Bio-inšpirované optimalizačné algoritmy

Vybrané typy bio-inšpirovaných optimalizačných algoritmov

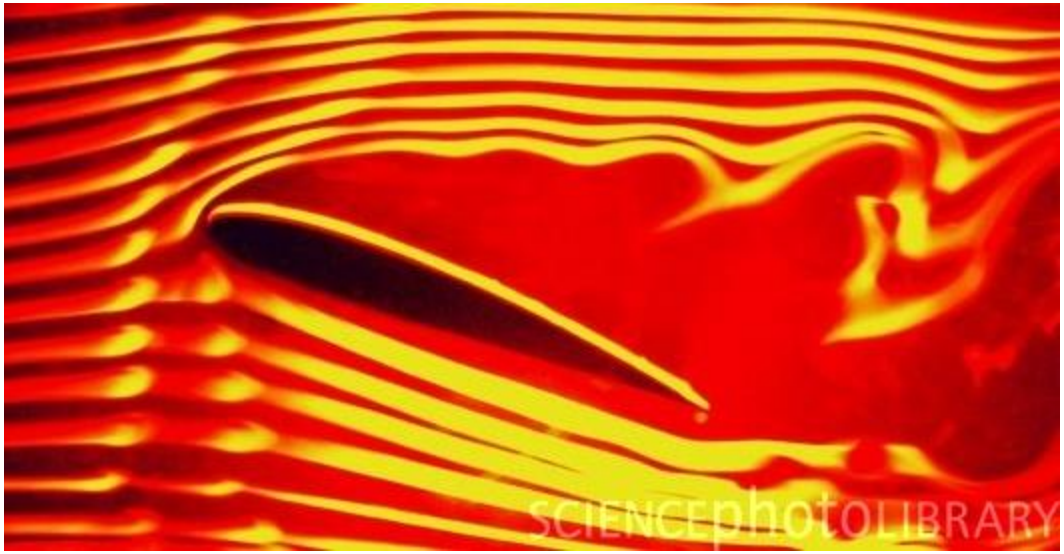
<u>Evolučné stratégie</u> (numerická optimalizácia = optim. parametrov)
<u>Genetický algoritmus</u> (optim. parametrov)
<u>Genetické programovanie</u> (optimalizácia štruktúry aj parametrov)
<u>Diferenciálna evolúcia</u> (optim. parametrov)
<u>Kŕdľový algoritmus (PSO)</u> (optim. parametrov)
Umelý imunitný systém (optim. parametrov)
Gramatická evolúcia (optimalizácia štruktúry aj parametrov)
Včelie algoritmy (optim. parametrov)
Kukučí algoritmus (optim. parametrov)
<u>Mravčie algoritmy (ACO)</u> (optim. dráhy, cesy)

mnohé iné...

1.4 Evolučné stratégie

I. Rechenberg a H.P.Schwefel, 60. roky
Technická univerzita v Berlíne, NDR.
Prvý evolučný prístup.

Ich pôvodnou aplikačnou oblasťou bola
optimalizácia aerodynamických tvarov


SCIENCEphotoLIBRARY

Algoritmus ES

Jediný aktuálny jedinec je reprezentovaný dvojicou vektorov

$$X=\{x, \sigma\}=\{x_1,x_2,\ldots,x_n, \sigma_1,\sigma_2,\ldots,\sigma_n\}$$

x - vektor hľadaných parametrov
(súradnice bodu v prehľadávanom priestore)

σ- vektor smerodajných odchýlok možných zmien prvkov vektora *x*.

Zmeny boli uskutočňované vďaka jedinému operátoru – mutácii

$$x^{t+1}=x^t+Norm(0, \sigma)$$

Norm - vektor nezávislých náhodných čísel, normálne rozdelenie pravdepodobnosti s nulovou strednou hodnotou a smerodajnou odchýlkou
σ - Funkcia hustoty normálneho rozdelenia pravdepodobnosti náhodnej premennej, ktorá je v tvare

$$h(x)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{-(x^2/2\sigma^2)}$$

σ- ovplyvňuje šírku Gaussovej krivky, zjednodušene sa nazýva veľkosť „krok mutácie“.

Zmeny jedincov sú v súlade s pozorovaním, že v prírode prebiehajú malé zmeny častejšie než veľké.

Do nasledujúceho kroku (generácie) sa dostane ten z dvojice rodič + potomok, ktorý dosahuje lepšiu hodnotu účelovej funkcie, prípadne ten, ktorý spĺňa všetky obmedzenia. Táto verzia algoritmu sa nazývala „(1+1)“ ..... 1 rodič + 1 potomok.

Zo začiatku bolo používané: σ - vektor konštánt

Na základe štatistických pozorovaní ale bolo zistené, že počet mutácií, ktoré vedú k zlepšeniam účelovej funkcie, predstavuje zo všetkých mutácií asi 1/5. To viedlo k nasledovnej úprave algoritmu. Po každých k generáciách, počas ktorých sa vyhodnocuje miera úspešnosti mutácií φ, sa uskutoční korekcia parametra σ

$\sigma^{t+1} = \sigma^{t}.c$, ak $\varphi < 1/5$

$\sigma^{t+1} = \sigma^{t} / c$, ak $\varphi > 1/5$

$\sigma^{t+1} = \sigma^{t}$, ak $\varphi = 1/5$

kde $0.817 < c < 1$ - empiricky získaná konštanta

Táto úprava zapríčinila, že ak mutácie sú úspešné, krok zmien sa zväčšuje, ak nie, krok zmien sa zmenšuje.

Napriek tejto úprave algoritmus často viedol k <u>uviaznutiu riešenia v lokálnom optime</u>.

-Preto bola <u>veľkosť populácie zväčšená z 1 na n jedincov</u>, čo prinieslo viac paralelných smerov hľadania.

-Okrem toho bol zavedený aj <u>operátor kríženia</u>, ktorý sa aplikoval ako nad zložkami vektora x, tak súčasne aj nad vektorom σ. Kríženie pre dvoch rodičov X_a a X_b

$$X_a=\{x_a,\ \sigma_a\}=\{x_{a1},x_{a2},\dots,x_{an},\ \sigma_{a1},\sigma_{a2},\dots,\sigma_{an}\}$$
$$X_b=\{x_b,\ \sigma_b\}=\{x_{b1},x_{b2},\dots,x_{bn},\ \sigma_{b1},\sigma_{b2},\dots,\sigma_{bn}\}$$

vyprodukuje potomka

$$X_p=\{x_p,\ \sigma_p\}=\{x_{p1},x_{p2},\dots,x_{pn},\ \sigma_{p1},\sigma_{p2},\dots,\sigma_{pn}\},$$

ktorého prvky sú kombináciou rodičov R_a a R_b.

Spôsob kríženia oproti pôvodnému typu použitému v GA bol zmodifikovaný na diskrétne kríženie:

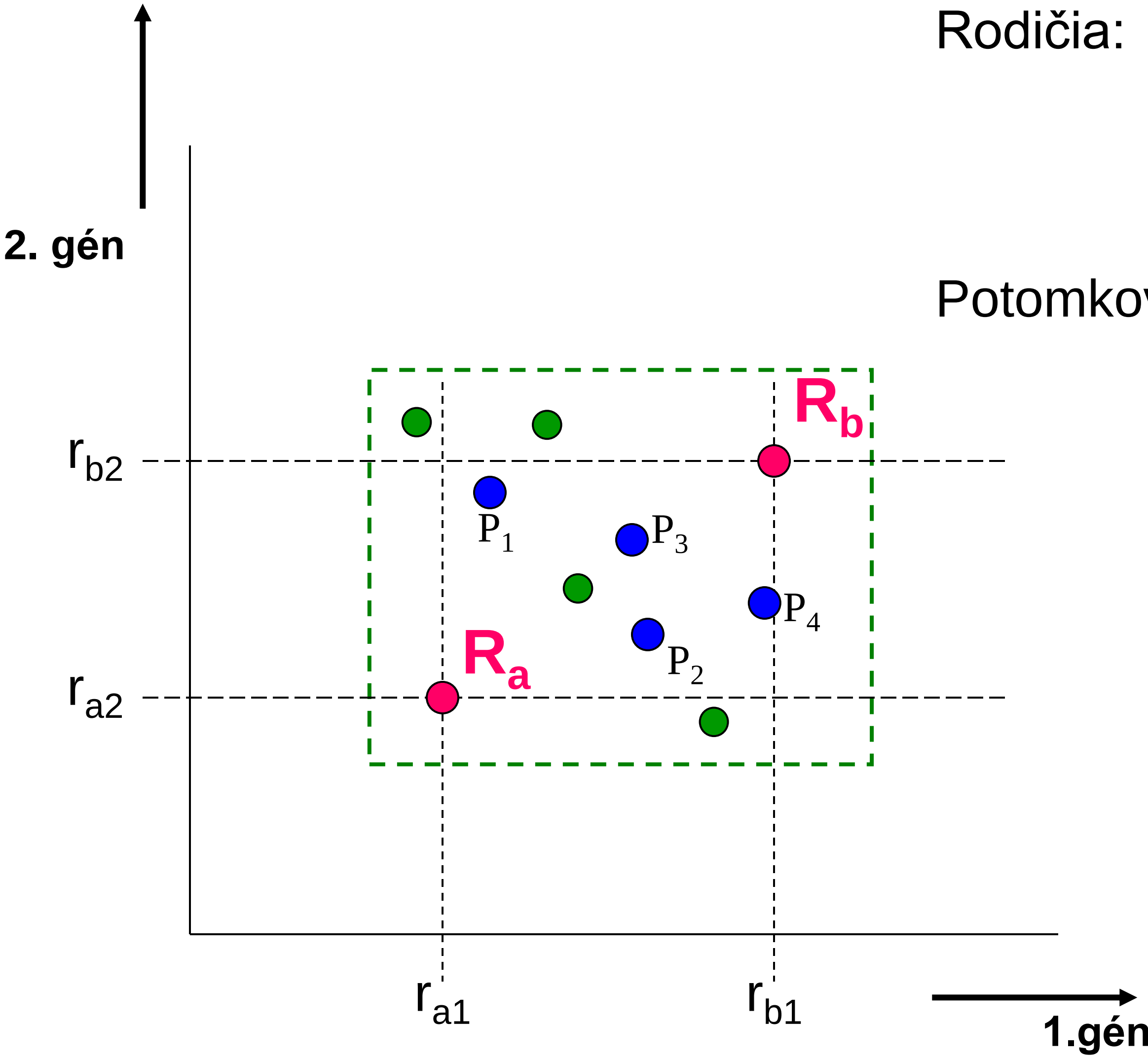
$R_a=[a_1\ a_2\ a_3\ a_4\ a_5]$ $M_1=[1\ 1\ 3\ 2\ 2] \rightarrow P_1=[a_1\ a_2\ c_3\ b_4\ b_5]$

$R_b=[b_1\ b_2\ b_3\ b_4\ b_5]$ $M_2=[2\ 1\ 1\ 1\ 2] \rightarrow P_2=[b_1\ a_2\ a_3\ a_4\ b_5]$

$R_c=[c_1\ c_2\ c_3\ c_4\ c_5]$ $M_3=[3\ 3\ 2\ 2\ 1] \rightarrow P_3=[c_1\ c_2\ b_3\ b_4\ a_5]$

alebo medziľahlé kríženie:

Medziľahlé kríženie



Rodičia: $R_a=[r_{a1}, r_{a2}]$

$R_b=[r_{b1}, r_{b2}]$

Potomkovia: P_1, P_2, P_3, P_4

$$P=R_1+a(R_2-R_1)$$

$0<a<1$

$-0.25<a<1.25$

Geometrická interpretácia medziľahlého kríženia

Neskôr sa tieto typy kríženia začali používať aj v GA...

Pri verziách ES s viacerými jedincami v populácii sa prestala používať deterministická adaptácia vektora σ s pravidlom založeným na 1/5 úspešnosti.

Namiesto toho sa aj táto časť reťazca začala podrobovať mutácii

$$x^{t+1}=x^{t}+Norm(0, \sigma)$$

Pri viacprvkových populáciách sa začali používať dve stratégie výberu:

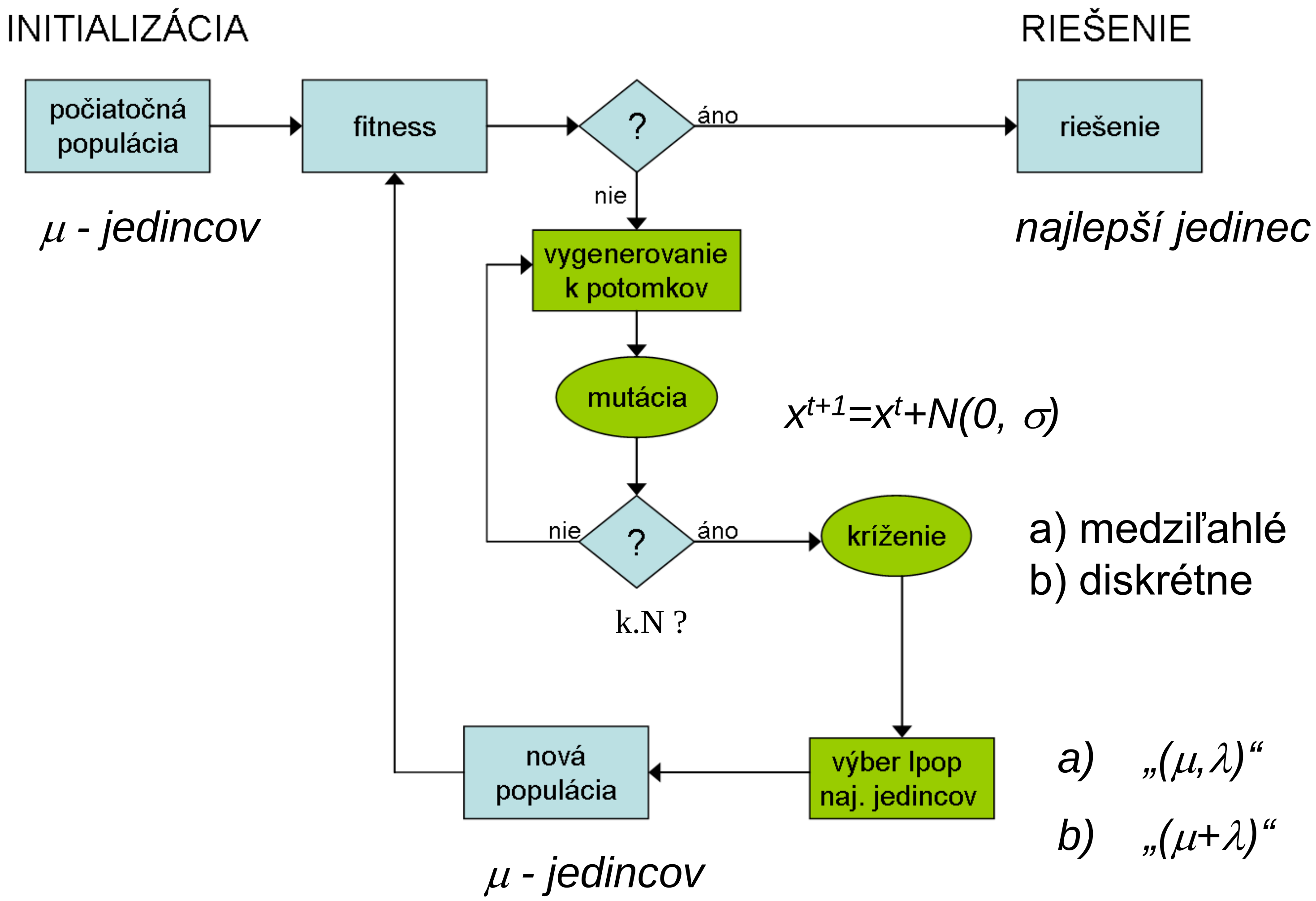
Najprv sa z náhodne vybraných dvojíc rodičov krížením a mutáciou vyprodukuje λ (λ>μ) potomkov.

Pomer μ:λ sa bežne doporučuje voliť okolo 1:7 (napr. 15 rodičov ku 100 potomkom). Ak chceme urýchliť lokálnu konvergenciu môžeme tento pomer zmeniť až na hodnotu 1:20 (5 rodičov, 100 potomkov).

a) Pri tzv. type „(μ+λ)“ sa rodičia a potomkovia spoja do jednej skupiny, z ktorej sa deterministicky opäť vyberie μ najúspešnejších jedincov.

b) Pri type „(μ,λ)“ sa vyberie tiež μ najúspešnejších jedincov, ale iba zo skupiny λ potomkov. Vďaka tomuto spôsobu výberu sa život každého jedinca obmedzí iba na 1 generáciu. Naviac sa nezaručí prechod najlepšieho jedinca predchádzajúcej generácie do populácie v novej generácii. Na druhej strane ale tento typ vykazuje isté prednosti pri úlohách, kde sa poloha globálneho optima v čase mení, prípadne je zašumená.

Evolučné Stratégie



Evolučné stratégie sú určené predovšetkým na optimalizáciu numerických problémov.

<u>Poznámka:</u>

Skutočnosť, že súčasťou optimalizovaného reťazca sú aj strategické parametre σ, ktoré ovplyvňujú proces evolúcie vlastne spôsobuje, že u ES sa prejavuje jav, ktorý sa zvykne nazývať <u>samoadaptácia</u>.

Toto bolo pôvodne špecifikom ES, ale postupne sa podobné mechanizmy adaptácie riadiacich parametrov algoritmu začali v rôznych formách používať aj v iných evolučných prístupoch.

1.5 Diferenciálna evolúcia

Diferenciálna evolúcia - jednoduchý, ale výkonný optimalizačný algoritmus.

Napriek tomu, že používa len málo riadiacich parametrov, je schopný úspešne riešiť numerické optimalizačné úlohy.

Pracuje podobne ako iné evolučné prístupy s populáciou potenciálnych riešení (jedincov):

$$x_i=\{x_{i,1}, x_{i,2},...,x_{i,n}\}, \quad i = 1,2,...,N \qquad N \text{ - veľkosť populácie}$$

Základom algoritmu je špecifický spôsob tvorby tzv. „skušobných jedincov“. Sú to novovytvorení potomkovia, ktorí súperia o prežitie so svojimi rodičmi.

Algoritmus DE:

1. Pre jedinca x_i aktuálnej populácie o veľkosti N reťazcov, kde i je poradové číslo jedinca v populácii sa vygeneruje vektor v

$$v = x_{r1} + F(x_{r2} - x_{r3})$$

kde r_1, r_2, r_3 sú rôzne náhodné poradové čísla jedincov v populácii (od 1 do N), súčasne sú rôzne aj od i a 0<F<2 je konštanta. Slovne vyjadrené: vyberieme jeden náhodný reťazec populácie iný ako x_i a pripočítame k nemu rozdiel vektorov dvoch ďalších, náhodne vybraných reťazcov vynásobený číslom F.

2. Vytvoríme tzv. skúšobný reťazec u. Pre každý index j=1,...,n

$u_j = v_j$ ***ak*** $\rho < CR$ ***alebo***

$u_j = x_{i,j}$ ***ak*** $\rho \geq CR$

kde j je poradové číslo prvkov vektora (génov) u a v, ρ je náhodné číslo z intervalu (0;1) a CR je parameter pravdepodobnosti kríženia z rovnakého intervalu, n je počet génov reťazca. Jedná sa vlastne o vytvorenie reťazca u pomocou diskrétneho skríženia reťazcov v a x_i.

