TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Optimalizácia vykresľovania grafov pomocou Kohonenovej neurónovej siete

Samuel LEFKOVITŠ

DIPLOMOVÁ PRÁCA

2009

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra kybernetiky a umelej inteligencie

Optimalizácia vykresľovania grafov pomocou Kohonenovej neurónovej siete

DIPLOMOVÁ PRÁCA

Samuel Lefkovitš

Vedúci diplomovej práce: Ing. Rudolf Jakša, PhD.

Konzultant diplomovej práce: Ing. Rudolf Jakša, PhD.

Košice 2009

Analytický list

Autor:	Samuel Lefkovitš
Názov práce:	Optimalizácia vykresľovania grafov pomocou Kohonenovej neurónovej siete
Podnázov práce:	
Jazyk práce:	slovenský
Typ práce:	Diplomová práca
Počet strán:	68
Akademický titul:	Inžinier
Univerzita:	Technická univerzita v Košiciach
Fakulta:	Fakulta elektrotechniky a informatiky (FEI)
Katedra:	Katedra kybernetiky a umelej inteligencie (KKUI)
Študijný odbor:	Umelá inteligencia
Študijný program:	Špecializácia
Mesto:	Košice
Vedúci práce:	Ing. Rudolf Jakša, PhD.
Konzultanti práce:	Ing. Rudolf Jakša, PhD.
Dátum odovzdania:	7. máj 2009
Dátum obhajoby:	28-29.máj 2009
Kľúčové slová:	Vykresľovanie grafov, Graphviz, samoorganizujúce sa mapy, Kohonenová neurónová sieť,
Kategória konspekt:	Výpočtová technika; Umelá inteligencia
Citovanie práce:	Lefkovitš, Samuel: Optimalizácia vykresľovania grafov pomocou Kohonenovej neurónovej siete. Diplomová práca. Košice: Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky, 2009. 68 s.
Názov práce v AJ:	Graphs Layout Optimization Using Kohonen Neural Network
Podnázov práce v AJ:	
Kľúčové slová v AJ:	Graph drawing, Graphviz, Self-organizing maps, Kohonen neural network

Abstrakt v SJ

Táto diplomová práca navrhuje spôsob využitia Kohonenovej neurónovej siete pri vykresľovaní grafov – rozmiestňovaní vrcholov v rovine. Práca úspešne implementuje navrhnutý spôsob najskôr vo vlastnom testovacom prostredí a potom v jednom z najrozšírenejších softvérových produktov pre vizualizáciu grafov - v Graphviz-e. Experimentálne overuje možnosť použitia navrhnutej metódy a prezentuje výsledky porovnaní nášho algoritmu s ostatnými algoritmami z Graphviz-u.

Abstrakt v AJ

This diploma thesis proposes the usage of Kohonen neural network for graphs drawing – the layout of nodes. We implement this proposed method first in our own test environment and then in one of the most popular graphs visualization software – Graphviz. The work experimentally evaluates possibility of using proposed method and compares results of the new algorithm with other algorithms in the Graphviz.

T E C H N I C K Á U N I V E R Z I T A V K O Š I C I A C H

Fakulta elektrotechniky a informatiky Katedra kybernetiky a umelej inteligencie

D I P L O M O V Á P R Á C A

Študent: **Samuel Lefkovitš**

Študijný odbor: **Umelá inteligencia**

Akademický rok: 2008/2009

Názov práce v slovenskom a anglickom jazyku:

Optimalizácia vykresľovania grafov pomocou Kohonenovej neurónovej siete
Graphs Layout Optimization Using Kohonen Neural Network

Pokyny na vypracovanie:

1. Vypracovať úvod do vizualizácie grafov v programovom prostredí graphviz.
2. Vypracovať úvod do Kohonenových neurónových sietí.
3. Navrhnúť systém optimalizácie vykresľovania grafov pomocou Kohonenovej neurónovej siete.
4. Implementovať navrhnutý systém.
5. Realizovať experimenty na navrhnutom systéme za účelom overenia jeho výkonnosti.
6. Zhodnotiť realizované experimenty.
7. Vypracovať dokumentáciu podľa pokynov vedúceho diplomovej práce.

Vedúci diplomovej práce: Ing. Rudolf Jakša, PhD.
Konzultant diplomovej práce:

Dátum odovzdania diplomovej práce: 7.5.2009

prof. Ing. Ján Sarnovský, CSc.
vedúci zadávajúceho vedecko-pedagogického pracoviska

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY -10-

prof. Ing. Liberios Vokorokos, PhD.
dekan

V Košiciach, dňa 31.10.2008

Čestné vyhlásenie

Vyhlasujem, že som celú diplomovú prácu vypracoval/a samostatne s použitím uvedenej odbornej literatúry.

Košice, 7. máj 2009

.....

vlastnoručný podpis

Poďakovanie

Chcem poďakovať vedúcemu práce Ing. Rudolfovi Jakšovi, PhD. za usmernenie a podporu, a všetkým, ktorí ma akýmkoľvek spôsobom motivovali a podporili v mojej práci.

Obsah

Úvod

Táto diplomová práca navrhuje novú metódu, ktorá vychádza z neurónových sietí s nekontrolovaným učením, používa Kohonenovu neurónovú sieť. Táto práca navrhnutú metódu implementuje do prostredia Graphviz-u. Ďalej sa snaží porovnávať dosiahnuté výsledky jednotlivých rozvrhovacích algoritmov. Nová metóda by mala poskytnúť nové rozloženie vrcholov, mala by rovnomerne rozprestrieť vrcholy na celé kresliace plátno. Naša metóda zavádza aj jednu novinku, tvar kresliaceho plátna môže určiť používateľ. Môže to byť napríklad úzky obdĺžnik, trojuholník a pod.

Táto práca vychádza po teoretickej stránke zo všeobecných znalostí o neurónových sietiach. Pri implementácii do prostredia Graphviz-u sme vychádzali z popisu a príručiek uverejnených na domovskej stránke Graphviz-u.

Existuje niekoľko softvérov, ktoré sa zaberajú vykresľovaním grafov: uDraw, JGraph, GVF - The Graph Visualization Framework, aiSee, Tom Sawyer Software a ďalšie. V kategórií open source dominuje Graphviz svojou všeobecnosťou, multiplatformou, rozhraním pre programátorov.

Graphviz disponuje niekoľkými rozvrhovacími algoritmami. Základná všeobecná metóda DOT používa hierarchické usporiadanie vrcholov. Ostatné metódy sú trochu špecifickejšie a vhodné pre typy grafov s určitými vlastnosťami.

V prvej kapitole sa táto práca venuje všeobecnému popisu programu Graphviz, popisuje možnosti a spôsob použitia tohto softvérového produktu. Popisuje vstupný popisný súbor a jeho formátovacie vlastnosti pre grafy, vrcholy a hrany. Stručne vysvetľuje rozdiel medzi jednotlivými rozvrhovacími algoritmami a napokon vysvetľuje ovládanie programu z povelového riadku.

Nasledujúca kapitola je stručným úvodom do Kohonenovych neurónových sietí. Začiná pri nekontrolovanom učení, vysvetľuje špecifiká konkurenčného učenia až nakoniec popipsuje ako funguje Kohonenova neurónová sieť.

V tretej kapitole táto práca všeobecné znalosti o Kohonenovej neurónovej sieti aplikuje pre problém vykresľovania grafov. Vysvetľuje každý použitý vzťah, ktorý je potrebný pre implementáciu.

Ďalšia kapitola podrobne popisuje testovacie prostredie, ktoré bolo vytvorené na testovanie nového algoritmu a porovnávanie výsledkov starých algoritmov. Venuje sa

podrobnejšie výpočtu vzdialeností vrcholov v grafe a výpočte súradníc vrcholov, pretože aplikácia pre prostredie Graphviz-u je odlišná.

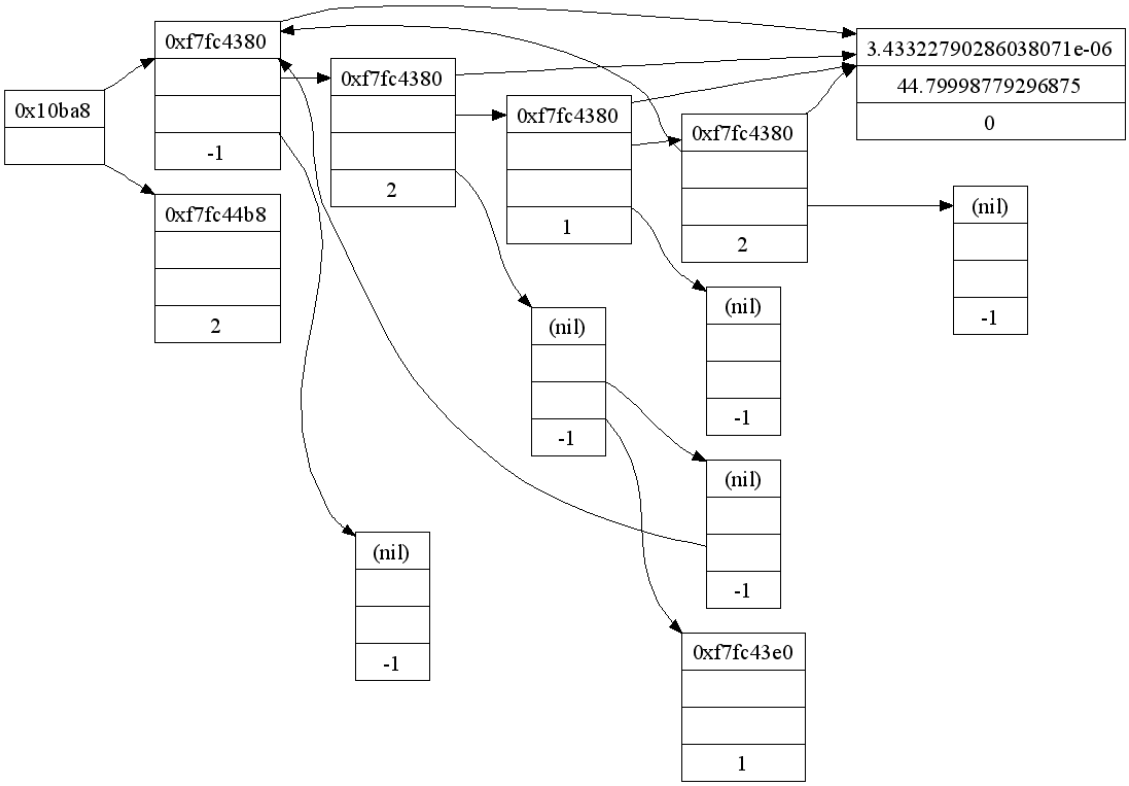
Piata kapitola podrobne rozoberá problémy pri implementácii do prostredia Grpahviz. Hľadá vstupný bod v programovom projekte, popisuje rozhranie medzi našimi programovými funkciami a existujúcim programom. Detailne popisuje algoritmus na výpočet vzdialeností vrcholov v grafe a výpočet súradníc vrcholov pomocou vývojových diagramov a výpisu zdrojových kódov.

Posledná časť je venovaná experimentom. Najskôr práca uvádza výsledky, ktoré sme dosiahli pri overovaní funkčnosti algoritmu v testovacom prostredí. Potom zobrazuje dosahované výsledky nového algoritmu oproti existujúcich algoritmov a snaží sa zhodnotiť klady a nedostatky navrhovaného riešenia. Nakoniec v tejto kapitole sú prezentované výsledky pre použitie rôzneho tvaru kresliaceho plátna.

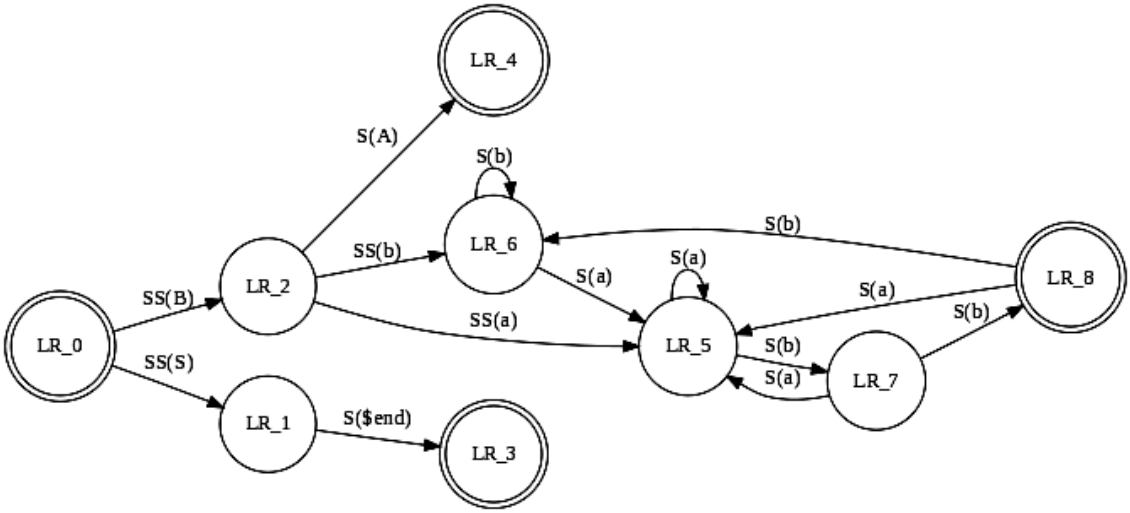
1 Úvod do vizualizácie grafov v programovom prostredí Graphviz

Vizualizácia grafov je spôsob ako reprezentovať štruktúrované informácie do diagramov. Automatické vykresľovanie grafov má mnoho dôležitých aplikácií v softvérovom inžinierstve, databázach, web dizajne, sietiach, tvorí vizuálne rozhranie pre mnoho iných odborov. Nasledujúci text vychádza zo zdroja [1].

