EVA 1 (10-11)

R. Neruda

Evoluční algoritmy

- Hollandovy genetické algoritmy (binární řetězce)
- Fogelovo evoluční programování (automaty)
- Kozovo genetické programování (stromy)
- Schwefelovy evoluční strategie (parametry funkcí)

- Rayův umělý život (sebekopírující assembler)
- Hollandovy klasifikační systémy (pravidla)

Odkazy:

- Holland: Adaptation in natural and artificial systems, MIT Press, 1992.
- Goldberg: Genetic algorithms in Optimization, Search and Learning, Addison-Wesley, 1989.
- Koza: Genetic Programming, I-III, MIT Press, 1992, 1994, 1999.
- Mitchell: Introduction to GA, MIT Press, 1996.
- Michalewicz: GA+DS=EP, Springer, 2001
- Hitch-hiker's guide to EC

ÚVOD

Obecný evoluční algoritmus

- Populace P(t) jedinců $\{x_1(t), \ldots x_n(t)\}$
- Každý jedinec je zakódované řešení problému
- Každý jedinec má fitness (jak dobře řeší problém)
- Vytváření populace P(t+1):
  - Výběr (selekce)
  - Křížení apod
  - Mutace apod

OEA pokr.

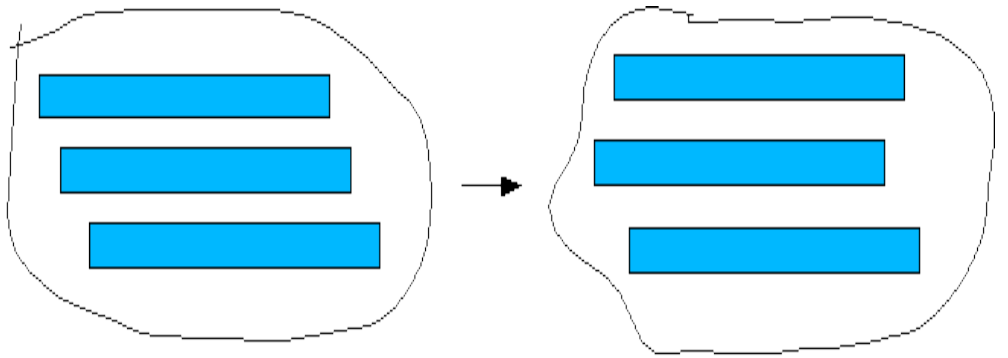
```
Evolution program() {
    t=0; initialze P(t); evaluate P(t);
    while (! end) {
        t++;
        select P(t) from P(t-1);
        alter P(t);
        evaluate P(t);
    }
}
```

Jednoduchý genetický algoritmus

- genetická reprezentace řešení
- vytvoření iniciální populace
- ohodnocovací/účelová/fitness funkce
- genetické operátory:
  - výběr, křížení, mutace
- parametry GA:
  - velikost populace, pravděpodobnosti operátorů

JGA pokr.

- Reprezentace:
  - Binární řetězce
- Mutace:
  - Náhodná změna bitu
- Křížení
  - Jednobodové
- Selekce
  - Ruletová



TEORIE SCHÉMAT

Schémata

- Schéma je slovo v abecedě 0, 1 a* (don't care)
- Schéma reprezentuje množinu řetězců
- Schéma s r * reprezentuje 2^r řetězců
- Řetězec délky m je reprezentován 2^m schématy
- Je 3^m schémat délky m
- V populaci velikosti n je 2^m až $n.2^m$ schémat

Schémata pokr.

- Řád schématu S: o(S)
  - Počet 0 a 1 (pevných pozic)
- Definující délka: d(S)
  - Vzdálenost mezi první a poslední pevnou pozicí
- Fitness schématu: F(S)
  - Průměrná fitness odpovídajících řetězců v populaci

Věta o schématech

- Krátká nadprůměrná s malým řádem schémata se v populaci během GA exponencielně množí.
- Hypotéza o stavebních blocích:
  - GA hledá suboptimální řešení problému rekombinací krátkých, nadprůměrných s malým řádem schémat (building blocks).
  - “just as a child creates magnificent fortress through arrangement of simple blocks of wood, so does a GA seek near optimal performance ...”

