GENETICKÉ PROGRAMOVANIE

Kódovanie syntaktických stromov a ich využitie v genetickom programovaní

Kódovanie syntaktických stromov

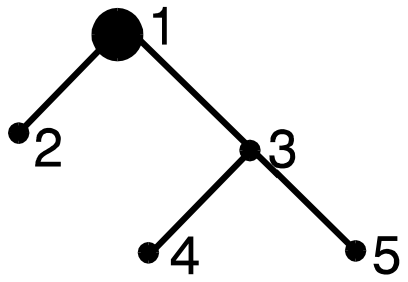
- Nech Readov lineárny kód stromu ***T*** je code(***T***)=**α**=(α_1,α_2,...,α_p),
- potom lineárny kód syntaktického stromu ***t*** môže byť zostrojený tak, že Readov kód koreňového stromu je rozšírený o zobrazenie ϕ

$$\text{code}(\mathbf{t}) = ((\alpha_1, \phi_1), (\alpha_2, \phi_2), ..., (\alpha_p, \phi_p))$$

- kde ϕ_i je ohodnotenie *i*-teho vrcholu buď funkcionálnym alebo terminálovým symbolom.

Kódovanie syntaktických stromov

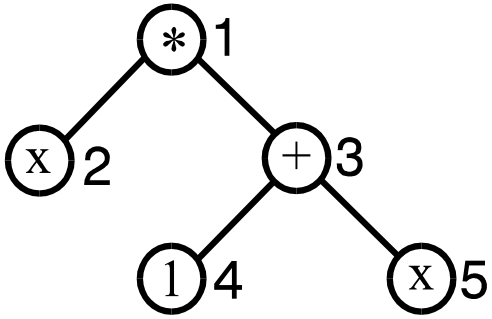
koreňový strom ***T***



code(***T***)=(20200)

A

syntaktický strom ***t***



code(***t***)=((2,*),(0,x),(2,+),(0,1),(0,x))

B

Úloha regresnej analýzy

- Majme tréningovú množinu obsahujúcu *n* bodov

$$A_{train} = \{x_i, y_i; i = 1,2,...,n\}$$

- Cieľom štandardnej regresnej analýzy je nájsť také optimálne parametre modelovej funkcie $G(x; \boldsymbol{w})$, kde $\boldsymbol{w}$ sú parametre funkcie G, také, že nasledujúca účelová funkcia je minimalizovaná

$$E(w) = \sum_{i=1}^{n} |G(x_i; w) - y_i|$$

Úloha regresnej analýzy

- Táto funkcia má minimum v bode $w_{opt} = \underset{w}{\operatorname{argmin}} E(w)$
- Hovoríme, že adaptovaná funkcia $G(\boldsymbol{x}, w_{opt})$ modeluje tréningovú množinu A_{train}
- Symbolická regresia ide ďalej ako štandardná regresná analýza, hľadá v množine T takú funkciu, že nasledujúca účelová funkcia (funkcionál) je minimalizovaná

$$E(t) = \sum_{i=1}^{n} |t(x_i) - y_i|$$

- pričom táto účelová funkcia má minimum v "bode"

$$t_{opt} = \underset{t \in T}{\operatorname{argmin}} E(t)$$

Rekonštrukcia stromov s požadovanou vlastnosťou

- Verzia genetického programovania, ktorá je schopná rekonštruovať stromy (súvislé acyklické grafy) s požadovanými vlastnosťami.

- **Pod vlastnosťou stromu** budeme rozumieť reálne číslo, ktoré je priradené stromu ***G*** z množiny prípustných stromov T, formálne $t: \mathcal{T} \rightarrow R$.
- Úloha rekonštrukcie spočíva v tom, že hľadáme v množine prípustných stromov T taký strom ***G***, ktorého vlastnosť $t(\mathbf{G})$ je blízka požadovanej vlastnosti t_{req}. Definujme účelovú funkciu

$$E(\mathbf{G}) = \left| t(\mathbf{G}) - t_{req} \right|$$

Rekonštrukcia stromov s požadovanou vlastnosťou

- Optimálny strom, ktorého vlastnosť minimalizuje funkcionál je určený ako riešenie nasledujúceho minimalizačného problému

$$\mathbf{G}_{opt} = \underset{\mathbf{G}\in\mathcal{T}}{\operatorname{argmin}} E(\mathbf{G})$$

- Riešenie tohto optimalizačného problému sa realizuje pomocou genetického algoritmu nad populáciou stromov, ktoré sú kódované pomocou Readovho lineárneho kódu a operácie mutácie a kríženia sa vykonávajú spôsobom popísaným ďalej.

Topologické indexy

- Akým spôsobom popísať vlastnosti grafu $\boldsymbol{G}=(V,E)$? Najjednoduchší spôsob je pomocou tzv. *topologických indexov* $\chi_1(\boldsymbol{G})$, $\chi_2(\boldsymbol{G})$,...., ktoré sú charakterizované ako funkcie - zobrazenia priraďujúce grafu reálne číslo.