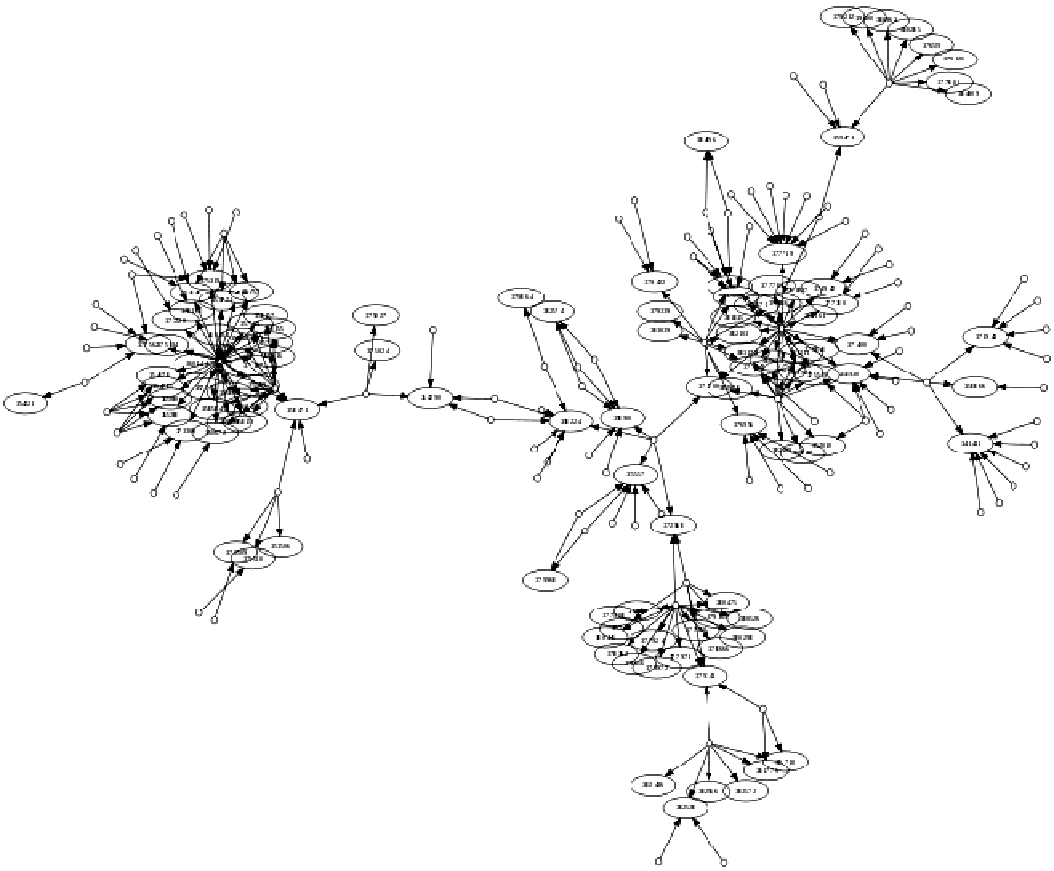
Graphviz je open source softvér pre vizualizáciu grafov. Má niekoľko hlavných metód pre rozvrhovanie. Vie poskytnúť webové a interaktívne rozhranie, pomocné nástroje, knižnice pre programátorov.



Obr. 1 - 1 Z galérie Graphviz-u. Príklady grafov, formátovania vrcholov a hrán. (Obr. prevzatý z [1]).



Obr. 1 - 2 Z galérie Graphviz-u. Príklady grafov, formátovania vrcholov a hrán. (Obr. prevzatý z [1]).



Obr. 1 - 3 Z galérie Graphviz-u. Príklady grafov, formátovania vrcholov a hrán. (Obr. prevzatý z [1]).

Rozvrhovacie programy Graphviz-u používajú na popis grafov jednoduchý textový jazyk (skript, vstupný popisný súbor), vytvárajú grafy v niekoľkých užitočných bežných formátoch ako SVG pre webové stránky, postcript pre zahrnutie do PDF alebo iných dokumentov, PNG, JPG a ďalšie.

Graphviz má veľa užitočných čŕt pre konkrétne grafy, ako napríklad: nastavovanie farieb, fontov, prehľadné rozvrhnutia vrcholov, štýly čiar, hyperlinky a používateľské tvary.

V praxi, grafy sú bežne generované z nejakých externých dát, ale tiež môžu byť vytvorené a editované manuálne ako surový text alebo pomocou nejakého grafického editora.

1.1 Úvod do vykresľovania grafov

Graf G je usporiadaná dvojica (V, H), kde V je nejaká neprázdna množina a H je množina dvojprvkových podmnožín množiny V. Prvky množiny V nazývame vrcholy grafu G a prvky množiny H nazývame hrany grafu G. Zápis G = (V,H) znamená, že graf G má množinu vrcholov V a množinu hrán H. [2]

Grafy sa znázorňujú kreslením do roviny. Vrcholom grafu sa priradia body roviny a hrany sa vyjadrujú spojením príslušných dvojíc bodov priamymi alebo zakrivenými čiarami. Toto znázornenie sa nazýva diagram grafu. Platí, že dva rôzne grafy môžu mať zhodný diagram, ale tiež ten istý graf je možné znázorniť rôznymi diagramami. [2]

Existujú špecifické typy grafov, ktoré majú špecifické vlastnosti. Napríklad: stromy, planárne neorientované grafy, planárne orientované grafy. Existujú špecifické prvky grafov, ako: most, kružnica. Pre hore uvedené typy grafov existujú presne prepracované metódy na vykresľovanie diagramov. Prvky ako most a kružnica majú niekedy zásadný vplyv na tieto metódy. [3]

Táto práca navrhuje metódu, ktorá je úplne nezávislá od typu grafu. Preto týmto typom grafov a uvedeným prvkom grafov sa nemusíme ďalej venovať.

Z hľadiska hodnotenia diagramov grafov sú pre nás dôležité hlavne všeobecné estetické kritéria. Výkonové kritérium nie je zaujímavé a je vysvetlené v závere tejto práce.

Diagram grafu je lepší, ak lepšie vyjadruje informácie grafu, ak je lepšie čitateľný. Estetické kritéria sú požiadavky, ktoré chceme pri vykresľovaní grafov dosiahnuť:

- zachytiť hierarchickú štruktúru grafu,
- minimalizovať križovania hrán,
- maximalizovať najmenšie uhly susedných hrán k vrcholom,
- minimalizovať kresliace plátno, veľkosť diagramu,

- maximálne pokryť (využiť) kresliace plátno,

- minimalizovať dĺžku hrán

- dosiahnuť symetriu a vyváženosť.

Aj na základe týchto kritérií tiež budeme hodnotiť dosiahnuté výsledky. [3]

1.2 Úvod do programového prostredia Graphviz

Vstupný popisný súbor pre Graphviz

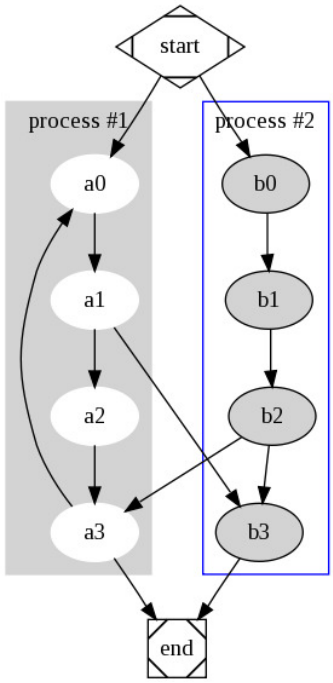
Vstupom do programu Graphviz je vstupný popisný súbor s príponou dot.

```
digraph G {

        subgraph cluster_0 {
                style=filled;
                color=lightgrey;
                node [style=filled,color=white];
                a0 -> a1 -> a2 -> a3;
                label = "process #1";
        }

        subgraph cluster_1 {
                node [style=filled];
                b0 -> b1 -> b2 -> b3;
                label = "process #2";
                color=blue
        }
        start -> a0;
        start -> b0;
        a1 -> b3;
        b2 -> a3;
        a3 -> a0;
        a3 -> end;
        b3 -> end;

        start [shape=Mdiamond];
        end [shape=Msquare];
}
```



Obr. 1 - 4 Jednoduchý príklad vstupného popisného súboru. (Obr. prevzatý z [1]).

Jednoduchými príkazmi sa popisujú formátovacie vlastnosti grafov, podgrafov, hrán, vrcholov. Niektoré vlastnosti sú špecifické len pre určitú rozvrhovaciu metódu. Ostatné rozvrhovacie metódy ich ignorujú.

Príklady formátovacích vlastností:

Vrcholy: shape, height, width, label, fontisize, font name, fontcolor, style, color, pos ...

Hrany: weight, label, fontsize, fontname, fontcolor, style, color, len, dir, decorate, id ...

Grafy: start, size, page, margin, label, fontsize, fontname, fontcolor, orientation, center, ...

Rozvrhovacie metódy Graphviz-u

Dot - Používa hierarchické alebo vrstvené kreslenie orientovaných grafov. Metóda sa snaží umiestňovať hrany v tom istom smere (zhora nadol, zľava doprava) a potom sa pokúša vyhnúť križovaniu hrán a redukovať dĺžku hrán.

Neato - Používa na rozvrhnutie pružinový model, algoritmus Kamada-Kawai, čo je obdoba štatistického multi-dimenzionálneho scaling-u.

Fdp - Používa pružinový model, heuristiku Fruchterman-Reingold, zahŕňa multigridové riešenie, ktoré rieši väčšie grafy a husté, natlačené neorientované grafy.

Twopi - Predstavuje radiálne (hviezdicovité, lúčovité) rozvrhnutie podľa Graham Wills 97. Vrcholy sú umiestnené do sústredných kružníc, ktoré sú viazané na ich vzdialenosti od daného koreňového vrcholu.

Circo - Kruhovité rozvrhnutie je riešené podľa Six and Tollis 99, Kauffman and Wiese 02. Toto rozvrhnutie je vhodné pre diagramy s viacnásobnými kruhovými štruktúrami, napríklad niektoré telekomunikačné siete.

Ovládanie programu Graphviz

Graphviz možno používať ako dynamické knižnice pre vývojárov do ich aplikácii. Častejšie sa však používa ako skompilované binárne súbory pre požadovanú platformu. Pre účely testovanie v tejto práci sa tiež používajú skompilované binárne súbory. Požadovaná rozvrhovacia metóda sa volá z príkazového riadku nasledovne:

dot [−(G|N|E)name=value][−Tlang][−l libfile][−o outfile][−v][−V][files]

Používateľ môže zvoliť meno vstupného a výstupných súborov, formát výstupu a ďalšie vlastnosti.

Podporované výstupné formáty:

−Tps (PostScript)

−Tsvg −Tsvgz (Structured Vector Graphics)

−Tfig (XFIG graphics)

−Tmif (FrameMaker graphics)

−Thpgl (HP pen plotters)

−Tpcl (Laserjet printers)

−Tpng −Tgif (bitmap graphics)

−Tdia (GTK+ based diagrams)

−Timap (imagemap files for httpd servers for each node or edge that has a non(hynull "href" attribute.)

−Tcmapx (client-side imagemap for use in html and xhtml)

Podporované platformy:

Linux (Redhat, Centos, Fedora, Ubuntu),

Windows,

Mac OS,

Unix (FreeBSD, Solaris).

2 Úvod do Kohonenovych neurónových sietí

Kohonenova sieť je špecifickým typom neurónových sietí s nekontrolovaným konkurenčným učením. Nasledujúci text vychádza z práce [5].

Nekontrolované učenie

Nekontrolované učenie predstavuje prípady, keď neurónovej sieti (NN) poskytujeme len vstupy a očakávame, že NN s tým niečo urobí. Tento proces môžeme nazvať samo-organizácia, preto sa tieto siete nazývajú tiež samo-organizujúce sa mapy (SOM) alebo samo-organizujúce sa siete. Učenie v tomto prípade, tak ako pri kontrolovanom učení, znamená zmenu synaptických váh (SV). Učenie sa končí ak NN sa dostane do stavu globálnej stability (GS) nie konvergencie. Tento stav môže popisovať nasledujúci vzťah 2.1

$$\left|\Delta W(t) - \Delta W(t+1)\right| \leq \varepsilon . \tag{2.1}$$

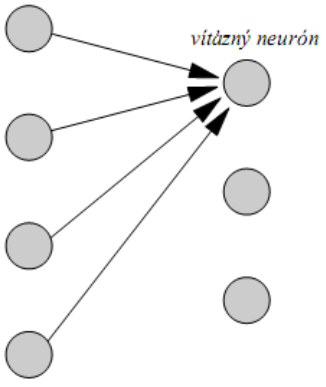
ΔW(t) – zmena synaptických váh v iterácií t,

ε – dostatočne malá hodnota.

V týchto sietiach existuje riziko nekonečného učenia. Stav GS môže reprezentovať aj oscilácia NN v nejakých pevných bodoch.

Konkurenčné učenie

Konkurenčné učenie je jedným z typov patriacim pod nekontrolované učenie. Najčastejšie slúži k spracovaniu dát za účelom zhlukovej analýzy. Typológia pre konkurenčné učenie je na nasledujúcom Obr. 2-1. Neurón, ktorý má najvyššiu výstupnú hodnotu sa stáva víťazom a SV k nemu sa menia najvýraznejšie alebo len tie. Vtedy hovoríme o pravidle WTA – víťaz berie všetko (winner takes all).