Důkaz věty o schématech 1

- Populace P(t), P(t+1), ... n jedinců délky m
- Co se děje s konkrétním schématem S při:
  - Selekci
  - Křížení
  - Mutaci
- C(S,t) ... četnost schématu S v populaci P(t)

(počet řetězců v reprezentovaných S v P(t))

- Postupně odhadujeme C(S,t+1)

Důkaz věty o schématech 2:

- Selekce:
  - Řetězec v má pravděpodobnost vybrání:
    $p_s(v)=F(v)/F(t)$, kde $F(t)$ je suma $F(u)$, u z $P(t)$
  - Schéma S má pravděpodobnost vybrání:
    $p_s(S)=F(S)/F(t)$
  - Tedy: $C(S,t+1) = C(S,t)\ n\ p_s(S)$
  - Jinými slovy: $C(S,t+1)=C(S,t)\ F(S)/F_{prum}(t)$
  - $F_{prum}(t)=F(t)/n$ průměrná fitness v $P(t)$

Důkaz věty o schématech 3:

- ... ještě selekce:
  - Shrňme si to: $C(S,t+1)=C(S,t)\ F(S)/F_{prum}(t)$
  - Kdyby bylo schéma “nadprůměrné” o e%:
    - $F(S,t)=F_{prum}(t) + e\ F_{prum}(t)$, pro t=0, ...
  - $C(S,t+1)=C(S,t)\ (1+e)$
  - $C(S,t+1)=C(S,0)\ (1+e)^t$
- Četnost nadprůměrných schémat roste geometrickou řadou (v populacích (po selekci)).

Důkaz věty o schématech 4:

- Křížení:
  - Pravděpodobnost, že schéma křížení (ne)přežije:

    $p_d(S) = d(S)/(m-1);$ $\quad p_s(S) = 1 - d(S)/(m-1)$

  - Křížení má pravděpodobnost pc:

    $p_s(S) >= 1 - p_c \cdot d(S) / (m-1)$

- Selekce a křížení dohromady:

$$C(S,t+1) >= C(S,t) \cdot F(S)/F_{prum}(t) \, [1 - p_c \cdot d(S) / (m-1)]$$

Důkaz věty o schématech 5:

- Mutace:
  - 1 bit (ne)přežije: p_m; $1 - p_m$
  - Schéma nepřežije ($p_m << 1$):
  - $p_s(S) = (1 - p_m)^{o(S)}$ = asi tak = $1 - p_m . o(S)$
- Selekce, křížení a mutace dohromady:

  $C(S,t+1) >= C(S,t).F(S)/F_{prum}(t) [1-p_c.d(S)/(m-1)-p_m.o(S)]$
- QED.

Význam VoS a HoSB

- Na zakódování záleží
- Na velikosti záleží
- Předčasná konvergence škodí
- Co GA nejde:
  - (111*******) a (********11) jsou nadprůměrní
  - Ale F(111*****11) << F(000*****00)
  - Ideál je (1111111111); GA ho těžko najde
- Podmínka na výběr je divná