- Medzi najznámejšie topologické indexy patria:

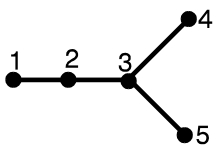
(1) *Wienerov topologický index*

(2) *Randičov topologický index*

Wienerov topologický index

- $$\chi_W(\boldsymbol{G}) = \sum_{i<j} d_{ij}$$
- kde suma obsahuje všetky rôzne dvojice vrcholov a d_{ij} je vzdialenosť medzi *i*-tým a *j*-tým vrcholom v grafe ***G.***

Schematické znázornenie výpočtu Wienerovho indexu pre jednoduchý strom znázornený diagramom A. Diagram B obsahuje maticu vzdialeností D, suma jej zložiek pod diagonálou tvorí Wienerov index. Nájdite chybu.



A

$$D=\begin{pmatrix} 0 & & & & \\ 1 & 0 & & & \\ 3 & 2 & 1 & 0 & \\ 3 & 2 & 1 & 2 & 0 \end{pmatrix} \begin{matrix} \ldots 0 \\ \ldots 1 \\ \ldots 6 \\ \ldots 8 \end{matrix}$$

$\chi_w=18$

B

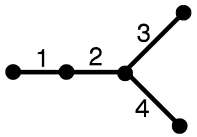
Randičov topologický index

- $$\chi_R(\boldsymbol{G}) = \sum_{[v,v']\in E} \frac{1}{\sqrt{\mathrm{val}(v)\cdot \mathrm{val}(v')}}$$

kde suma obsahuje všetky hrany $[v, v'] \in E(\boldsymbol{G})$

Výpočet Randičovho topologického indexu pre strom znázornený diagramom A, kde jednotlivé hrany stromu sú indexované. V diagrame B je priradený ku každej hrane odpovedajúci výraz

$$1/\sqrt{\mathrm{val}(v)\cdot \mathrm{val}(v')}$$



A

$$\chi_R = \overset{e_1}{\frac{1}{\sqrt{1\cdot 2}}} + \overset{e_2}{\frac{1}{\sqrt{2\cdot 3}}} + \overset{e_3}{\frac{1}{\sqrt{1\cdot 3}}} + \overset{e_4}{\frac{1}{\sqrt{1\cdot 3}}}$$

B

Lineárne stromy

- Pre lineárne stromy (t.j. taký strom, ktorý neobsahuje tzv. vetviace vrcholy, každý vrchol je buď valencie 1 alebo 2) jednoduchými úvahami (matematickou indukciou) je možné zostrojiť explicitné výrazy pre Wienerov a Randičov topologický index

$$\chi_W = \frac{p\left(p^2-1\right)}{6}\left(\text{pre}\, p \geq 1\right)$$

$$\chi_R = \sqrt{2} + \frac{p-3}{2}\left(\text{pre}\, p \geq 3\right)$$

Kombinácia top. indexov

- Postulujme, že vlastnosť $t(\mathbf{G})$ grafu $\mathbf{G}$ je popísaná ako konvexná kombinácia Wienerovho a Randičovho indexu

$$t(\mathbf{G}) = \omega\chi_R(\mathbf{G}) + (1-\omega)\chi_W(\mathbf{G})$$

- pre $0 \leq \omega \leq 1$. Táto "vlastnosť" pre lineárne grafy má tvar

$$t(\mathbf{G}) = \omega\left(\sqrt{2} + \frac{p-3}{2}\right) + (1-\omega)\left(\frac{p(p^2-1)}{6}\right)$$

Randičov Index

```
function Randic_topological_index(α):real;
begin χ:=0; branch_0:=1; val_0:=0; d:=0; i:=1;
    while d>=0 do
    if branch_d>0 then
    begin branch_d:=branch_d-1; i:=i+1; d:=d+1;
        branch_d:=i; val_d:=α_i+1;
        χ:=χ+1/sqrt(val_{d-1}*val_d)
    end else d:=d-1;
    Randic_topological_index:=χ;
end;
```

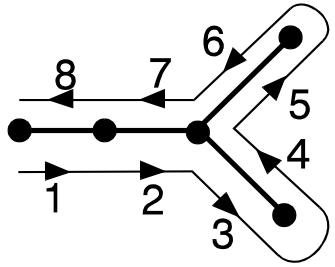
Algoritmus. Pseudopascalovská implementácia výpočtu Randičovho indexu stromu zadaného pomocou Readovho lineárneho kódu, tento algoritmus je jednoduchou modifikáciou backtrack algoritmu.