Obr. 2 - 1 Zmena váh, ktoré smerujú k víťaznému neurónu. (Obr. prevzatý z [5]).

Pre nasledujúce vzťahy bude platiť:

i - index neurónu vo výstupnej vrstve;
N_c - počet neurónov vo výstupnej vrstve;
j - index neurónu vo vstupnej vrstve;
M - počet neurónov vo vstupnej vrstve
w_{ij} - synaptické váhy od neurónu j do neurónu i;
x - vstupná hodnota na vstupných neurónoch
ou - výstupná hodnota na výstupných neurónoch
t - čas alebo iterácia

Ako **aktivačná funkcia** neurónov vo výstupnej vrstve sa používa vzťah 2.2:

$$ou_i(t) = \sum_{j=1}^{M} w_{ij}(t) x_j(t) . \tag{2.2}$$

Tento vzťah si však vyžaduje normalizáciu. Preto sa tiež používa „euklidovský vzťah“ 2.3., ktorý je použitý aj v tejto práci a nevyžaduje si dodatočné úpravy výstupov.

$$ou_i(t) = \sum_{j=1}^{M} (w_{ij}(t) - x_j(t))^2 . \tag{2.3}$$

Určenie víťaza. Z výstupných hodnôt týchto neurónov určíme maximálnu hodnotu a jej neurón sa stáva víťazom. Vzťah 2.4.

$$\begin{aligned} &ou_i(t) = 1 \qquad ak \qquad ou_i = \max\{ou, \forall i = 1,..., N_C\}, \\ &ou_i(t) = 0 \qquad inak. \end{aligned} \tag{2.4}$$

V prípade ak používame euklidovskú vzdialenosť, víťaza určíme z minimálnej hodnoty. Použijem vzťah 2.5.

$$\begin{aligned} &ou_i(t) = 1 \qquad ak \qquad ou_i = \min\{ou, \forall i = 1,..., N_C\}, \\ &ou_i(t) = 0 \qquad inak. \end{aligned} \tag{2.5}$$

Zmena SV. V konkurenčnom učení sa snažíme jednotlivé vstupy x premietnuť do hodnôt SV. Cieľom je premapovať vstupy do SV. Rôzne vstupy x budú mať rôznych víťazov, ktorí budú reprezentovať a ich SV budú reprezentovať centrá zhlukov. Výsledok samotného procesu zhlukovania je skrytý v hodnotách jednotlivých SV, ktoré reprezentujú centrá zhlukov, ktoré má systém nájsť.

Odvodenie adaptačného pravidla je analogické ako pri kontrolovanom učení. Pre každý výstupný neurón budeme mať chybovú funkciu 2.6:

$$J(t) = 0{,}5\sum_{j=1}^{M}(w_{ij}(t) - x_j(t))^2 . \tag{2.6}$$

Zmena synaptických váh, potom vyzerá nasledovne:

$$\Delta w_{ij}(t) = -\gamma \frac{\partial J(t)}{\partial w_{ij}(t)} . \tag{2.7}$$

Kde γ je učiaci pomer. Pre nevíťazné neuróny je zmena váh rovná nule, pre víťazný neurón po zderivovaní platí:

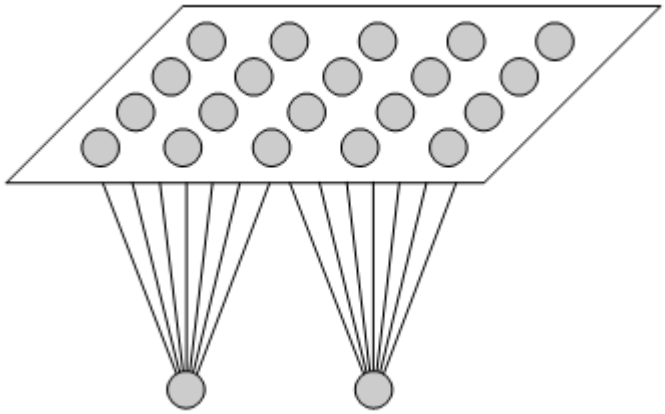
$$\Delta w_{ij}(t) = -\gamma(w_{ij}(t) - x_j(t)) . \tag{2.8}$$

A nasledujúca iterácia potom pre i-tý víťazný neurón vyzerá:

$$w_{ij}(t+1) = w_{ij}(t) + \gamma(x_j(t) - w_{ij}(t)) . \tag{2.9}$$

Kohonenova vrstva a funkcia susednosti

Výstupná vrstva Kohonenovej NN je usporiadaná geometricky do nejakého útvaru (napr. obdĺžnik). Táto vrstva sa volá Kohonenova vrstva.

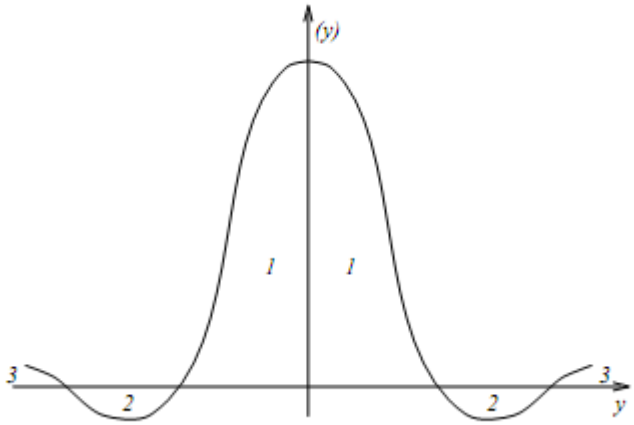


Obr. 2 - 2 Typické usporiadanie v Kohonenovej NN. (Obr. prevzatý z [5]).

Kohonenove siete sú rozšírením konkurenčného učenia. Zavádza sa princíp viacerých víťazov – multiply WTA. Zmeny váh sa nerealizujú len pre víťaza, ale zavedením funkcie susednosti (2.10) sa aktualizujú aj váhy najbližších susedov.

$$\Lambda_{ij} = h(t).e^{-\frac{(d_j)^2}{r_i(t)}} . \tag{2.10}$$

h(t) je adaptačná výška, d_j je vzdialenosť medzi neurónmi v Kohonenovej vrstve a *r(t)* predstavuje polomer priestorového susedstva v iterácií *t*. Táto funkcia môže byť vyjadrená v tvare mexického klobúka.



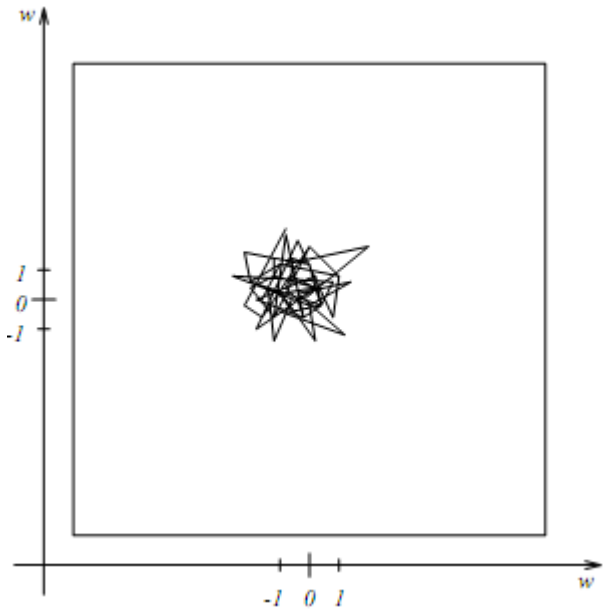
Obr. 2 - 3 Možný tvar funkcie susednosti Λ_{ij}. (Obr. prevzatý z [5]).

Potom pre zmeny SV platí:

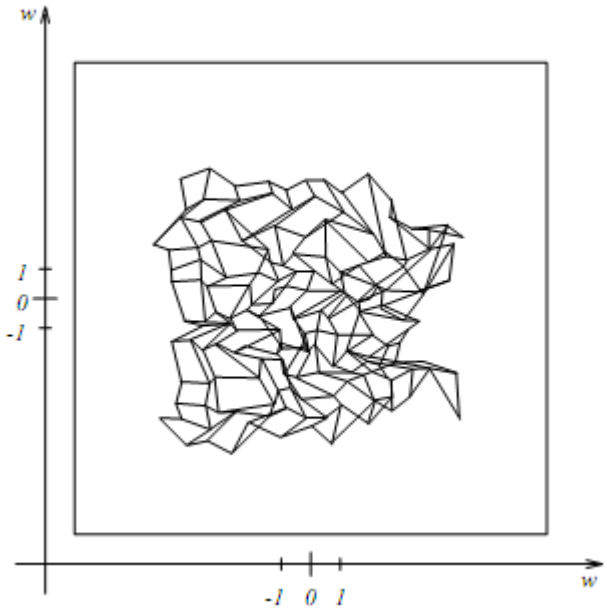
$$\Delta w_{ij}(t) = -\gamma \Lambda_{ij}(w_{ij}(t) - x_j(t)), \tag{2.11}$$

$$w_{ij}(t+1) = w_{ij}(t) + \gamma \Lambda_{ij}(x_j(t) - w_{ij}(t)). \tag{2.12}$$

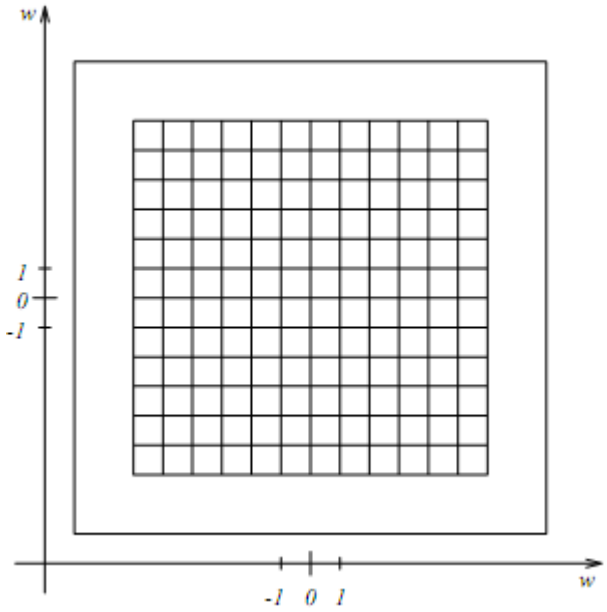
Proces učenia s dvoma vstupnými neurónmi, ktoré reprezentujú súradnice x a y v priestore, potom bude vyzerať nasledovne:



Obr. 2 - 4 Stav SV pri inicializácii z intervalu <-1, 1>. Obrázok predstavuje Kohonenovu vrstvu. Vrchol reprezentuje neurón zakreslený na základe hodnôt synaptických váh ku vstupným neurónom (x a y súradnica). Hrana reprezentuje susednosť zhlukov. (Obr. prevzatý z [5]).



Obr. 2 - 5 Stav SV po určitej dobe učenia t. (Obr. prevzatý z [5]).



Obr. 2 - 6 Stav SV na konci učenia, stav GS, keď vstupné vzorky boli rovnomerne rozdelené. (Obr. prevzatý z [5]).

Staršie implementácie Kohonenovych neurónových sieti pre vykresľovanie grafov.

Podarilo sa nám nájsť dva články, v ktorých sa autori pokúšali implementovať samoorganizujúce sa mapy (SOM) pre vykresľovanie grafov [7][8]. Články sú staršie ako 10 rokov. Oba články popisujú spôsob použitia SOM. Rozdiely oproti našej metóde sú minimálne, predovšetkým vo funkcii susednosti.

Prvý článok [7] pre toto použitie zavádza nové termíny: samoorganizujúce sa grafy (Self-Organizing Graphs) a inverzné samoorganizujúce sa mapy ISOM (Inverted Self-Organizing Maps). Článok letmo porovnáva stochastické metódy. Spomína tiež pokus z roku 1994 keď, pomocou evolučných algoritmov, dobré rozvrhnutie grafu s 12 vrcholmi trvalo 4 minúty. Článok popisuje výsledky experimentálneho overenia funkčnosti metódy v 2D a 3D. Naša práca poukazuje na výhodu možnosti použitia ľubovoľného tvaru kresliaceho plátna v 2D v kapitole 6.3. Uvádzaný článok prezentuje výsledky vykresľovania v 3D na povrch gule.

Druhý článok [8] sa venuje veľkým grafom (121 vrcholov). Prezentované výsledku sú takmer rovnaké ako v našej práci.

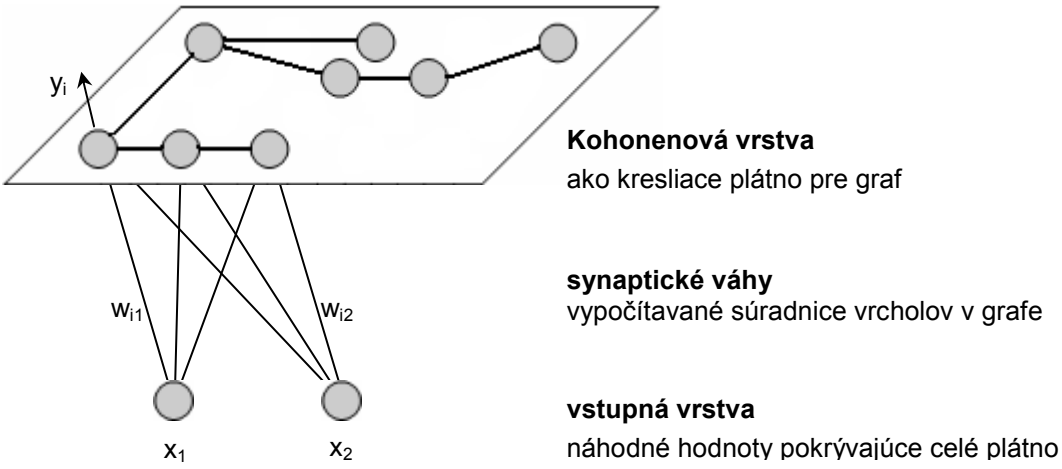
Hodnotenia metódy oboch článkov sú podobné. Metóda SOM je extrémne jednoduchá oproti bežne používaným sofistikovaným metódam, je silno prispôsobivá pre rôzne typy grafov a tvary vykresľovacieho priestoru. Má malé výpočtové nároky. Druhý článok odporúča použiť SOM ako predbežné spracovanie a následne použiť viac sofistikovanú metódu na rozvrhovanie.

