Genetické programování

- Vyvinuto v USA v 90. letech J. Kozou
- Typické problémy:
  - Predikce, klasifikace, aproximace, tvorba programů
- Vlastnosti
  - Soupeří s neuronovými sítěmi apod.
  - Potřebuje značně velké populace (tisíce)
  - Velice pomalá
- Zvláštnosti:
  - Ne-lineární chromosomy: stromy, grafy
  - Mutace možná ale ne nutná

Problém automatického programování

"How can computers learn to solve problems without being explicitly programmed? In other words, how can computers be made to do what is needed to be done, without being told exactly how to do it?"

---Attributed to Arthur Samuel - about 1959

Problém automatického programování

"Genetic programming *is* automatic programming. For the first time since the idea of automatic programming was first discussed in the late 40's and early 50's, we have a set of non-trivial, non-tailored, computer-generated programs that satisfy Samuel's exhortation: 'Tell the computer what to do, not how to do it.' "

– John Holland, University of Michigan, 1997

Ukázkový příklad

- Banka chce rozlišit dobré od špatných úvěrových zákazníků
- Je potřeba model, který dobře popíše existující záznamy (data)

ID	Počet dětí	Příjem	Stav	OK?
ID-1	2	45000	Ženatý	ANO
ID-2	0	30000	Svobodný	NE
ID-3	1	40000	Rozvedený	NE
…				

Ukázkový příklad

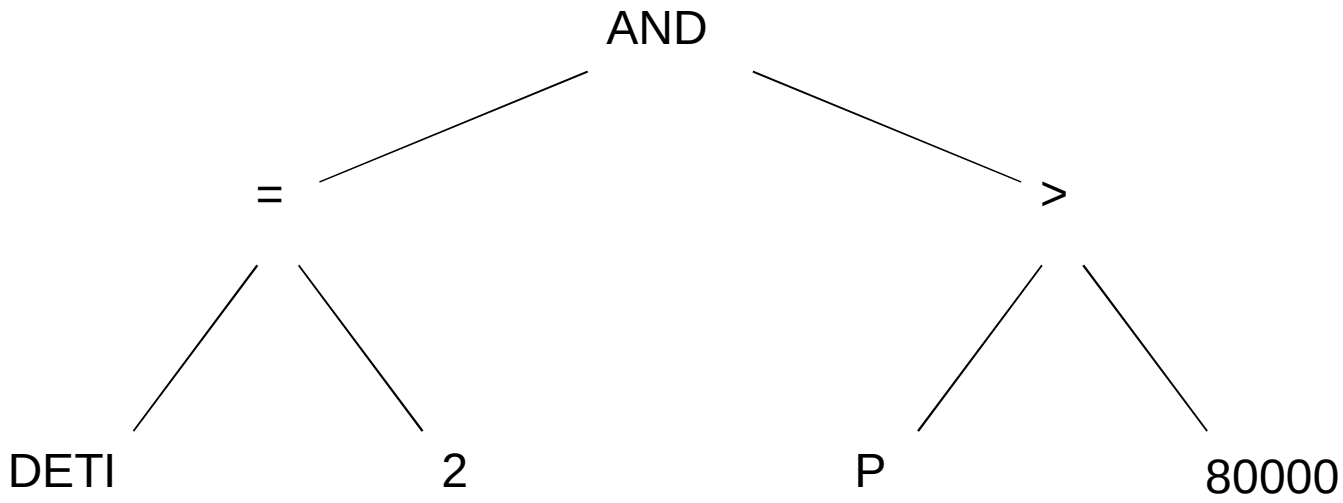
- Možný model:

  IF (DETI = 2) AND (P > 80000) THEN ANO ELSE NE
- Obecně:

  IF logický výraz THEN ANO ELSE NE
- Jediná neznámá je logický výraz, proto je náš prohledávací prostor (genotyp) prostor všech logických výrazů
- Cílová funkce je počet dat, která jsou dobře popsána nalezeným logickým výrazem
- Přirozená reprezentace logických výrazů: stromová struktura

Ukázkový příklad

- IF (DETI = 2) AND (P > 80000) THEN ANO ELSE NE
- Může být reprezentován následujícím stromem:



Stromová reprezentace

- Stromová struktura je univerzální, např.
- Aritmetický výraz

$$2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$$

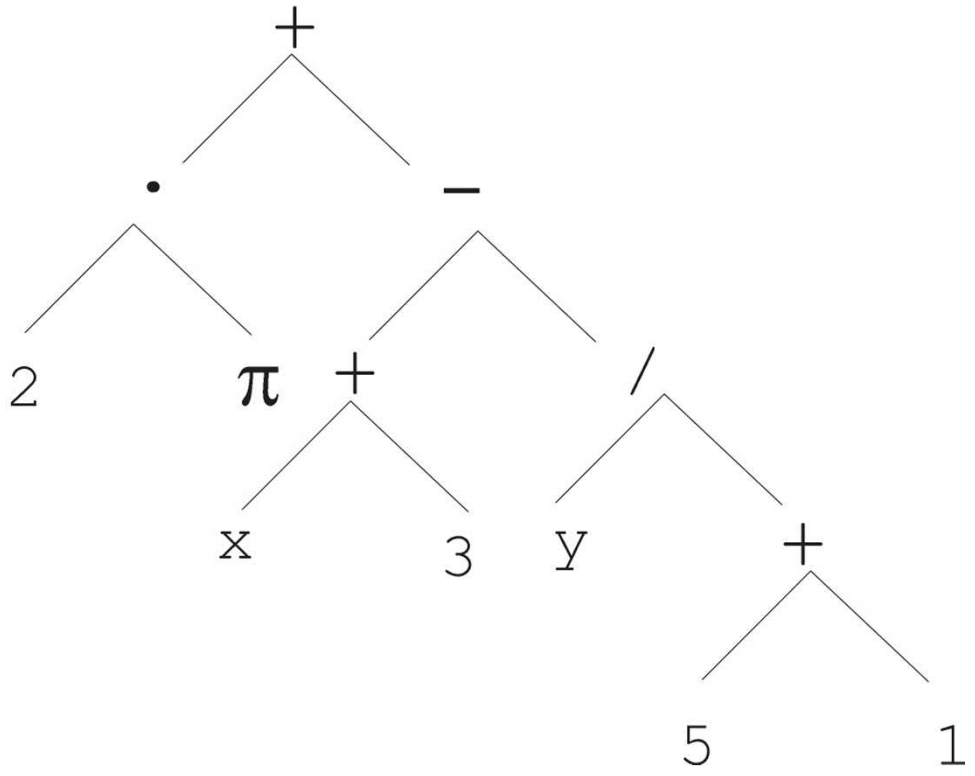
- Logický výraz

$$(\mathrm{x} \wedge \mathrm{true}) \rightarrow ((\ \mathrm{x} \vee \mathrm{y}\ ) \vee (\mathrm{z} \leftrightarrow (\mathrm{x} \wedge \mathrm{y})))$$

- Program

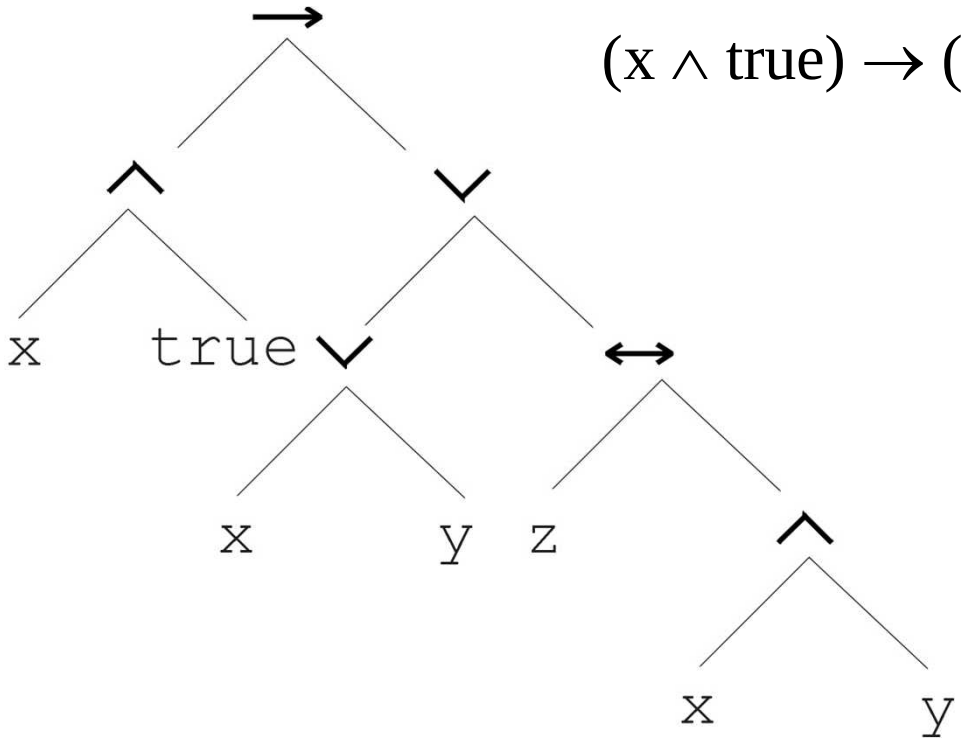
```
i =1;
while (i < 20)
{
        i = i +1
}
```

Stromová reprezentace



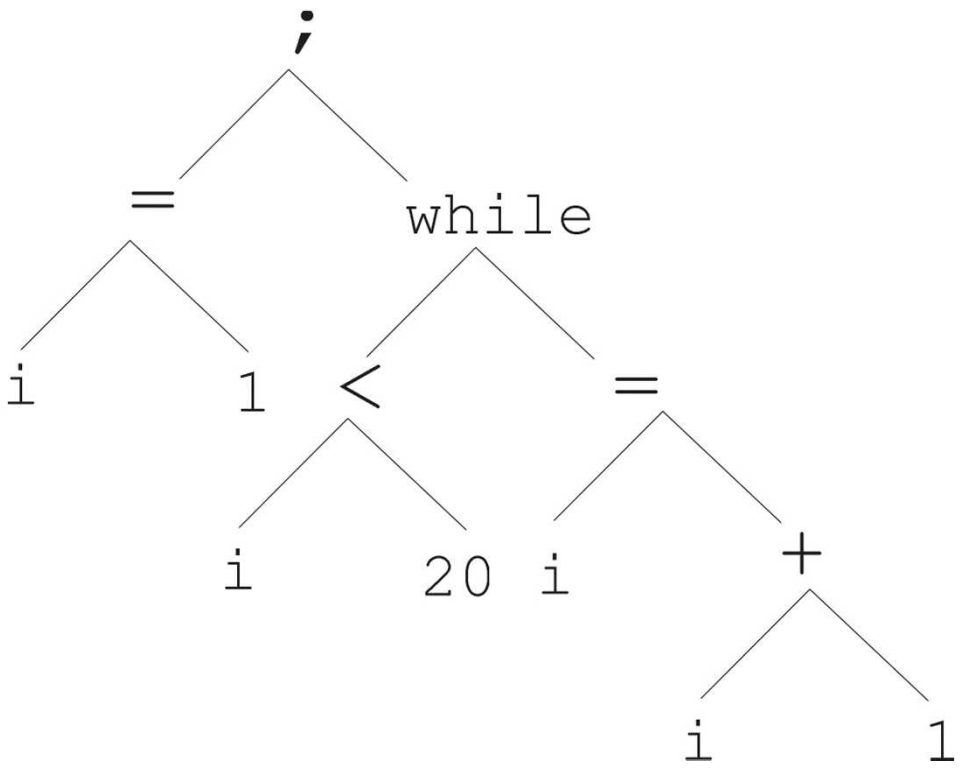
$$2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$$

