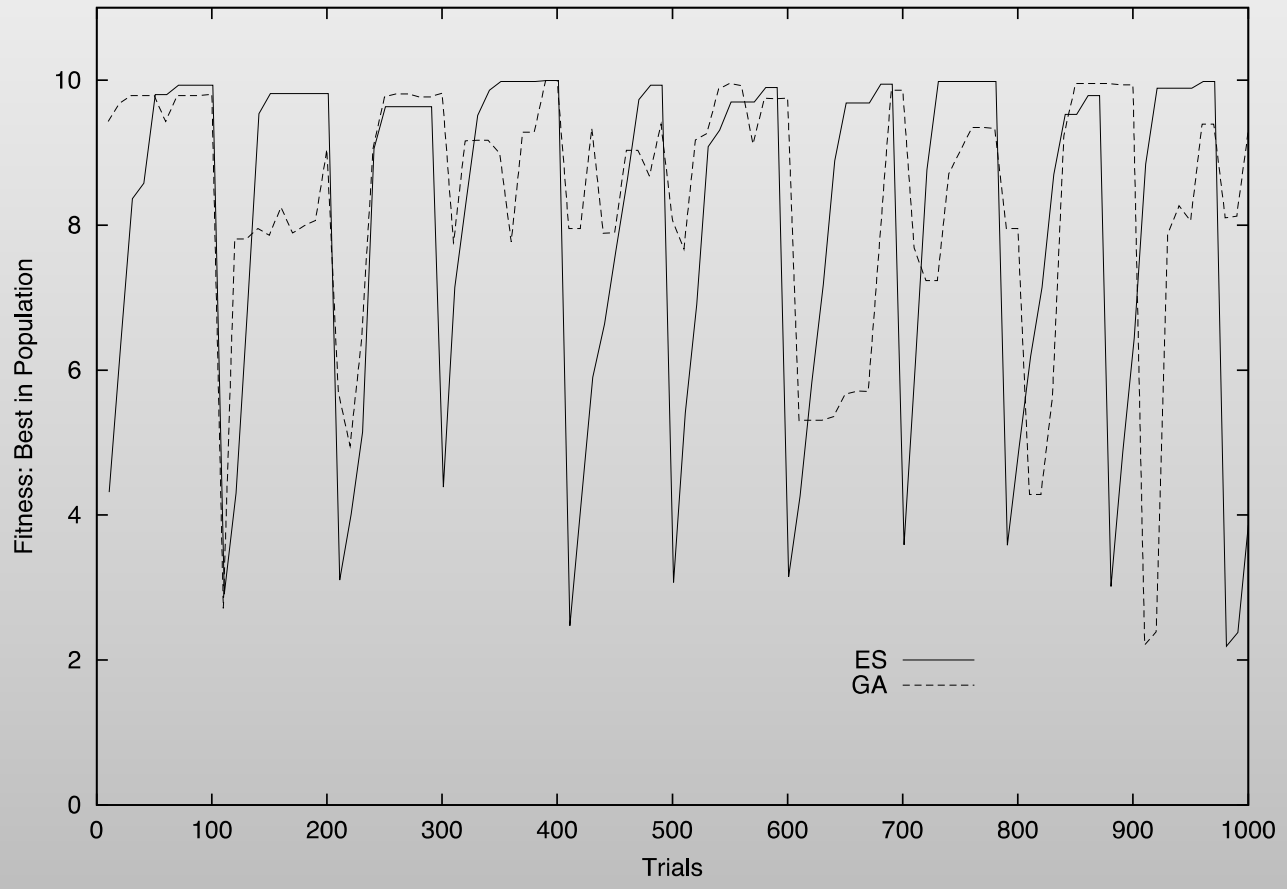
Jak dynamické jsou krajiny

- Když se bude fitness moc měnit, těžko to vyřešit obecně, kvůli No Free Lunch thm
- Malé změny
  - Robot se opotřebuje, v chemické továrně se trochu změní suroviny
- Výrazné morfologické změny
  - Vrcholy zanikají, nové vznikají, “emerging markets”
- Cyklické změny
  - Roční období, spotřeba elektřiny, …
- Nespojité katastrofické změny
  - Bouchla elektrárna, nehoda na dálnici, …

Reakce na oscilující změnu



Reakce na rychlou oscilaci



BIOLOGICKY VĚRNĚJŠÍ EVOLUCE

Druhy

- I při malém selekčním tlaku se populace rychle zbaví diverzity
- Zavední druhů a jiných mechanismů nenáhodného rozmnožování tomu možná zabrání
- Dvojí cíl:
  - Lepší reakce na změny (Darwinovské)
  - Paralelní průzkum více oblastí prostoru

Niching

- Cíle:
  - Zpomalit konvergenci k jednomu optimu
  - Vytvořit paralelní subpopulace konvergující v různých oblastech
- Omezení rekombinací na podobné jedince
- Výpočet fitness relativní pro skupinu podobných jedinců
- Crowding: turnaje podobných jedinců

Výhody a nevýhody

- Plní to ty zadané cíle
- je to citlivé na nastavení parametrů:
- Jak počítat podobnost jedinců
- Co je jedna nika

Non-random mating

- Křížení dochází jen u jedinců, kteří jsou si blízko
- Nutno definovat topologii jedinců
- Artificial life
- Ostrovní model GA
- Explicitní druhy a jen vnitrodruhové křížení

Koevoluce

- Vývoj strategií pro hry – dáma apod
- Kolektivní úkol – práce v týmu, predator-prey
- Kontextová fitness
  - Závisí na jiných jedincích v populaci
  - Nebo na jiných druzích
  - V tom se liší biologové od informatiků
- Často komplikovaná dynamika vztahů mezi ko-populacemi