3 Návrh systému optimalizácie vykresľovania grafov pomocou Kohonenovej neurónovej siete

Cieľom tejto práce je použiť Kohonenovu neurónovú sieť ako novú metódu pre optimalizáciu vykresľovania grafov v prostredí programu Graphviz. Kohonenova neurónová sieť dokáže vyriešiť rozmiestňovanie vrcholov do roviny (na kresliace plátno), vypočítať konečné súradnice vrcholov grafu v rovine. Ostatné problémy ako kreslenie hrán, formátovanie vrcholov názvov a ponechávame na riešenie samotného programu Graphviz.

Táto práca používa Kohonenovu metódu nasledovne. Vstupná vrstva neurónovej siete je tvorená dvoma neurónmi. Váhy z týchto dvoch neurónov ku každému neurónu v Kohonenovej vrstve potom predstavujú súradnice x1 a x2 vrcholov v grafe, ktoré chceme zakresliť v rovine. Kohonenova vrstva predstavuje kresliace plátno a každý neurón v nej predstavuje jeden vrchol grafu, ktorý chceme vizualizovať. Hrany grafu majú vplyv na výpočet zmeny váh v procese učenia prostredníctvom funkcie susednosti.

Platí ďalej, že hodnoty na vstupných neurónov sa snažia premietnuť do synaptických váh s ohľadom víťaza v každej iterácií a funkcie susednosti.



Obr. 3 - 1 Označovanie jednotlivých prvkov NN pre vykresľovanie grafov.

V nasledujúcom budeme uvažovať:

j - index vstupného neurónu,

x_j - hodnota vstupného neurónu - x_1: x-ová súradnica; x_2 :y-ová súradnica,

i - index neurónu v kohonenovej vrstve - index vrcholu v grafe,

y_i - výstupná hodnota neurónu v kohonenovej vrstve - vrchol v grafe,

w_{ji} - synaptická váha medzi vstupným a kohonenovym neurónom - w_{1i}: x-ová súradnica i-tého vrcholu; w_{2i}: y-ová súradnica i-tého vrcholu.

Proces učenia neurónovej siete predstavuje výpočet súradníc vrcholov grafu. Po ukončení procesu učenia synaptické váhy predstavujú definitívne súradnice vrcholov.

Inicializácia NN

Na začiatku výpočtu sa váhy inicializujú na náhodné hodnoty, avšak do stredu kresliaceho plátna. Celé kresliace plátno má rozmer 1 krát 1 palec a inicializačný štvorec v strede plátna má jednu desatinu dĺžky a šírky celého plátna.

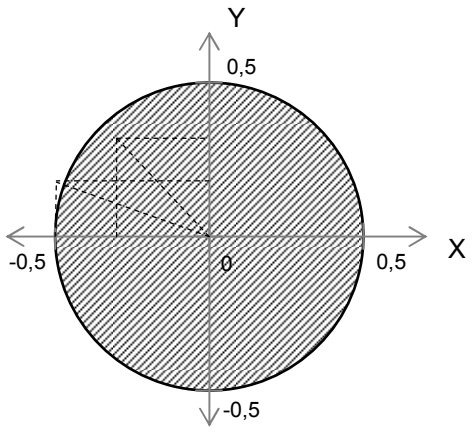
Kresliace plátno

Učenie spočíva v generovaní náhodných vstupných súradníc, ktoré majú pokryť celé plátno. Veľkosť kresliaceho plátna je počas výpočtu je 1 krát 1 palca. Jednotky ‘palce‘ sú dané implementovaným rozhraním v Graphviz-e. Na konci výpočtu sa vypočíta výsledná veľkosť kresliaceho plátna a prepočítajú sa pomerne tiež súradnice vrcholov.

Graf je možné rozprestrieť na kresliace plátno, ktoré môže mať rôzny tvar, napr. kruh, trojuholník a pod. Stačí vstupné náhodne generované hodnoty eliminovať na požadovaný tvar.

Pre kruh musí platiť (súradnice (0, 0) sú v strede plátna):

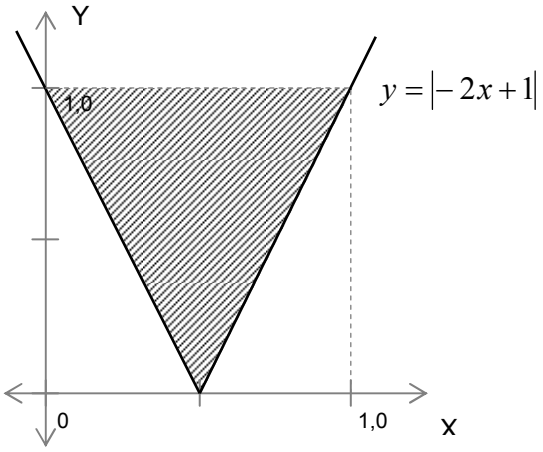
$$\sqrt{x^2 + y^2} < 0{,}5\,. \tag{3.1}$$



Obr. 3 - 2 Kresliace plátno - kruh. Generované vstupné hodnoty musia byť vnútri kruhu.

Pre trojuholník musí platiť (súradnice (0,0) sú v ľavo dole):

$$y > |-2x+1|. \tag{3.2}$$



Obr. 3 - 3 Kresliace plátno - trojuholník. Generované vstupné hodnoty musia byť vnútri trojuholníka.

Učenie NN

Parametre pre proces učenia sú: počet iterácií, koeficient učenia γ a polomer priestorového susedstva r. Tieto hodnoty sme určili na základe experimentov nasledovne:

počet iterácií = 1000,
γ = <1.0; 0,0001>,
r = <1.0; 0.0>.

Koeficient učenia a polomer priestorového susedstva sa lineárne mení každou iteráciou od začiatočnej po konečnú hodnotu.

V každej iterácií vypočítame výstupné hodnoty pre i-tý neurón v Kohonenovej vrstve ako euklidovskú vzdialenosť:

$$y_i = (w_{i1} - x_1)^2 + (w_{i2} - x_2)^2. \tag{3.3}$$

Minimálna výstupná hodnota určí víťaza v danej iterácii. Zmena váh pre obe súradnice sa potom vypočíta nasledovne:

$$\Delta w_{i1} = -\gamma \Lambda_i (w_{i1} - x_1), \tag{3.4}$$

$$\Delta w_{i2} = -\gamma \Lambda_i (w_{i2} - x_2). \tag{3.5}$$

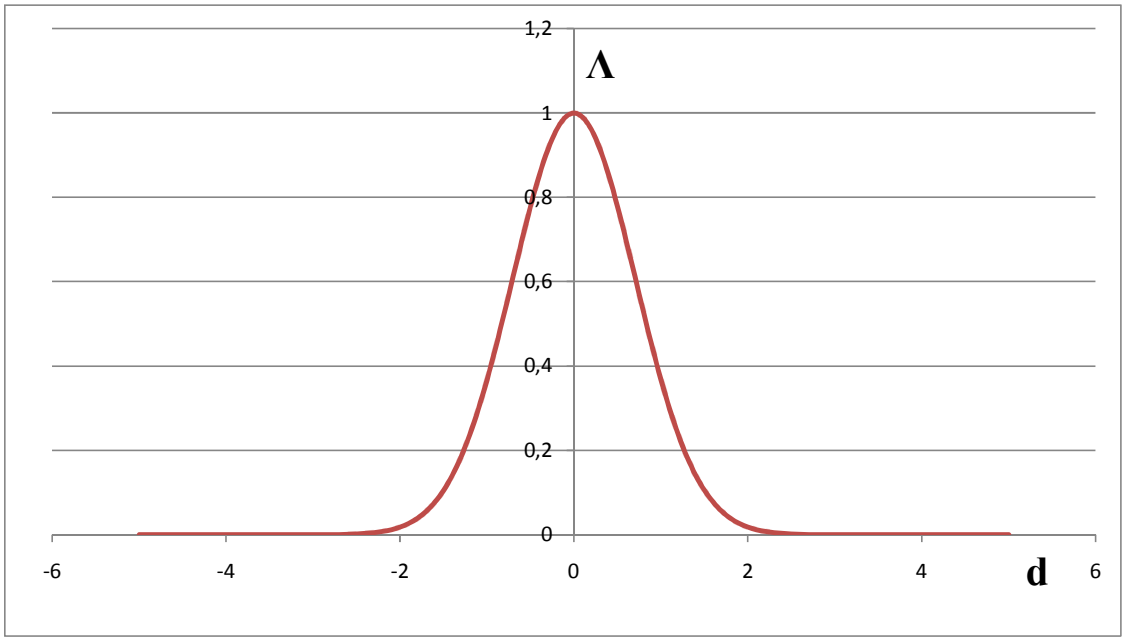
kde Λ_{ij} je funkcia susednosti závislá predovšetkým od vzdialenosti neurónu od víťaza.

Funkcia susednosti

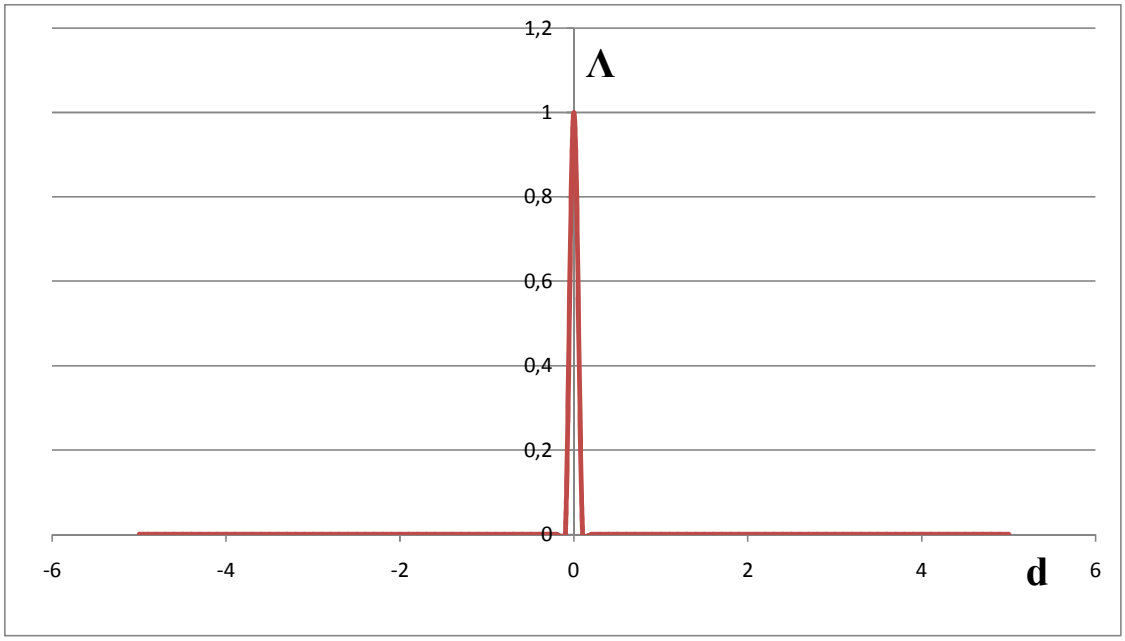
Tvar použitej funkcie susednosti je:

$$\Lambda = h(t).e^{-\frac{d_i^2}{r(t)}}. \tag{3.6}$$

Adaptačnú výšku *h(t)* sme v našom prípade ignorovali, teda jej hodnota sa rovná 1. Polomer priestorového susedstva r sa lineárne mení od začiatočnej po konečnú hodnotu podľa predchádzajúcej stati. Vzdialenosť medzi i-tým a víťazným neurónom je d_i a je daná maticou vzdialenosti.



Obr. 3 - 4 Graf funkcie susednosti pre r = 1 na začiatku učenia.



Obr. 3 - 5 Graf funkcie susednosti pre r = 0,0001 na konci učenia

Matica vzdialeností vrcholov grafu

Vzdialenosť pre výpočet funkcie susednosti chápeme ako počet vrcholov v ceste od štartovacieho k cieľovému vrcholu.

Najskôr sa zostrojí cenová matica grafu. Nech náš graf, ktorý chceme vizualizovať, je $\vec{G} = (V, H)$. Potom cenová matica je C = (c_{ij}). Keďže graf nemá ohodnotené hrany uvažujeme, že každá hrana grafu má hodnotu váhy 1. Pre jej prvky potom platí:

$$c_{ij} = \begin{cases} 1, & ak \quad (v_i, v_j) \in H, \\ 255, & ak \quad (v_i, v_j) \notin H, \\ 0, & ak \quad i = j. \end{cases} \tag{3.7}$$

V nasledujúcom kroku hľadáme najkratšiu cestu z každého do každého vrcholu vlastnou heuristikou. Iný algoritmus je použitý v testovacom prostredí a iný pre prostredie Graphviz-u, kde prioritou je optimalizácia počtu operácii. Obe algoritmy sú popísané v nasledujúcich kapitolách.