3. V poslednom kroku sa vygeneruje i-ty jedinec novej populácie (uvažujme prípad minimalizácie)

$x_i^{t+1}=u$ ***ak*** $f(u) < f(x_i^t)$ ***alebo***

$x_i^{t+1}=x_i^t$ ***ak*** $f(u) \geq f(x_i^t)$

kde t je poradové číslo aktuálnej generácie. To znamená, že do novej generácie postúpi ako i-ty jedinec úspešnejší z dvojice x_i^t ***– pôvodný jedinec a u – skúšobný reťazec .***

4. postupnosť krokov 1-3 sa opakuje pre všetky reťazce aktuálnej populácie (i=1,2,...,N)

Diferenciálna evolúcia

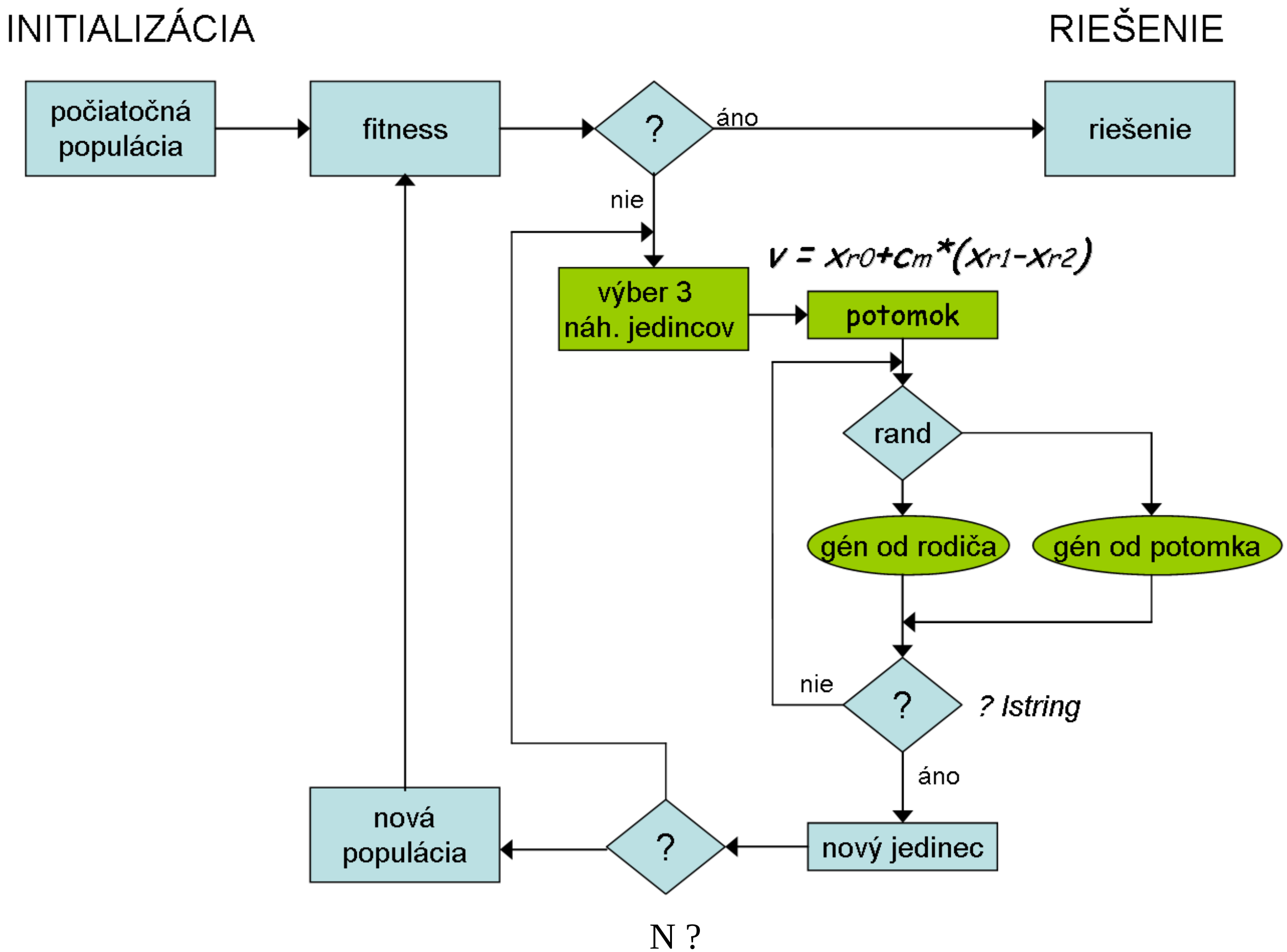
- ***Algoritmus DE je jednoduchý, vyžaduje iba dva parametre CR a F, ktoré sa určujú experimentálne.***

- ***Ich hodnoty ovplyvňujú správnu činnosť a rýchlosť konvergencie algoritmu.***

- ***DE je vhodná pri numerickej optimalizácii spojitých funkcií, pričom väčšinou pomerne rýchlo konverguje.***

- ***Niekedy, najmä v prípade veľmi členitých, multimodálnych účelových funkcií môže uviaznuť v niektorom lokálnom extréme.***

Diferenciálna Evolúcia



1.6 Rojenie častíc
(Particle swarm optimisation -PSO, kŕdľový algoritmus)

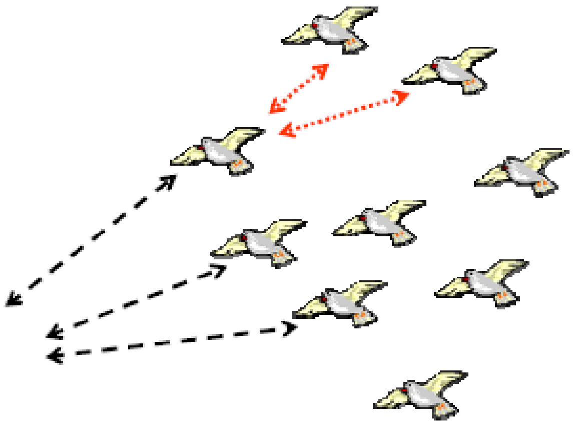
Algoritmus PSO

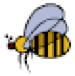
Algoritmus pozostáva z pohybu častíc (particles) kŕdľa (swarm), ktoré „lietajú“ nad povrchom účelovej funkcie.

Každá nesie v sebe informáciu o svojej polohe a vektore rýchlosti.

Jej pozícia predstavuje potenciálne riešenie optimalizačného problému.

Každá častica ovplyvňuje svoj pohyb zosúľaďovaním svojej pamäti (svojou najlepšou pozíciou) s polohou aktuálneho „lídra“ (najlepšia aktuálne dosiahnutá poloha celým kŕdľom).





Algoritmus PSO

- Inicializuj všetkých jedincov kŕdľa (populáciu) náhodnými vektormi $x_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,n}]$, $i=1,2,\ldots,N$; *N=počet častíc (jedincov)*
- Inicializuj všetky hodnoty $pbest_i$ na hodnoty x_i
- Inicializuj všetky hodnoty rýchlosti (prírastku polohy) v_i na náhodné hodnoty
- Do splnenie ukončovacích podmienok vykonávaj pre každú časticu $i=1\ldots N$:

1. Aktualizuj vektor hodnôt častice i

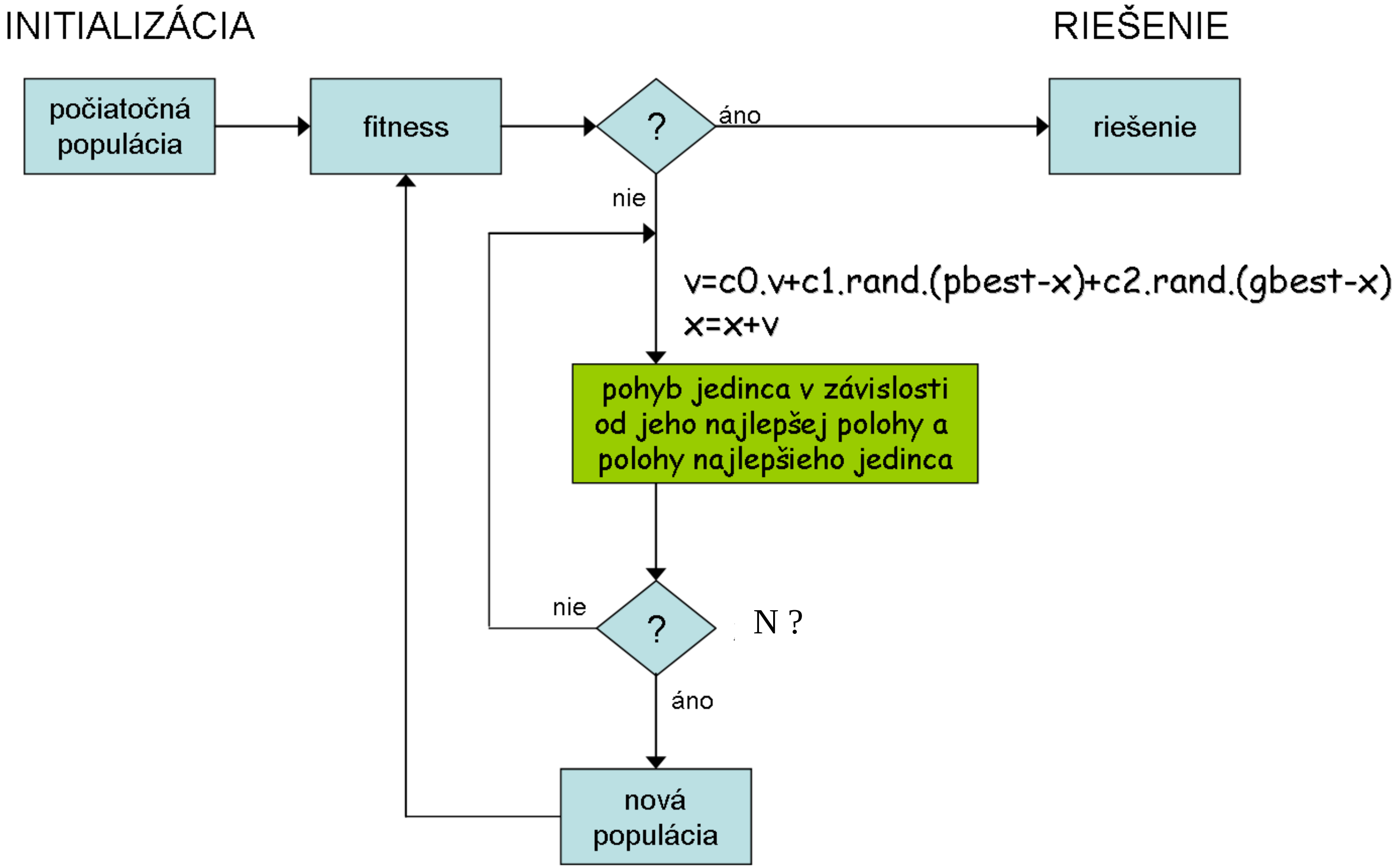
$$x_{i,new} = x_{i,old} + v_{i,new}$$

$$v_{i,new} = c_0 \cdot v_{i,old} + c_1 \cdot rand \cdot \left(pbest_i - x_i\right) + c_2 \cdot rand \cdot \left(gbest - x_i\right)$$

2. Ak x_i je lepšie než $pbest_i$, nahraď $pbest_i \leftarrow x_i$
3. Ak $pbest_i$ je lepšie než *gbest*, nahraď $gbest \leftarrow pbest_i$

$pbest_i$ – *najlepšia dosiahnutá pozícia i-tej častice v prehľadávanom priestore*
gbest – najlepšia aktuálna pozícia dosiahnutá celým kŕdľom
c_0, c_1, c_2 – *vektory zvolených konštánt, obyčajne z (0;2)*
(c_0 – zotrvačnosť častice, c_1 – individuálny a c_2 - sociálny faktor)
rand - vektory náhodných čísel z (0;1)

Particle Swarm Optimization

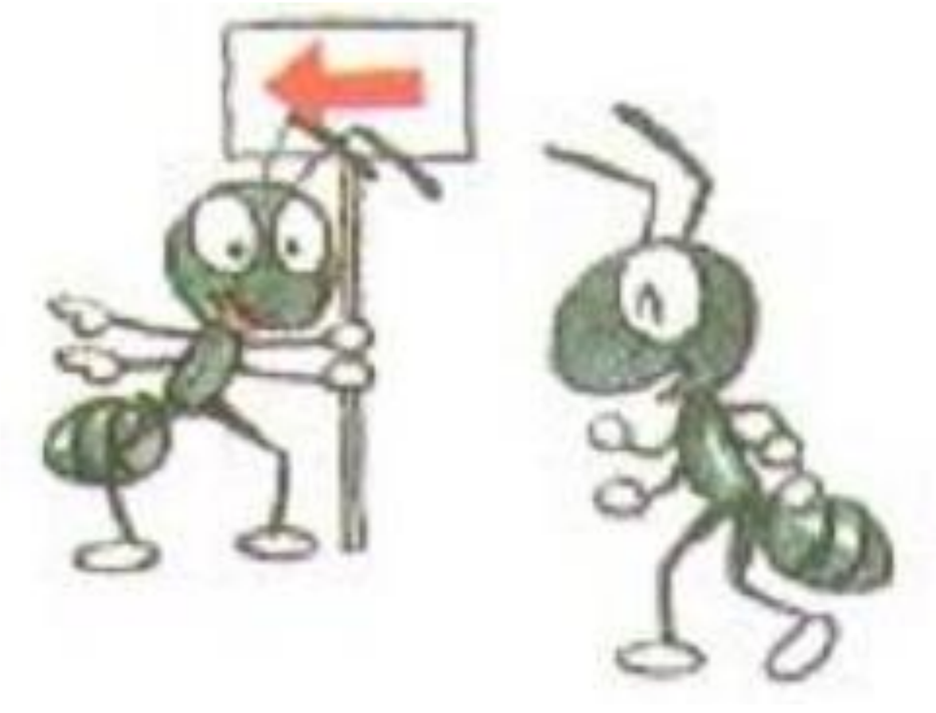


Rojenie častíc zaručuje rýchlu konvergenciu riešenia, je výpočtovo nenáročný,

avšak náchylný k uviaznutiu v lokálnom extréme.

Obmedzenie tohto javu sa dá dosiahnuť vnorením mutácie z GA do jadra PSO.

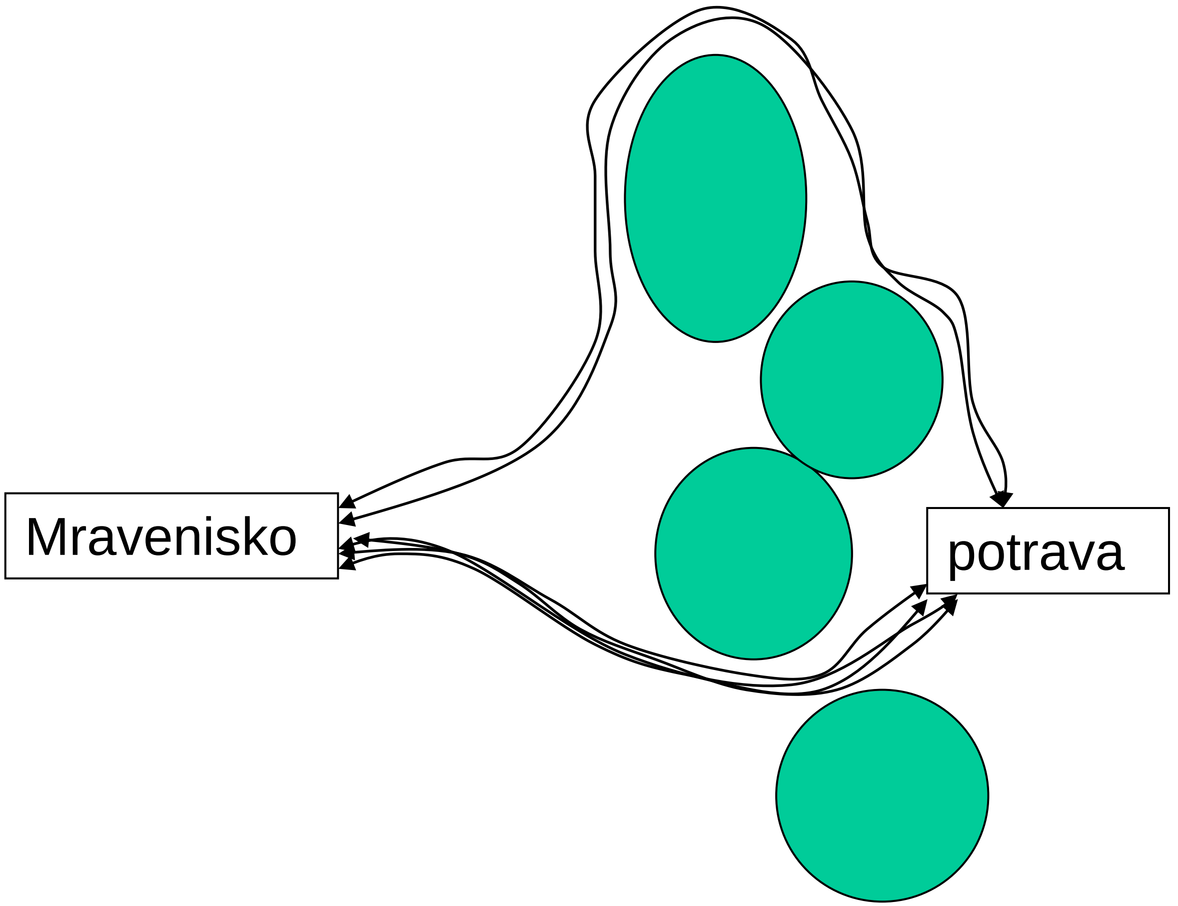
1.7 Optimalizácia pomocou kolónie mravcov (Ant colony optimisation – ACO)