Stromová reprezentace



$$(x \wedge \text{true}) \rightarrow ((\ x \vee y\ ) \vee (z \leftrightarrow (x \wedge y)))$$

Stromová reprezentace




```
i =1;
while (i < 20)
{
        i = i +1
}
```

Stromová reprezentace

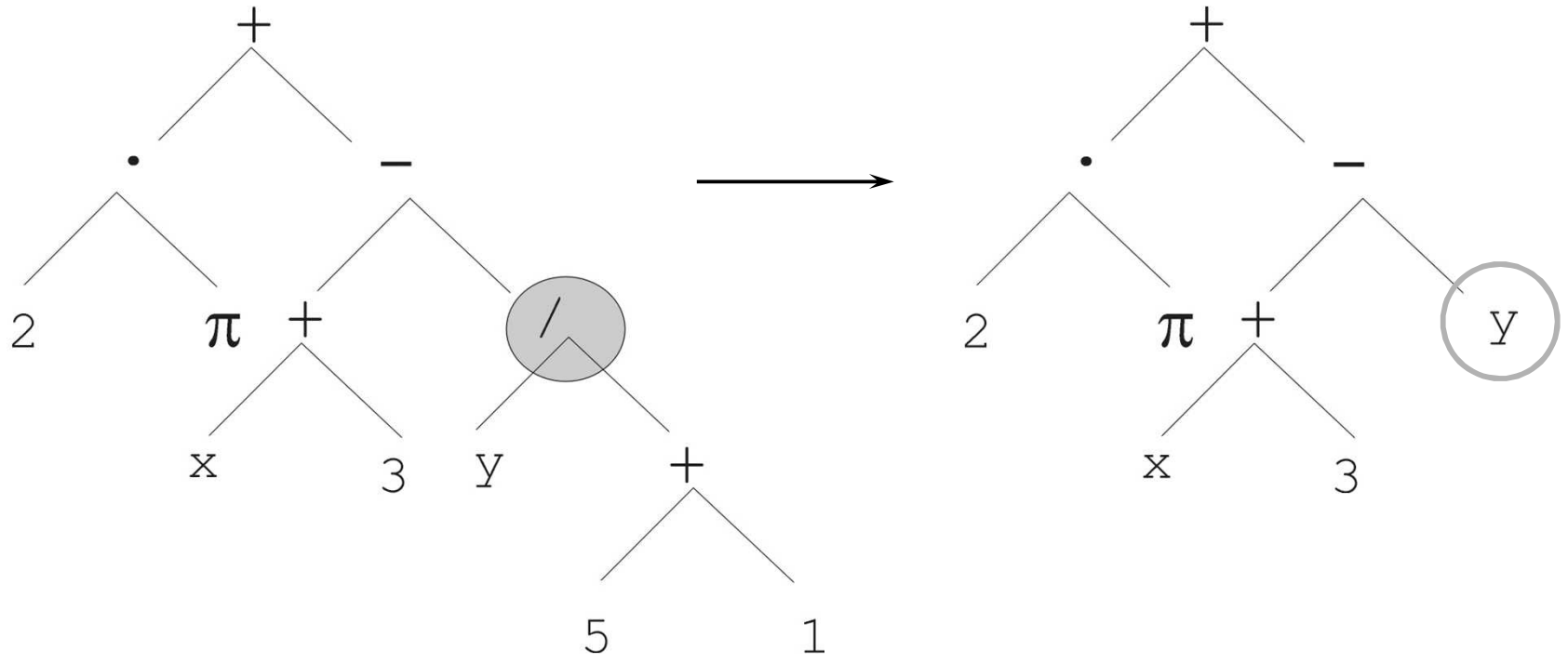
- Má proměnnou velikost
- Symbolický výraz může být vyjádřen:
  - Množinou terminálních symbolů T
  - Množinou funkcí F (s aritami funkčních symbolů)
- Pokud zavedeme následující rekurzivní definici:
  1. Každý $t \in T$ je korektní vyjádření
  2. $f(e_1, \ldots, e_n)$ je korektní vyjádření pokud $f \in F$, arita(f)=n a $e_1, \ldots, e_n$ jsou korektní vyjádření
  3. Další možné korektní vyjádření neexistují
- Výrazy v GP nejsou obvykle striktně předepsány (uzavřenost: jakákoliv $f \in F$ může použít jakoukoliv $g \in F$ jako svůj argument)

Algoritmus

- K porovnání:
  - GA používá křížení A mutaci následovně (náhodně)
  - GP používá křížení NEBO mutaci (náhodně vybrané)