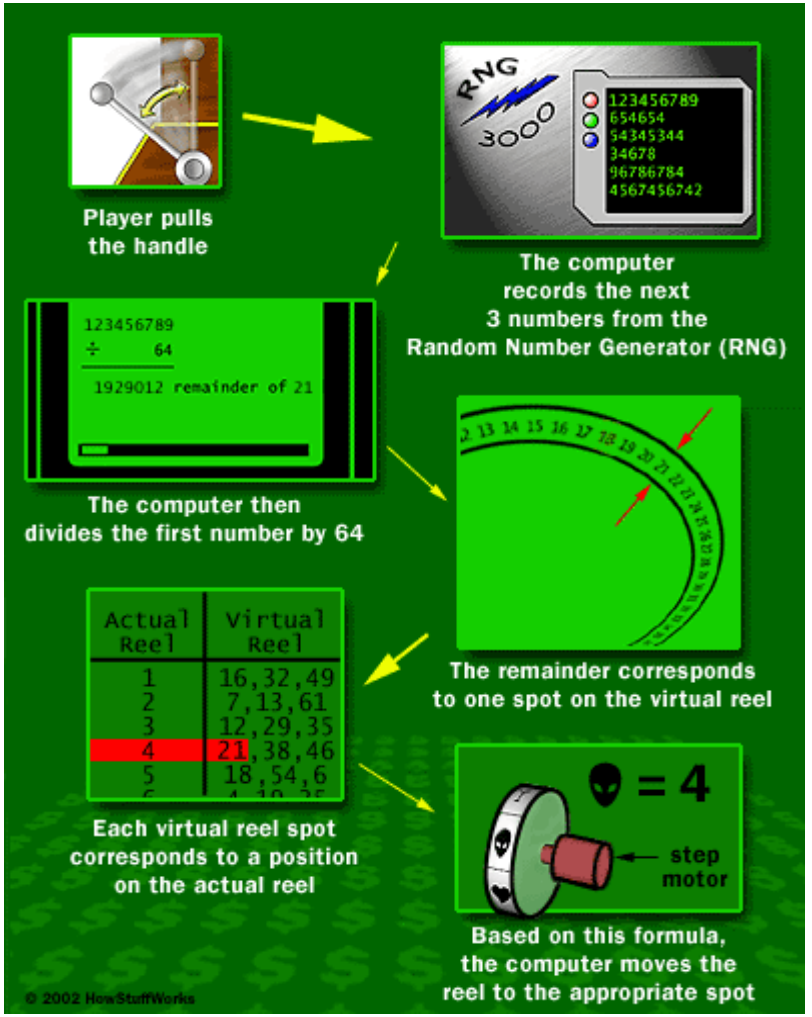
Implicitní paralelismus

- GA pracuje s n jedinci, ale “implicitně” vyvíjí (mnohem více) schémat: 2^m až $n.2^m$.
- Kolik schémat se v GA zpracuje efektivně:
- Holland (a další): (Za různých přepokladů, např., že $n = 2^m$, schémata zůstávají nadprůměrná, ...) Počet schémat, kterým se v GA dostává exponenciálního růstu, je úměrný n^3.
- Jediný příklad kombinatorické exploze na straně dobra. (Asi ani to ne.)

Exploration vs. Exploitation

- Původní motivace: GA je “adaptivní plán” vyvažující při hledání řešení napětí mezi
  - explorací (nacházení nových oblastí hledání)
  - exploatací (využití stávajících znalostí)
- Jen explorace: náhodné procházky, nevyžívá se předchozích znalostí
- Jen exploatace: uvíznutí v lokálních extrémech, rigidita

1-ruký bandita



2-ruký bandita

- Mám N mincí, stojím před dvourukým banditou (ruce vyplácejí se střední hodnotou m_1, m_2 a rozptylem s_1, s_2). N-n alokuji lepší ruce, n horší.
- Cíl: maximalizovat zisk / minimalizovat ztrátu.
- Analytické řešení: alokovat exponenciálně více pokusů pro právě vyhrávající ruku.
- $N-n = O(\exp(c\ n))$; c závisí na m_1, m_2, s_1, s_2

... a GA

- GA taky alokuje exponenciálně mnoho nadějným schématům
- Řeší tedy optimálně problém expl. vs. expl.
- Schémata spolu hrají mnohoruké bandity
  - Výhra je počet pozic v populaci
  - Je odhadnut hodnotou fintess schématu
  - Nejdříve se myslelo: GA hraje 3^m rukého banditu,
  - Všechny schémata jsou konkurenční ruce

Ale ...

- Hraje se paralelně více her
- Schémata “soutěží” o “konfliktní” pevné pozice
- Schémata řádu k soutěží vždy o těch konkrétních k pevných pozic – hrají spolu 2^k rukého banditu
- Takže vždy nejlepší z této hry dostane exp. mnoho míst v populaci
- A to ještě jen tehdy, když GA může v populaci správně odhadnout fitness schémat

Takže, 'blbá' úloha pro GA je ...

- = 2; pro x ~ 111*... *
- f(x) = 1; pro x ~ 0*...*
- = 0; jinak
- Pro schémata platí:
  - F(1*...*) = ½ ; a F(0*...*) = 1
- Jenže GA odhadnou F(1*...*) ~ 2,
- Protože 111*...* v populaci převáží

- GA zde nesampluje schémata nezávisle, takže neodhadne jejich skutečnou fitness.