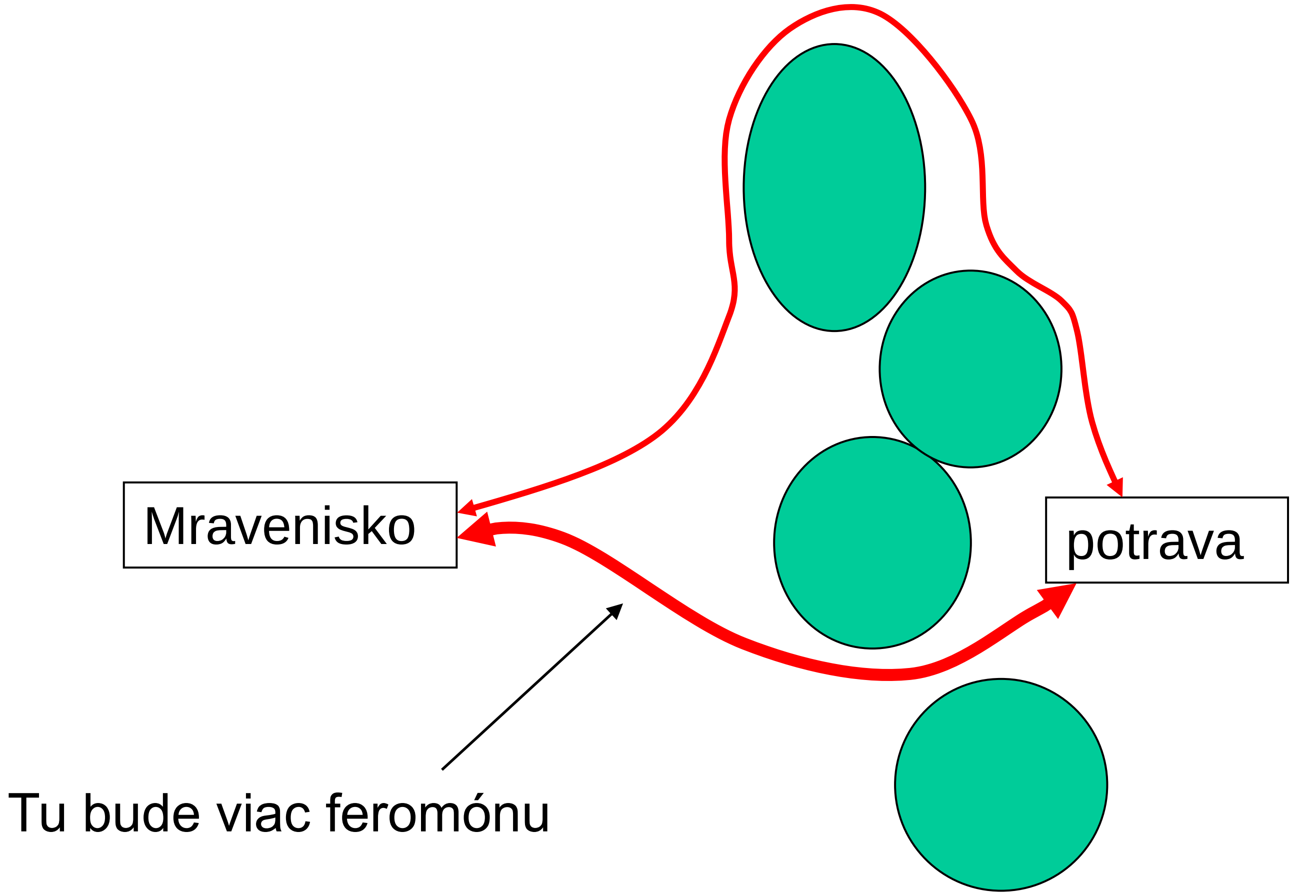


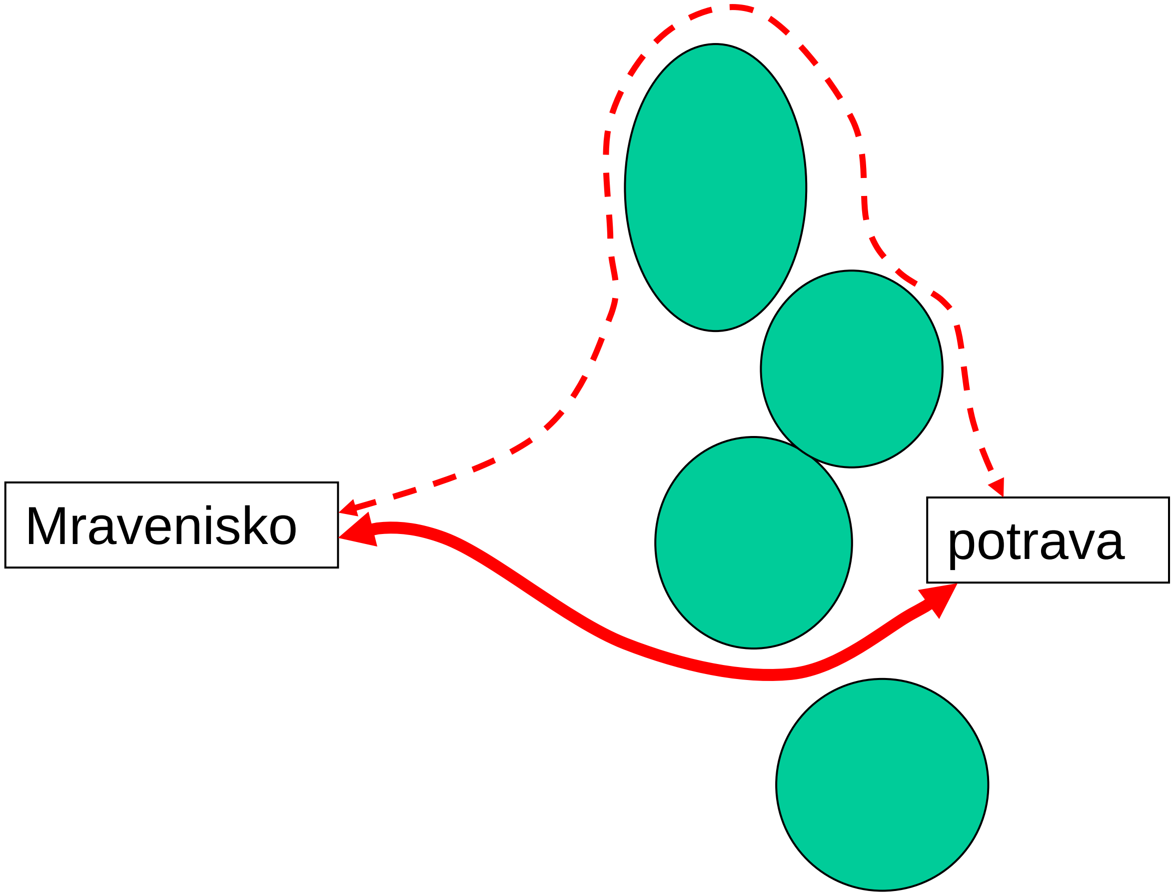
mravenisko - potrava - mravenisko

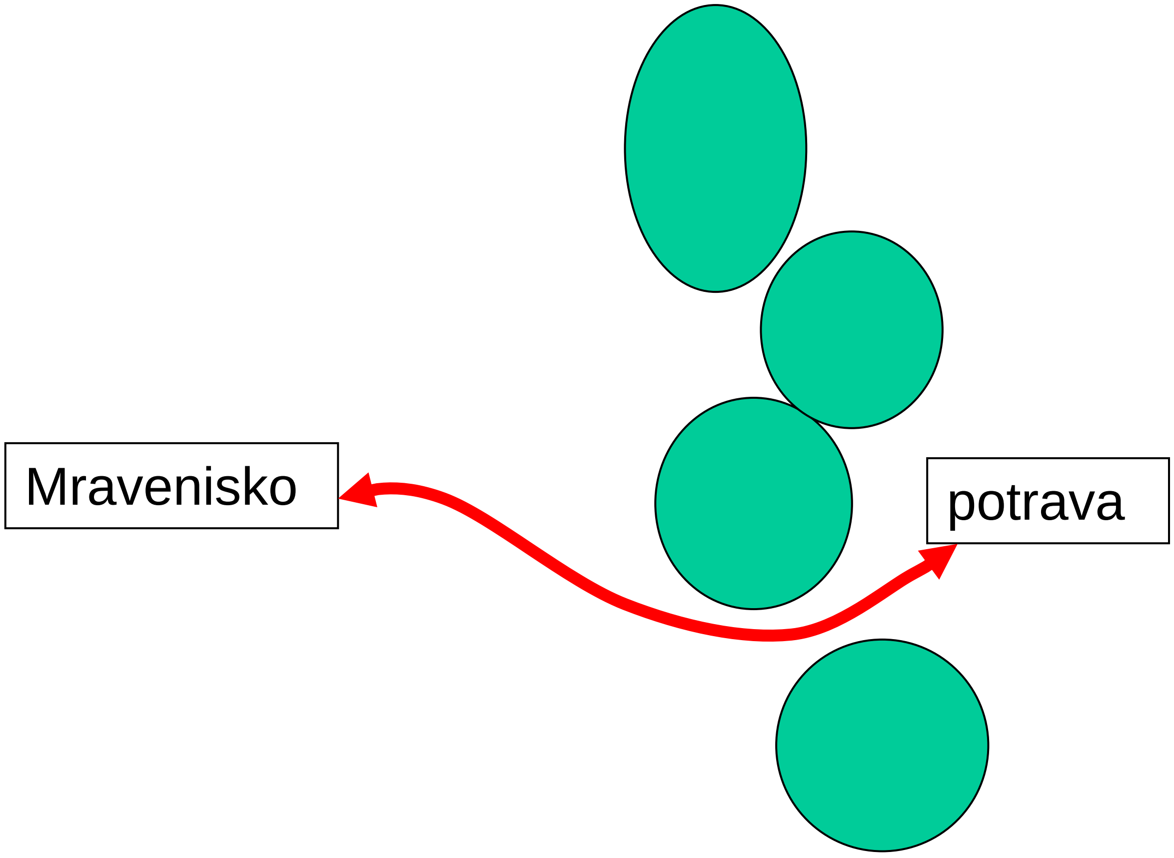
Vyparovanie feromónu

za rovnaký čas prejde po kratšej trase viac mravcov

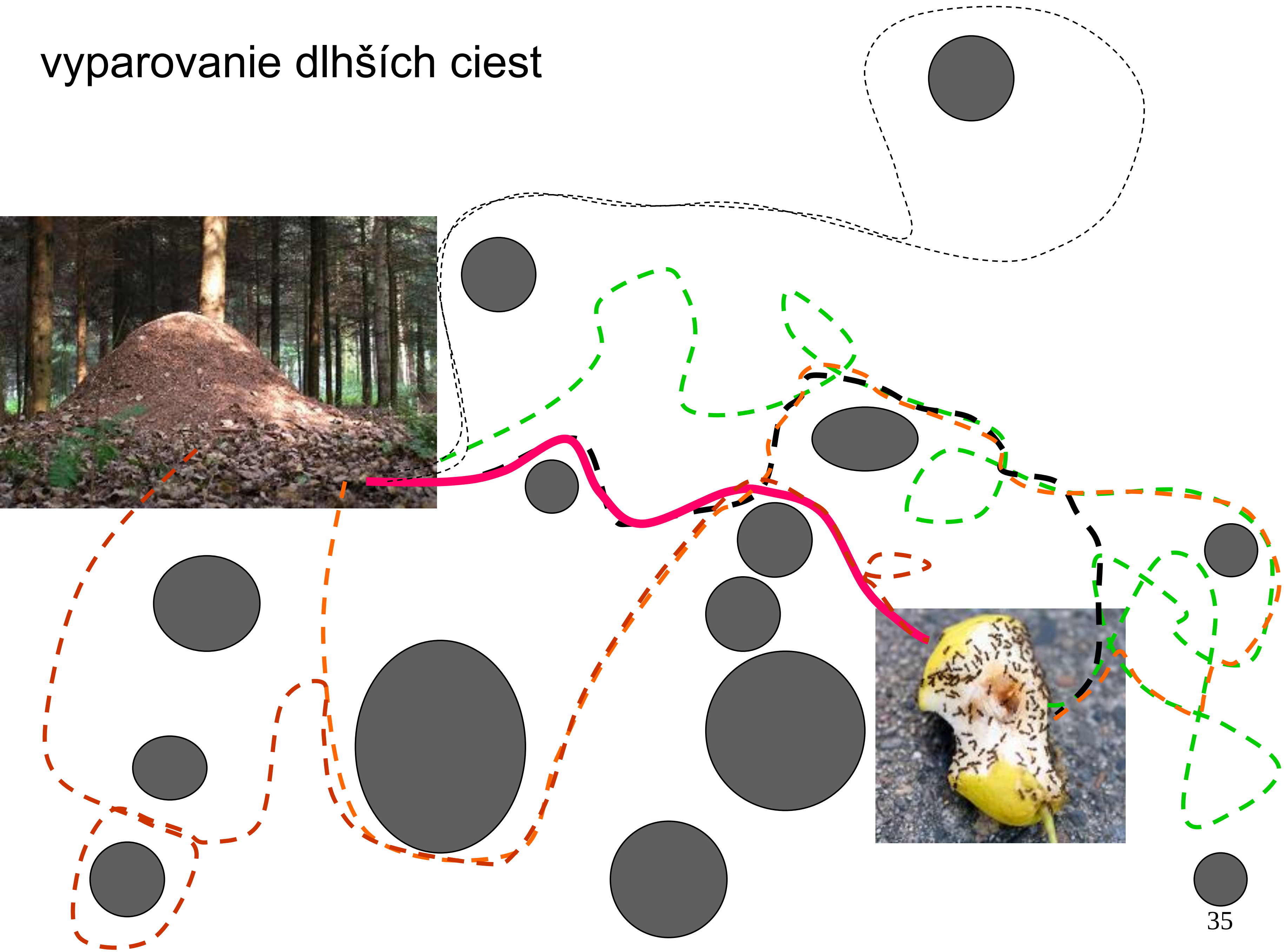



Mravenisko
potrava
Tu bude viac feromónu


Mravenisko
potrava


Mravenisko
potrava

vyparovanie dlhších ciest



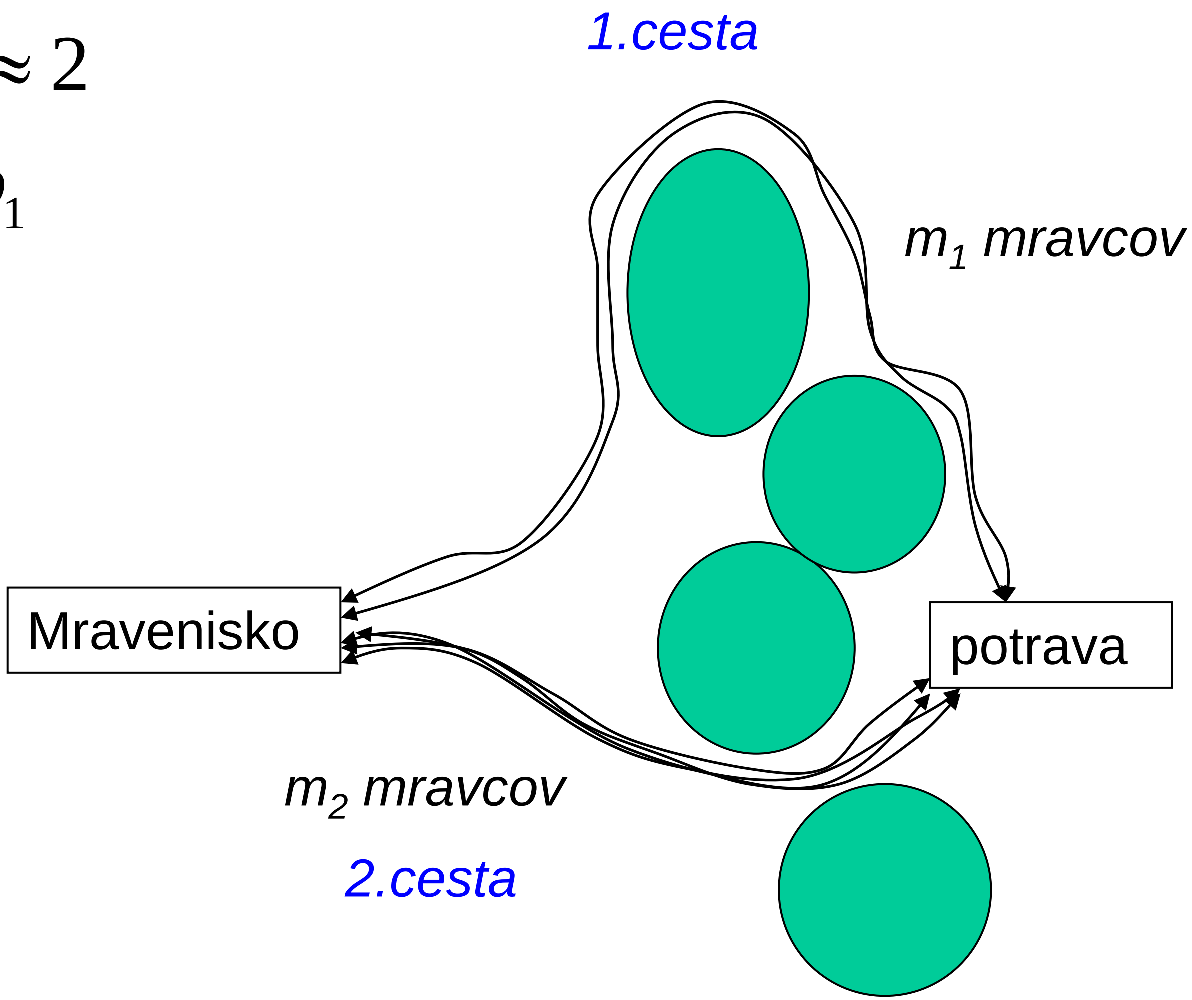
nakoniec zostane najkratšia cesta,
ostatné sa vyparia

Pravdepodobnosť, že mravec pôjde 1. cestou je:

$$p_1 = \frac{(m_1 + k)^h}{(m_1 + k)^h + (m_2 + k)^h}$$

$$k \approx 20, h \approx 2$$

$$p_2 = 1 - p_1$$

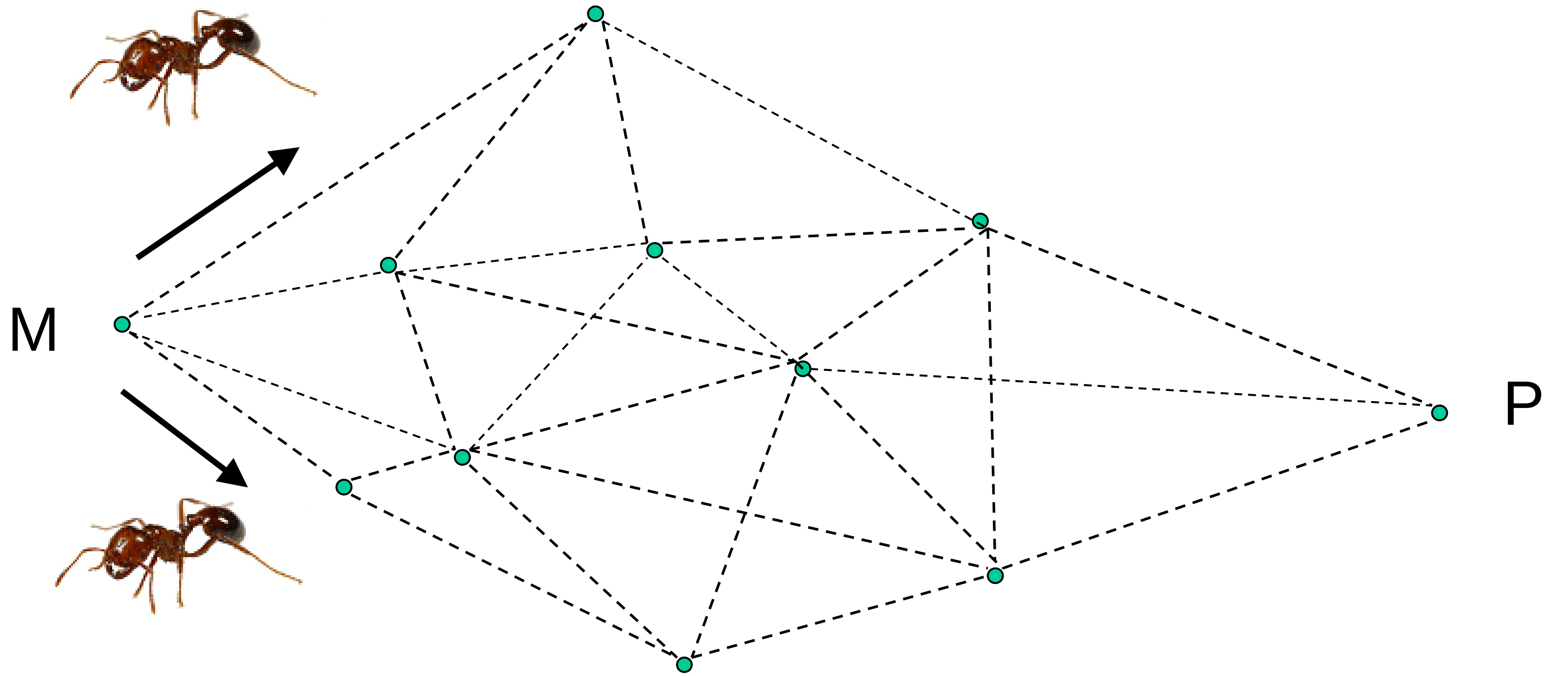


ACO algoritmus

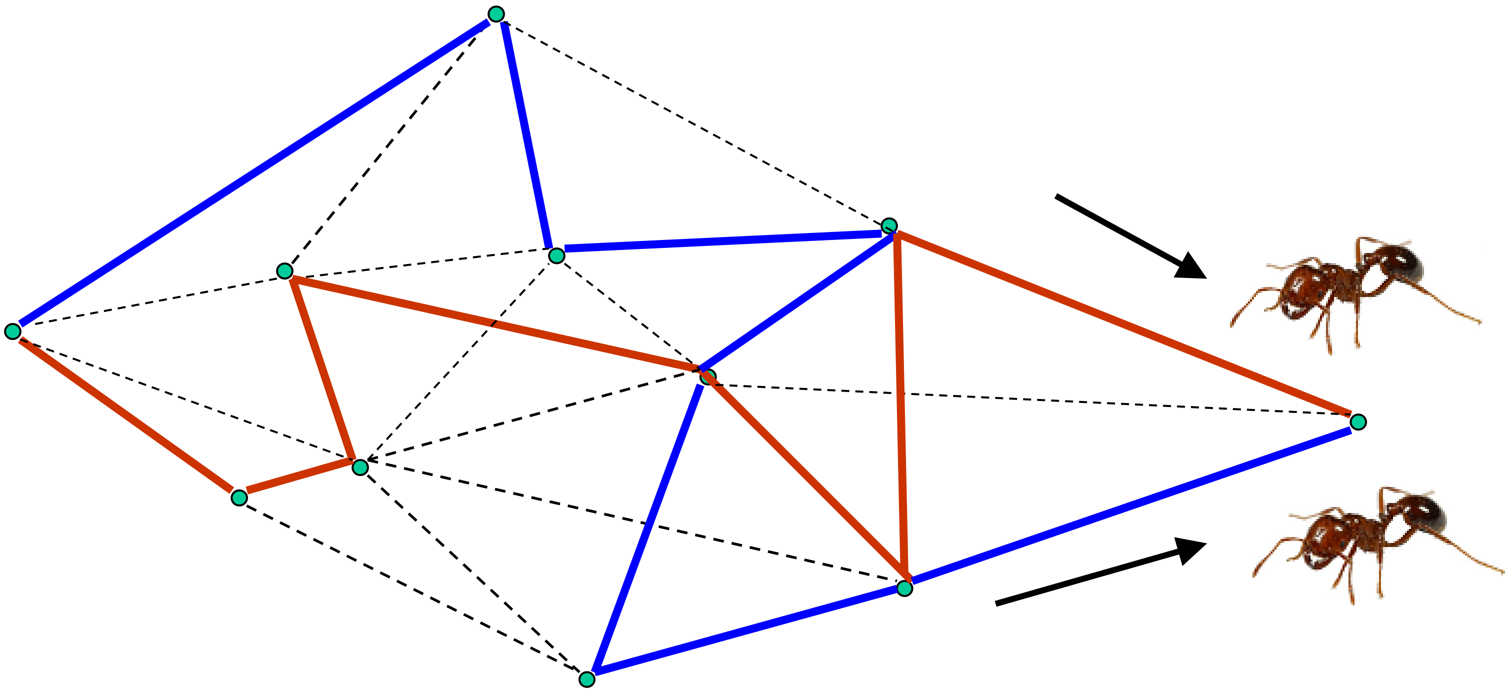
(základná verzia)

pre úlohu obchodného cestujúceho

0. Predpokladajme množinu (kolóniu) M mravcov, ktorý hľadajú potravu pre mravenisko po definovaných cestách.



1. V každej iterácii prejde každý z M mravcov zo začiatku na koniec v ľubovoľnom poradí vrcholov. Žiadny vrchol neopakuje.

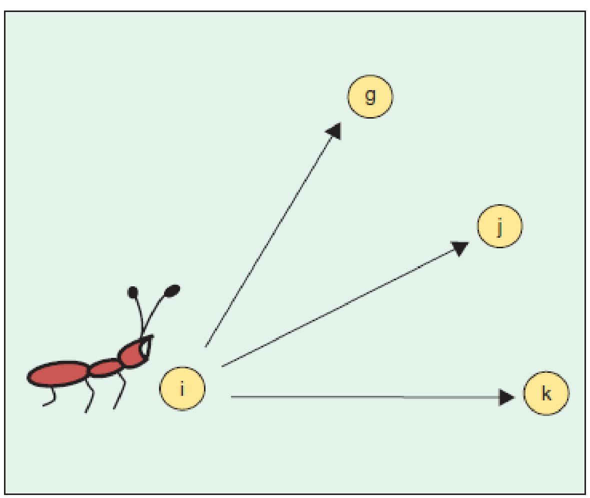


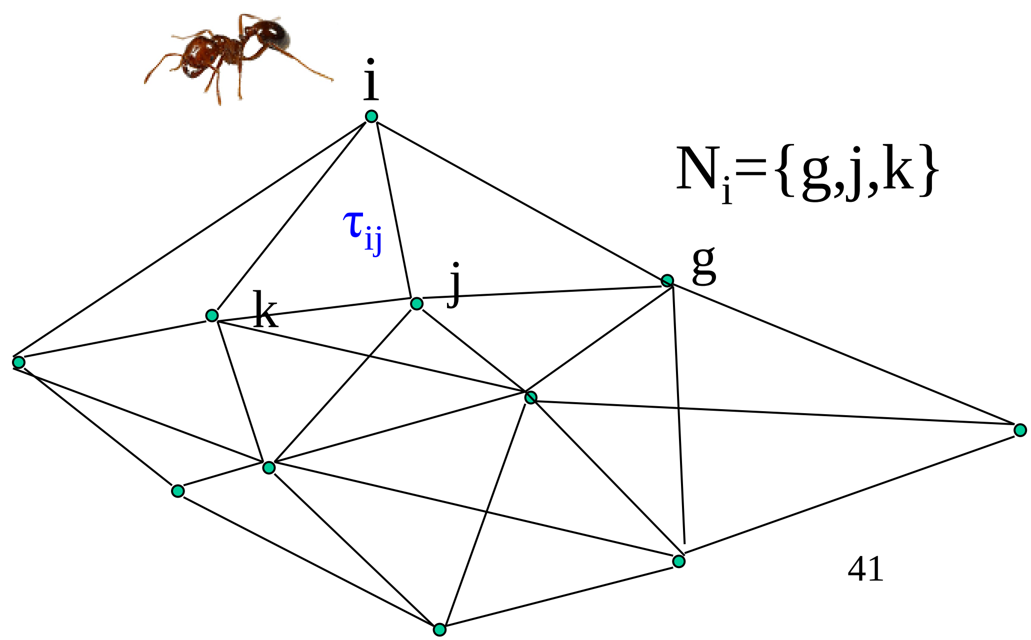
1.1 Každý nasledujúci vrchol ***j*** trasy k-teho mravca z vrcholu ***i*** je vyberaný s pravdepodobnosťou podľa vzťahu

$$p_{ij}^{k} = \frac{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}{\sum_{j}\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}$$

$$j \in N_i$$

N_i – Prípustný nenavštívený vrchol z vrcholu ***i***
$\boldsymbol{\tau_{ij}}$ **–** množstvo feromónu na hrane *ij*
$\boldsymbol{\eta_{ij}}$ **– 1/**$\boldsymbol{d_{ij}}$ **,** $\boldsymbol{d_{ij}}$ **–** dĺžka hrany
α, β **–** voliteľné parametre





2. Po prechode všetkých mravcov sa obnovuje hodnota feromónu každej hrany podľa vzťahu

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \sum_{k=1}^{M} \Delta\tau^{k}{}_{ij}(t)$$

$$\forall(i, j)$$

$\rho < 1$ koeficient vyparovania

$\Delta\tau^{k}{}_{ij} = 1/L_k$ ak mravec k použije hranu i-j

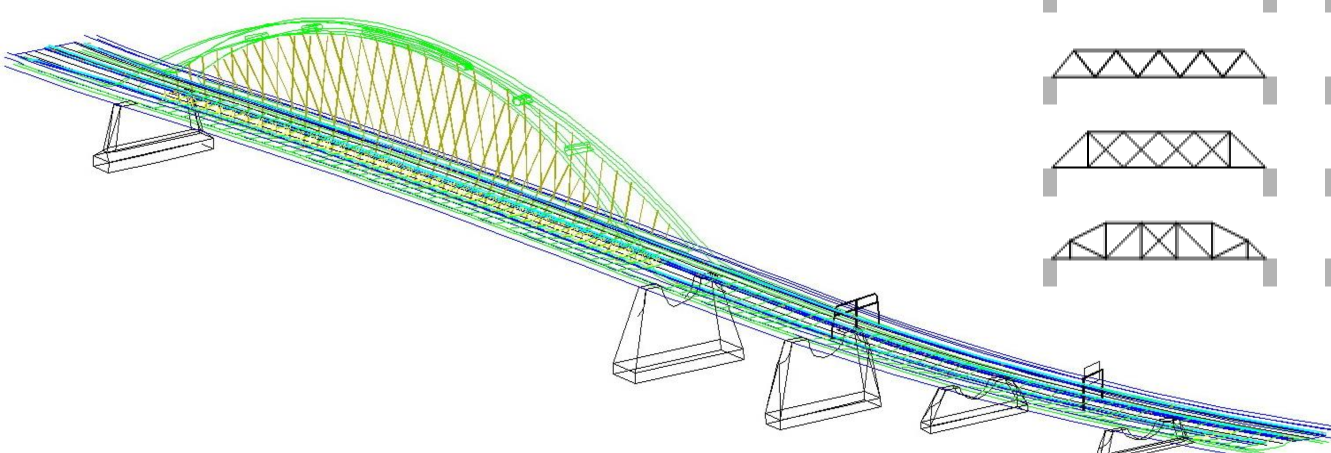
$\Delta\tau^{k}{}_{ij} = 0$ inak

L_k – dĺžka (ohodnotenie) celej cesty mravca k

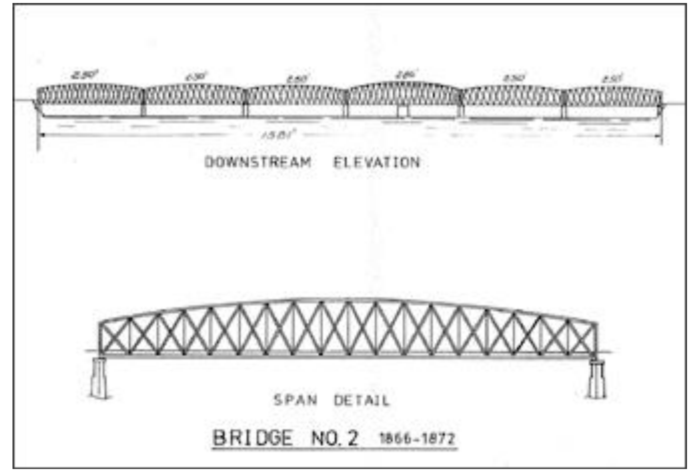
3. Opakuj iterácie (kroky 1 a 2) do splnenia ukončovacích podmienok.
   Napr: splnenie požadovanej kvality riešenia atď.