Wiennerov index

function Wiener_topological_index(α):real;
begin χ:=0; $branch_0$:=1; $index_0$:=1; d_{11}:=0; d:=0; i:=1;
while d>=0 do
if $branch_d$>0 then
begin $branch_d$:=$branch_d$-1; i:=i+1; d:=d+1;
$branch_d$:=i; $index_d$:=i;
for j:=1 to i-1 do
begin $d_{ij} := d_{index_{d-1},j} + 1$; d_{ij}:=d_{ji}; χ:=χ+d_{ij} end;
d_{ii}:=0;
end else d:=d-1;
Wiener_topological_index:=χ;
end;

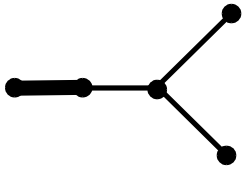
Algoritmus. Pseudopascalovská implementácia výpočtu Wienerovho indexu stromu zadaného pomocou Readovho lineárneho kódu, tento algoritmus je jednoduchou modifikáciou backtrack analýzy. V priebehu analýzy kódu sa postupne zostrojí matica vzdialeností (d_{ij}).

V priebehu analýzy koreňového stromu (diagram A) sú identifikované hrany grafu. V priebehu každého kroku tejto identifikácie vykoná sa čiastočný výpočet Randičovho a Wienerovho indexu (diagram B).