Problémy

- V banditovi jsou ruce nezávislé, GA ale nesampluje schémata nezávisle
- Selekce nepracuje “ideálně” jako ve VoS, je dynamická.
- GA maximalizuje on-line performance, jsou velmi vhodné pro adaptivní případy. (Škoda je zastavovat ;-)
- (Paradoxně) se nejčastěji používají “jen” pro hledání nejlepšího řešení.

“Statická hypotéza o blocích”

- Grafenstette, 91: Lidi předpokládají, že GA konverguje k řešením se skutečně nejlepší statickou průměrnou fitness; a ne k těm existujícím v populacích s nejlepší pozorovanou fitness.
- To pak lidi může zklamat:
  - Kolaterální konvergence
  - Velký rozptyl fitnees

Kolaterální konvergence

- Když už se někam začne konvergovat, nemusejí být schémata samplována stejnoměrně.
- Je-li např. schéma 111***...* dobré, převáží v populaci po pár generacích (skoro všechny řetězce tak začínají).
- Pak ale skoro všechny samply schématu ***000...* jsou i samply schématu 111000*...*.
- Takže GA neodhadne F(***000*...*) správně.

Velký rozptyl fitness

- Má-li statická průměrná fitness schématu velký rozptyl, GA ji zase nemusí odhadnout dobře.
- Viz schéma 1*...* z našeho blbého příkladu.
- Rozptyl jeho fitness je velký, takže GA nejspíš bude konvergovat jen na těch částech prostoru, kde má fitness velkou hodnotu.
- To ale zatíží další samplování schématu chybou, takže se neodhadne správně jeho statická fitness.

REPREZENTACE

Kódování

- Binární
  - Odjakživa
  - Teoretické výsledky
  - Holland: binární řetězce délky 100 jsou lepší než desítkové délky 30, protože mají víc schémat ($2^{100}>2^{30}$).
  - ale to nestačí!
  - často nepřirozené a neintuitivní pro daný problém.

A jiná kódování

- Víceznakové abecedy
- Floating point
- Úplně jiné:
  - Stromy,
  - Matice,
  - Neuronové sítě
  - Zvířátka

Floating point-Křížení

- Jen na hranicích čísel
- Jedno ... vícebodové
- Uniformní:
  - náhodně se u každé pozice zvlášť rozhodne, ze kterého rodiče se vezme
- Aritmetické:
  - Pi = (Xi + Yi) / 2
  - Nebo Pi = rnd [Xi,Yi]

Floating point-Mutace

- Bitová mutace nevýhodná
- Nezatížená mutace
  - Pi = rnd [DMi, HMi]
  - Výrazná změna, nebere v úvahu předch. hodnotu
- Zatížená mutace
  - A) Pi = Xi +/- rnd_Norm;
  - B1) hod korunou, padne + nebo -;
  - Pi = Xi + F(t, HMi-Xi) anebo = Xi - F(t,Xi-DMi)

EVOLUČNÍ STRATEGIE

Evoluční strategie

- Rechenberg, Schwefel, 60.léta
- optimalizace multidimenzionálních funkcí
- 'evoluce evoluce'
- evolvovaný jedinec:
  - *Genetické* parametry ovlivňující chování
  - *Strategické* parametry ovlivňující evoluci
- nový jedinec akceptován jen je-li lepší
- na vzniku se může podílet více jedinců

ES notace

- Důležité parametry:
  - M počet jedinců v populaci
  - L počet vznikajících potomků
  - R počet 'rodičů'
- Zvláštní notace souvisí se selekcí:
  - (M+L) ES – M jedinců do nové populace je vybráno z M+L starých i nových jedinců
  - (M,L) ES – M nových jedinců je vybráno jen z L nových potomků
- Jedinec C(i)=[Gn(i),Sn(i)]