4 Implementované testovacie prostredie pre realizáciu experimentov.

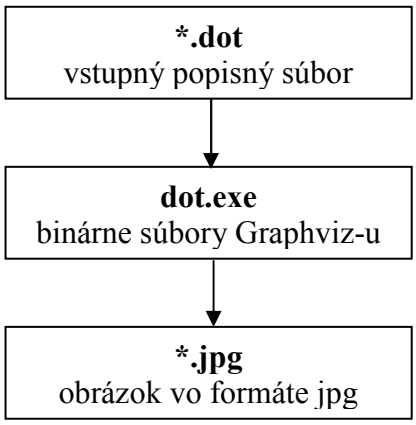
Program testovacieho prostredia slúžil na:

- prezentáciu a porovnávanie starých a nových rozvrhovacích metód,
- testovanie nových metód,
- na sledovanie priebehu výpočtov,
- na hľadanie optimálnych koeficientov výpočtov.

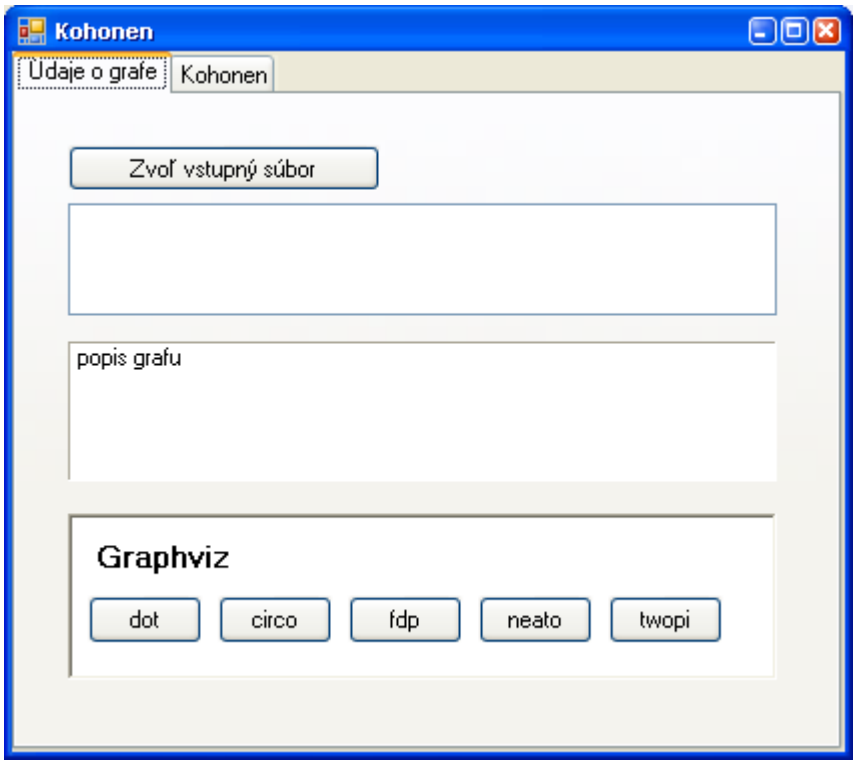
Testovacie prostredie pre realizáciu experimentov bolo vytvorené v jazyku C# pomocou MS Visual Studio 2008. Jazyk C# bol zvolený kvôli jednoduchosti a rýchlosti vývoja.

Prezeranie a porovnávanie výsledkov rozvrhovacích metód

Prvá časť programu slúži na porovnávanie výsledkov. Vstupom do programu v tomto prípade je vstupný popisný súbor s príponou dot. Výstupom je vygenerovanie obrázku vo formáte jpg pomocou existujúcich rozvrhovacích metód Graphviz-u, teda volaním binárnych súborov Graphviz-u. Testovací program následne otvorí obrázok v aplikácii, ktorá je v prostredí Windows asociovaná s danou príponou. Takto je umožnené rýchle porovnávanie výsledkov.



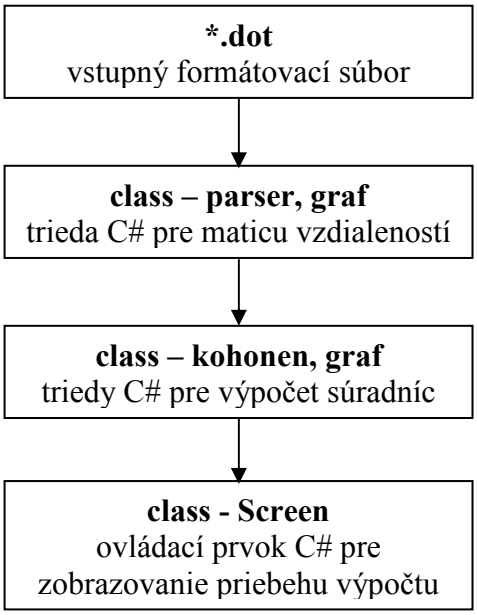
Obr. 4 - 1 Blokový diagram – porovnávanie existujúcich rozvrhovacích metód.



Obr. 4 - 2 Používateľské rozhranie – načítanie vstupného súboru a prezeranie výsledkov.

Testovanie Kohonenovej rozvrhovacej metódy

Druhá časť programu slúži na prácu s Kohonenovou rozvrhovacou metódou.



Obr. 4 - 3 Blokový diagram – práca s Kohonenovou rozvrhovacou metódou.

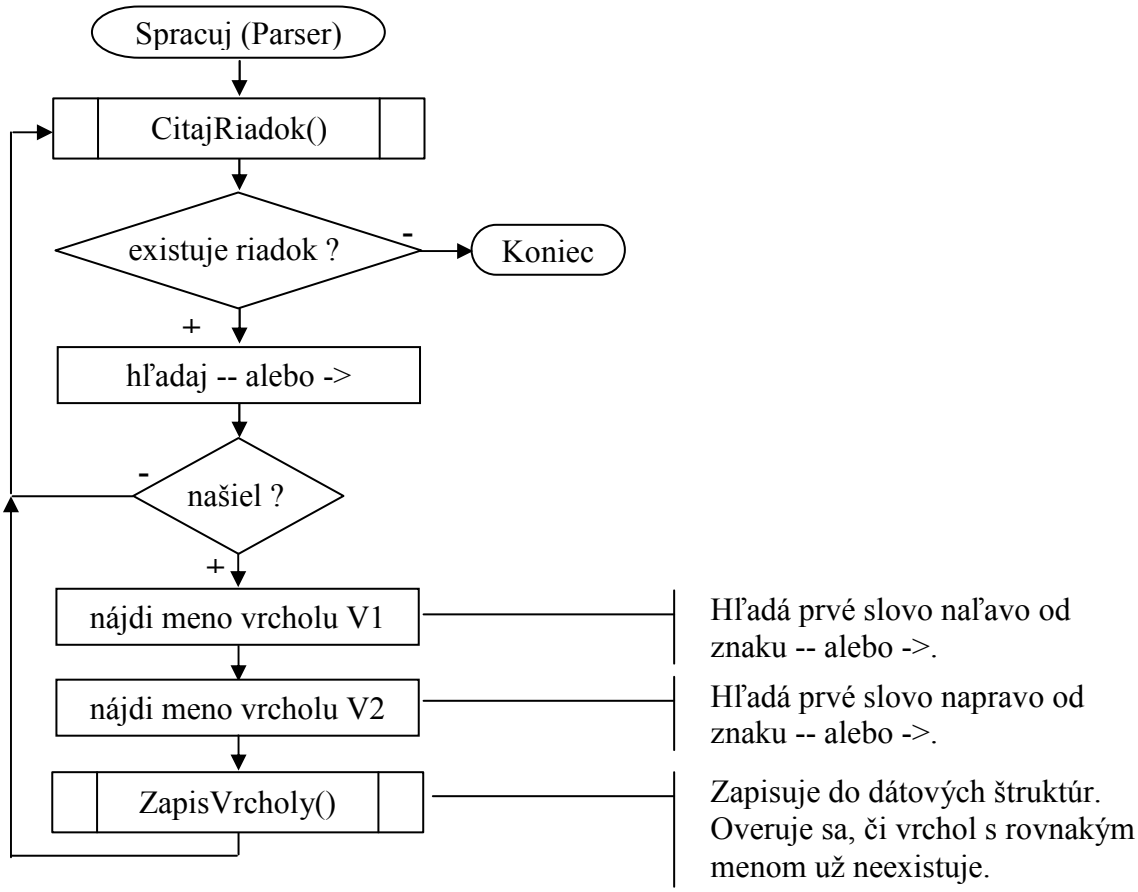
Vstupom do programu je vstupný popisný súbor s príponou dot. Druhá časť číta vstupný textový súbor a vytvára najskôr cenovú maticu grafu a potom maticu vzdialenosti. Ďalší blok – Kohonen rieši samotný problém výpočtu súradníc vrcholov

grafu. Posledný blok programu slúži na zobrazovanie výsledku, teda vykresľuje pozíciu vrcholov na kresliace plátno po inicializácii NN, počas výpočtu a po skončení výpočtu.

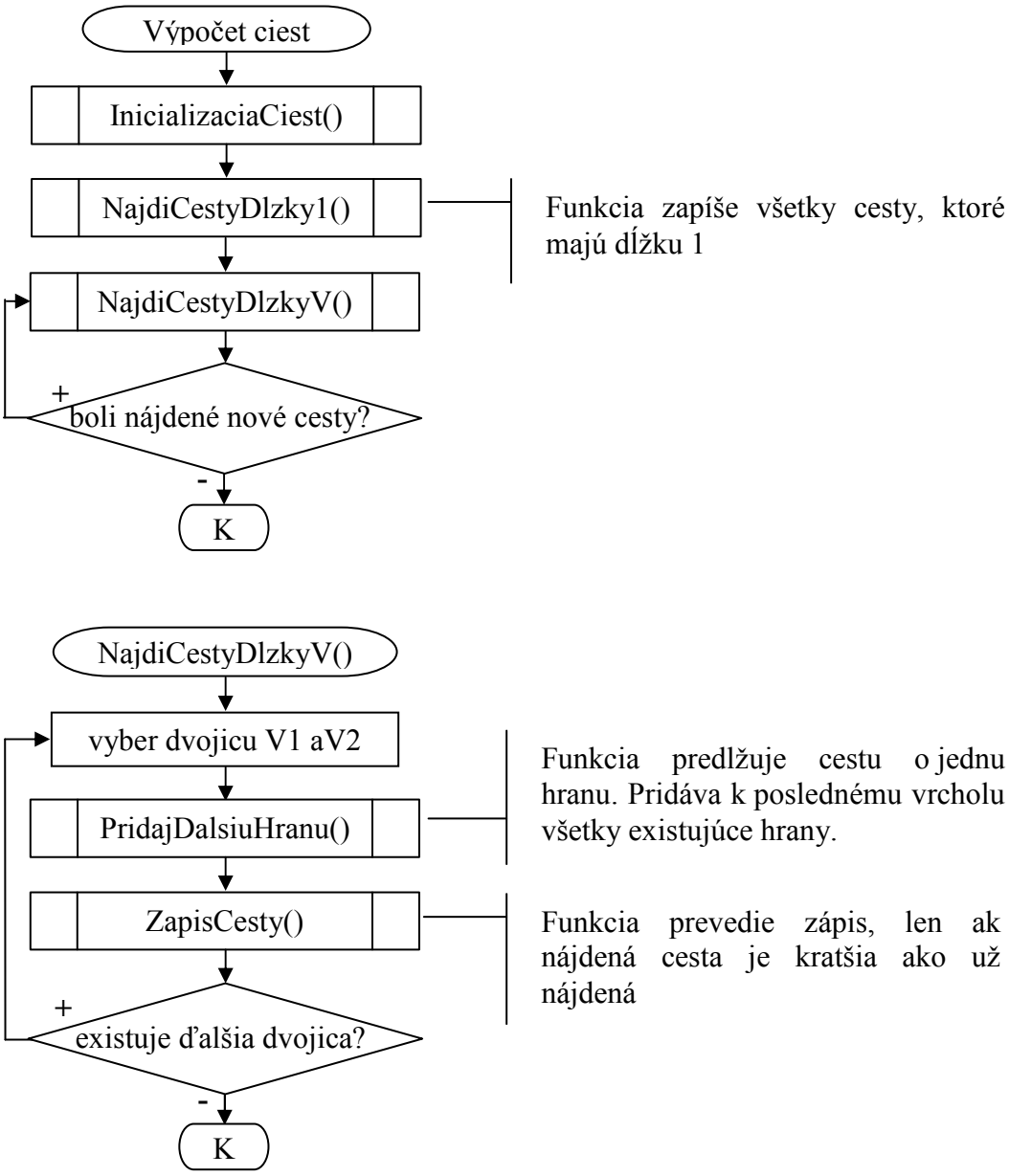
Parser, matica vzdialeností vrcholov v grafe

Program číta vstupný popisný súbor ako textový súbor po riadkoch. Zisťuje údaje o grafe (názov, typ) a eviduje zoznam vrcholov a hrán. Pozri vývojový diagram na Obr. 4-4. Objekt triedy CGraf následne vytvára cenovú maticu (váha každej hrany je 1). V nasledujúcom kroku hľadá najkratšiu cestu z každého do každého vrcholu vlastnou heuristikou a zapisuje sa do matice vzdialeností v objekte triedy CGraf, ktorý následne používajú triedy pre výpočet súradníc. Pozri vývojový diagram na Obr. 4-5.