A

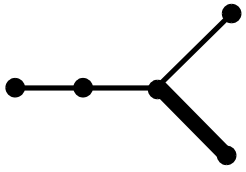
Krok 1



$$\chi_R = \chi_R + \frac{1}{\sqrt{1 \cdot 2}}$$

$$\chi_W = \chi_W + 1$$

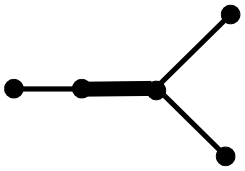
Krok 3



$$\chi_R = \chi_R + \frac{1}{\sqrt{1 \cdot 3}}$$

$$\chi_W = \chi_W + 3 + 2 + 1$$

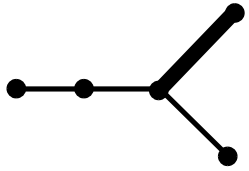
Krok 2



$$\chi_R = \chi_R + \frac{1}{\sqrt{2 \cdot 3}}$$

$$\chi_W = \chi_W + 2 + 1$$

Krok 5

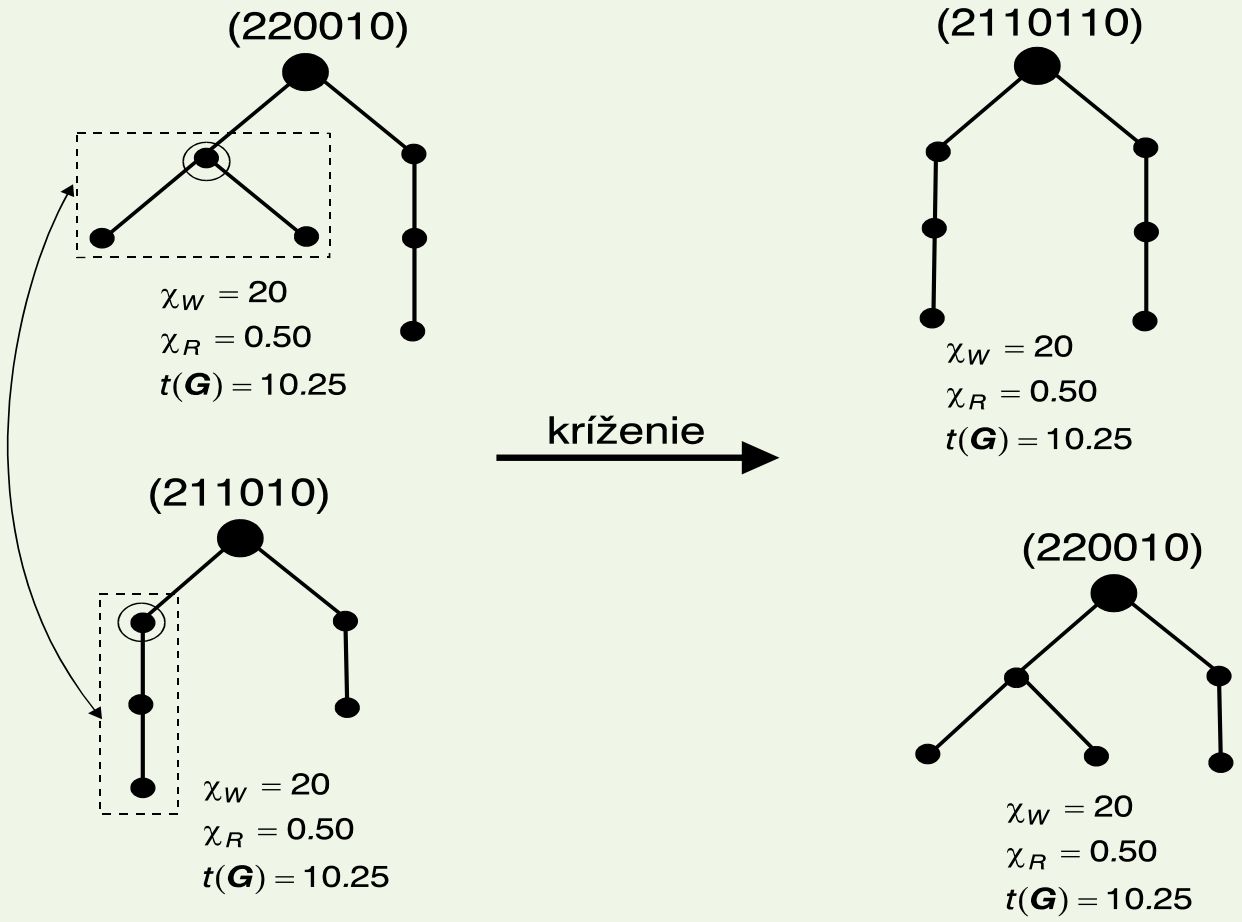


$$\chi_R = \chi_R + \frac{1}{\sqrt{1 \cdot 3}}$$

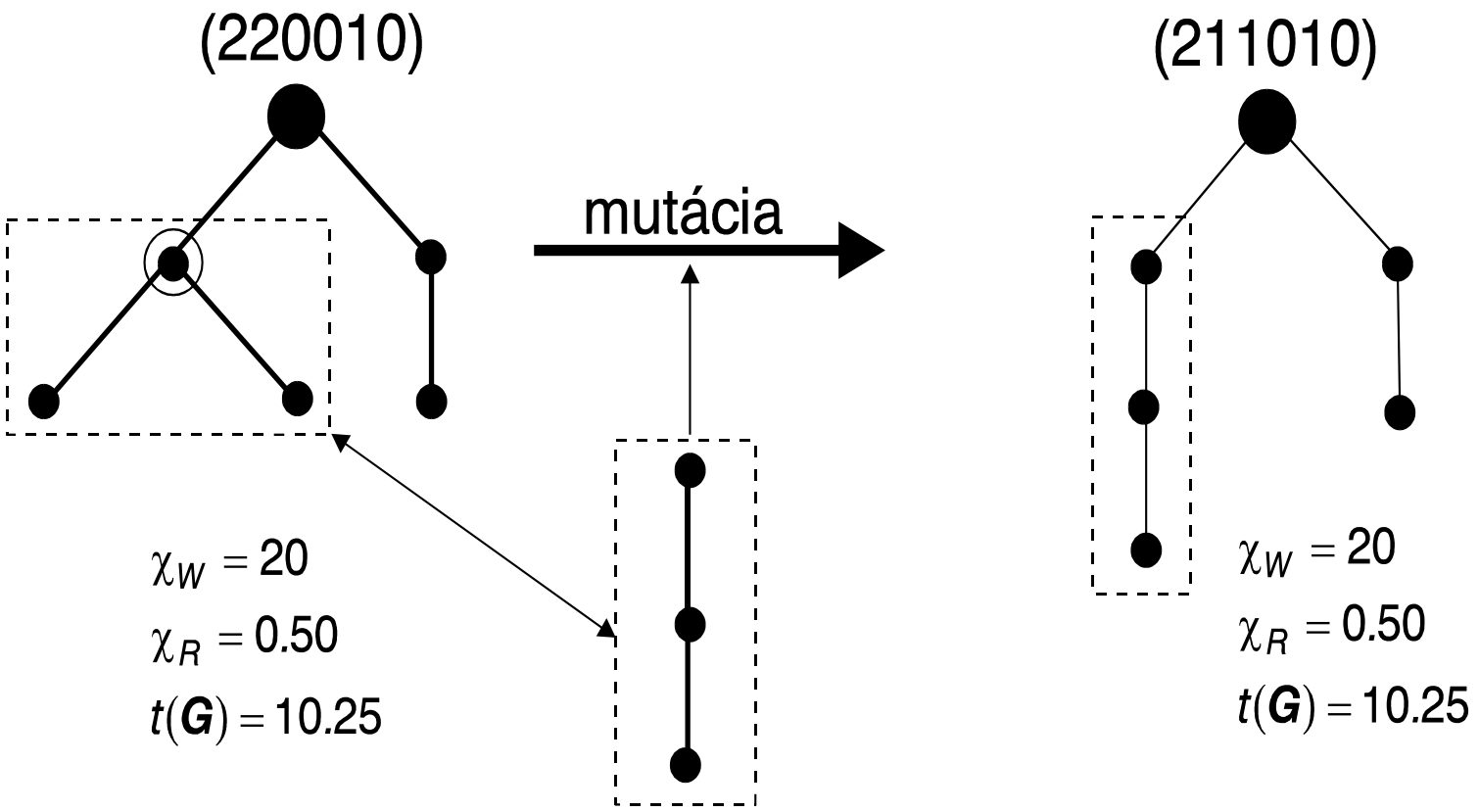
$$\chi_W = \chi_W + 3 + 2 + 1 + 2$$

B

Ilustračný príklad kríženia dvoch koreňových stromov. V každom strome je náhodne vybraný vrchol (zakrúžkovaný), príslušné podstromy si koreňové stromy medzi sebou vymenia. Príslušné hodnoty topologických indexov a odpovedajúcich vlastností sú uvedené pod grafmi.