ES cyklus

1. n=0;Náhodně inicializuj populaci Pn M jedinců
2. Ohodnoť jedince Pn pomocí fitness
3. Dokud není řešení dostatečně dobré:
   a) Opakuj L krát:
      i. vyber R rodičů,
      ii. zkřiž, mutuj, ohodnoť nového
   b) Vyber M nových (podle typu ES)
   c) ++n

ES jedinec a mutace

- C(i)=[Gn(i),Sn(i)]
- Sn jsou:
  - Standardní odchylky floating point mutací
  - Nekorelované mutace
  - Případně vylepšené o „rotace“:
  - Korelované mutace
    - Nemutuje se jen podle os dimenzí
    - Násobí se maticí rotace pro určení optimálních směrů
    - (stačí vektor: n-rozměrný)
    - Takže metaparametrů je 2n

ES mutace

- Genetické parametry:
  - Přičtení náhodného čísla z normálního rozdělení (s příslušnou odchylkou (a rotací))
- Odchylky:
  - Zvětšovat nebo zmenšovat podle úspěšnosti mutace
  - 1/5 pravidlo heuristika
- Rotace:
  - Přičtení náhodného čísla z N(0,1)

ES Křížení

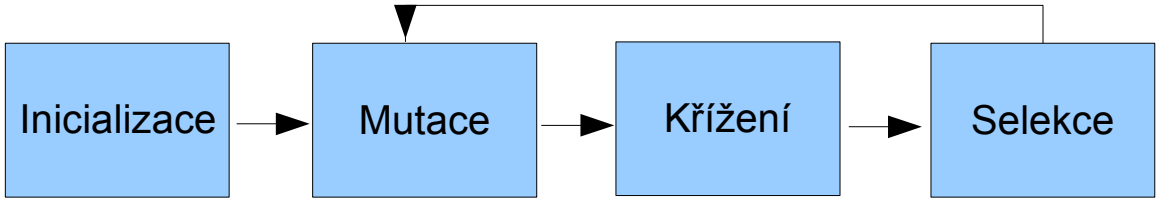
- Uniformní
- Gang bang více rodičů
  - Lokální (R=2)
  - Globální (R=M)
- Dvě verze
  - Diskrétní
  - Aritmetické (průměr)

Diferenciální evoluce

Diferenciální evoluce

- Storn, Price – 1996
- Stochastický, populační optimalizační alg.
- Hledá optimální reálné parametry reálné funkce
- Jedinec $\mathbf{x}_{ip} = [x_{1,i,p}, x_{2,i,p} \ldots x_{D,i,p}]$
  - D je počet parametrů funkce
  - i = 1, ... , N, kde N>3 je velikost populace
  - p je číslo populace

DE – schéma, inicializace



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

Mutace

- Každý jedinec v populaci projde mutací, křížením a selekcí
- Pro jedince $\mathbf{x}_{i,p}$ zvolme tři různé jedince $\mathbf{x}_{a,p}$, $\mathbf{x}_{b,p}$, $\mathbf{x}_{c,p}$
- Definujme *donora* **v**: $\mathbf{v}_{i,p+1} = \mathbf{x}_{a,p} + F.(\mathbf{x}_{b,p} - \mathbf{x}_{c,p})$
- 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 $\mathbf{u}_{i,p+1}$:
  - $u_{j,i,p+1} = v_{j,i,p+1}$; iff $rand_{ji} <= C$ or $j=I_{rand}$
  - $u_{j,i,p+1} = x_{j,i,p+1}$; 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 **x** a **v** a lepšího vezmeme:
  - $x_{i,p+1} = u_{i,p+1}$; iff $f(u_{i,p+1}) <= f(x_{i,p})$
  - $x_{i,p+1} = x_{i,p}$; jinak
  - pro i=1,2, ... , N
- Mutaci, křížení a selekci opakujeme dokud není splněno nějaké ukončovací kriterium, (typicky např. fitness nejlepšího už je dost dobrá)

PSO

Particle swarm optimization

- Populační prohledávací algoritmus
- Eberhart, Kennedy, 1995
- Inspirace hejny hmyzu/ryb
- Jedinec je typicky vektor reálných čísel
- Říká se mu *částice*
- Nejsou zde operace křížení, mutace
- Jedinci se pohybují v hejnu prostorem parametrů