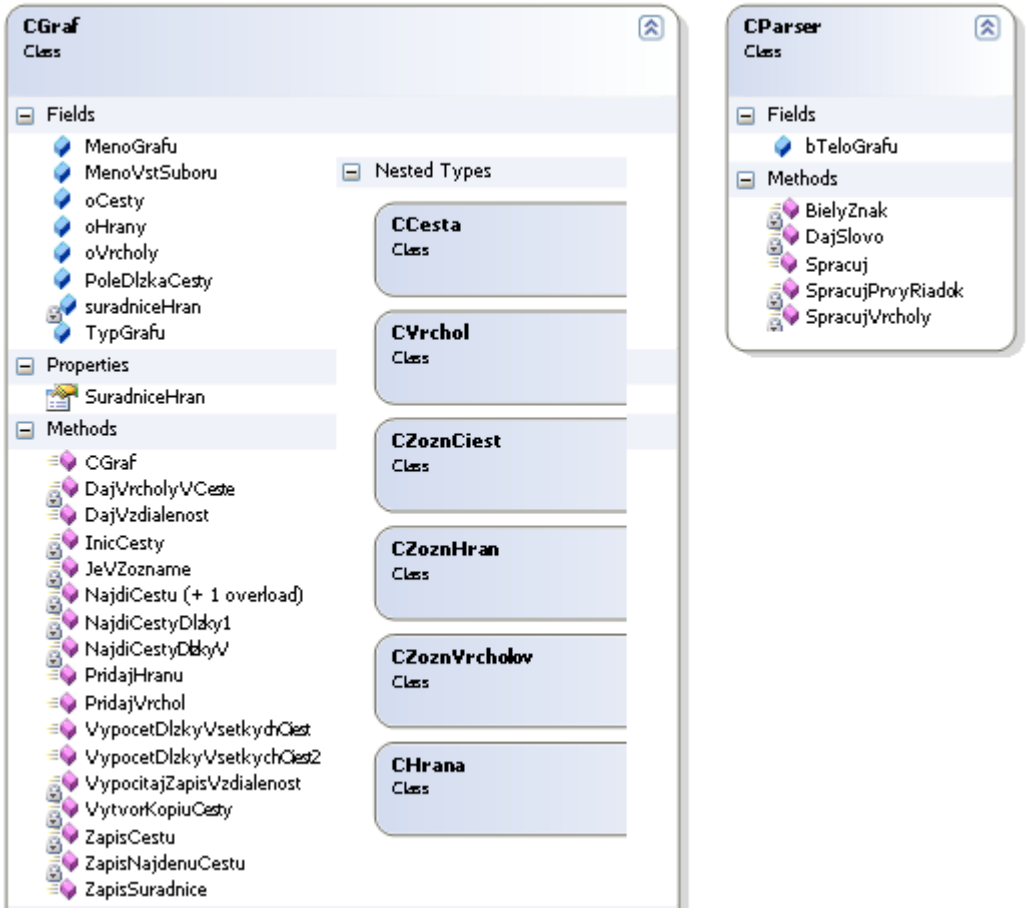
Matica vzdialeností sa vytvára postupne hľadaním cesty z každého do každého vrcholu. Najkratšia cesta jednej dvojice sa hľadá postupným pridávaním existujúcich hrán, pričom sa odpamätávajú všetky cesty. Pridávanie hrán sa vykonáva rekurziou. Po dosiahnutí cieľového vrcholu v každej vetve volania rekurzie sa porovnáva dĺžka práve nájdenej cesty a najkratšej zapísanej cesty. V prípade potreby sa prepíše najkratšia cesta aktuálne nájdenou.



Obr. 4 - 4 Parser - čítanie vstupného popisného súboru, zápis nájdených hrán a vrcholov.



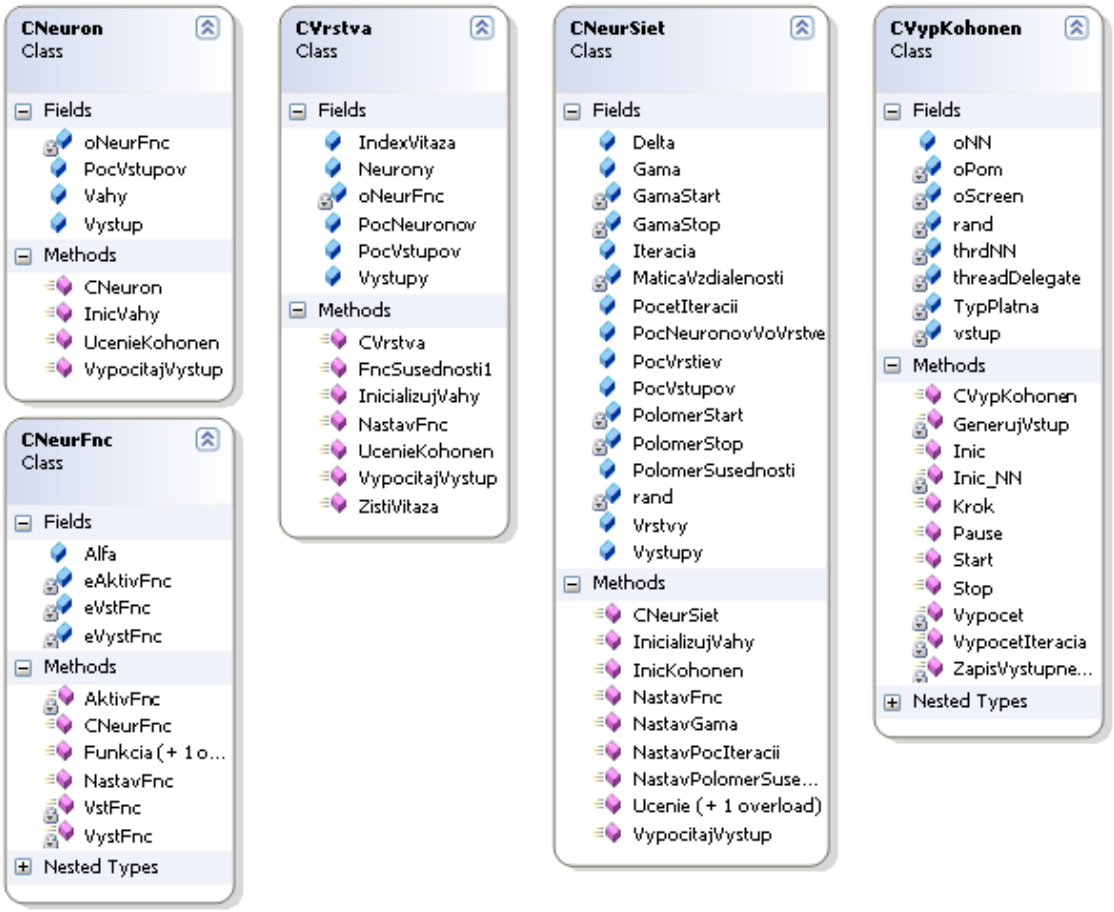
Obr. 4 - 5 Výpočet vzdialeností vrcholov v grafe.



Obr. 4 - 6 UML diagram tried bloku pre výpočet matice vzdialeností. CParser, CGraf sú hlavné triedy pre čítanie údajov zo vstupného popisného súboru a pre výpočet vzdialeností vrcholov v grafe. Diagram zobrazuje dátové prvky – Fields a Properties, funkcie – Methods a vnorené triedy.

Výpočet súradníc vrcholov grafu

Pod výpočtom súradníc teda rozumieme prácu novej rozvrhovacej metódy, ktorá pracuje ako Kohonenova neurónová sieť. V predchádzajúcom kroku program zapísal maticu vzdialeností do objektu triedy CGraf. Rozvrhovacia Kohonenova metóda sa nachádza v triede Kohonen a jej podtriedach. Výpočet uvažuje s koeficientmi, ktoré zadal používateľ v hlavnom pracovnom okne. Celý výpočet je možné krokovať po jednej iterácií. Po každej iterácii program prekreslí aktuálnu pozíciu vrcholov. V tejto časti uvádzame len UML diagram tried a ich členov potrebných na výpočet. Podrobný vývojový diagram algoritmu uvádzame až v kapitole 5.3 na Obr. 5-4. Algoritmus je úplne totožný. Rozdiel je v tom, že v testovacom prostredí sme použili objektové programovanie a riešili sme problémy testovania, napríklad dostupnosť výsledkov v priebehu výpočtu, možnosť zastavenia výpočtu, možnosť krokovania výpočtu a pod.



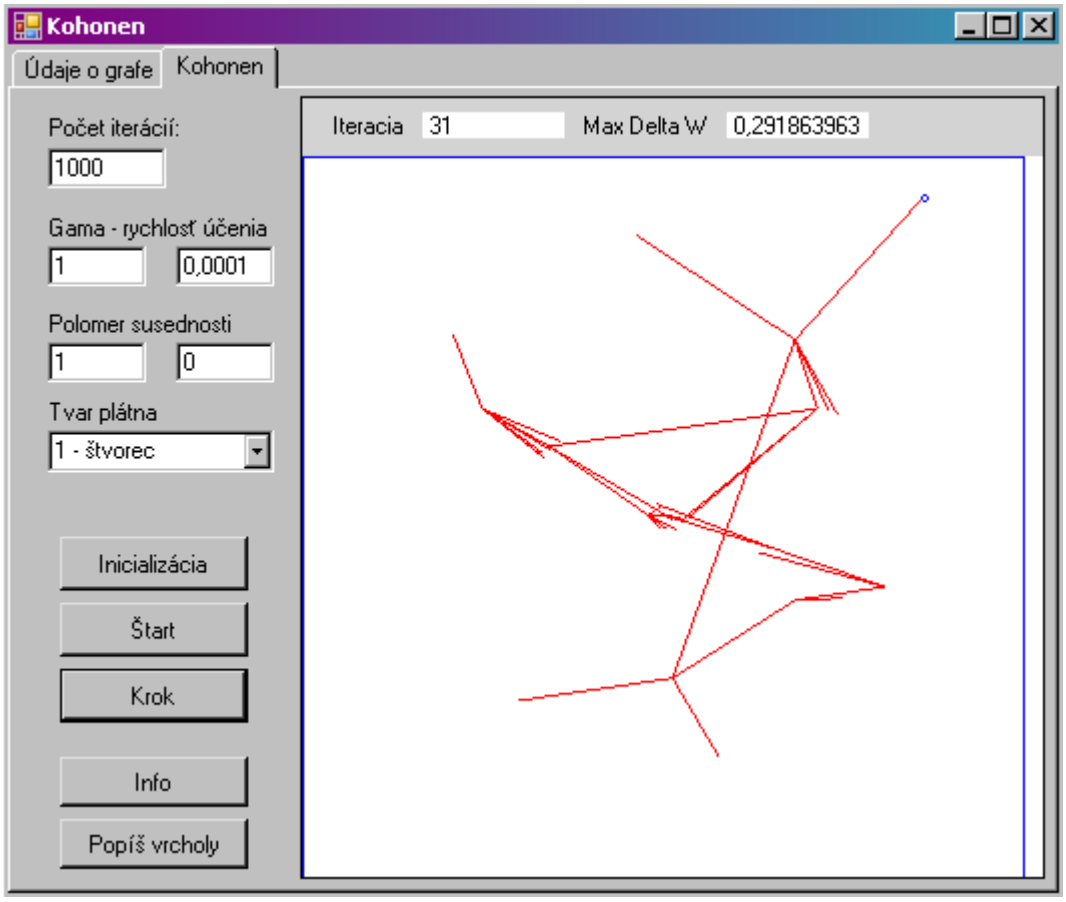
Obr. 4 - 7 UML diagram tried bloku pre Kohonenovu metódu. Uvedené triedy sú navrhnuté všeobecne pre neurónové siete. V našom prípade ide o výpočet súradníc vrcholov v grafe. Diagram zobrazuje dátové prvky – Fields a funkcie – Methods.

Parametre výpočtu a testovacie prostredie

Používateľské rozhranie programu umožňuje jednoducho meniť parametre výpočtu. Ovládacie prvky na zápis parametrov sú: počet iterácií, gama – učiaci pomer alebo rýchlosť učenia, štartovacia a konečná hodnota, polomer susednosti alebo polomer priestorového susedstva, štartovacia a konečná hodnota, tvar plátna na ktoré chceme graf rozprestrieť.

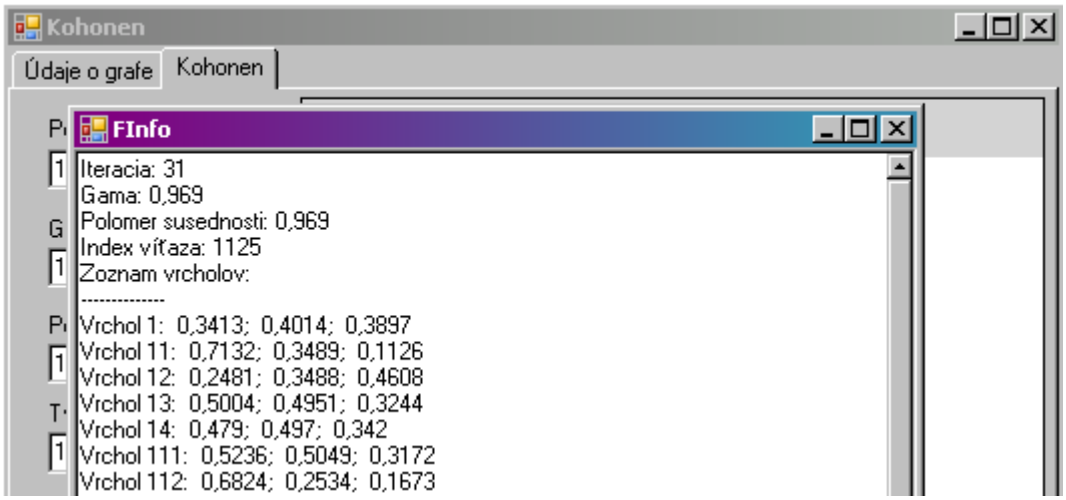
Tlačidlo "Inicializácia", zmaže všetky hodnoty a inicializuje NN, teda jej synaptické váhy, do stredu kresliaceho plátna.