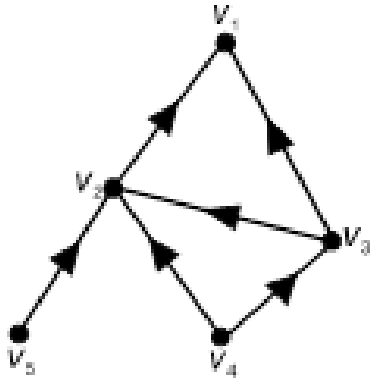


Ilustračný príklad mutácie koreňového stromu, podstrom susedný s náhodne vybraným vrcholom (zakrúžkovaný) sa vymení za náhodne vygenerovaným podstromom. Hodnoty topologických indexov a príslušná vlastnosť sú uvedené pod grafmi.



Kódovanie funkcií pomocou acyklických orientovaných grafov

- alternatívny prístup ku kódovaniu funkcií pomocou acyklických orientovaných grafov, ktoré môžu byť chápané ako zovšeobecnenie koreňových stromov



$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

- Nájdite chybu v matici A.
- *Indexovanie* vrcholov orientovaného grafu G je realizované pomocou zobrazenia φ, ktoré priradí 1-1-značne každému vrcholu celé číslo $\varphi: V \rightarrow \{1,2,\ldots, p\}$

Kódovanie funkcií pomocou acyklických orientovaných grafov

Veta. Orientovaný graf ***G***=(*V*,*E*) je acyklický vtedy a len vtedy, ak jeho vrcholy môžu byť indexované tak, že platí

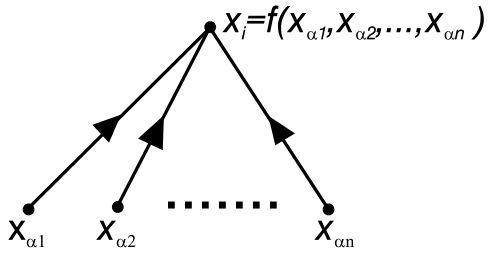
$$\forall(v,v') \in E : \varphi(v) > \varphi(v')$$

- Indexovanie φ, ktoré vyhovuje podmienke sa nazýva ***kanonické indexovanie***. Každý orientovaný acyklický graf môže byť kanonicky indexovaný.
- Jednoduchý ilustračný príklad kanonického indexovania orientovaného acyklického grafu je uvedený na obr., kde vrcholy sú už tak označené, aby podmienka bola splnená. Vrcholy kanonicky indexovaného grafu môžu byť rozdelené na tri disjunktné množiny:
- (1) *Vstupné vrcholy*, tieto vrcholy susedia len s vychádzajúcimi hranami,
- (2) *Prechodné vrcholy*, tieto vrcholy súčasne susedia tak s vychádzajúcim, ako aj s vchádzajúcou hranou.
- (3) *Výstupné vrcholy*, tieto vrcholy susedia len s vchádzajúcimi hranami.

Syntaktické grafy

- Kanonicky indexovaný orientovaný acyklický graf sa nazýva ***syntaktický graf***.
- Tento druh orientovaných grafov má veľký význam pre implementáciu symbolickej regresie, pretože tieto grafy môžu slúžiť ako efektívna reprezentácia funkcií - programov.

Každý prechodný a výstupný vrchol syntaktického grafu je ohodnotený funkčnou hodnotou s argumentmi odpovedajúcim funkčným hodnotám vrcholov, ktoré sú incidentné s hranami vychádzajúcimi z nich a vchádzajúcimi do uvažovaného vrcholu. Vrcholy, ktoré sú vstupné sú ohodnotené konštantami.