PSO - algoritmus

For each particle: Initialize particle END
Do
    For each particle
        Calculate fitness value
        If the fitness value is better than the best fitness value (pBest) in history
            set current value as the new pBest
    End

    Choose the particle with the best fitness value of all the particles as the gBest
    For each particle
        Calculate particle velocity according equation (a)
        Update particle position according equation (b)
    End
While maximum iterations or minimum error criteria is not attained

PSO – rovnice pohybu

- V[] : = v[] +

  + c1 * rand() * (pbest[] - present[]) + c2 *
  + rand() * (gbest[] - present[]) (a)

- present[] = persent[] + v[] (b)

v[] je rychlost částice, present[] je pozice částice.

pbest[], gbest[] viz alg.

rand () náhodné číslo z (0,1). c1,

c2 konstanty (učící faktory) často c1 = c2 = 2.

PSO - diskuse

- **Společné s GA:**
  - Začínají z náhodné konfigurace, prohledávají prostor, mají fitness jako ohodnocení, používají stochastické metody
- **Odlišné:**
  - Nejsou zde genetické operace
  - Částice mají paměť
  - Výměna informací je jen od nejlepších částic ostatním

Evoluční strojové učení

Strojové učení

- Učení pravidel na základě předkládaných dat
  - Data mining
  - Expertní systémy
  - Učení agentů, robotů (posilované učení)
- Základní přístupy:
  - Michiganský (Holland): pravidlo je jedinec
    - Hollandovy LCS: učící se klasifikační systémy (vhodné pro posilované učení)
  - Pittsburský: jedinec je množina pravidel

Michigan

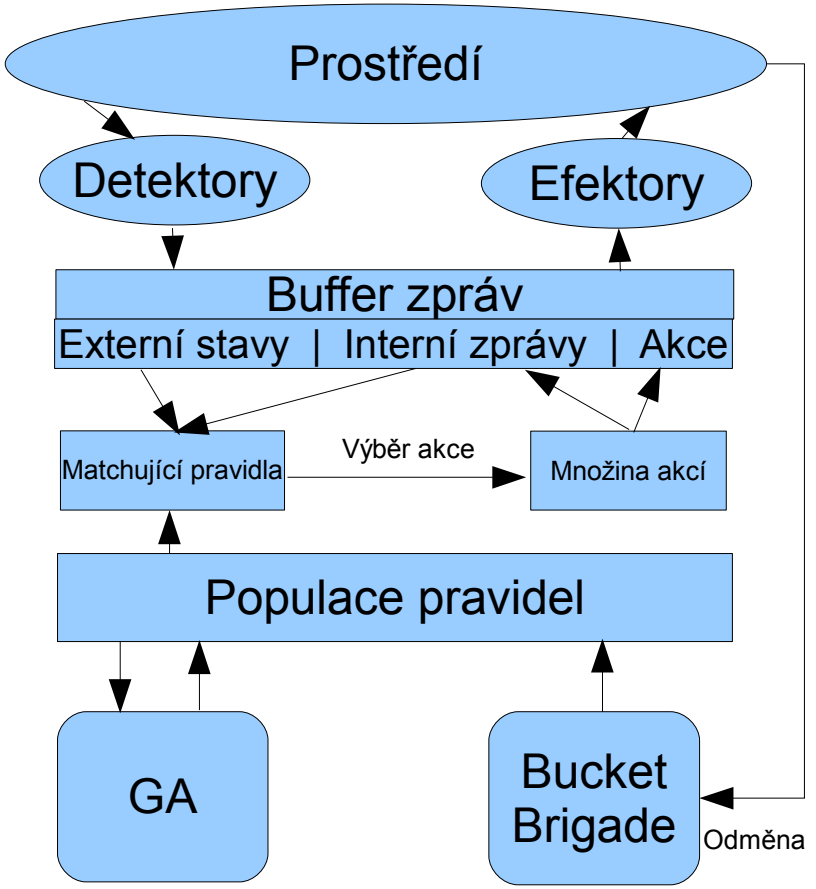
- Holland v 80.letech: learning classifier systems
- Jedinec GA je pravidlo, celá populace pak funguje jako řídící či expertní systém
- Jednoduchá pravidla:
  - Pravá strana: příznak nastal/ne/don't care
  - Levá strana: kód akce či klasifikace kategorie
- Pravidla mají váhu (úspěšnost),
- Ta určuje fitness v evoluci
- Evoluce nemusí probíhat v generacích