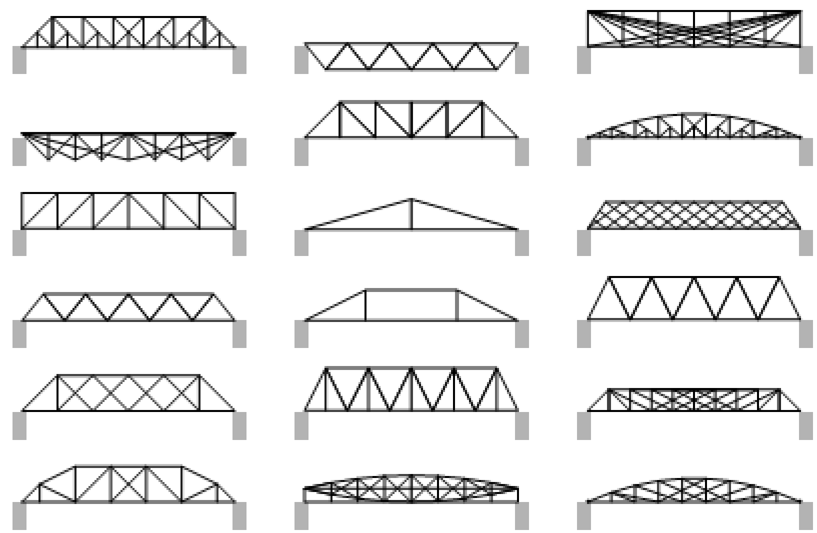
1.8 Genetické programovanie

Známa štruktúra objektu a neznáme parametre



Neznáma štruktúra objektu aj neznáme parametre



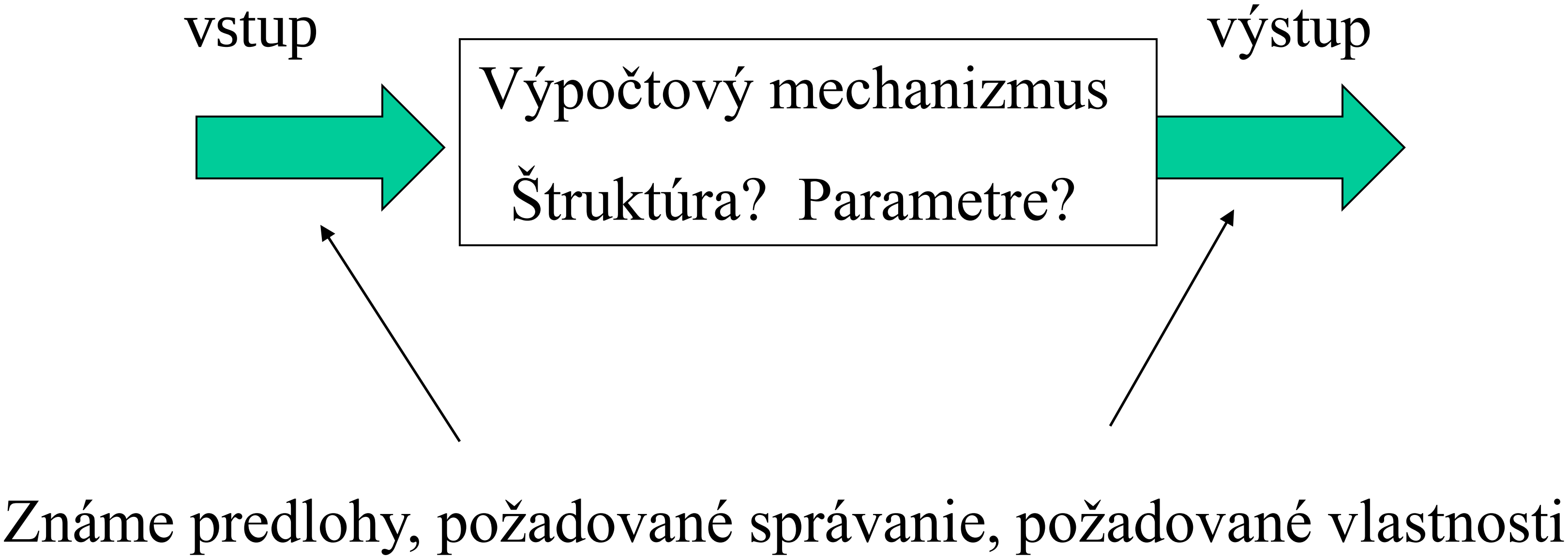


Neznáma štruktúra objektu aj neznáme parametre

Genetické programovanie nástroj na tvorbu (evolúciu)

- Algoritmov, programov
- Schém, zapojení
- Grafov, štruktúr
- Konštrukcií
- Strojové učenie
- Hľadanie riešení (ne)obmedzenej zložitosti
- ...

GP


vstup
Výpočtový mechanizmus
Štruktúra? Parametre?
výstup
Známe predlohy, požadované správanie, požadované vlastnosti

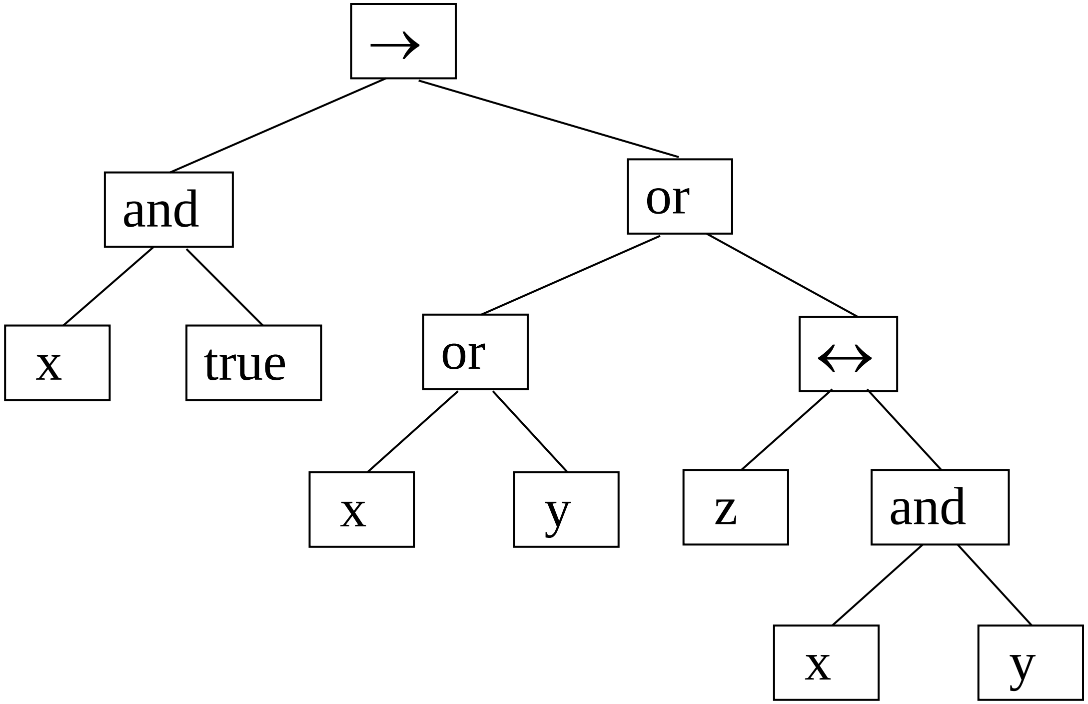
Genetické programovanie - aplikačné oblasti a príklady

- Symbolická regresia – matematické výrazy
- Získavanie dát, analýza dát
- Syntéza logických funkcií
- Návrh schém (elektrických obvodov, iných)
- Návrh štruktúr (konštrukcie, grafy)
- Návrh pravidiel (finančníctvo, obchod, rozhodovanie)
- Automatizácia programovania
- Robotika, regulačné obvody
- Biochémia, medicína
- iné

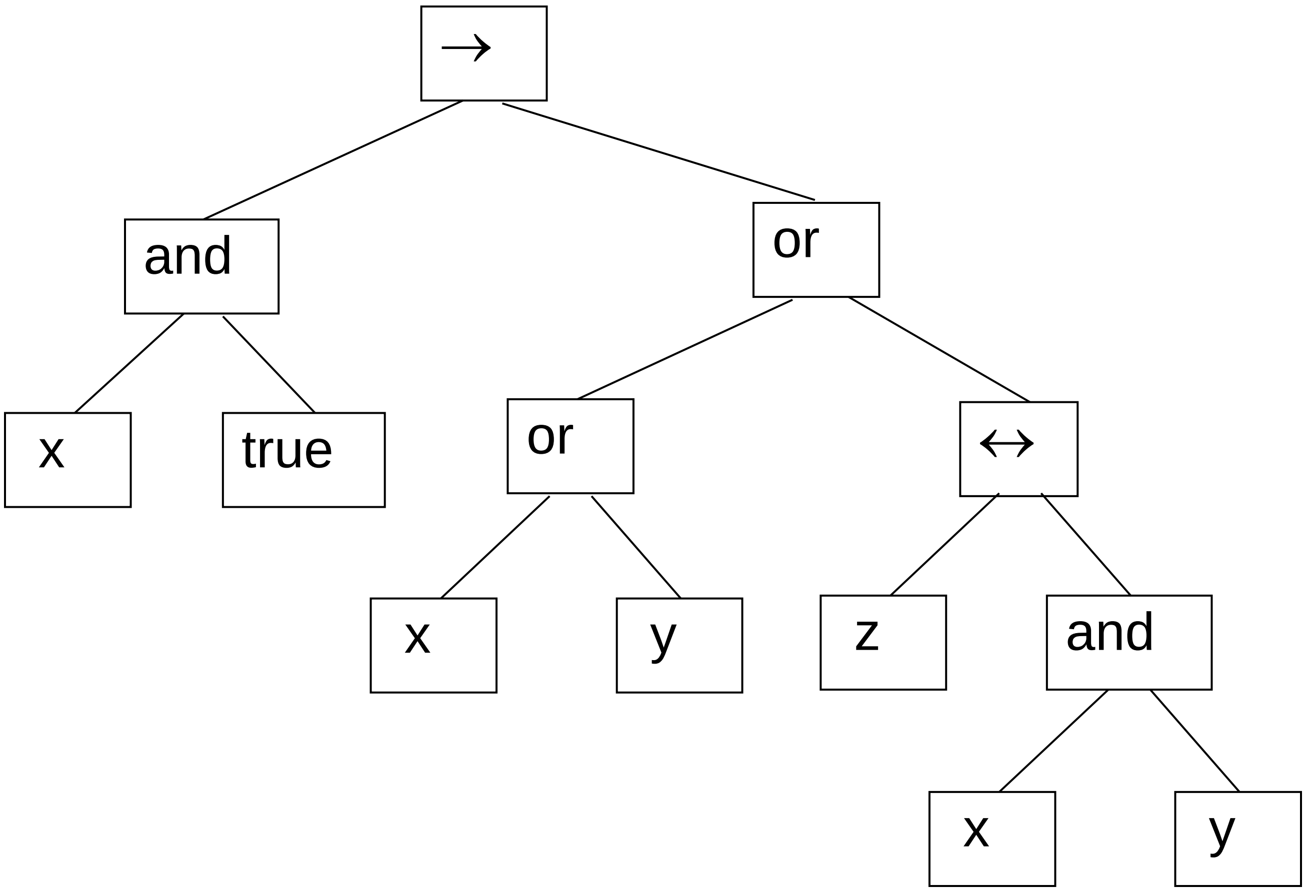
Reprezentácia jedinca

- stromová
- tabuľková
- lineárna
- grafová

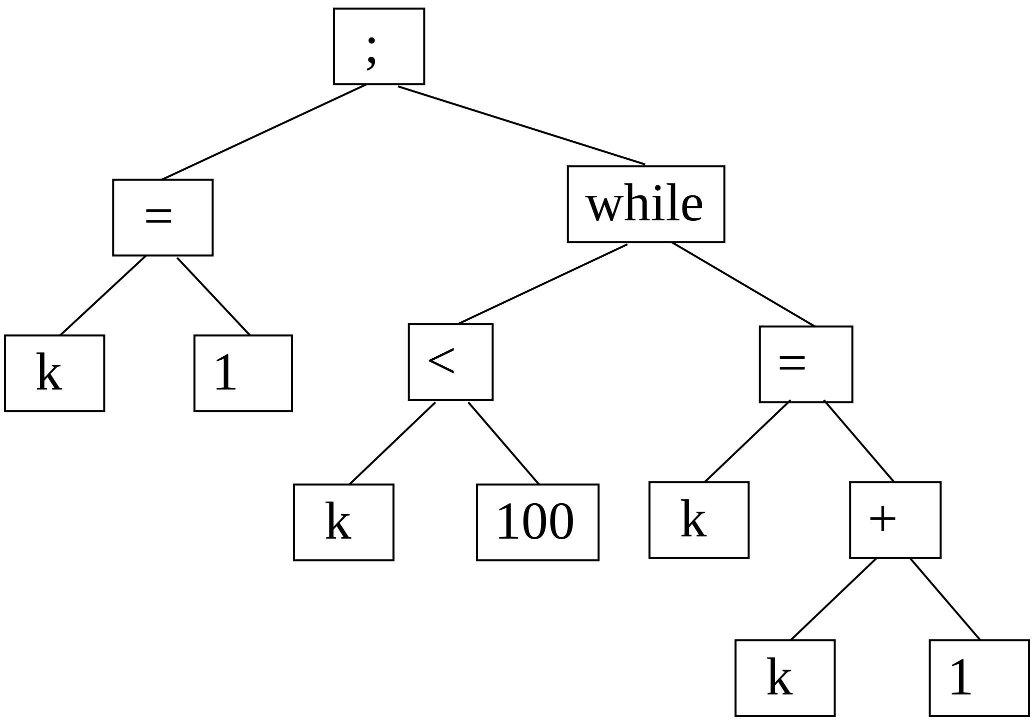
Stromová reprezentácia
logickej funkcie



$$(a \wedge true) \rightarrow ((a \vee b) \vee (z \leftrightarrow (a \wedge b)))$$



Stromová reprezentácia programu



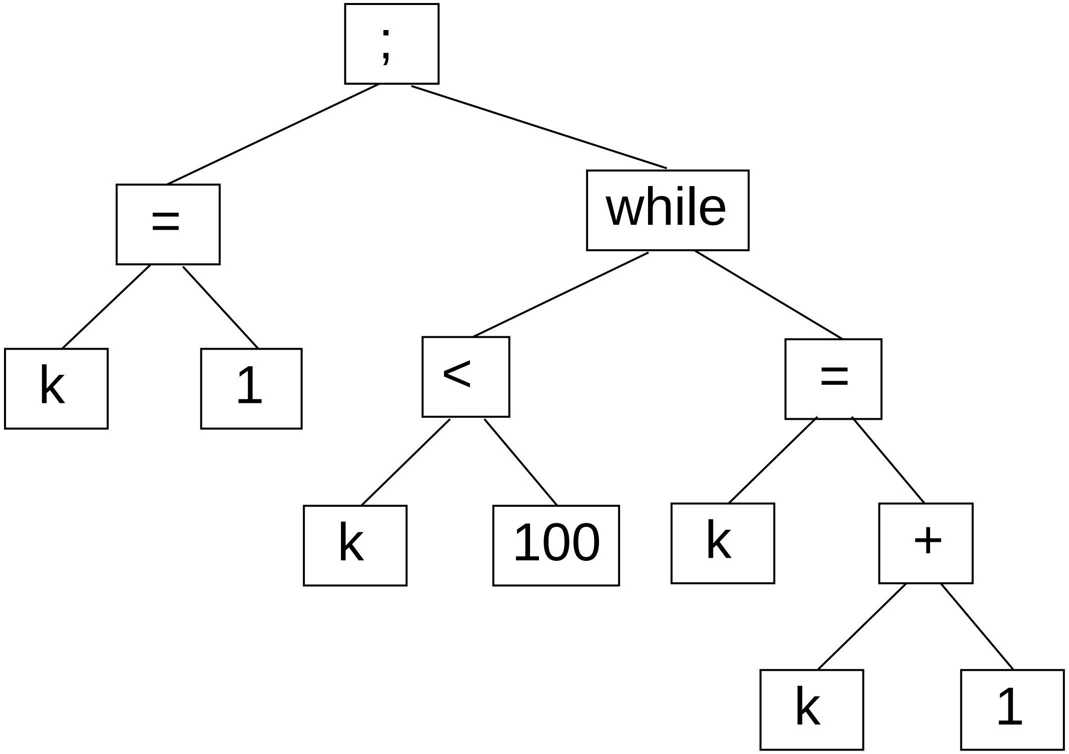
k=1;

while (k<100)