Mutace

- Nejčastěji: vyměň náhodně vybraný podstrom jiným, náhodně vytvořeným



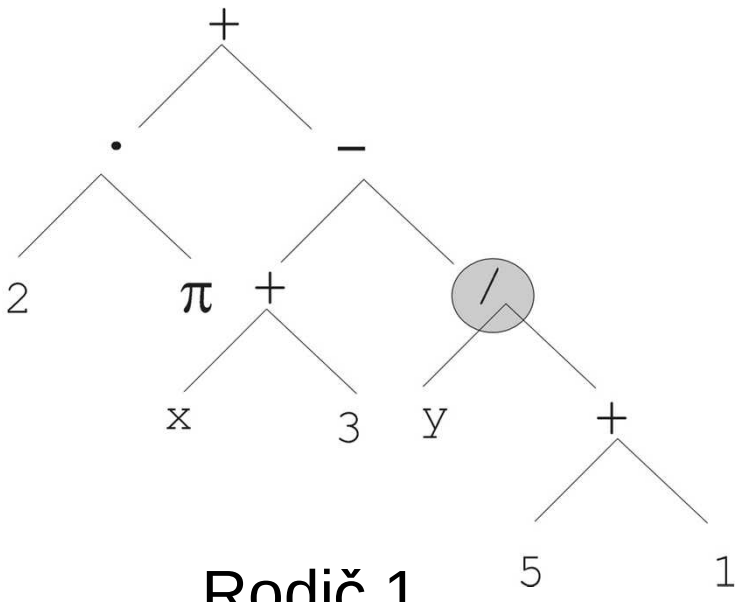
Mutace

- Mutace má dva parametry:
  - Pravděpodobnost p_m že bude vybraná mutace na úkor křížení
  - Pravděpodobnost vybrání uzlu jako kořene pro začátek mutace
  - Pro zajímavost, je doporučeno p_m=0

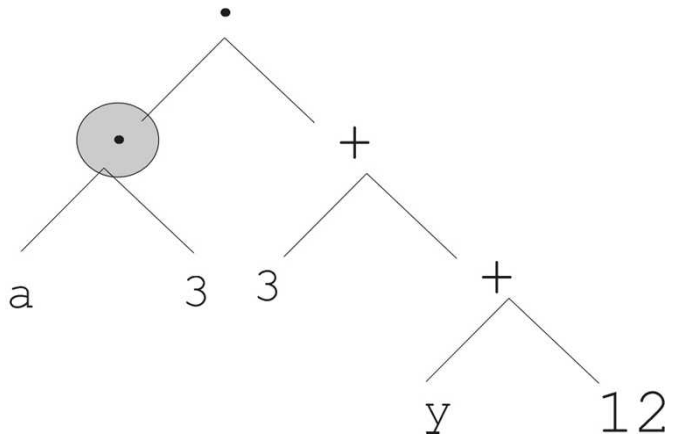
    [Koza, 1992] nebo velmi malé, např. 0,05 [Banzhaf et al., 1998]
- Velikost potomků může přesáhnout velikost rodičů!

Křížení

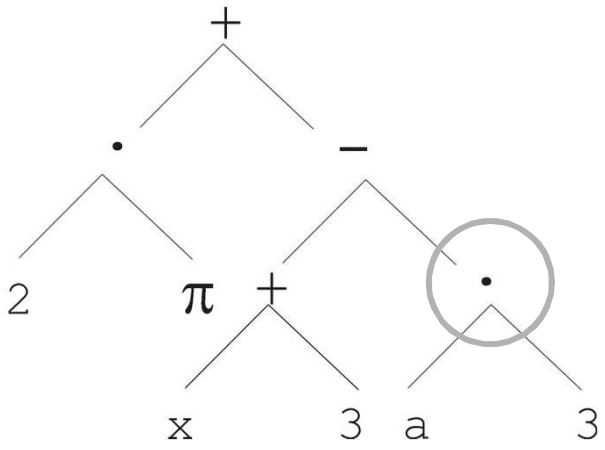
- Nejčastěji: výměny dvou náhodně vybraných podstromů mezi rodiči
- Před-selekce (over-selection) ve velkých populacích:
  - Podle fitness je populace rozdělena na dvě části:
  - skupina 1: nejlepších x%, skupina 2 zbytek (100-x)%
  - 80% výběru bude ze skupiny 1, 20% ze skupiny 2
  - Pro populaci = 1000, 2000, 4000, 8000: x = 32%, 16%, 8%, 4%