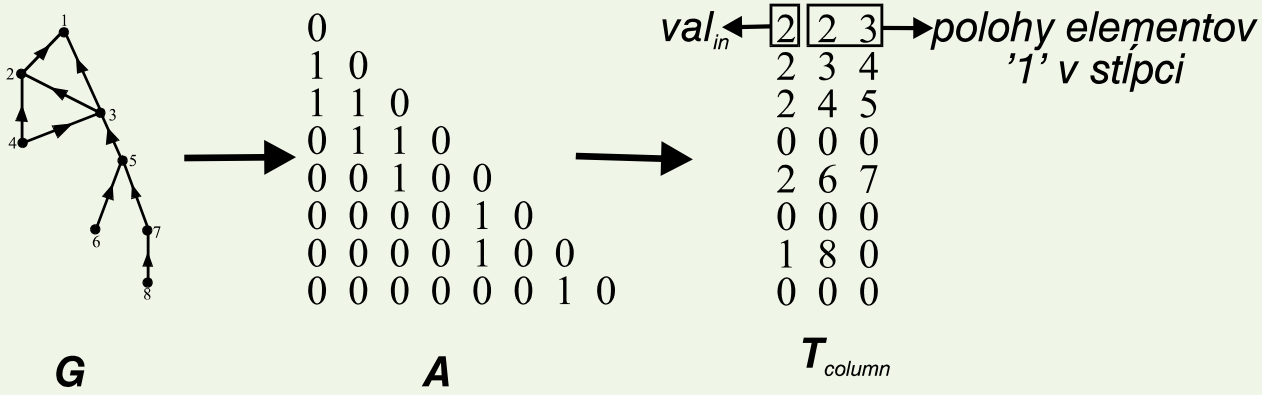


Syntaktické grafy

- Graf ***G*** je ***syntaktický graf*** vtedy a len vtedy, ak matica susednosti ***A*** je dolno-trojuholníkovou maticou, pričom každý riadok až na prvý obsahuje aspoň jeden jednotkový prvok '1'.
- Podmienka, že v každom riadku až na prvý je aspoň jeden element '1' odpovedá podmienke, že syntaktický graf obsahuje práve jeden výstupný vrchol.
- Ak v stĺpci pod diagonálou sú len nulové elementy '0', potom v odpovedajúci vrchol je vstupný (t.j. neobsahuje predchodcov).
- Počet orientovaných hrán je určený pomocou vstupných (počet hrán, ktoré vchádzajú do vrcholu) a výstupných valencií (počet hrán, ktoré vzchádzajú z vrcholu) všetkých vrcholov grafu
- $$q = \sum_{v \in V} val_{in}(v) = \sum_{v \in V} val_{out}(v)$$

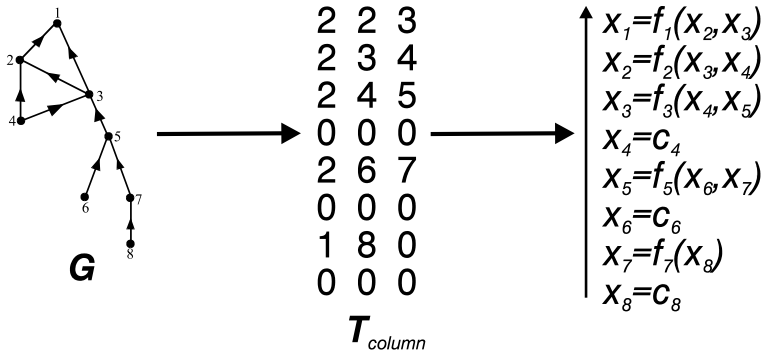
Syntaktické grafy

Ilustračný príklad kódovania syntaktického grafu pomocou stĺpcovej tabuľky. Syntaktický graf G je kódovaný maticou susednosti A, táto matica je v ďalšom kroku "kondenzovaná" do tvaru stĺpcovej tabuľky T_{column}.