{

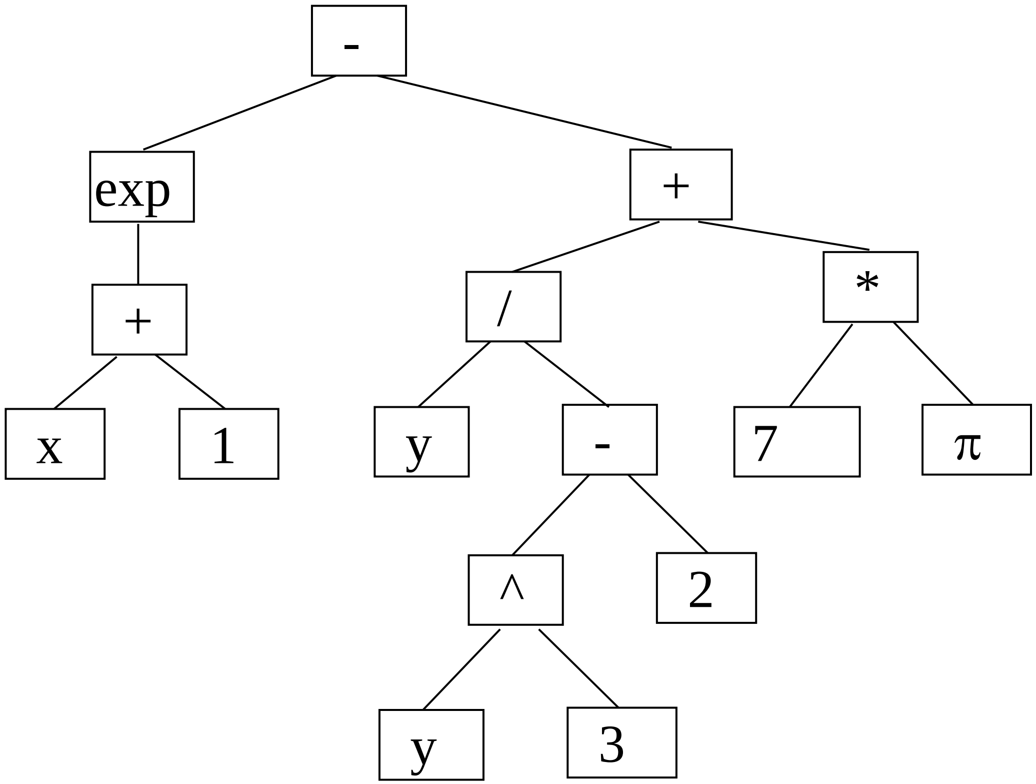
k=k+1

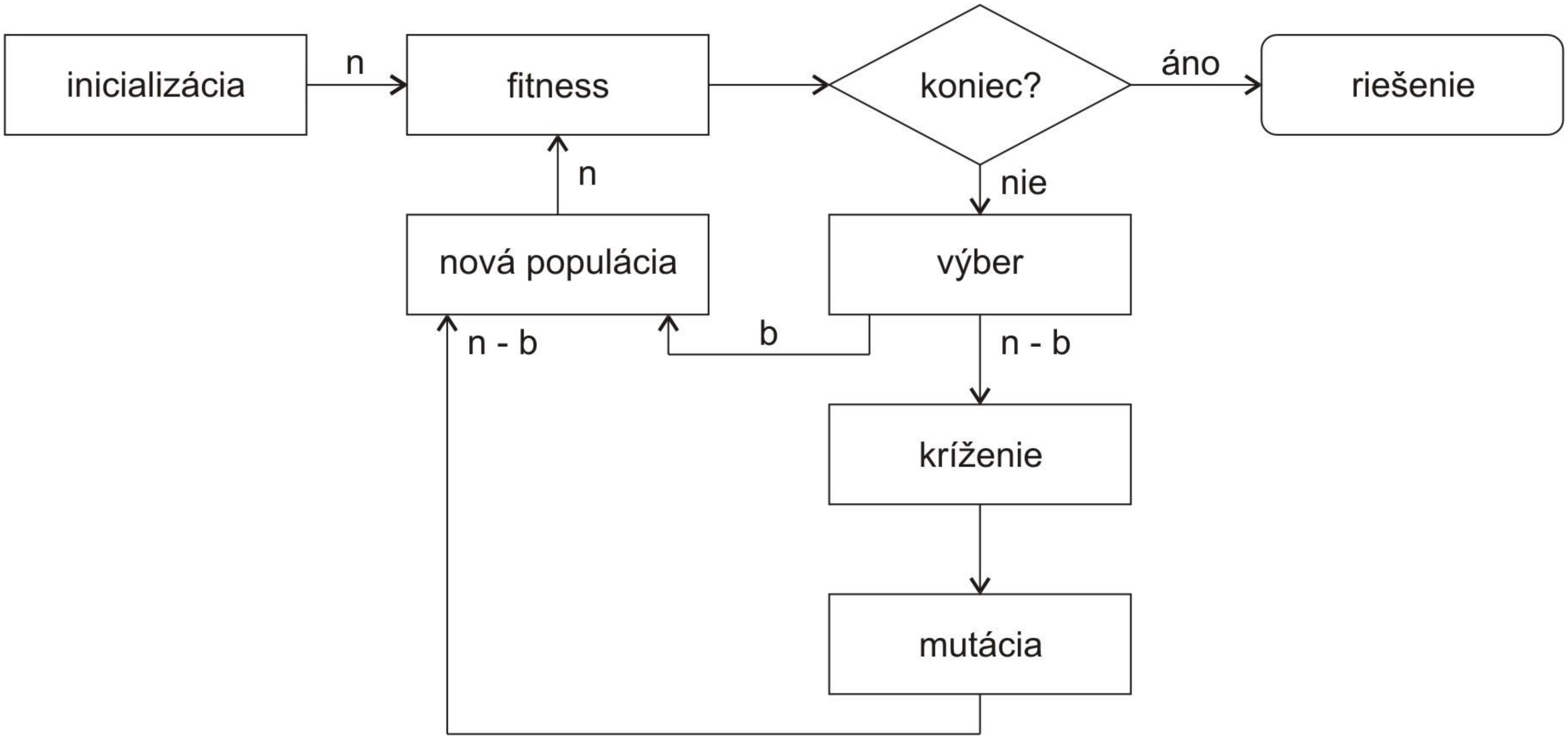
}



Stromová reprezentácia
matematického výrazu

$$\exp(x+1) - \frac{y}{y^3 - 2} + 7\pi$$



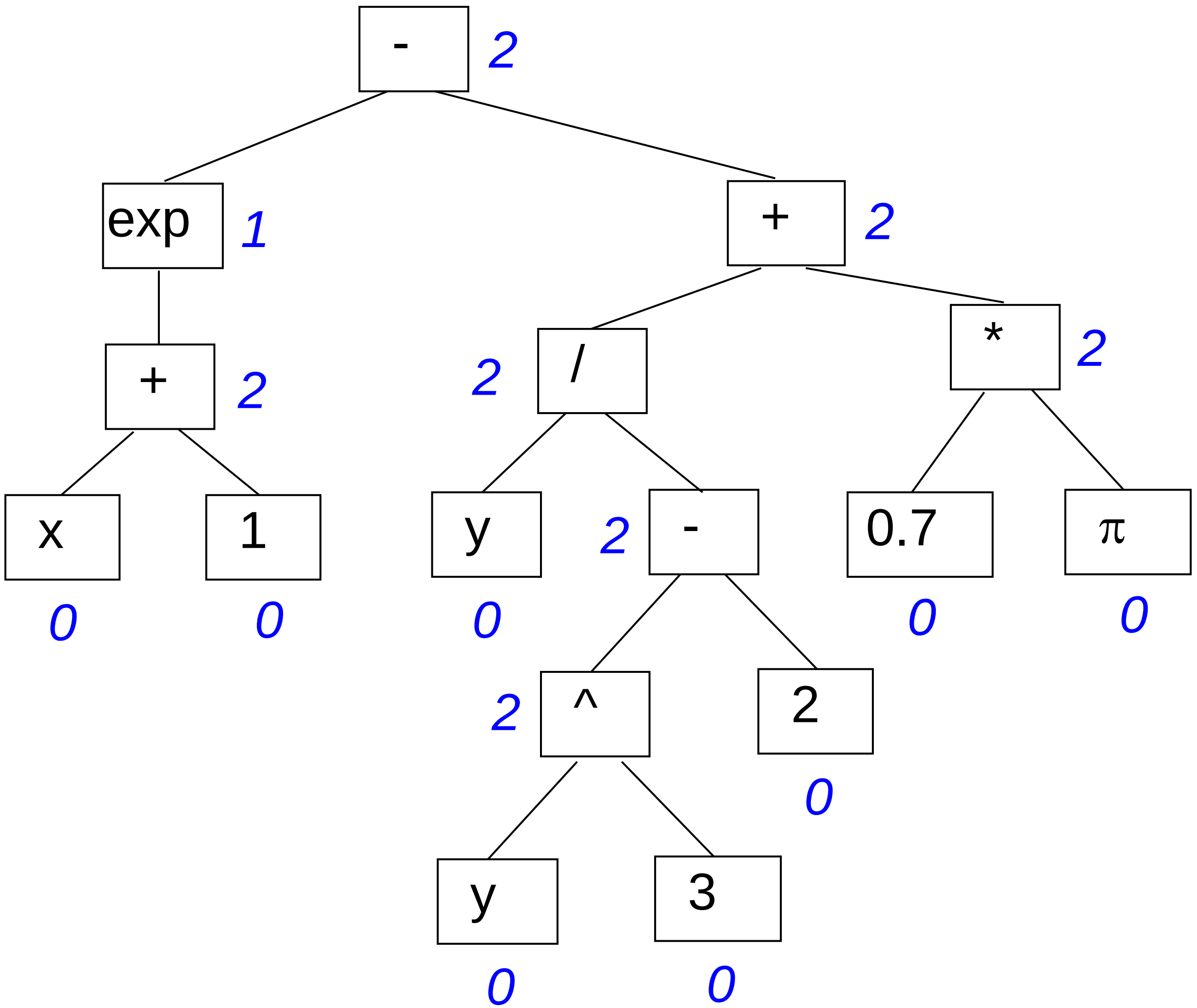


Tento (evolučný) algoritmus narába s reťazcom (lineárnou postupnosťou) znakov, symbolov.

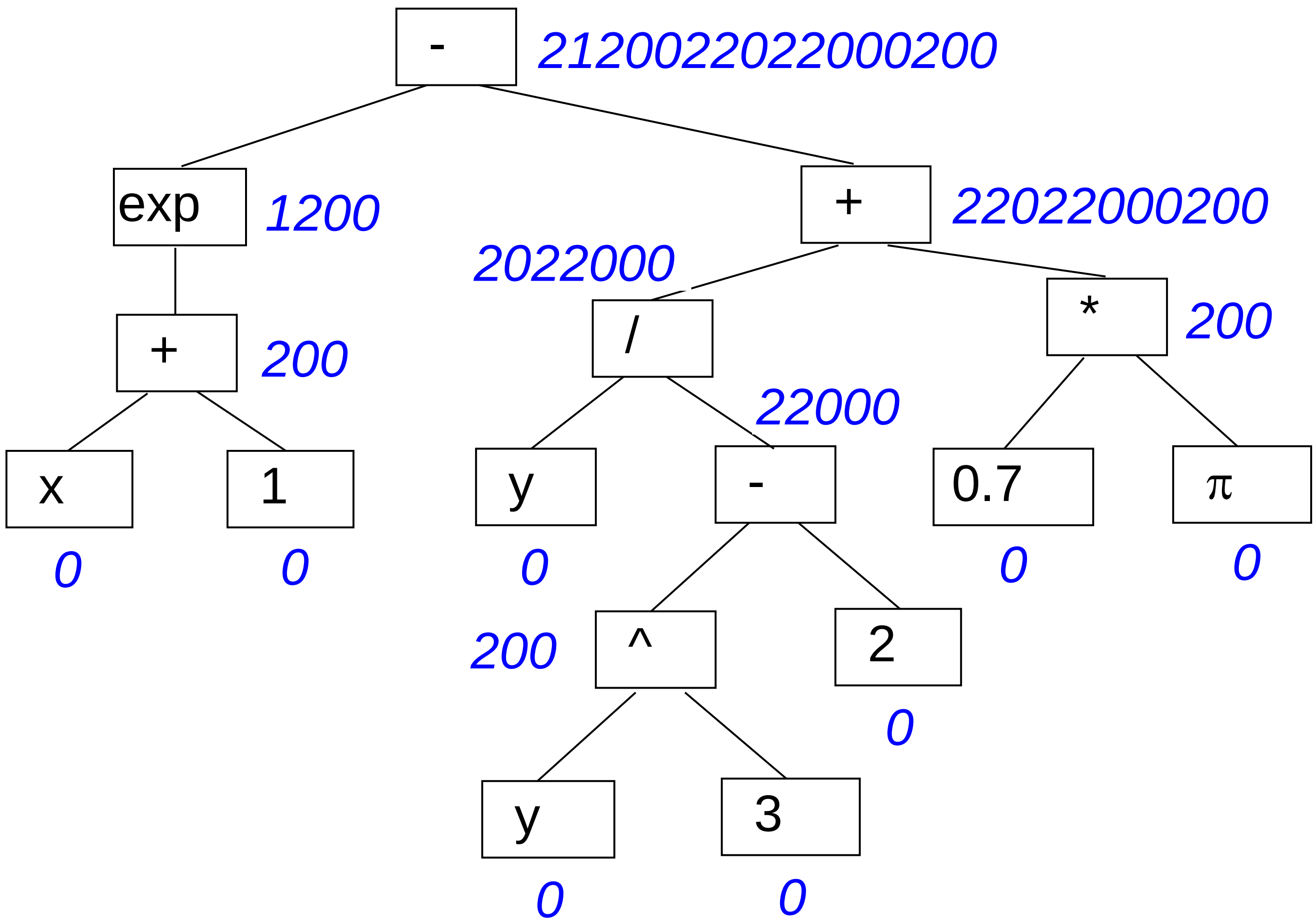
Readov lineárny kód

transformácia stromovej štruktúry do lineárneho reťazca znakov

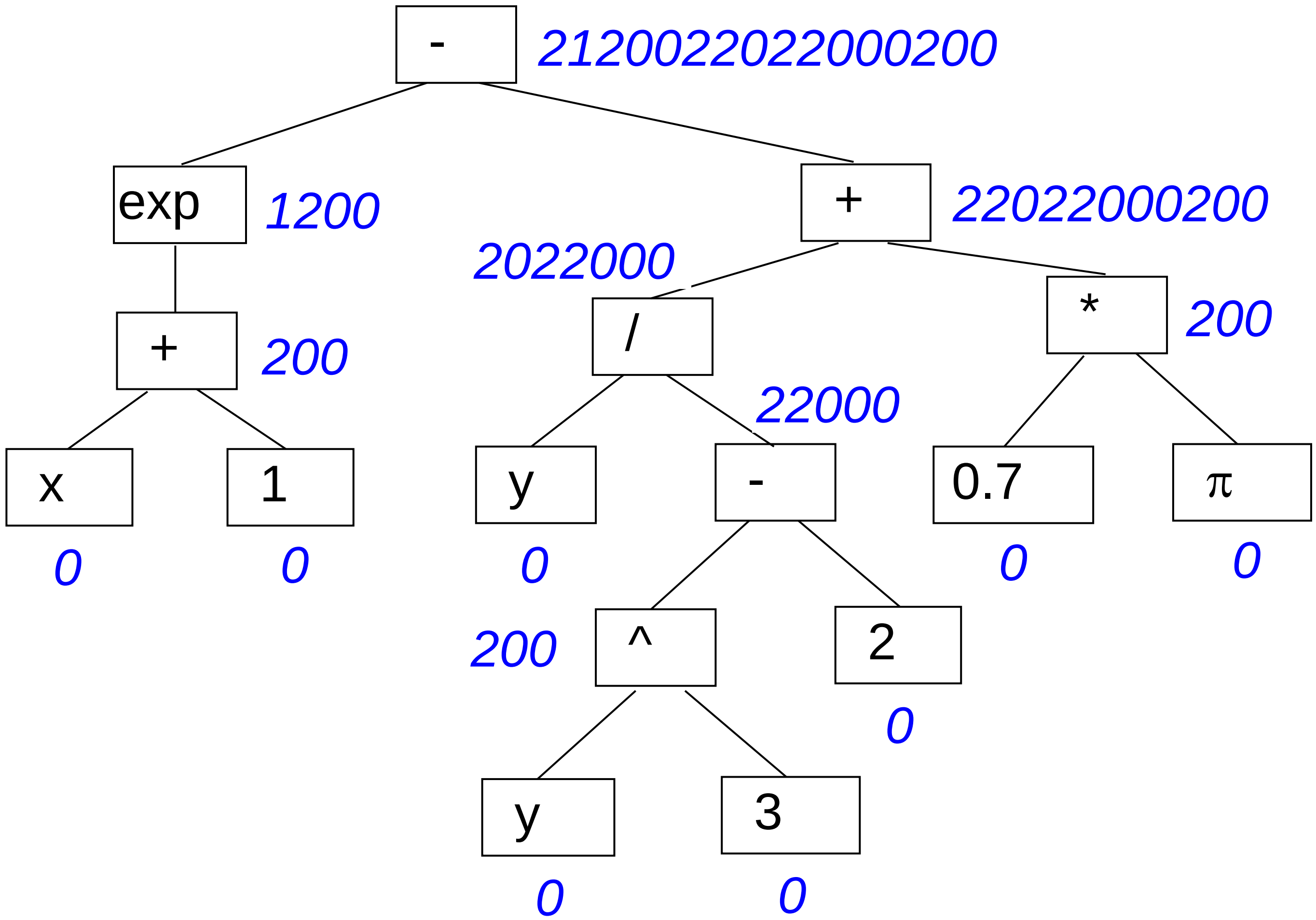
Valencia uzlov stromu



Readov kód štruktúry stromu



Kompletný readov kód stromu



[(2,-),(1,exp),(2,+),(0,x),(0,1),(2,+),(2,/),(0,y),(2,),(2,^),(0,y),(0,3),(0,2),(2,*),(0,0.7),(0,π)]

alebo [2,1,2,0,0,2,2,0,2,2,0,0,0,2,0,0, -,exp,+,x,1,+,/,y,-,^,y,3,2,*,0.7,π]

Lineárna reprezentácia jedinca

pomocou registrových operácií

$$\exp(x+1) - \frac{y}{y^3 - 2} + 7\pi$$

a=x

a=a+1

a=exp(a)

b=y

b=b^3

b=b-2

c=y

b=c/b

c=7

*c=c**π

a=a-b+c

Inicializácia jedincov v GP

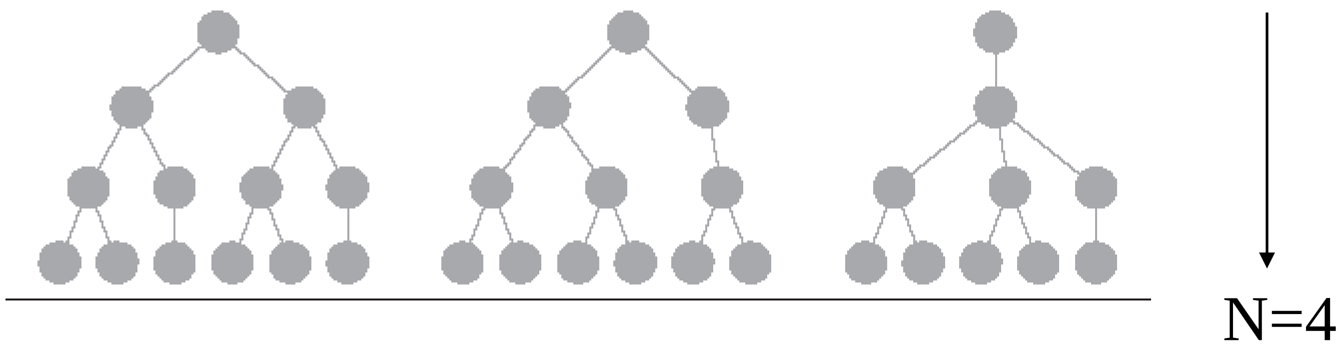
Počet jedincov v GP (veľkosť populácie) sa volí väčší než v GA alebo v iných typoch EA aby bola zabezpečená dostatočná diverzita bežne: 100-1000

Dimenzia prehľadávaného priestoru v úlohách GP býva veľmi veľká a je potrebná veľká diverzita jedincov.

Inicializácia populácie stormových jedincov

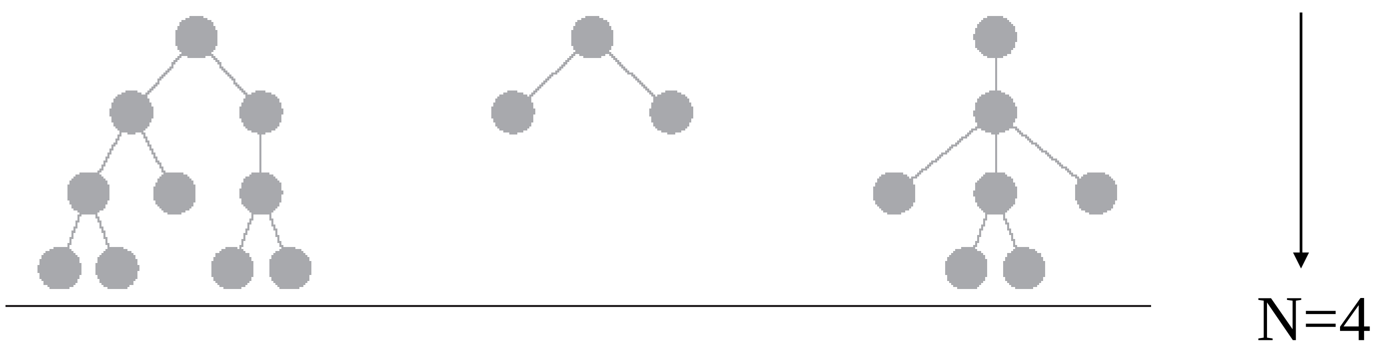
a) Plná meóda

Vygeneruje populáciu jedincov, ktorých hĺbka je práve N hrán (úrovní)



b) Rastová metóda

Vygeneruje populáciu jedincov, ktorých hĺbka sa zväčšue náhodne do N hrán (úrovní)



c) Kozova metoda
(John Koza, Ramped half-and-half)

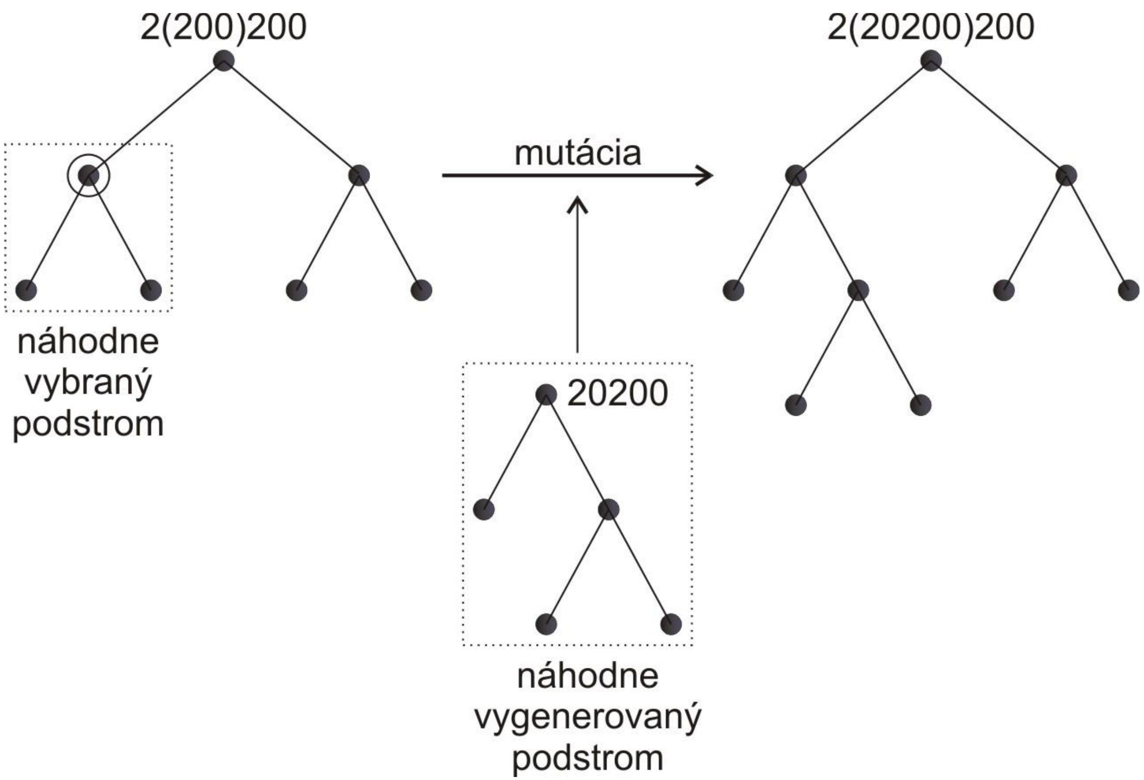
1. Pre každého jedinca sa vygeneruje náhodné číslo
$r<N$
2. Potom sa buď rastovou alebo plnou metódou vygeneruje strom o veľkosti max r

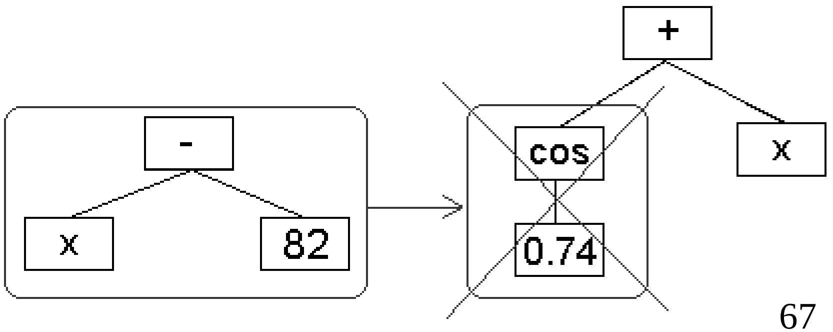
Gény v GP

- aritmetické operácie: +,-,*,/
- funkcie: *sin*, *cos*, *exp*, *log*, *ln* ...
- logické funkcie: *and*, *or*, *not*, *xor*
- priradenie hodnoty premennej: napr. *x=1*
- inštrukcie programu príslušného jazyka: *if*, *else*, *then*, *switch*, *goto*, *call*, *while*-*do*, *for*-*do*, ...
- podprogramy, makroinštrukcie alebo výkonné časti programu: načítanie informácie zo snímača, akčný zásah o určenej veľkosti, zatočenie doprava, zatočenie doľava
- prvky (orientovaných) grafov, prvky schém
- zapojenia, spojenia medzi prvkami schém
- konštrukčné prvky ...