Michigan - LCS

- Evoluce probíhá jen občas a jen na části populace
- Problém reaktivnosti (absence vnitřní paměti)
  - Pravidla mohou mít na pravé straně – kromě klasifikačního výstupu – další vnitřní příznaky - „zprávy“
  - a na levé straně „receptory“ na jejich příjem,
  - Systém má pak frontu zpráv a musí realizovat algoritmus odměn pro pravidla

LCS – Bucket brigade

- Jen některá pravidla vedou k akci, za kterou následuje odměna/trest od prostředí,
- Rozdělění odměny – pro celý řetěz úspěšných pravidel
- Pravidla musejí dát část své síly (jakoby peněz), když chtějí soupeřit o možnost být v cestě k řešení
- *Bucket brigade algoritm*, v praxi velmi komplikované a těžkopádné



Pitt

- Jedinci jsou množiny pravidel
- Ohodnocení komplikovanější
  - Priority pravidel, konflikty
  - False positives, false negatives
- Genetické operátory komplikovanější
  - Typicky vede k desítkám operátorů pracujících na úrovni celých množin, jednotlivých pravidel, termů v pravidlech
  - Důraz na bohatou reprezentaci domén (množiny, výčtové typy, intervaly, ...)

Pitt – ukázka GIL (Janikow)

- Binární klasifikační úloha
- Jedinec klasifikuje implicitně do jedné třídy (nemá pravou stranu)
- Každý jedinec je disjunkce *komplexů*
- Komplex je konjunkce *selektorů* (z 1 proměnné)
- Selektor je disjunkce hodnot z domény proměnné
- Reprezentuje se bitovou mapou:

((X=A1)AND(Z=C3)) OR((X=A2)AND(Y=B2))

[001|11|0011 OR 010|10|111]

GIL pokr.

- Operátory na úrovní jedince:
  - výměna pravidel, kopírování pravidel, generalizace pravidla, smazání pravidla, specializace pravidla, zahrnutí jednoho pozitivního příkladu
- Operátory na úrovni komplexů:
  - rozdělení komplexu na 1 selektoru, generalizace selektoru (nahrazení 11...1), specializace generalizovaného selektoru, zahrnutí negativního příkladu
- Operace na selektorech:
  - Mutace 0<->1, rozšíření 0->1, zúžení 1->0,

NP-TĚŽKÉ ÚLOHY

EA řeší těžké úkoly

- Batoh (0-1 knapsack problem)
  - Jednoduché zakódování
  - Jednoduchá (ale problematická) fitness
  - Standardní operátory
- Obchodní cestující - POC (TSP)
  - Jednoduchá fitness
  - Problematické kódování a příslušné op (křížení)
- rozvrhování, plánování, dopravní problémy ...

Batoh v. 1

- Máme:
  - batoh kapacity CMAX
  - a N věcí,
    - každá má cenu v(i)
    - a objem c(i)
- Úkolem je vybrat z nich takové, abychom
  - maximalizovali sumu v(i)
  - a zároveň se vešli do batohu, t.j. suma c(i) <= CMAX

Batoh v. 1

- Zakódování přímočaré – bitová mapa:
  - 0110010 – bereme věci 2,3 a 6
  - je to až triviální
  - ale jedinci nemusejí splňovat podmínku o CMAX
- Operátory:
  - Přímočaré křížení, mutace, selekce
- Fitness: dva členy:
  - max [suma v(i)] vs. min [CMAX – suma c(i)]

Batoh v.2

- Problém s multi-účelovou optimalizací lze řešit různě:
  - Oba členy fitness zvážit a sečíst
  - Použít 1 z obecných EA pro multi-účelové funkce
- Anebo chytře změnit zakódování:
  - 1 znamená: DEJ věc do batohu POKUD se nepřeplní
  - takhle dosáhneme i toho, že všechny řetězce jsou přípustná řešení