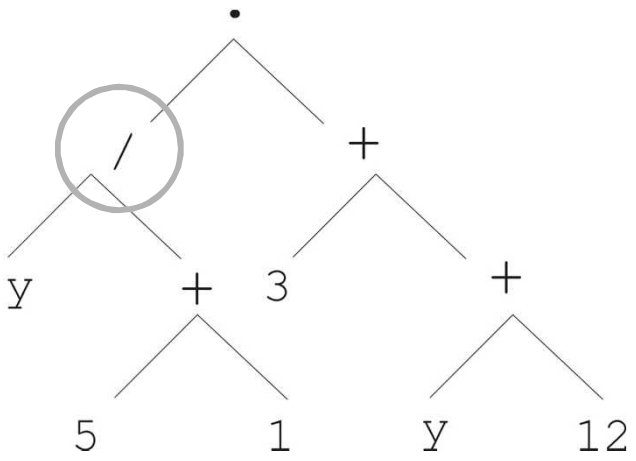
Rodič 1



Rodič 2



Potomek 1



Potomek 2

Tvorba nových řešení

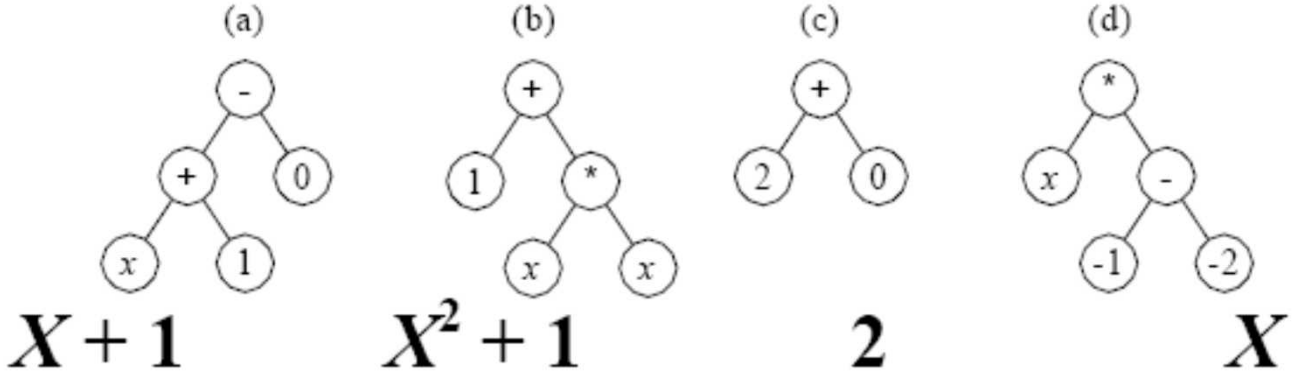
- Nejprve je nastavena maximální hloubka stromu D_{max}
- Plnící metoda (každá větev má délku = D_{max}):
  - Uzly v hloubce $d < D_{max}$ jsou náhodně vybrány z množiny funkcí F
  - Uzly v hloubce $d = D_{max}$ jsou vybrány z množiny terminálů T
- Růstová metoda (každá větev má délku $\leq D_{max}$):
  - Uzly v hloubce $d < D_{max}$ náhodně vybrány z $F \cup T$
  - Uzly v hloubce $d = D_{max}$ náhodně vybrány z T
- Běžně je počáteční populace tvořena půl-na-půl oběmi metodami

Bloat

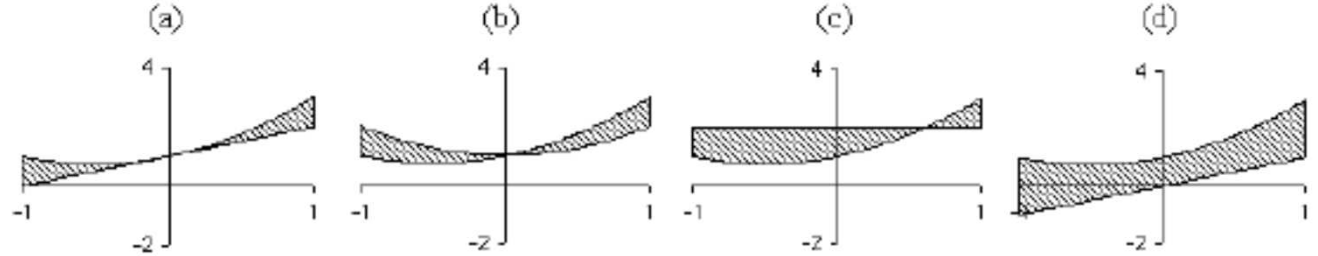
- Bloat = “survival of the fattest”, nebo-li, průměrná délka řešení narůstá přibližně lineárně s počtem iterací
- Možná řešení:
  - Upravit operátory aby netvořily dlouhá řešení
  - Penalizovat dlouhá řešení
  - Vícekriteriální optimalizace

Příklad: Symbolická regrese x^2+x+1

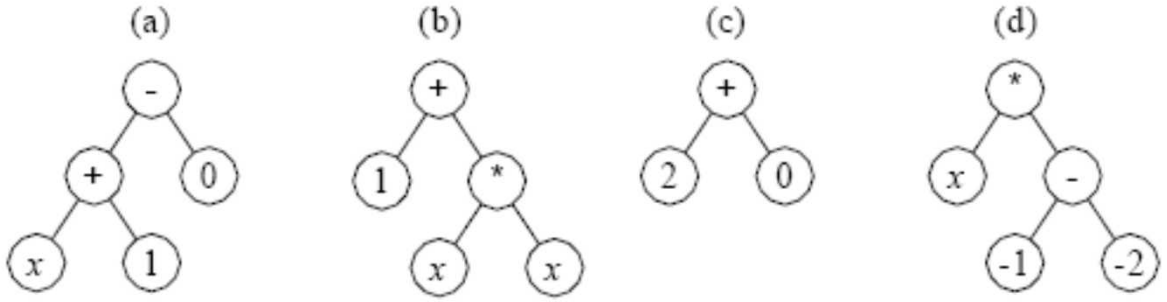
Generation 0



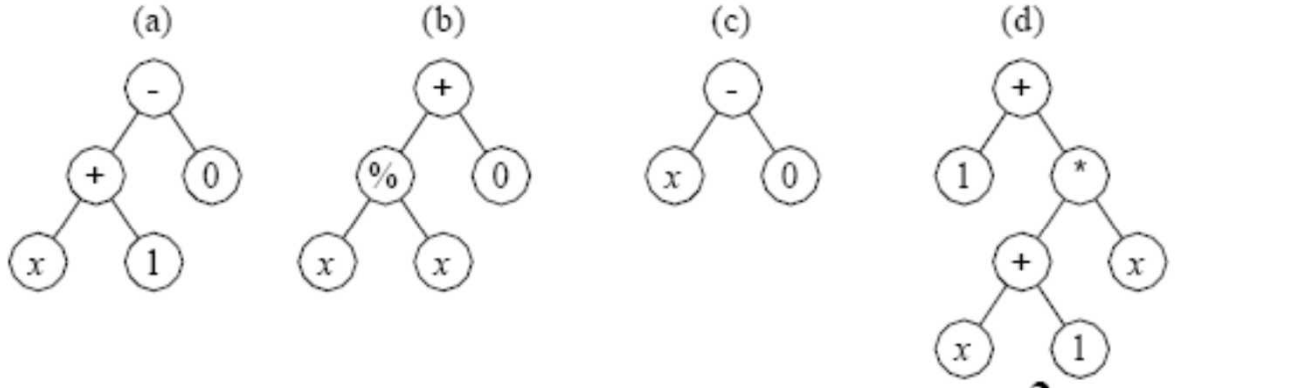
FITNESS



Příklad: Symbolická regrese x^2+x+1



Generation 1



$x+1$ 1 X x^2+x+1

Kopie z (a) Mutant z (c) Potomci z (a) a (b)