Tlačidlo "Štart" spustí výpočet. Opätovným stlačením daného tlačidla je možné výpočet pozastaviť. Tlačidlom "Krok" je možné výpočet krokovať po jednej iterácii. Tlačidlom "Popíš vrcholy" je možné zobraziť mená vrcholov na kresliacom plátne.



Obr. 4 - 8 Používateľské rozhranie – parametre a priebeh výpočtu Kohonenovej NN. Na ľavej strane sú koeficienty výpočtu, ktoré môže používateľ meniť. Pod nimi sú príkazové tlačidlá na ovládanie výpočtu – odštartovanie, pozastavenie, krokovanie. Nad kresliacim plátnom sa zobrazujú aktuálne hodnoty prebiehajúceho výpočtu.

Tlačidlom "Info" je možné zobraziť pomocnú obrazovku s presnými hodnotami o stave výpočtu: číslo iterácie, aktuálny učiaci pomer, aktuálny polomer priestorovej susednosti, meno víťaza a pre každý vrchol jeho aktuálne váhy (súradnice v rovine) a výstupnú hodnotu.

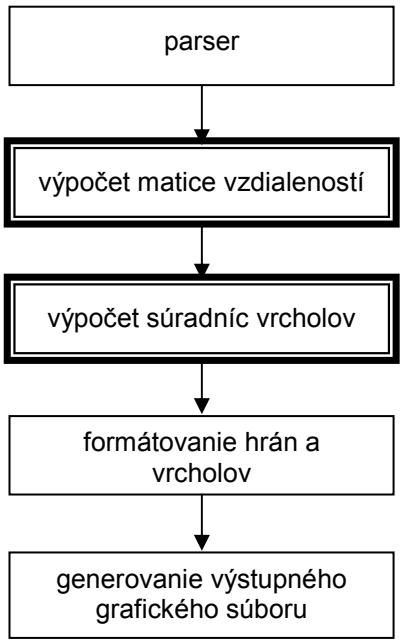


Obr. 4 - 9 Používateľské rozhranie – Informatívna obrazovka o stave výpočtu. Obrazovka zobrazuje, ktorá iterácia práve prebehla. Zobrazuje aktuálne koeficienty: koeficient učenia, polomer priestorového susedstva a index víťazného neurónu. Potom okno zobrazuje ku každému neurónu vypočítanú výstupnú hodnotu a hodnotu synaptických váh zo vstupných neurónov.

5 Implementácia Kohonenovej metódy v prostredí Graphviz-u

Program Graphviz je písaný v programovacom jazyku ANSI C. Nepodporuje objektové programovanie. Má byť skompilovateľný pre väčšinu bežných platforiem. Algoritmus pre výpočet vzdialeností v grafe a algoritmus pre výpočet súradníc musel byť napísaný znovu v neobjektovom jazyku a optimalizovaný pre výkonnosť.

Na nasledujúcom obrázku 5-1 sú znázornené jednotlivé bloky úloh, ktoré musí Graphviz riešiť.



Obr. 5 - 1 Bloková schéma úloh. Zvýraznené bloky rieši táto práca, ostatné sú súčasťou Graphviz-u a používajú ich aj ostatné rozvrhovacie metódy.

Vstupný bod v existujúcom projekte pre implementáciu novej rozvrhovacej metódy je súbor *gvlayout_neato_layout.c*. V ňom je treba rozšíriť enumeračný typ *layout_type* a pridať definíciu:

```
gvlayout_engine_t kohgen_engine = {
          koh_layout,
          koh_cleanup,
    };
```

Potom je treba vytvoriť nový adresár, v ktorom budú implementované dve vstupné funkcie *void koh_layout(Agraph_t * g)* a *void koh_cleanup(Agraph_t * g)*.

Graphviz má zadefinované niektoré externé funkcie, ktoré uľahčujú prístup k údajom o grafe. V našom prípade sa tieto funkcie stali rozhraním medzi Graphviz-om a našou prácou. Pozri tab. 5-1. Vypočítané súradnice sa zapíšu do štruktúry *Agraph_t* pomocou zadefinovaných makier. Pozri tab. 5-2

Tab. 5 - 1 Interfejs medzi Graphviz-om a touto prácou. Funkcie pre čítanie vlastností vrcholov a hrán.

Počet vrcholov v grafe
int agnnodes(Agraph_t *);
Funkcie pre vrcholy v grafe
extern Agnode_t *agnode(Agraph_t *, char *);
extern Agnode_t *agfindnode(Agraph_t *, char *);
extern Agnode_t *agfstnode(Agraph_t *);
extern Agnode_t *agnxtnode(Agraph_t *, Agnode_t *);
extern Agnode_t *aglstnode(Agraph_t *);
extern Agnode_t *agprvnode(Agraph_t *, Agnode_t *);
Funkcie pre hrany v grafe
extern Agedge_t *agedge(Agraph_t *, Agnode_t *, Agnode_t *);
extern Agedge_t *agfindedge(Agraph_t *, Agnode_t *, Agnode_t *);
extern Agedge_t *agfstedge(Agraph_t *, Agnode_t *);
extern Agedge_t *agnxtedge(Agraph_t *, Agedge_t *, Agnode_t *);
extern Agedge_t *agfstin(Agraph_t *, Agnode_t *);
extern Agedge_t *agnxtin(Agraph_t *, Agedge_t *);
extern Agedge_t *agfstout(Agraph_t *, Agnode_t *);
extern Agedge_t *agnxtout(Agraph_t *, Agedge_t *);

Tab. 5 - 2 Interfejs medzi Graphviz-om a touto prácou. Makrá pre zápis vypočítaných súradníc vrcholov.

ND_pos(nod)[0];	prvá súradnica vrcholu nod
ND_pos(nod)[1];	druhá súradnica vrcholu nod

Pre potreby našich výpočtov sme zadefinovali jednu štruktúru a dve lokálne automatické premenné:

```
typedef struct {
    double w1, w2;     /* synaptické váhy */
    double y;          /* výstup z neurónu */
} Neuron;
Neuron *neu;           /* pole neurónov v Kohonenovej vrstve */
int *adjM;             /* matica susednosti - vzdialenosti */
```

5.1 Parser vstupného popisného súboru

Parser je časť Graphviz-u, ktorá číta vstupný popisný súbor. Všetky informácie o grafoch, vrcholoch, hranách a ich formátovacích vlastnostiach zapisuje do štruktúry *Agraph_t*.

V testovacej aplikácii sme túto časť museli riešiť my, pretože Graphviz neposkytuje riešenie takýchto dielčích problémov. Testovacia aplikácia číta vstupný textový súbor riadok za riadkom a hľadá kľúčové slová a symboly. Potom nájdené objekty ukladá do údajových štruktúr.

Túto časť programu neupravujeme, používame ju tak ako je implementovaná v Graphviz-e.

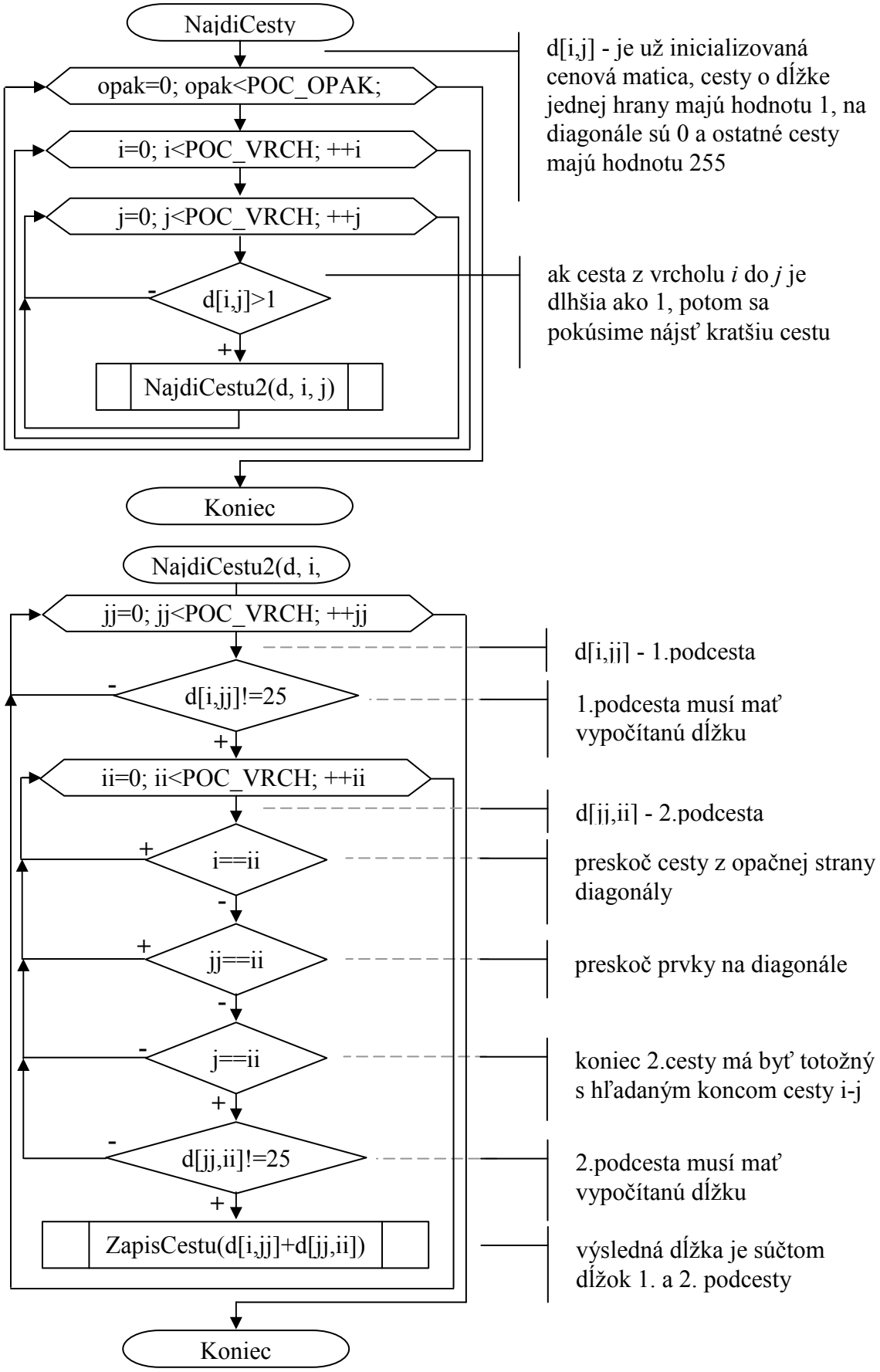
5.2 Výpočet matice vzdialeností vrcholov v grafe

Napriek tomu, že Graphviz už má implementované niektoré algoritmy na určovanie dĺžky cesty v grafe (Dijkstra), navrhli sme vlastný algoritmus kvôli optimalizácií výkonu.

Najskôr sme zostrojili cenovú maticu s tým, že váha každej hrany má hodnotu 1.

Algoritmus na výpočet vzdialeností je znázornený na obrázku 5-2. Algoritmus sa pokúša z dvoch podciest poskladať ďalšiu cestu pre iné vrcholy. Prechádza sa zaradom celá matica. Maticu stačí prejsť dvakrát, aby sme určite našli všetky cesty dĺžky dva. Podľa použitej funkcie susednosti (pozri Obr. 3-4) cesty s väčšou dĺžkou výpočet neovplyvnia.

Výsledky sú uložené do dvojrozmernej matice celých čísel - *adjM*.



Obr. 5 - 2 Výpočet vzdialeností vrcholov v grafe.

```
/* compute adjacency matrix */
void init_adjM(Agraph_t *g, int *d, int count)
{
    int i, j;
    for(i=0; i<count; ++i)
        for(j=0; j<count;++j)
            d[i*count+j] = ((i==j) ? 0 : 255);

    node_t *n1, *n2;
    for(i=0, n1=agfstnode(g); n1; ++i, n1=agnxtnode(g, n1)) {
        for(j=0, n2=agfstnode(g); n2; ++j, n2=agnxtnode(g, n2)) {
            if(n1 == n2)
                continue;
            if(agfindedge(g, n1, n2))
                d[i*count+j] = d[j*count+i] = 1;
        }
    }
}

/* find path for all pairs nodes of graph */
void findPaths(int *d, int count)
{
    int rep, i, j;
    for (rep = 0; rep < COUNT_REPEAT_FIND_PATH; ++rep)
        for (i = 0; i < count; ++i)
            for (j = i + 1; j < count; ++j)
                if (d[i*count+j] > 1) /* paths with lenght more than 1
*/
                    findPaths2(d, i, j, count);
}

/* find path from i to j; count - count nodes in graph */
/* d - adjacency matrix */
void findPaths2(int *d, int i, int j, int count)
{
    int jj, ii;
    for (jj = 0; jj < count; ++jj){
        /* 1st subpath */
        if (d[i*count+jj] != 255) {
            for (ii = 0; ii < count; ++ii){
                /* 2nd subpath */
                if (i == ii || jj == ii)    //skip opposite edge and diagonal
                    continue;
                /* 2nd subpath & for write have to first and last index equal */
                if (d[jj*count+ii] != 255 && j == ii)

writePath(d,i,j,d[i*count+jj]+d[jj*count+ii],count);
            }
        }
    }
}

/*write lenght of shortest path from i to j */
void writePath(int *d, int i, int j, int value, int count)
{
    if (value > 255)
        value = 255;
    if (d[i*count+j] > value)
        d[i*count+j] = d[j*count+i] = value;
}
```

Obr. 5 - 3 Úplný výpis kódu - výpočet vzdialenosti vrcholov v grafe.