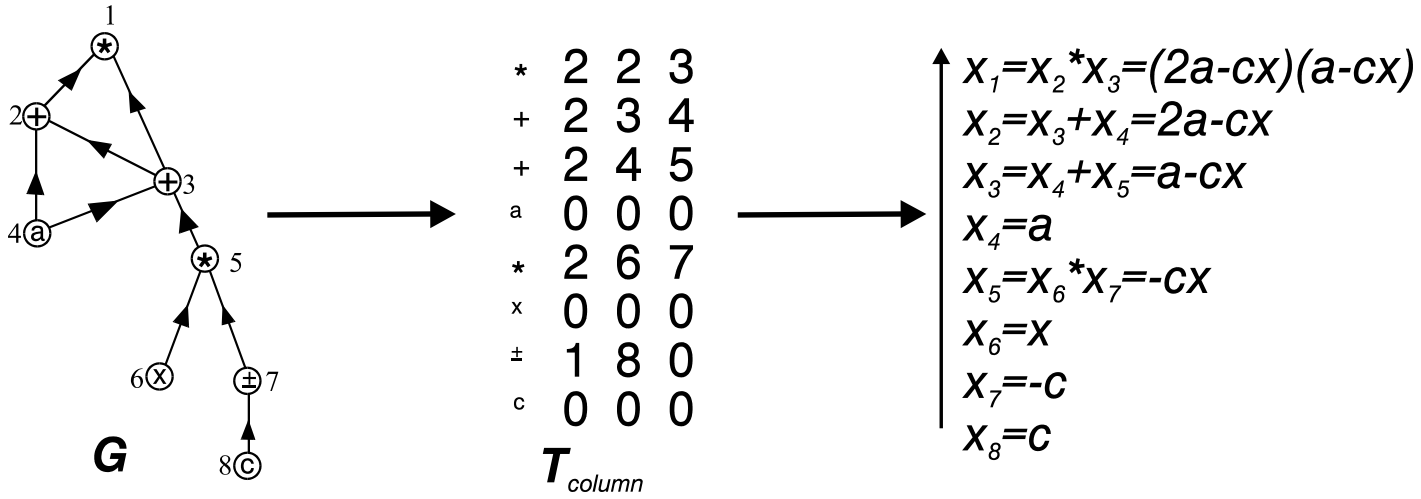
Syntaktické stromy

Ilustračný príklad výpočtu funkčných hodnôt syntaktického grafu G určeného pomocou stĺpcovej tabuľky T_{column}. Postupujúc zdola-hore cez všetky riadky tabuľky , funkčné hodnoty sú rekurentne počítané pomocou predchádzajúcich funkčných hodnôt.



Syntaktické stromy

Ilustračný príklad výpočtu funkčných hodnôt syntaktického grafu G z predchádzajúceho obr., ktorého jednotlivé vrcholy sú teraz ohodnotené algebraickými operáciami. Stĺpcová tabuľka T v tomto prípade je rozšírená o nový 0-tý stĺpec, ktorý špecifikuje funkcie priradené jednotlivým vrcholom. Výsledné ohodnotenie syntaktického grafu je funkčná hodnota výstupného vrcholu (2a-cx)(a-cx).



Syntaktické grafy

- Pre potreby symbolickej regresie, jednotlivé vrcholy syntaktického grafu musia byť ešte ohodnotené funkciami v súlade s ich vstupnou valenciou (aritou).
- Týmto spôsobom stĺpcové tabuľky sú plne špecifikované, predstavujú jednoduchý a efektívny prístup ku kódovaniu syntaktických grafov.
- Možno povedať, že predstavujú významné zovšeobecnenie prístupu syntaktických stromov pre kódovanie jednoducho vypočítateľných funkcií - procedúr.
- Zovšeobecnený pojem stĺpcovej tabuľky môže byť použitý ako "chromozóm" - elementárna informačná jednotka evolučného algoritmu a ukážeme elementárne operácie mutácie a kríženia nad týmito entitami.

Výpočet funkčnej hodnoty

```
function Eval_Table(input : i) : real; {i  index vrcholu}
begin if T_i1=0 then
      begin {input vertex}
            Eval_Table:=c_i
      end else
      if T_i1=1 then
      begin {intermediate or output unary vertex}
            Eval_Table:=f_i(Eval_Table(T_i2))
      end else
      if T_i1=2 then
      begin {intermediate or output binary vertex}
            Eval_Table:=f_i(Eval_Table(T_i2),Eval_Table(T_i3))
      end;
end;
```

Implementácia výpočtu funkčnej hodnoty výstupného vrcholu pomocou procedúry - funkcie s rekurziou, ktorá pôsobí nad stĺpcovou tabuľkou $T´=(T_{ij})$. Funkcia je inicializovaná príkazom Eval_Table(1), jej aktivácia končí na vstupnom vrchole, t.j. $T_{i1}=0$. Predpokladáme, že v syntaktickom grafe každý vrchol má maximálne dvoch predchodcov ($val_{in}^{max}=2$), tabuľka T má tri stĺpce a p riadkov.
Funkcie f_1, f_2, ...sú priradené jednotlivým vrcholom, reprezentujú požadované funkčné operácie (súčet, plus, krát, zmena znamienka, ...). V prípade vstupných vrcholov, tieto funkcie odpovedajú daným konštantám.

Syntaktické grafy

- Spôsob určenia stĺpcovej tabuľky, aby jej generovanie bolo dostatočne stochastické a pritom nevyžadovalo komplikovaný opravný proces transformácie náhodne generovanej tabuľky na semanticky korektný tvar.
- Budeme predpokladať, že počet riadkov v tabuľke je určený číslom (zadanou konštantou) p_{max}, a maximálna vstupná valencia vrcholov syntaktického grafu je v_{in}^{max} . Potom stĺpcová tabuľka obsahuje p_{max} riadkov a v_{in}^{max} stĺpcov (ktoré sú indexované 0,1,...).
- Použijeme rovnakú konvenciu, kde 0-tý stĺpec numericky kóduje typy funkcií priradených jednotlivým vrcholom, 1-vý stĺpec vyjadruje vstupnú valenciu vrcholu, t.j. počet vrcholov - predchodcov v syntaktickom grafe. Konečne, 2-hý a 3-tí stĺpec obsahujú indexy predchodcov, ich počet (včítane nuly) je určený elementmi 1-vého stĺpca (vstupnými valenciami).