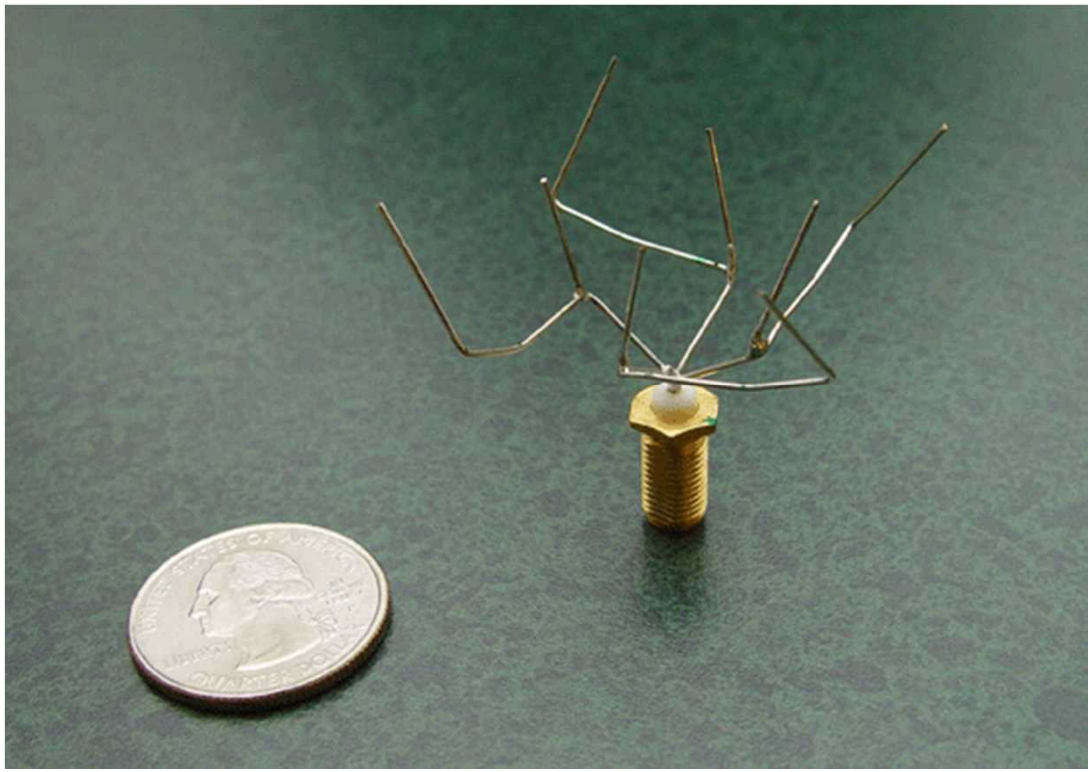
Symbolická regrese: problém reálných čísel

- Začlenění náhodných reálných čísel jakožto terminálních symbolů

- U lineárních funkcí lze využít lineární regrese

- Multimodální optimalizace

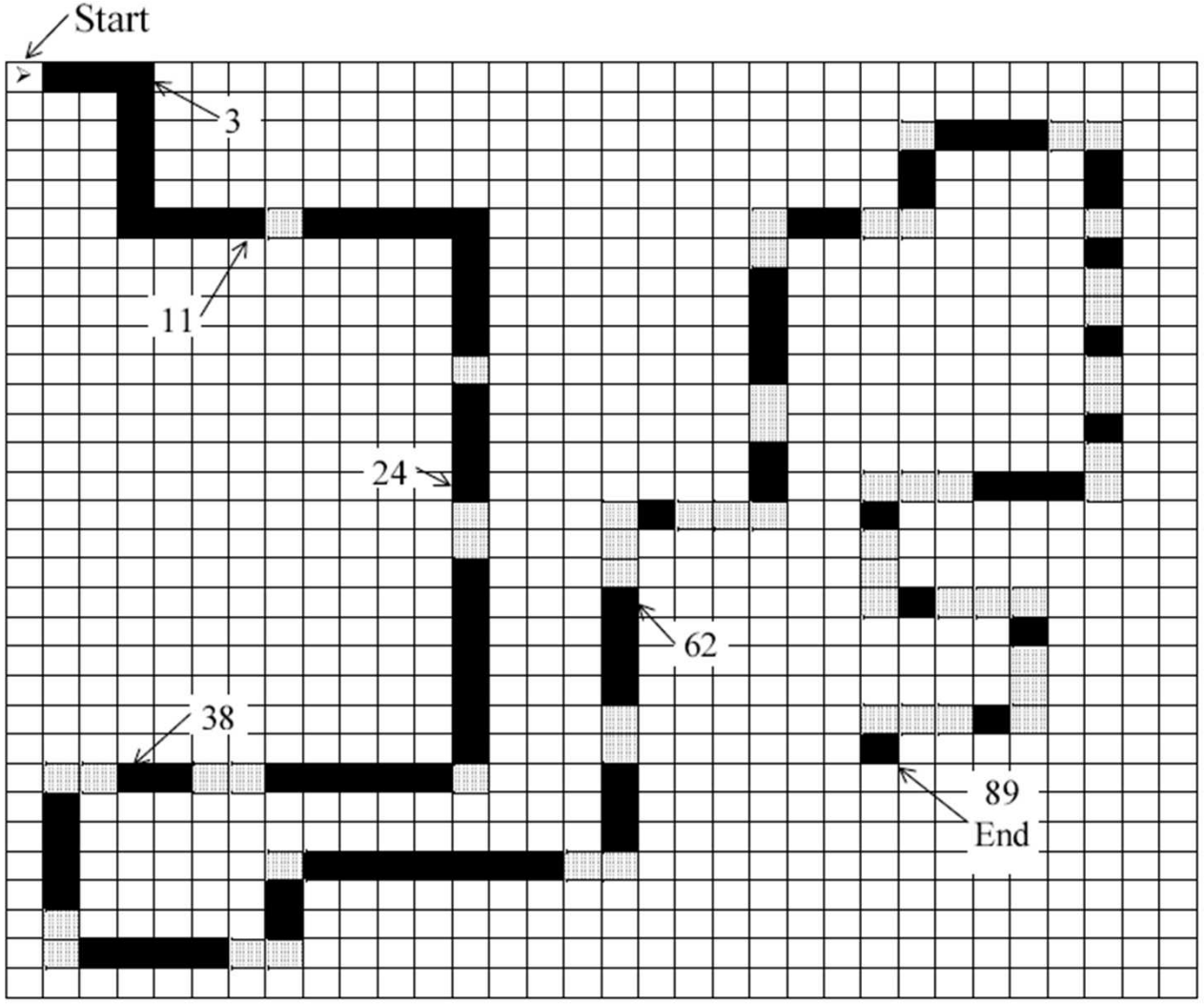
Příklad reálné aplikace: Tzv. „Human competitive results“