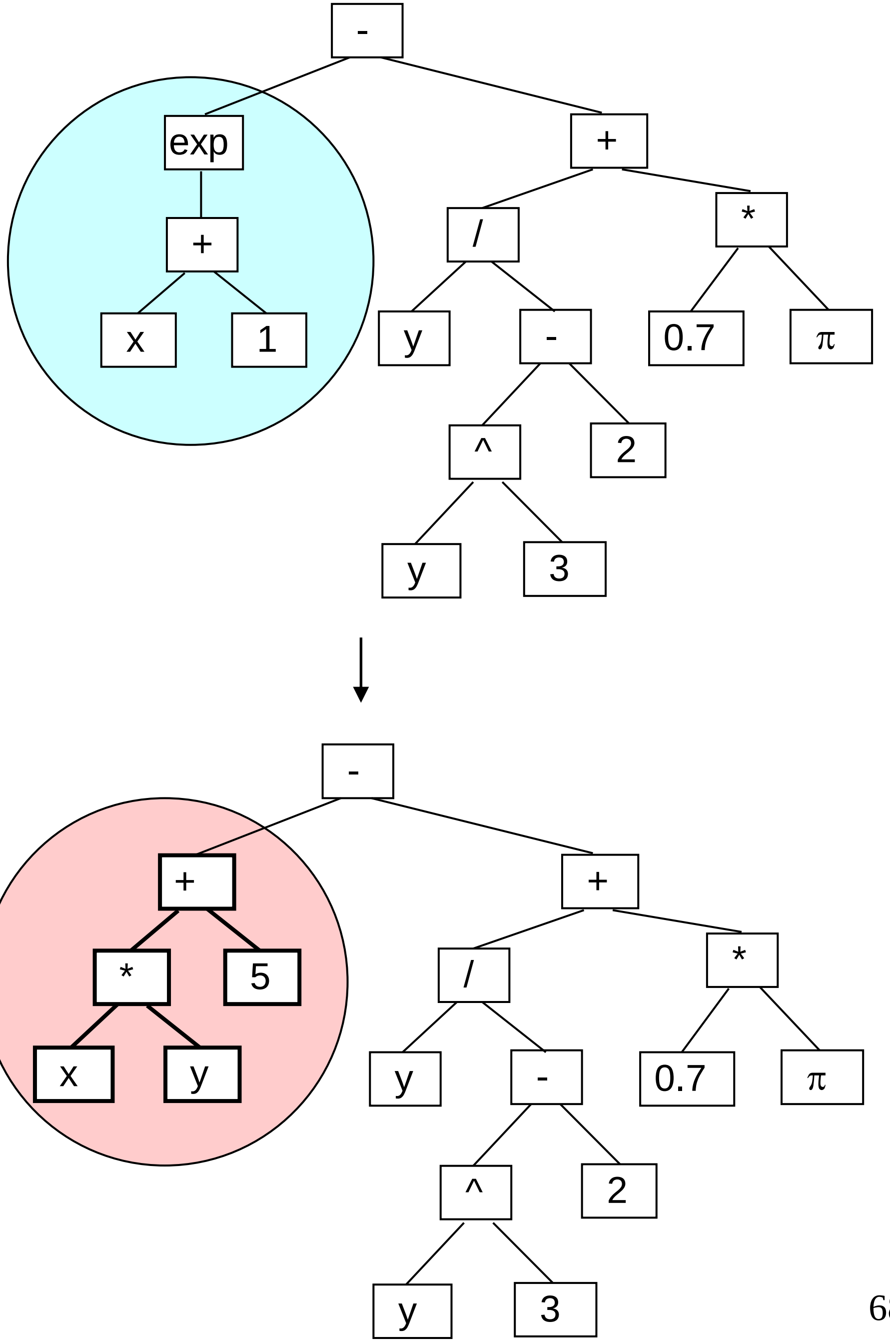
Genetické operácie

Mutácia – *náhrada podstromu*

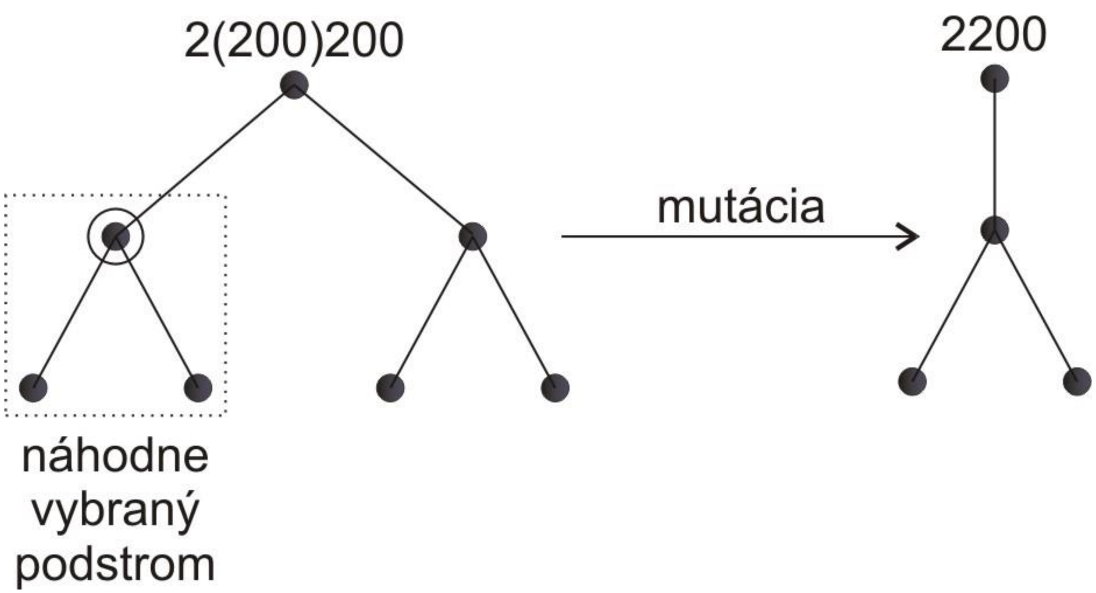


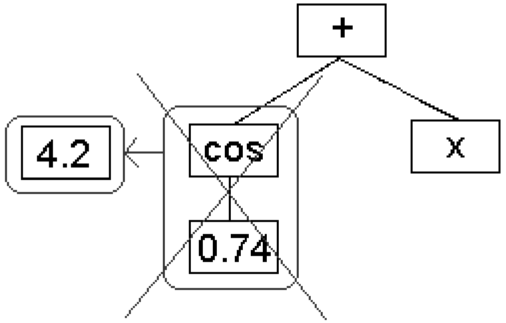


Náhrada podstromu iným podstromom

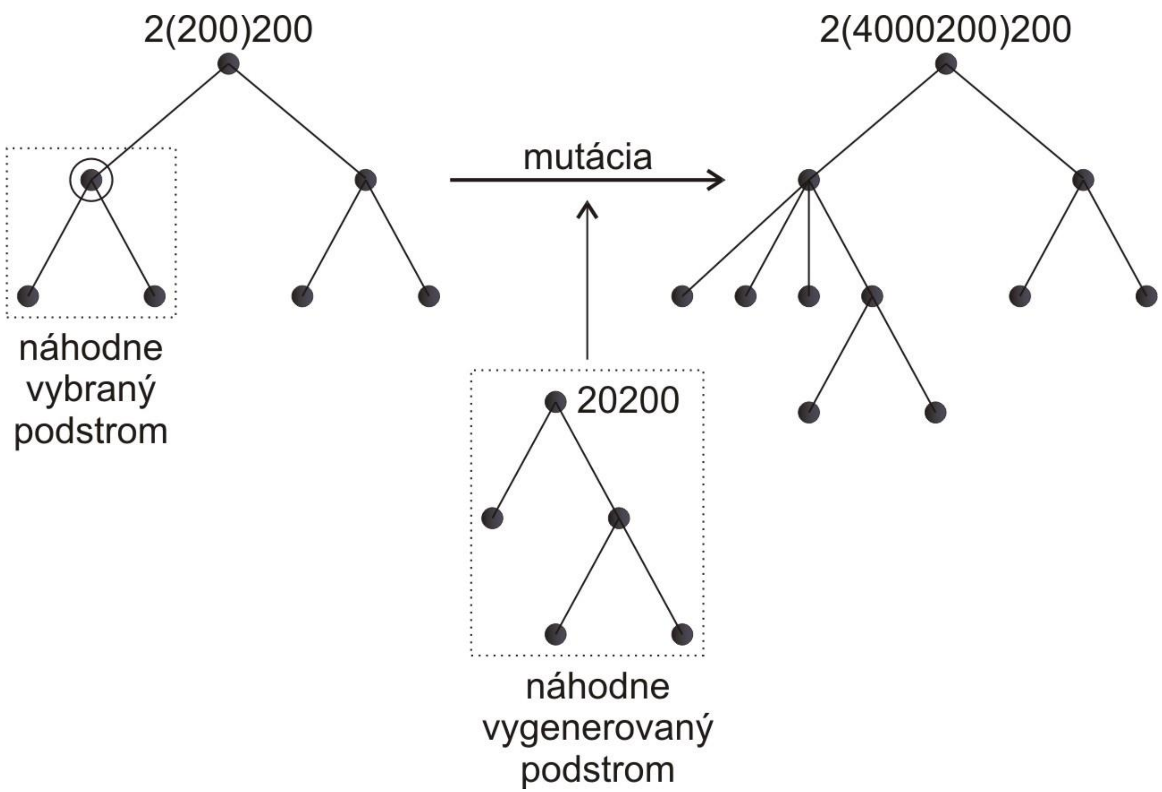


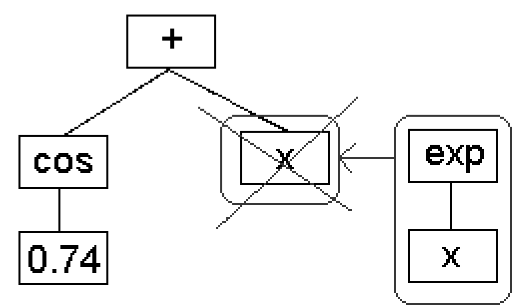
Mutácia - *odstránenie*





Mutácia - *doplnenie*



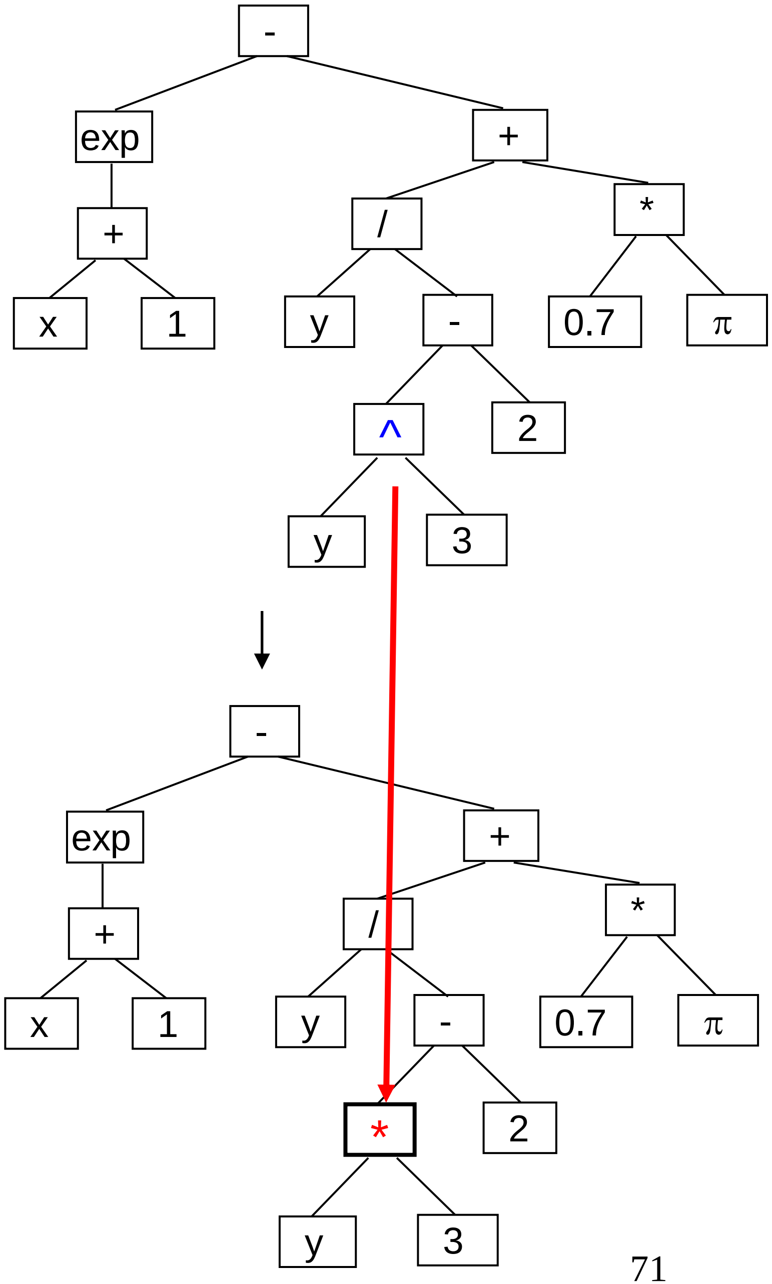


Náhrada uzla iným uzlom

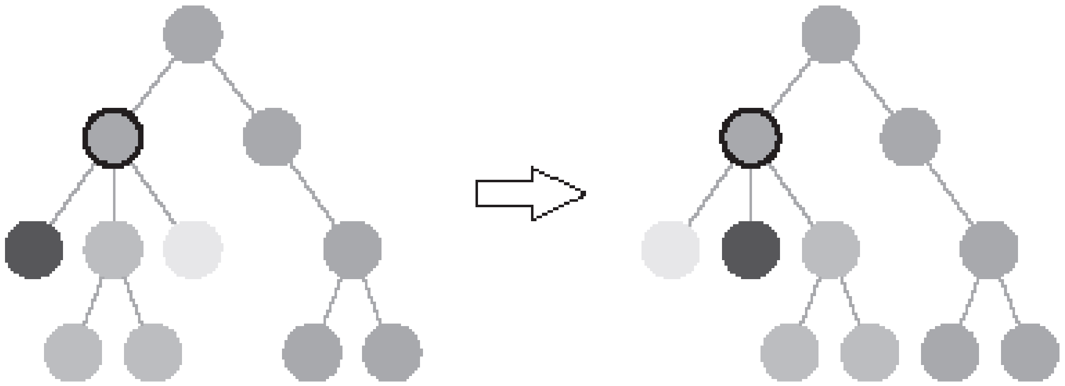
funkcia → funkcia
(operácia-opreácia)

premenná → premenná/konštanta

konštanta → premenná/konštanta

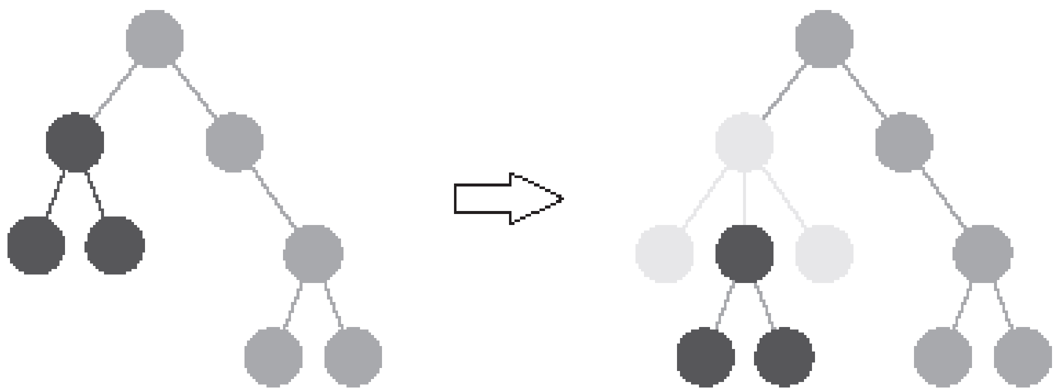


Mutácia – *posun (permutácia)*



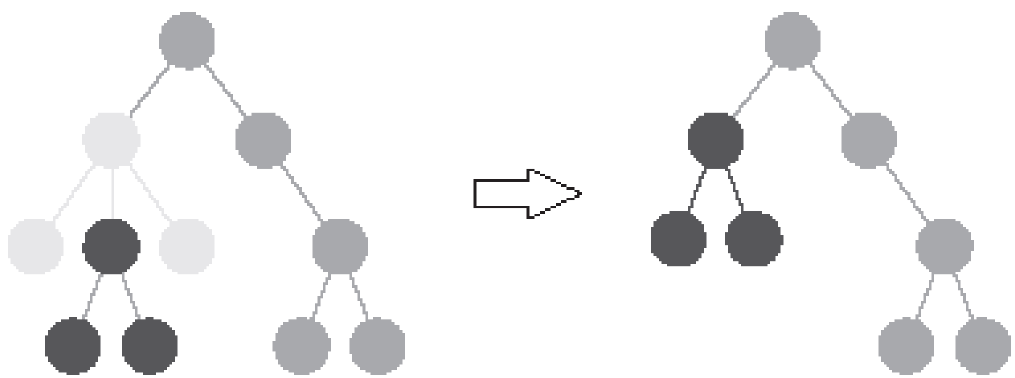
Vloženie podstromu

Na náhodne vybrané miesto (uzol) vo vnútri stromu sa vloží náhodne vygenerovaný podstrom.

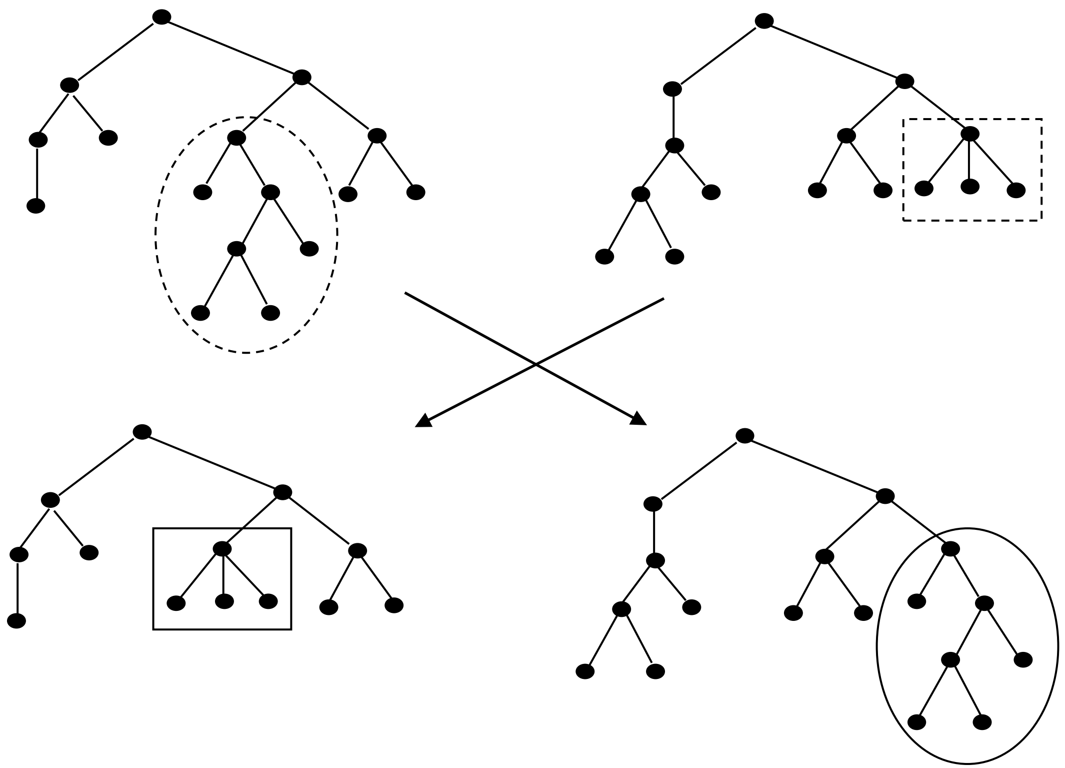


Vypustenie podstromu

Na náhodne vybranom mieste (uzle) vo vnútri stromu sa vystrihne a vynechá náhodne vygenerovaný podstrom.