Další nápady

- Morfogeneze a generativní reprezentace
  - Neuroevoluce
  - Celulární automaty
  - l-systémy
- Agentové přístupy
  - Holland a ECHO
  - Skoro cokoliv z Alife
- Lamarck – starý známý

DALŠÍ TÉMATA A PŘÍSTUPY V EVA

Tabu search

- Varianta lokálního prohledávání, která se snaží neprocházet místa, která už navštívila
- Algoritmus:
  - Jsem v řešení x(t),
  - vygeneruju náhodně x(t+1) řešení z okolí N(x(t))
  - Je-li f(x(t))<=f(x(t+1)), pokračuji s x(t+1),
  - jinak pokračuji s x(t)
  - N(x) je okolí bodu x, navíc si ale udržuji buffer posledních několika řešení x(t), které z N(x) vyřadím, tomu se říká *Tabu seznam*
- Tabu set lze přizpůsobit řešené úloze tak, aby reprezentoval i složitější omezení prostoru řešení

Tabu seznam

- Krátkodobá paměť
  - Seznam nedávno navštívených řešení
  - Nelze se do nich vracet, dokud nevyprší doba v seznamu
- Střednědobá paměť
  - Pravidla, která zintenzivní prohledávání směrem k nadějným oblastem prohledávacího prostoru
- Dlouhodobá pamět
  - Diverzifikační pravidla směřující do nových oblastí prohledávacího prostoru

Tabu search a TSP

- Efektivní zkoumání grafové struktury – ejection chain method
- Jednoduché použití:
  - Náhodné (nebo heuristika nejbližších sousedů) první řešení
  - Pak se náhodně prohodí dvě sousední města
  - Podle délky se rozhodne, které řešení se bere
  - Cyklům a uváznutí v lokálních optimech se zabrání tak, že se vytváří tabu seznam dobrých navštívených řešení
- Cyklí se do nějakého kritéria ukončení.

Scatter search

- Snaha zlepšit kvalitu řešení globálního prohledávání důrazem na diverzitu jedinců
- Algoritmus:
- Generuj počáteční populaci P(0)
- Vyber referenční podmnožinu R(0)
- V čase t:
  - Generuj nová kandidátská řešení P(t) pomocí aritmetického křížení jedinců z R(t-1)
  - Aplikuj lokální změny (mutace, hill-climbing, tabu search) na P(t)
  - Doplň (či vyměň) referenční množinu R(t) prvky z P(t)
- Pokud se R nemění, začni znovu

- Různé varianty úpravy R
- Kritériem přidání do R je typicky nějaká kombinace dobré fitness a různorodosti od ostatních členů z R (maximalizace minimální vzdálenosti)
- Upravuje se buď:
  - inkrementálně (1 až několik jedinců za populaci),
  - nebo se R zahodí a udělá znovu každou generaci,
  - nebo se dělá postupně a noví jedinci jsou rovnou již členy R a rekombinuje se s nimi.

Diferenciální evoluce

- **Inicializace**: náhodné hodnoty parametrů
- **Mutace**: „posun“ podle ostatních
- **Křížení**: uniformní „s pojistkou“
- **Selekece**: porovnání a případné nahrazení lepším potomkem

- INICIALIZACE->MUTACE->KŘÍŽENÍ->SELEKCE

Algoritmus

- Initialize all agents x with random positions in the search-space.
- Until a termination criterion is met (e.g. number of iterations performed, or adequate fitness reached), repeat the following:
- For each agent in the population do:
  - Pick three agents a,b,c from the population at random, they must be distinct from each other as well as from agent
  - Compute the agent's potentially new position y as follows:
    - y = a + F* (b-c)
  - If f(y)>f(x) then replace x with y.
- Pick the agent from the population that has the highest fitness or lowest cost and return it as the best found candidate solution.

Mutace

- Každý jedinec v populaci projde mutací, křížením a selekcí
- Pro jedince $\boldsymbol{x}_{i,p}$ zvolme tři různé jedince $\boldsymbol{x}_{a,p}$, $\boldsymbol{x}_{b,p}$, $\boldsymbol{x}_{c,p}$
- Definujme donora $\boldsymbol{v}$: $\boldsymbol{v}_{i,p+1} = \boldsymbol{x}_{a,p} + F.(\boldsymbol{x}_{b,p} - \boldsymbol{x}_{c,p})$
- F je parametr mutace, konstanta z intervalu <0;2>

Křížení

- Uniformní křížení původního jedince s donorem
- Parametr *C* určuje pravděpodobnost změny
- Ve výsledku zajištěno aspoň 1 prvek z donora
- Pokusný vektor *ui,p+1* :
- $\boldsymbol{u}_{j,i,p+1} = \boldsymbol{v}_{j,i,p+1}$; iff $rand_{ji} <= C$ or $j=I_{rand}$
- $\boldsymbol{u}_{j,i,p+1} = \boldsymbol{x}_{j,i,p+1}$; iff $rand_{ji} > C$ and $j \neq I_{rand}$
- $rand_{ji}$ je pseudonáhodné číslo z <0;1>
- I_{rand} náhodný int z <1;2; ... ; D>

Selekce

- Porovnáme fitness $\boldsymbol{x}$ a $\boldsymbol{v}$ a lepšího vezmeme:
  - $\boldsymbol{x}_{i,p+1} = \boldsymbol{u}_{i,p+1}$; iff $f(\boldsymbol{u}_{i,p+1}) <= f(\boldsymbol{x}_{i,p})$
  - $\boldsymbol{x}_{i,p+1} = \boldsymbol{x}_{i,p}$; jinak
  - pro $i=1,2, \ldots , N$
- Mutaci, křížení a selekci opakujeme dokud není splněno nějaké ukončovací kriterium, (typicky např. fitness nejlepšího už je dost dobrá)