Jason D. Lohn

Evolvable Systems Group
Computational Sciences Division
NASA Ames Research Center
Mountain View, CA USA

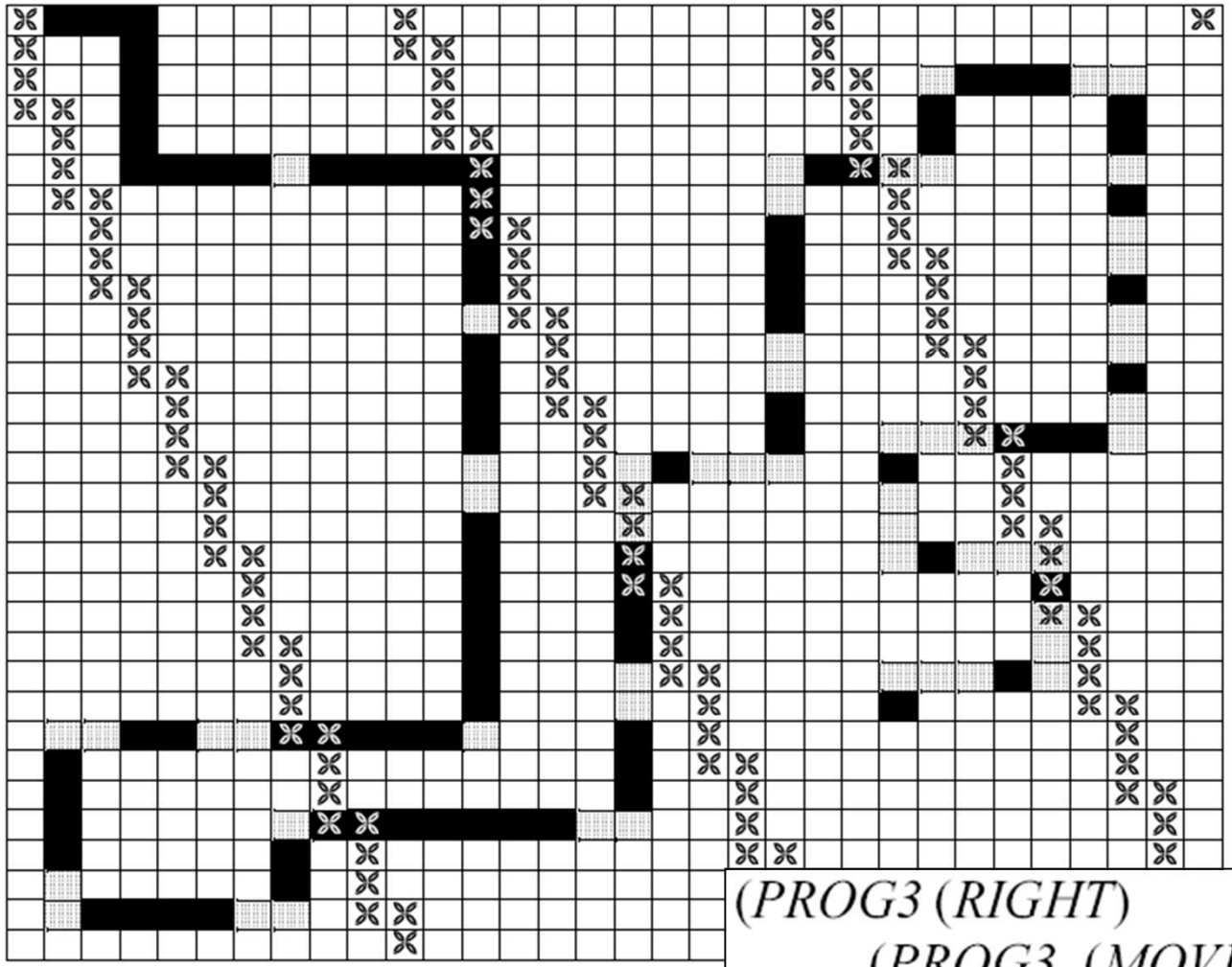
Příklad – stezka Santa Fe



Příklad – stezka Santa Fe

- Množina terminálů:
  - T = {MOVE,LEFT, RIGHT}
- Množina funkcí:
  - F = {IF-FOOD-AHEAD,PROG2,PROG3} s aritami {2,2,3}
- Fitness:
  - Počet snědené potravy za 400 kroků