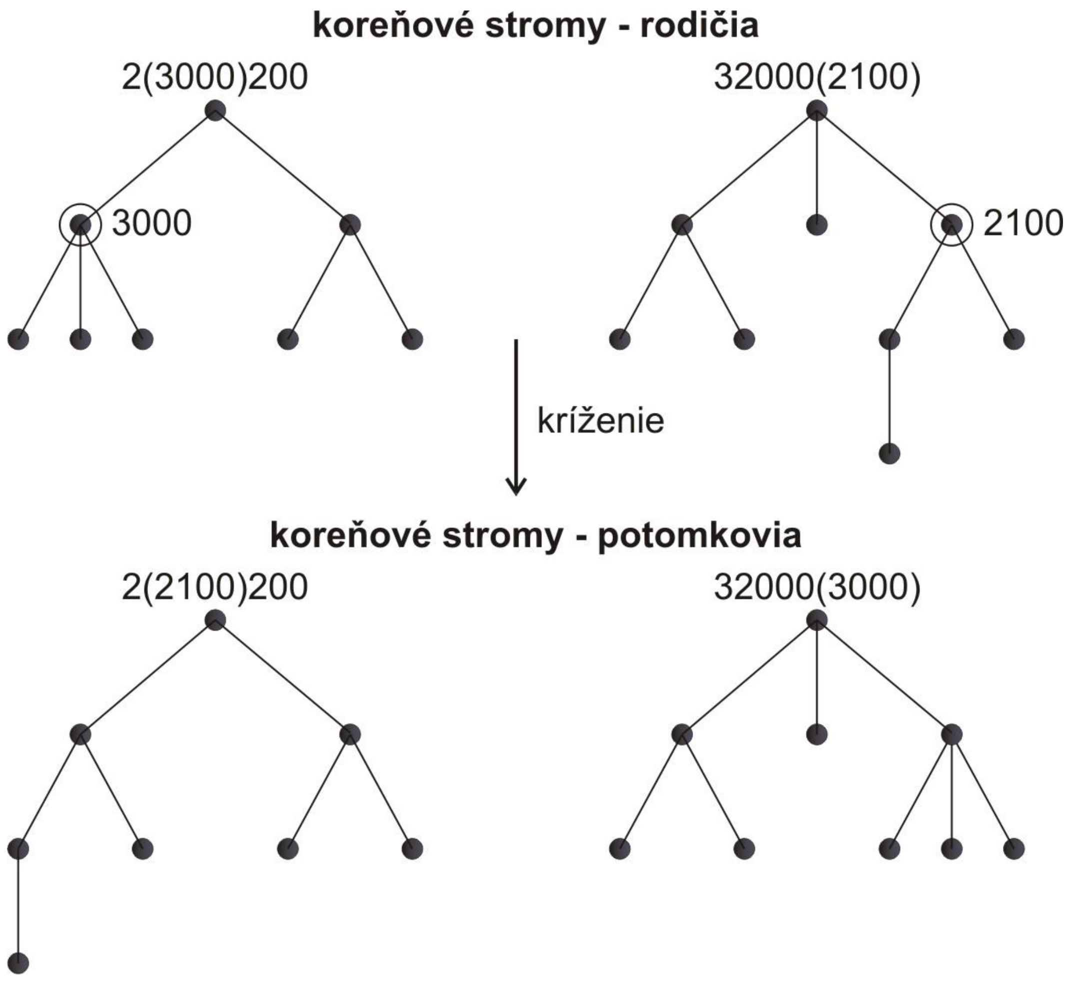
Kríženie stromov



Rodičia

Potomkovia

Kríženie stromov



Prerastanie stromov

V GP sa objavuje jav prerastania stromov („Bloat“). Je to spôsobené tým, že zlepšovanie fitness je možné dosahovať malými modifikáciami stromov, čím sa ale môže predlžovať (komplikovať) reťazec.

Príklad

vzor: $f(x)=x^2+1$

F(x)=(x*x*x)+1,78-0,85-0,2*x

F(x)=(x*x*x)/x+1,78-0,85-0,2*x

F(x)=(x*x*x)/x+1,78-0,85-0,2*x+0,1

F(x)=(x*x*x)/x+1,78-0,85-0,2*x+0.1+0,18*x

Prerastanie stromov

Prerastaniu stromov je možné zamedziť viacerými spôsobmi. Tie sú často závislé od aplikácie.

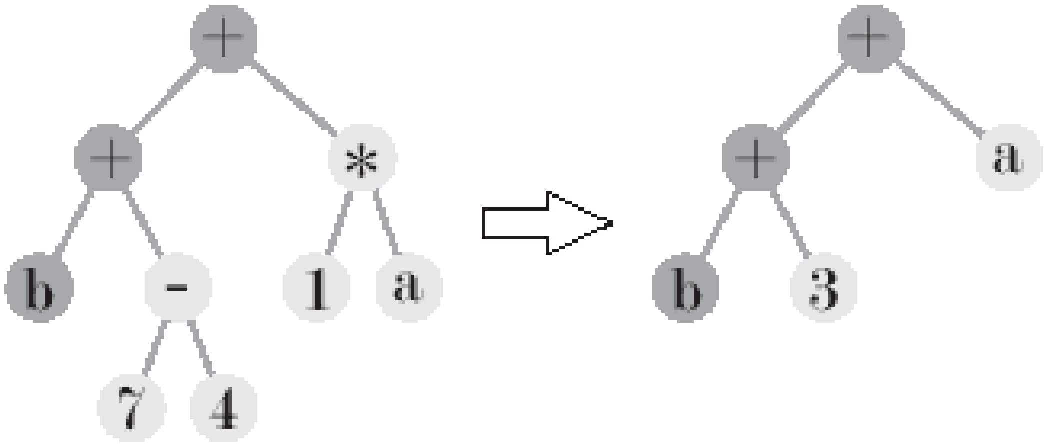
1. Základným a najjednoduchším spôsobom je obmedzenie veľkosti jedincov: - hĺbky stromu
   - počtu uzlov

   Tento spôsob nemusí byť vždy účinný.

2. Ďalším opatrením môže byť pokutovanie reťazcov vzhľadom na ich dĺžku.

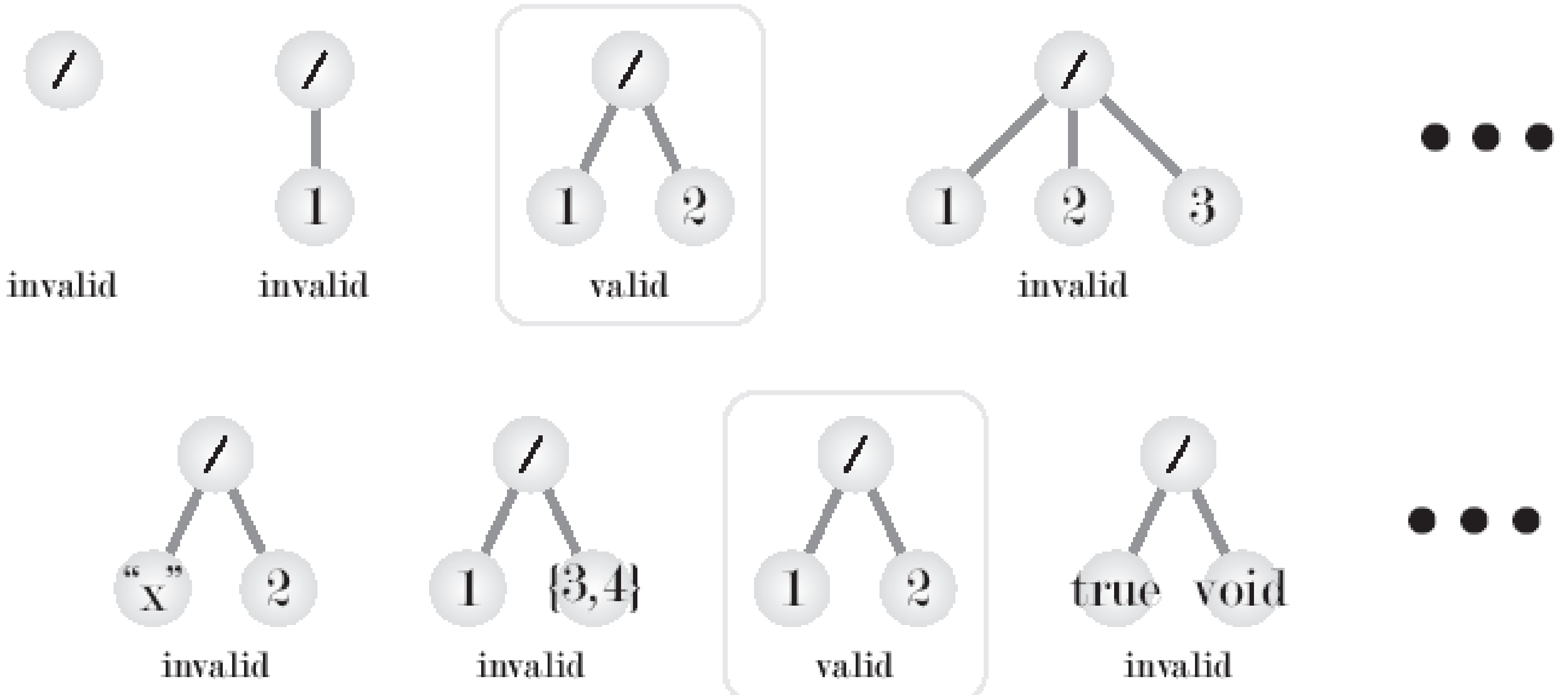
3. Editácia, zjednodušenie – úprava, zjednodušovanie reťazcov, odstraňovanie zbytočných, neúčinných častí reťazca.

Editácia, zjednodušovanie



Kontrola jedincov

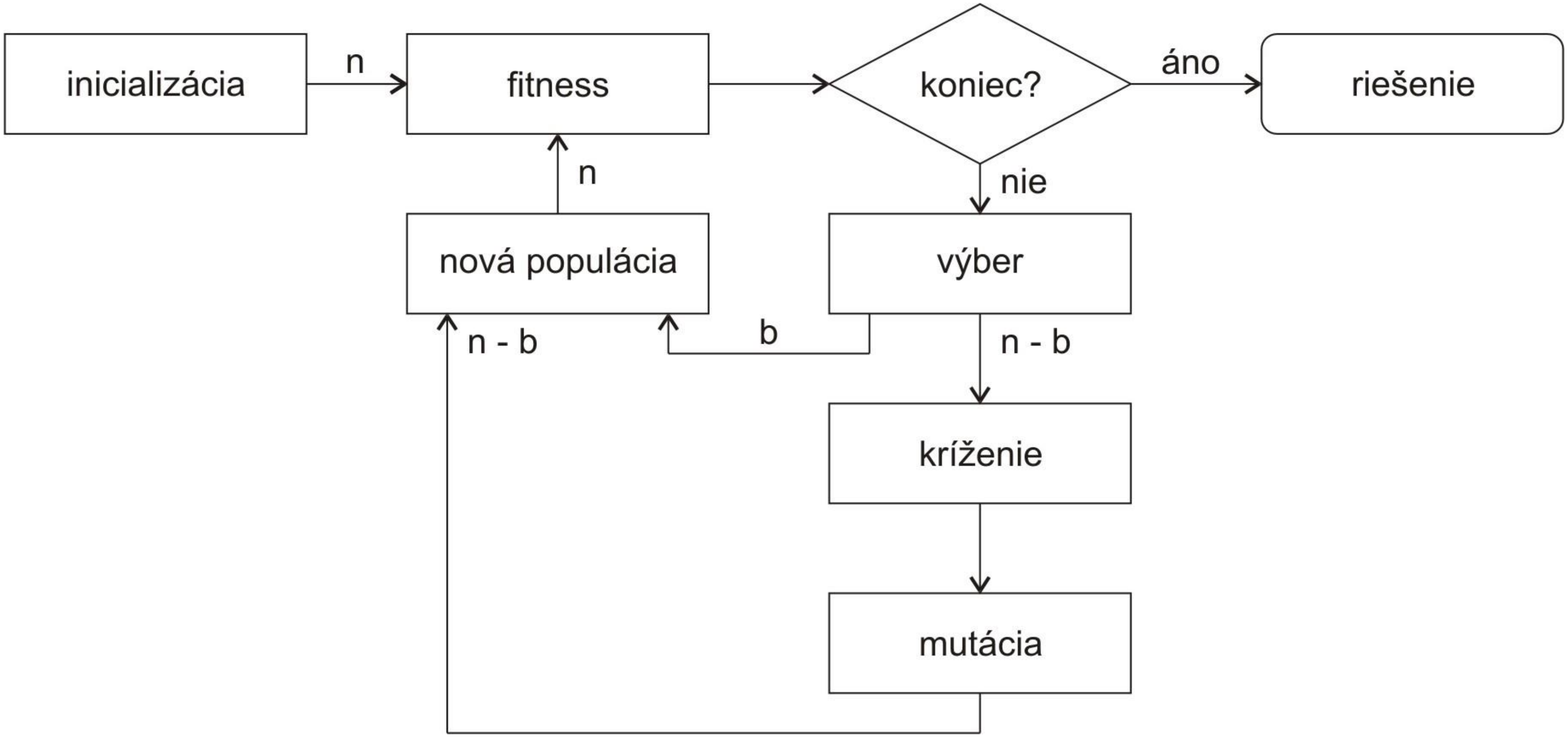
Po modifikácii jedincov (mutácii, krížení) vznikajú aj neprípustné, neplatné, neúčinné jedince. Tie môžu komplikovať vyhodnotenie fitness. Preto je vhodné, niekedy aj potrebné takéto jedince odstraňovať.

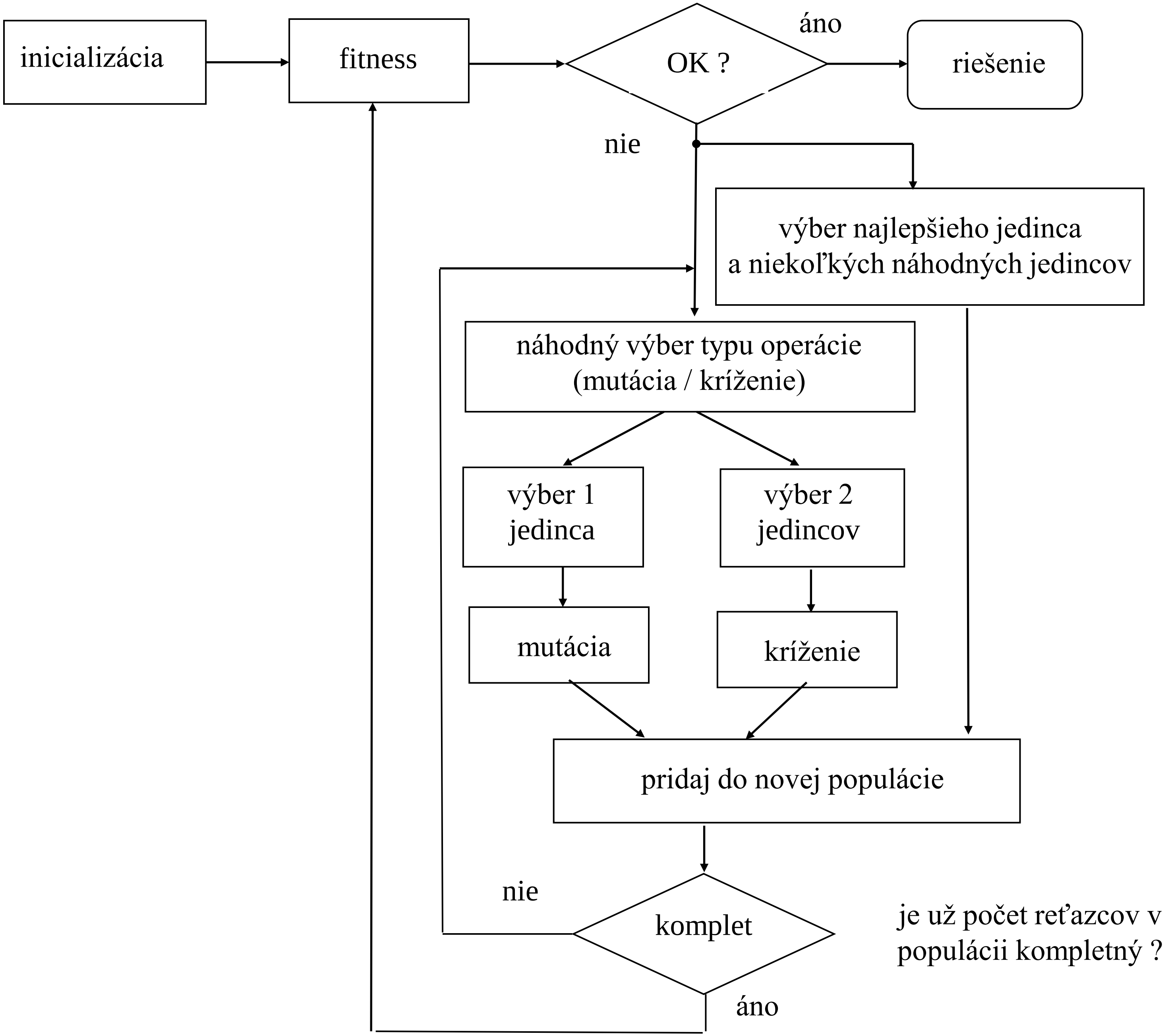


Niektoré prístupy na odstránenie neprípustných jedincov

1. Kompenzácia zakázaných operácii počas ich vyhodnocovania (v priebehu výpočtu fitness)
   Napr. pri delení bez deliteľa je výsledok =1
   Pri delení nulou bude operácia ignorovaná...
2. Opravný algoritmus skontroluje všetkých jedincov po reprodukcii (po ich modifikácii).
3. Doplnenie operácií modifikácií (mutácie a kríženia) tak, aby nemohli vzniknúť neprípustní potomkovia.
4. Použitím vhodného kódovania jedincov môžeme zamedziť vzniku neprípustných jedincov (podľa aplikácie).

Algoritmus GP


inicializácia
n
fitness
koniec?
áno
riešenie
n
nie
nová populácia
výber
n - b
b
n - b
kríženie
mutácia


inicializácia
fitness
OK ?
áno
riešenie
nie
výber najlepšieho jedinca
a niekoľkých náhodných jedincov
náhodný výber typu operácie
(mutácia / kríženie)
výber 1
jedinca
výber 2
jedincov
mutácia
kríženie
pridaj do novej populácie
nie
komplet
áno
je už počet reťazcov v
populácii kompletný ?

Aplikácie GP

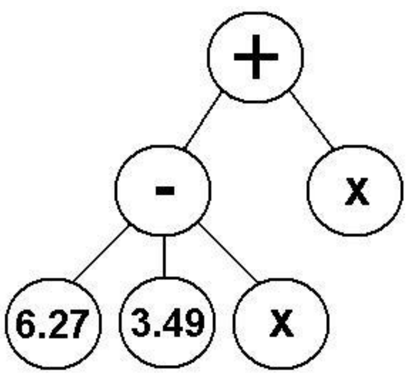
Symbolická regersia

<u>Príklad</u>

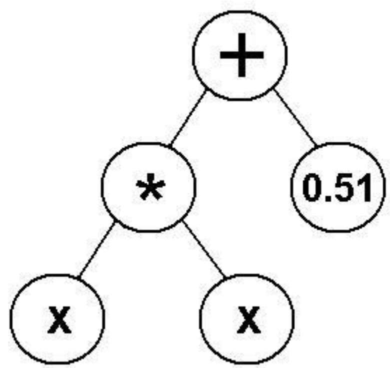
vzor: $f(x)=x^2+1$

$X=\{-2,-1,0,1,2\}$

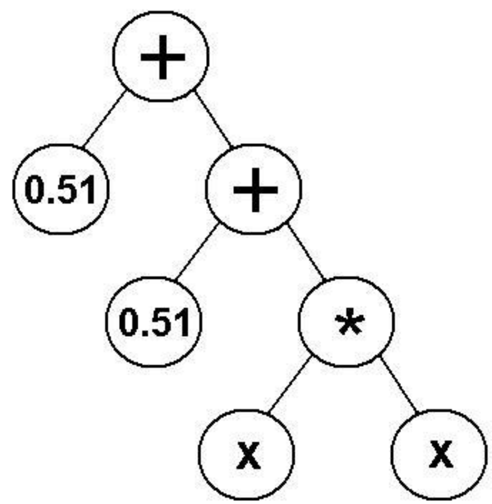
$f(x)=\{5,2,1,2,5\}$



1.generácia



50.generácia



100.generácia

Symbolická regersia

<u>Príklad</u> *vzor:* $f(x) = x^2 + 5\sin(2x) + 2$

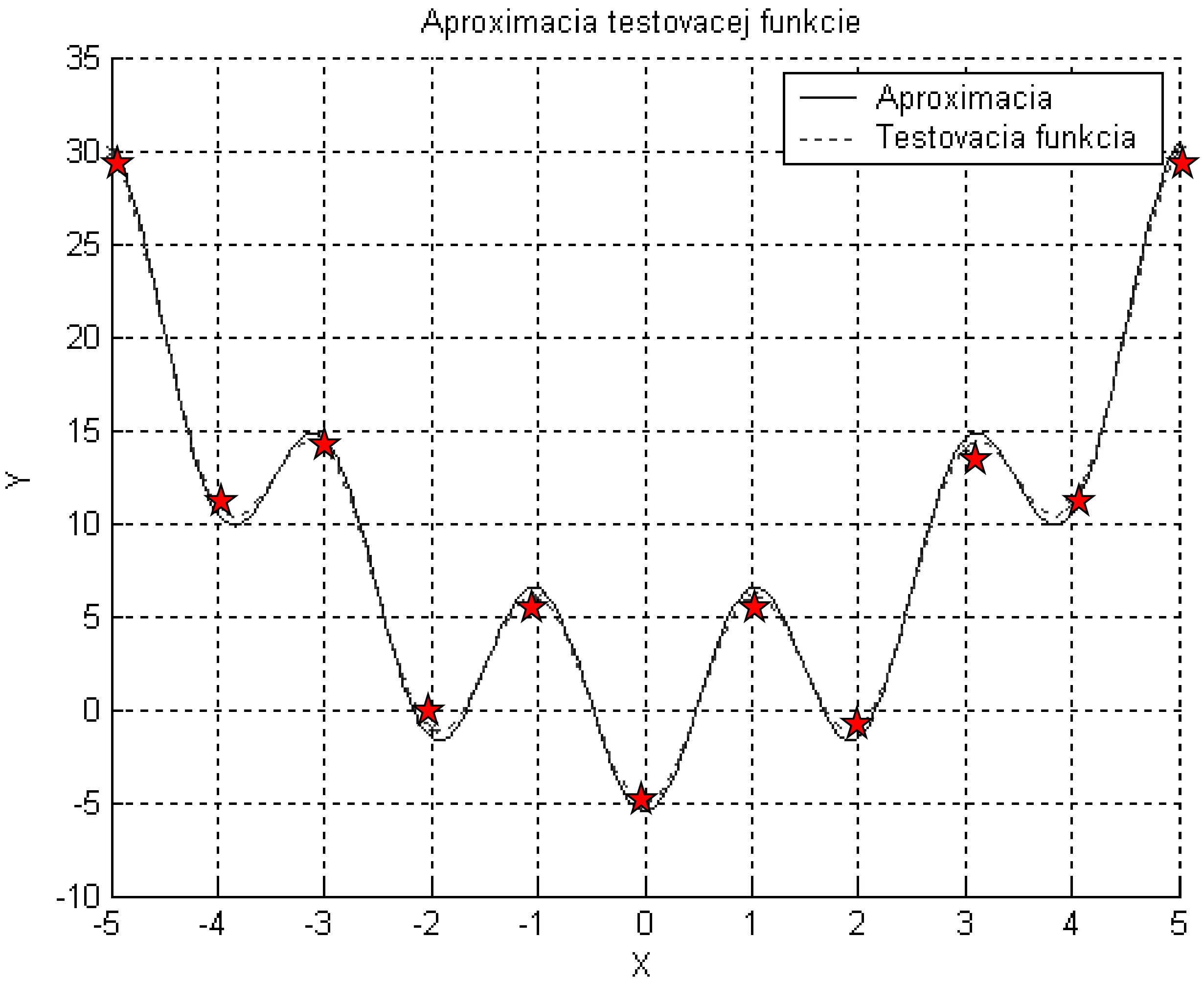
gény: +,-,*,/, sin(x), cos(x)
reálne čísla z intervalu od *0* do *10*