POC

- Máme N měst, chceme je objet s minimálními náklady
- Fitness je zřejmá – náklady na cestu
- Reprezentace jsou různorodé
  - Varianty reprezentací pomocí vrcholů
  - Reprezentace pomocí hran, ...
- Operátory jsou závislé na reprezentaci
- Křížení umožňuje využít heuristiky k řešení POC

Reprezentace sousednosti

- Cesta je seznam měst, město j je na pozici i, vede-li cesta z i do j
- Příklad
  - (248397156) odpovídá cestě 1-2-4-3-8-5-9-6-7
- Každá cesta má jen 1 reprezentaci, ale některé seznamy negenerují přípustnou cestu
- Klasické křížení nefunguje (různé opravy)
- Ale schémata ano:
  - Např (*3*...) značí všechny cesty s hranou 2-3

Reprezentace bufferem čili ordinální

- Motivována tím, aby šlo použít normální 1-bodové křížení (ale má smysl?)
- Mějme buffer vrcholů (třeba uspořádaný) a reprezentace představuje pořadí města v bufferu
- Města se z bufferu po použití mažou
- Příklad:
  - Buffer (123456789) a cesta 1-2-4-3-8-5-9-6-7 je repezentována jako (112141311)

Reprezentace cestou čili permutací

- Každého asi napadne první, má význam i pro kódování jiných úloh.
- cesta 5-1-7-8-9-4-6-2-3 je tedy repr. (517894623)
- Nefunguje křížení, proto byly a jsou vyvíjeny operátory, které produkují správné reprezentace a navíc obsahují nějakou ideu o tom, jak udělat dobré řešení
- PMX, CX, OX, ...

PMX

- Partially mapped crossover (Goldberg)
- Snaží se zachovat co nejvíce měst na pozicích z daných měst, jde-li to. (2-bodové křížení)
- (123|4567|89) PMX (452|1876|93) :
  - (...|1876|..) (...|4567|..) a zobr. 1-4 8-5 7-6 6-7
  - lze doplnit (.23|1876|.9) (..2|4567|93)
  - a dle zobr. (423|1876|59) (182|4567|93)

OX

- Order crossover (Davis)
- Zachovává relativní pořadí měst dle jednotl.cest
- (123|4567|89) OX (452|1876|93) :
  - (...|1876|..) (...|4567|..) a přeuspořádáme cestu 2 od druhého bodu křížení 9-3-4-5-2-1-8-7-6
  - vyhodíme překřížená města z 1, zbyde: 9-3-2-1-8
  - doplníme potomka 1: (218|4567|93)
  - obdobně potomek 2: (345|1876|92)

CX

- Cyclic crossover (Oliver)
- Snaží se zachovat absolutní pozici města v cestě
- (123456789) CX (412876935)
  - první pozice, náhodně třeba z prvního P1=(1.....),
  - teď musíme vzít 4, P1=(1..4....), pak 8, 3 a 2
  - P1=(1234...8.), dál už nelze, doplníme z druhého
  - P1=(123476985)
  - Obdobně P2=(412856739)

ER

- Edge recombination (Whitley et al)
- Předchozí křížení zachovají max 60% z obou rodičů, snaha zachovat jich co nejvíc
- Pro každé město si udělám seznamy hran, začnu někde (1 městem), vybírám města s méně hranami, v případě shod počtu hran se mezi nimi rozhodnu náhodně
- (123456789) ER (412876935)
  - Vytvořím seznam

ER pokr.

- 1: 9 2 4
- 2: 1 3 8
- 3: 2 4 9 5
- 4: 3 5 1
- 5: 4 6 3
- 6: 5 7 9
- 7: 6 8
- 8: 7 9 2
- 9: 8 1 6 3

ER posl.

- Začnu v 1, následníci jsou 9, 2, 4
- 9 vypadává, má 4 násl., z 2 a 4 náhodně vyberu 4
- násl. 4 jsou 3 a 5, beru 5, neboť 5 má 3 násl, zatímco 3 má 4 násl. (to je ale nepřehledné)
- Teď máme (145.....), podobně pokračujeme
- ... (145678239)
- Může dojít k tomu, že nelze hranu vybrat, a křížení selže, ale jen zřídka (1-1.5% případů)