Typické řešení: „prošívač“ (F=12)



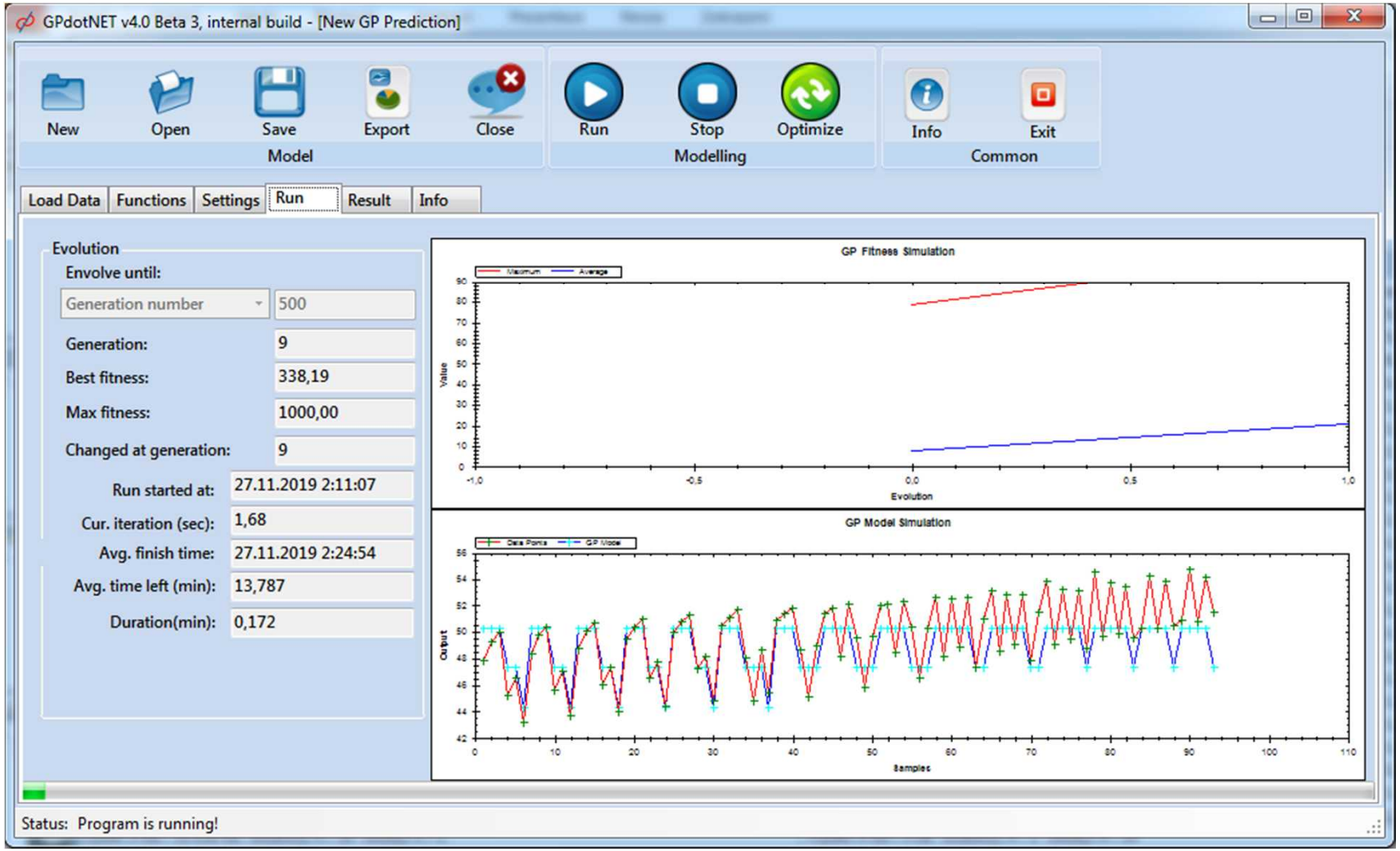
```
(PROG3 (RIGHT)
       (PROG3  (MOVE) (MOVE) (MOVE))
       (PROG2  (LEFT) (MOVE)))
```

GPLAB

- A Genetic Programming Toolbox for MATLAB
- Autorka Sara Silva
- http://gplab.sourceforge.net/

- Test: demoant.m
- [v,b] = gplab(počet_generací,počet_jedinců,parametry);

Symbolická regrese



GPdotNET https://gpdotnet.codeplex.com/

Reference

[1] Koza, J. R. (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press.

[2] Koza, J. R. (1994). Genetic Programming II: Automatic Discovery of Reusable Programs.

[3] Koza, J. R. (1999). Genetic Programming III: Darwinian Invention and Problem Solving.

[4] Koza, J. R. (2003). Genetic Programming IV: Routine Human-Competitive Machine Intelligence.

[5] www.genetic-programming.com

Reference

[6] Vladislavleva, E. (2008). Model-based Problem Solving through Symbolic Regression via Pareto Genetic Programming. PhD thesis, Tilburg University, Tilburg, The Netherlands.

[7] GPdotNET https://gpdotnet.codeplex.com/

[8] GPLab: A Genetic Programming Toolbox for MATLAB http://gplab.sourceforge.net/

Prosba. V případě, že v textu objevíte nějakou chybu nebo budete mít námět na jeho vylepšení, ozvěte se prosím na **matej.leps@fsv.cvut.cz**.

Oprava 25.11.2009: Přidán slide o reálných číslech při symbolické regresi
Oprava 16.12.2018: Přidány reference

Datum poslední revize: 16.12.2018
Verze: 002