Syntaktické grafy

- Pretože vrcholy syntaktického grafu musia byť indexované kanonicky, pre elementy i-tého riadku (T_{i1}, T_{i2}) musia vyhovovať podmienkam

$$i+1 \leq T_{i1} \leq p_{max}\ ,\ i+1 \leq T_{i2} \leq p_{max} \text{ a } T_{i1} \neq T_{i2}$$

- Graf určený takto definovanou tabuľkou nevyhovuje základnej podmienke pre určenie syntaktického grafu, aj keď je orientovaný acyklický graf, môže obsahovať viac ako jeden výstupných vrcholov.

Generovanie stĺpcovej tabuľky

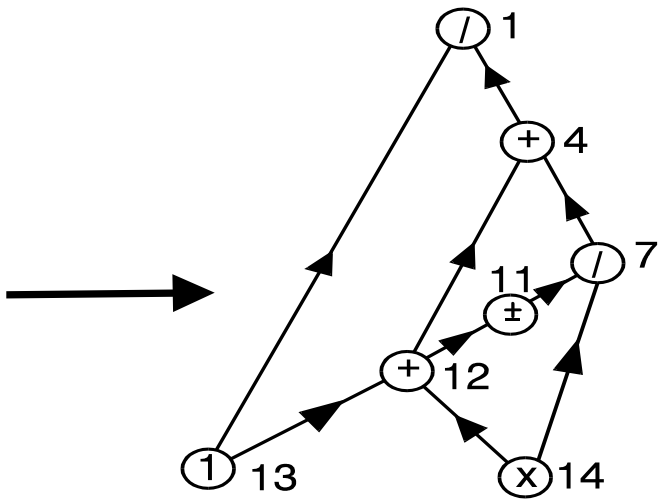
```
procedure Gener_Syntactic_Graph(input : p_max, val_in^max; output T);
begin for i:=1 to p_max do
    begin T_i1:=random(val_in^max+1);
        case T_i1 of
          0 : begin {input vertex}
                T_i0:=gener_type_function(0);
              end;
          1 : begin {unary intermediate/output vertex}
                T_i0:=gener_type_function(1);
                T_i2:=i+1+random(p_max-i);
              end;
          2 : begin {binary intermediate/output vertex}
                T_i0:=gener_type_function(2);
                repeat
                T_i2:=i+1+random(p_max-i);
                T_i3:=i+1+random(p_max-i);
                until T_i1=T_i2;
              end;
            .
            .
        end {of case};
    end {of for};
end;
```

Implementácia náhodnej generácie stĺpcovej tabuľky obsahujúcej p_{max} riadkov a (val_{in}^{max}+2) stĺpcov. Funkcia random(n) je náhodný generátor celých čísel z intervalu [0,n-1] s rovnomernou pravdepodobnosťou. cyklus repeat-until je aplikovaný tak dlho, až náhodne generované elementy T_{i2}, T_{i3} vyhovujú podmienke (5.42). Funkcia gener_type_function(val_{in}) náhodne generuje typ funkcie i-tého vrcholu v závislosti od hodnoty vstupnej valencie určenej $val_{in}=T_{i1}$. V prípade, že $T_{i1}=0$, potom táto funkcia určuje náhodne "vstupnú konštantu".

Náhodne generovaná tabuľka

Ilustratívny príklad náhodne generovanej stĺpcovej tabuľky obsahujúcej 15 riadkov a 4 stĺpce (p_{max}=15 a val_{in}^{max}=2). Hviezdičkou označené vrcholy sú aktívne pri konštrukcii syntaktického grafu obsahujúceho výstupný vrchol '1'. Ostatné vrcholy sa podieľajú na konštrukcii iného (alebo iných) syntaktických grafov, ktorých výstupné vrcholy sú iné ako vrchol '1'. Nakreslený syntaktický graf tiež vznikne aktiváciou funkcie Eval_Table(1) z algoritmu. Z týchto dôvodov môžeme pokladať náhodne generovanú tabuľku za korektný prístup k určeniu syntaktických funkcií, aj keď je nutné poznamenať, že veľká časť tabuľky môže byť nevyužitá pri konštrukcii syntaktickej funkcie.

	0	1	2	3
1*	/	2	13	4
2	x	0		
3	*	2	7	15
4*	+	2	12	7
5	1	0		
6	$()^2$	1	15	
7*	/	2	14	11
8	x	0		
9	-	2	15	14
10	±	1	15	
11*	±	1	12	
12*	+	2	13	14
13*	1	0		
14*	x	0		
15	3	0		



Spracované podľa knihy prof. Kvasničku a prof. Pospíchala Evolučné algoritmy.