$$F(x) = 4.179 * \sin(x + x) + (0.8165 * 0.7806 * 3.1551) + x * x$$

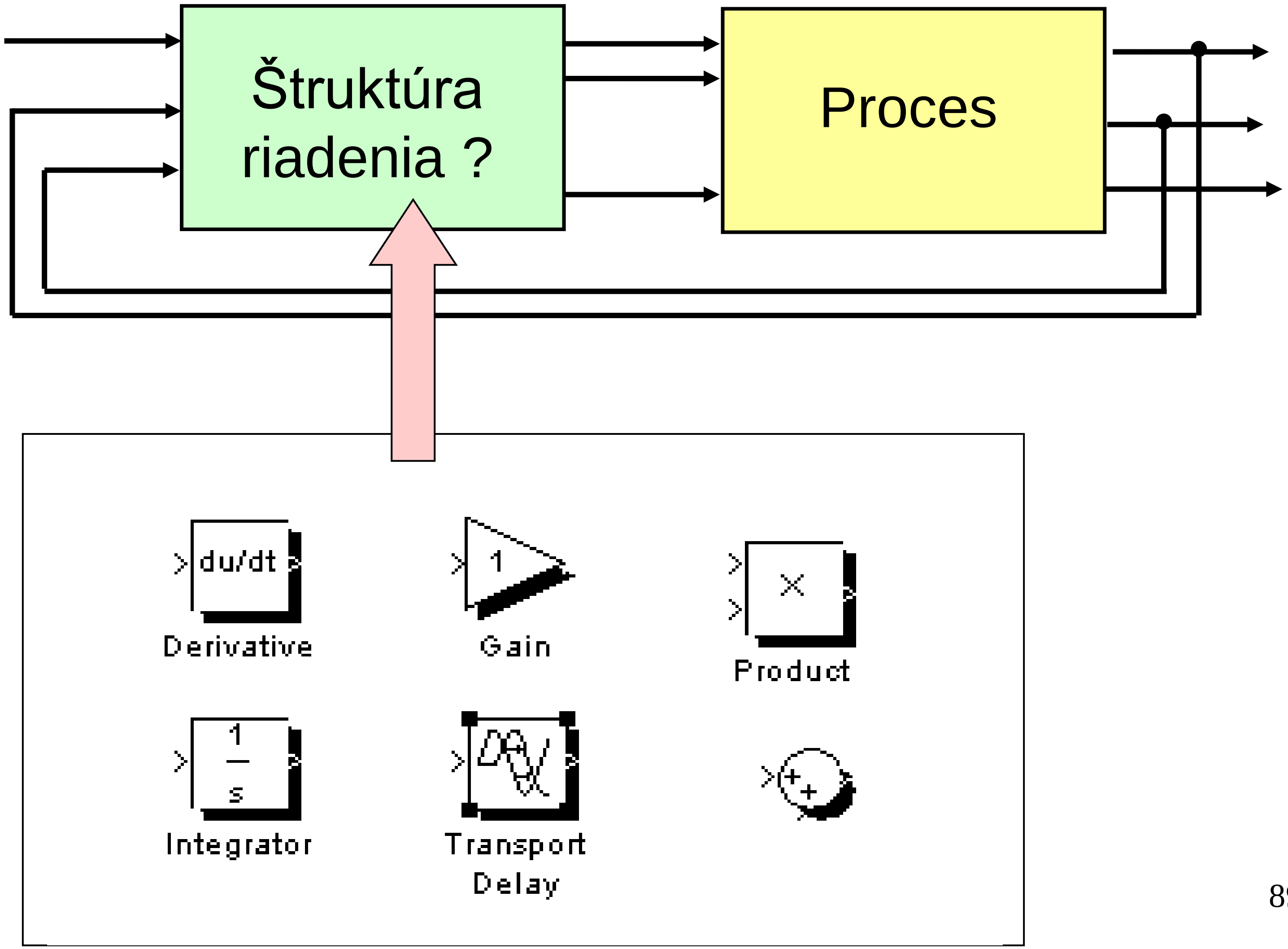
$$F(x) = x^2 + 4.9179 * \sin(2x) + (2.0109)$$

Symbolická regresia

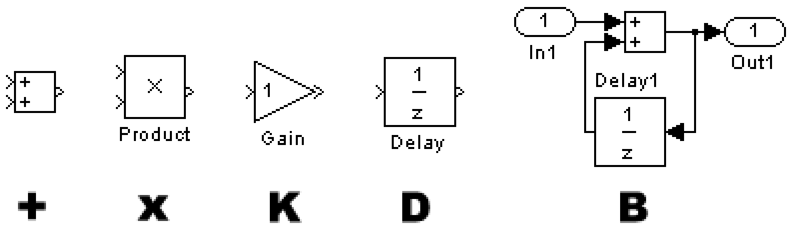
$$f(x) = x^2 + 5\sin(2x) + 2$$



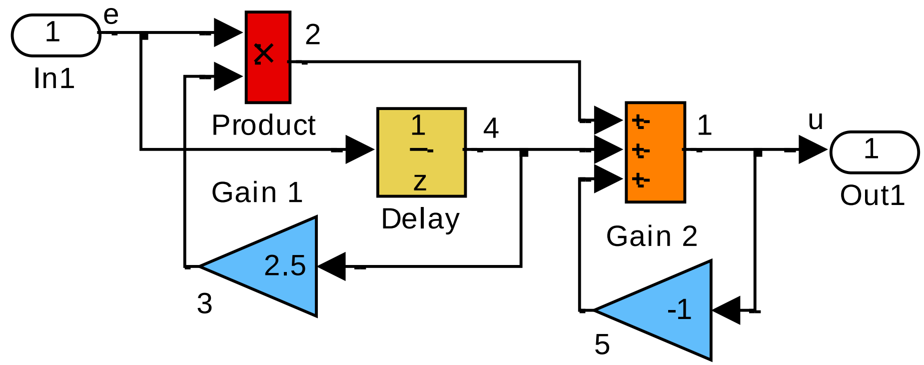
Návrh / optimalizácia štruktúry dynamického systému, (regulátora) jeho vnútorných väzieb a prvkov



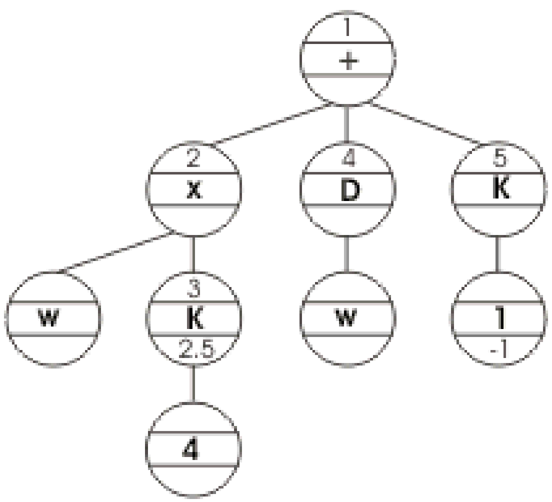
Evolúcia regulátora



Stavebné prvky



Regulátor v Simulinku - fenotyp

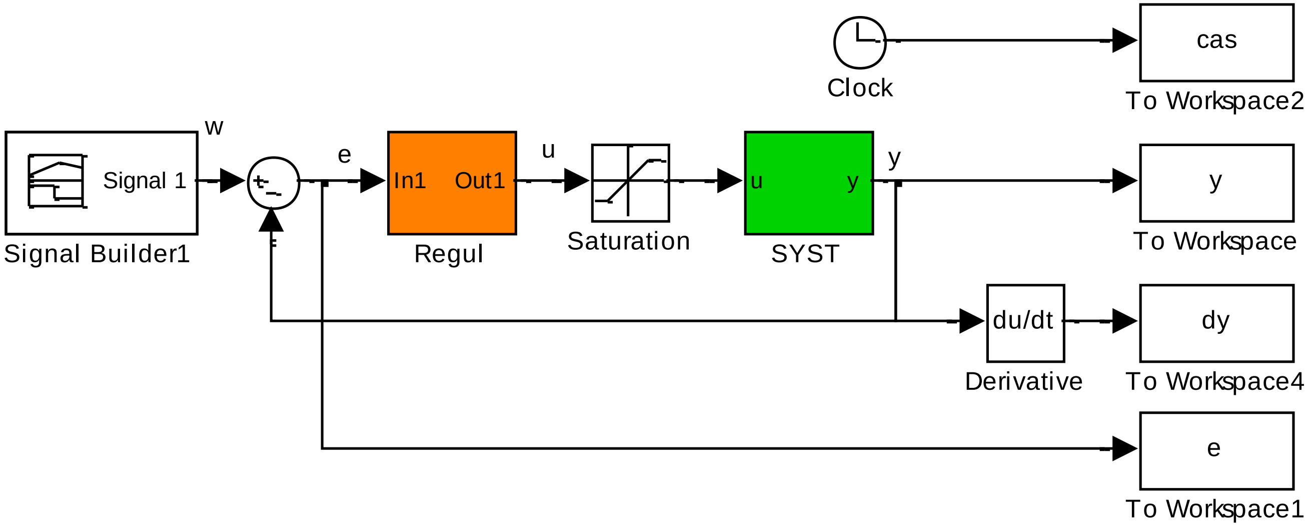


Stromová reprezentácia

$$\begin{pmatrix} 3 & 2 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ + & x & w & K & 4 & D & w & K & 1 \\ 1 & 2 & & 3 & & 4 & & 5 & \\ & & & 2.5 & & & & -1 & \end{pmatrix}$$

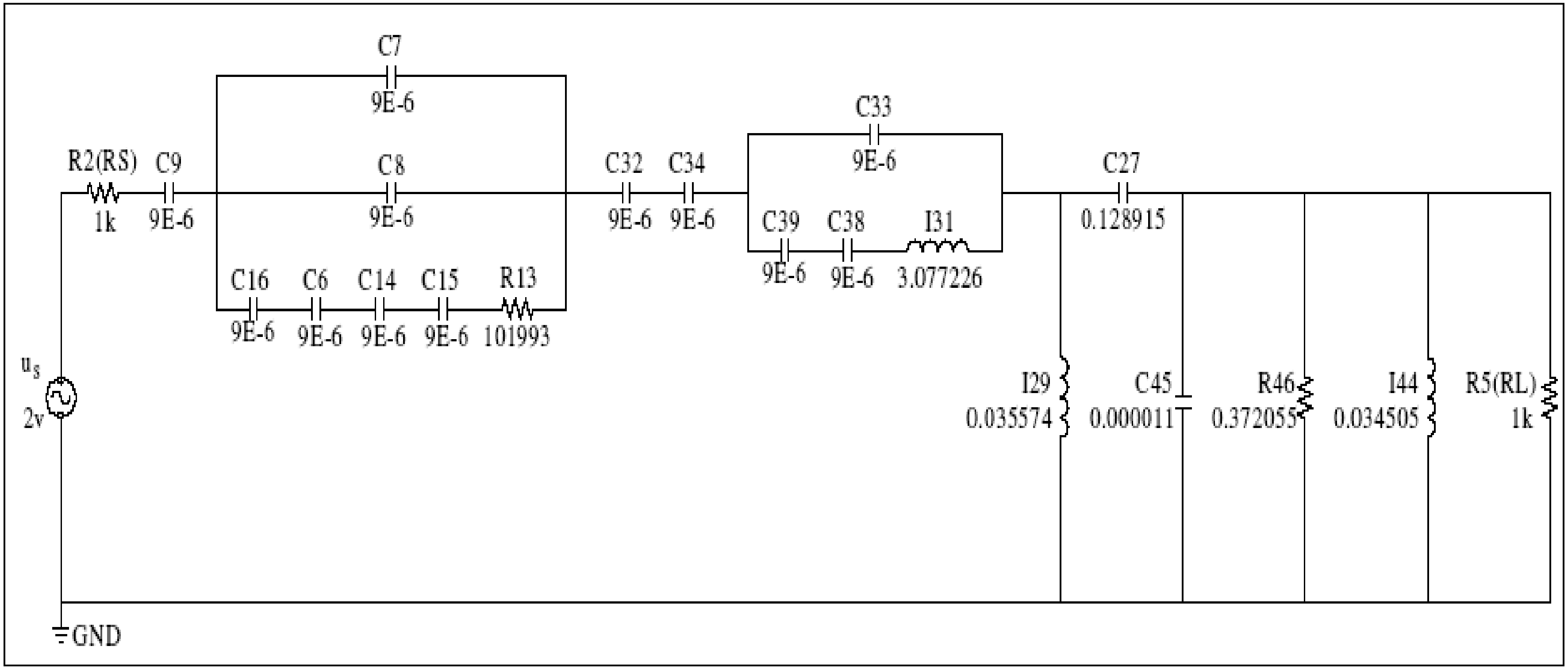
Tabulková reprezentácia - genotyp

Vyhodnotenie fitness regulátora - simulácia v Simulinku

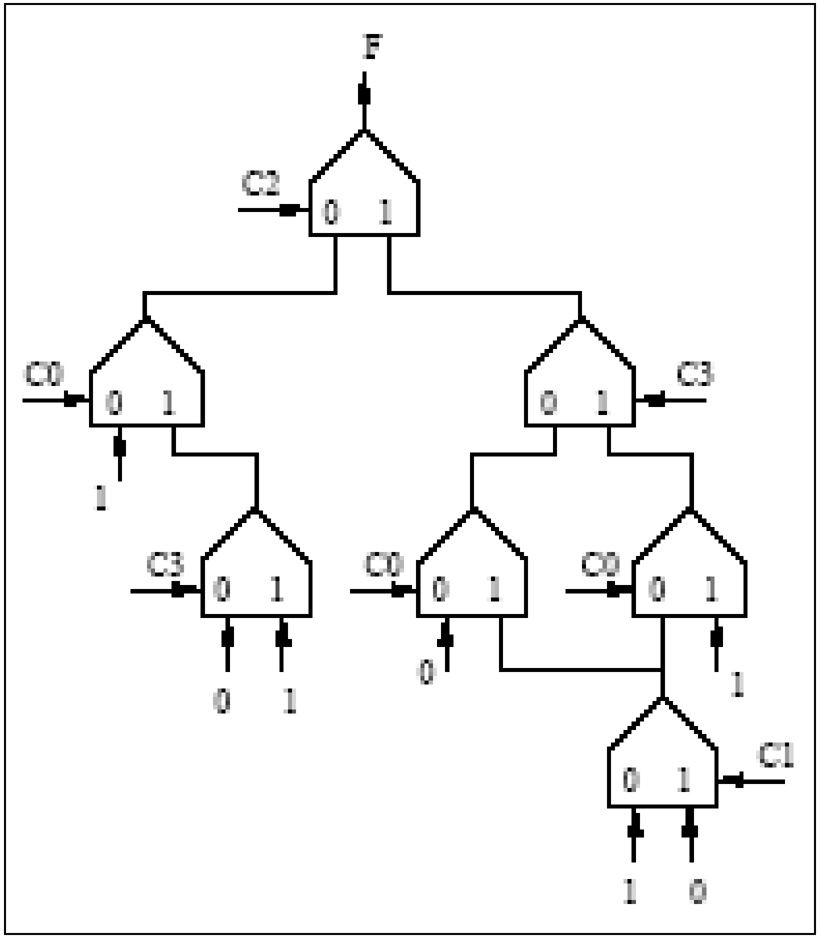


Fitness: $$J_{IAE} = \int_0^T |e(t)|dt \to \min$$

Evolúcia elektrických obvodov

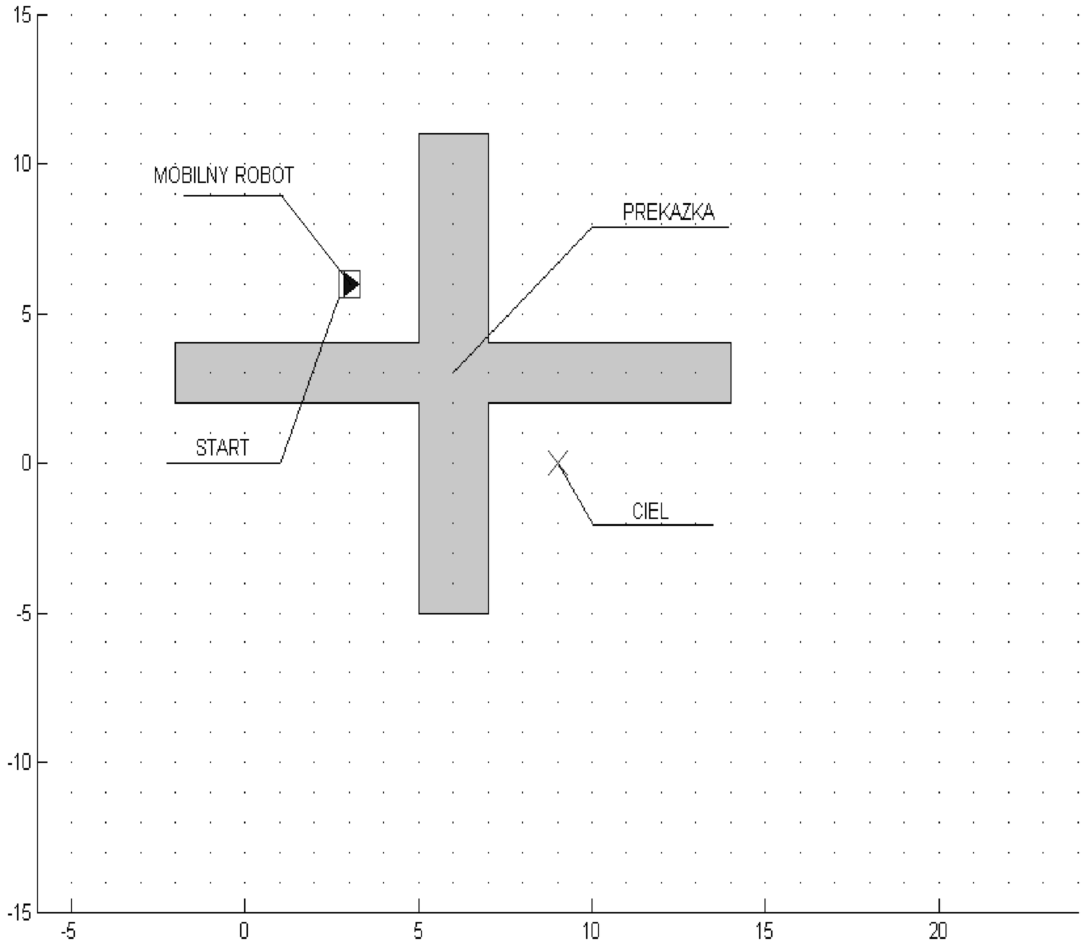


Evolúcia logických obvodov



Príklad ...

Evolúcia riadiaceho programu mobilného robota

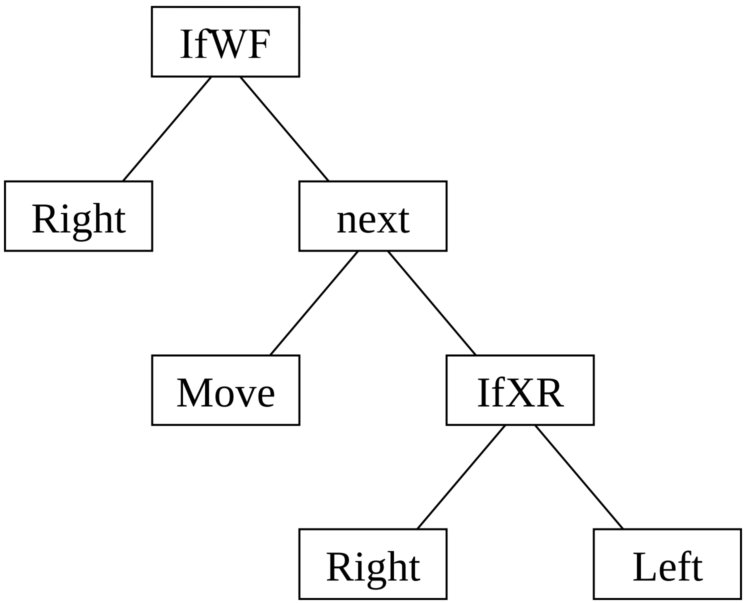


Inštrukčný súbor robota

	Symbol inštrukcie (gén)	Popis inštrukcie
Vetviace inštrukcie	*IfWF, IfWR, IfWL*	Ak je prekážka lokalizovaná snímačmi vpredu, napravo, naľavo od robota.
	IfXF, IfXR, IfXL	Ak sa nachádza cieľ smerom vpredu, napravo, naľavo od robota.
	Prog2	Sekvenčne vykonaj inštrukcie ľavého a potom pravého podstromu.
Výkonné inštrukcie	*Move*	Pohyb o jeden krok v smere natočenia robota.
	Right	Otočenie robota doprava o 90°.
	Left	Otočenie robota doľava o 90°.

Príklad riadiaceho programu a jeho stromová reprezentácia

```
IfWF
   Right
else
    Move
    IfXR
        Right
    else
        Left
    end
end
```



Výsledná trajektória robota

```
IfWF
    Right
           IfXF
        IfWL
                    IfWF
                  Left
            else
                IfWF
                    Move
                    Left
                else
                    Move
                end
                Left
            end
        else
            Move
            Right
        end
    else
        Move
    ...
```