5.3 Výpočet súradníc vrcholov - Kohonenov algoritmus

Táto časť programu je jadrom navrhovaného riešenia. Kohonenov algoritmus je navrhnutý s ohľadom na maximálnu optimalizáciu výkonu.

Maximálnu hodnotu zmeny váh nesledujeme, hoci sa zvykne používať ako jedna z ukončovacích podmienok. Ukončovacou podmienkou v našom prípade je počet iterácií.

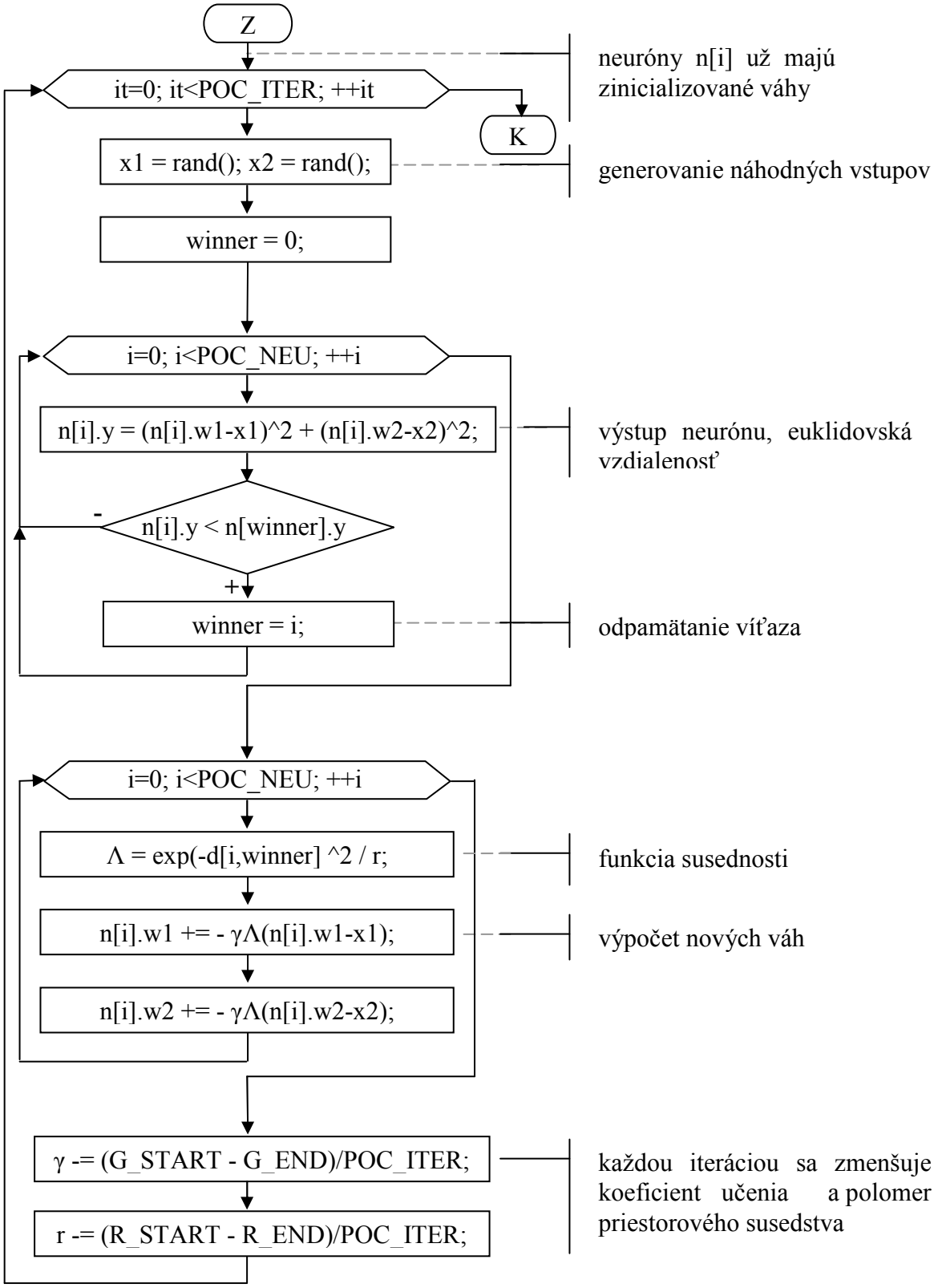
Polomer priestorového susedstva r a koeficient učenia γ sa lineárne menia počas celého výpočtu od začiatočných po konečné hodnoty zadefinované v konštantách.

Pre výpočet potrebujeme vzdialenosti medzi všetkými vrcholmi v grafe a tie máme uložené v premennej *ajdM*. Pre výpočet ešte potrebujeme pole štruktúr *SNeuron*. V jednej štruktúre *SNeuron* máme uložené potrebné údaje k jednému neurónu v Kohonenovej vrstve, teda váhy k dvom vstupným neurónom a výstupnú hodnotu neurónu.

Vývojový diagram algoritmu je na Obr. 5-4 a výpis zdrojového kódu na Obr. 5-5.

Vypočítané súradnice vrcholov sú rozmiestnené do štvorca jeden krát jeden palec. Tento rozmer je daný nasledovným spracovaním v Graphviz-e, konkrétne funkciou *spline_edges (Agraph_t*)*, kde sa palce prepočítavajú na pixely.

Na konci výpočtu kresliace plátno prenásobíme číslom, ktoré sme získali ako odmocninu z počtu vrcholov prenásobenú tromi. Toto číslo sme získali experimentálnym porovnávaním.



Obr. 5 - 4 Kohonenov algoritmus na výpočet súradníc vrcholov v grafe. Pred vstupom do tohto bloku, váhy sú už inicializované na náhodného hodnoty. V prvej časti tohto bloku sa vypočítavajú výstupné hodnoty neurónov a hľadá sa víťaz. V druhej časti tohto bloku sa prepočítavajú synaptické váhy na základe vzdialenosti vrcholov, koeficientu učenia a polomeru priestorového susedstva.

```
/* initialize weight of neurons */
void randomWeight(Neuron *neu, int count)
{
    #ifdef MSWIN32
        srand((unsigned int)time(NULL));
    #else
        srand(getpid() ^ time(NULL));
    #endif

    int i;
    for(i=0; i<count; ++i) {
        neu[i].w1 = (rand()%10000)/10000.0;
        neu[i].w2 = (rand()%10000)/10000.0;
        neu[i].w1 = (neu[i].w1*0.05)+0.475; /* only small rectangle in center
        neu[i].w2 = (neu[i].w2*0.05)+0.475; /* only small rectangle in center
    }
}
void learnKohonenNetwork(int *d, Neuron *n, int count)
{
    double gama = GAMA_START;          /* rate of learning */
    double radius = RADIUS_START;      /* radius for neighborhood function */
    int distance;                      /* distance between nodes - count edges
    double lambda;                     /* */
    double x1, x2;                     /* value of input neurons */
    int winner;                        /* index of winner */

    #ifdef MSWIN32
        srand((unsigned int)time(NULL));
    #else
        srand(getpid() ^ time(NULL));
    #endif

    int iteration, i;
    for(iteration=0; iteration<COUNT_ITERATION; ++iteration) {
        /* generate input of kohonen */
        x1 = (rand()%10000)/10000.0;
        x2 = (rand()%10000)/10000.0;
        /* compute  output */
        winner = 0;
        for(i=0; i<count; ++i) {
            /* Euclidean distance */
            n[i].y=(n[i].w1-x1)*(n[i].w1-x1)+(n[i].w2-x2)*(n[i].w2-x2)
            if (n[i].y < n[winner].y)
                winner = i;
        }
        /* change weight for all neurons*/
        for(i=0; i<count; ++i) {
            distance = d[i*count+winner];
            /* neighborhood function */
            lambda = exp(-1.0 * distance * distance / radius);
            n[i].w1 += -1.0 * (gama * lambda * (n[i].w1 - x1));
            n[i].w2 += -1.0 * (gama * lambda * (n[i].w2 - x2));
        }
        /* change gama, radius */
        if (gama > GAMA_END)
            gama -= (GAMA_START - GAMA_END) / COUNT_ITERATION;
        if (radius > RADIUS_END)
            radius -= (RADIUS_START - RADIUS_END) / COUNT_ITERATION;
    }
}
```

Obr. 5 - 5 Úplný výpis kódu - Kohonenov algoritmus na výpočet súradníc vrcholov v grafe.

6 Experimentálna analýza rozmiestňovania vrcholov grafu Kohonenovou sieťou.

6.1 Overenie funkčnosti rozmiestňovania vrcholov grafu Kohonenovou sieťou

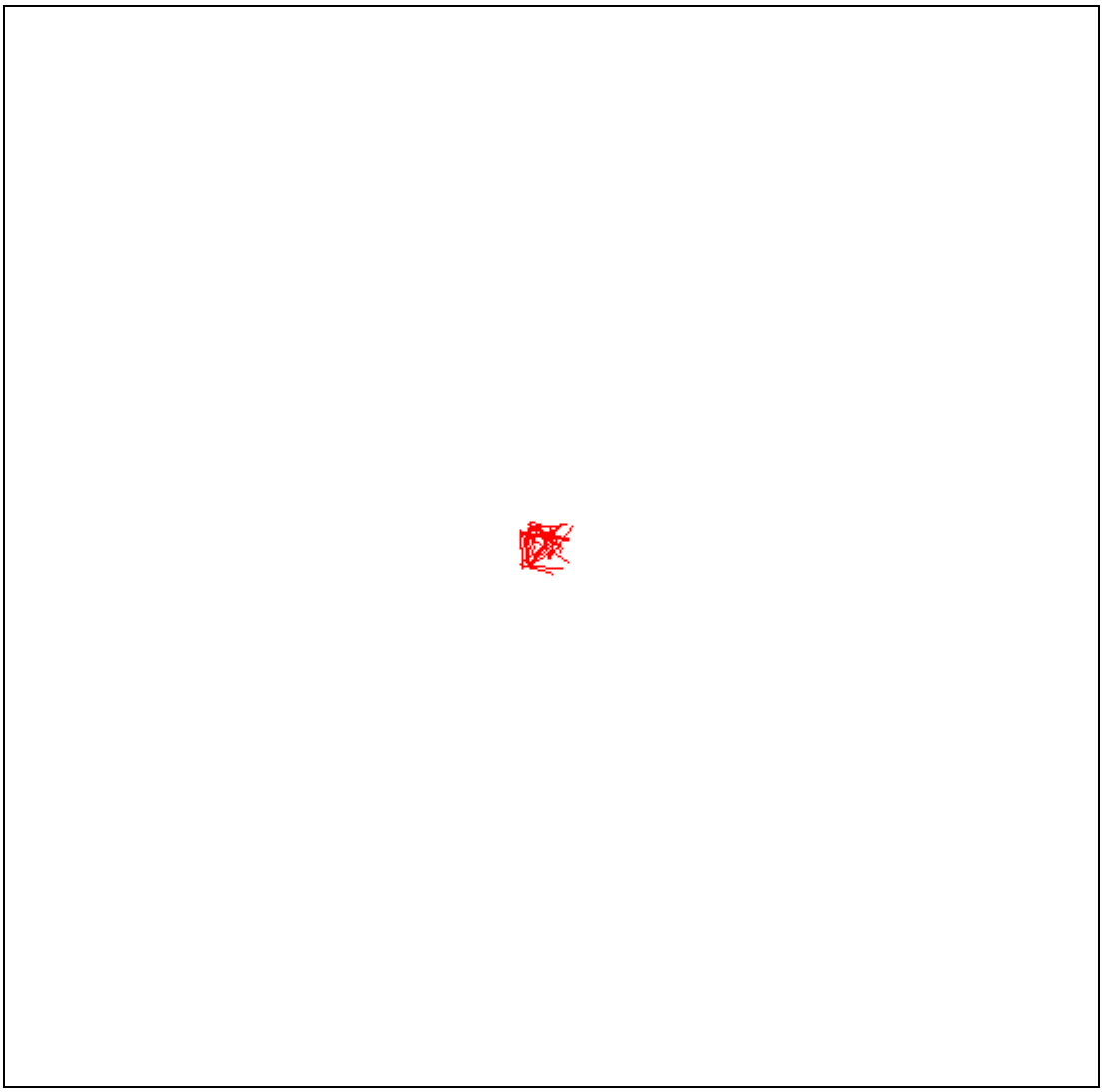
Prvou časťou experimentov je overenie funkčnosti novej rozvrhovacej metódy. Od metódy očakávame, že rozmiestni vrcholy na celé kresliace plátno, pričom vrcholy, ktoré majú spoločné hrany budú bližšie k sebe a očakávame, že metóda eliminuje križovania hrán.

Na overenie sme použili testovacie prostredie, ktoré sme vytvorili k tomuto účelu. Program používa jednoduché vykresľovanie grafu, vrchol je znázornený bodom a hrana je znázornená úsečkou, ktorá spája dva vrcholy.

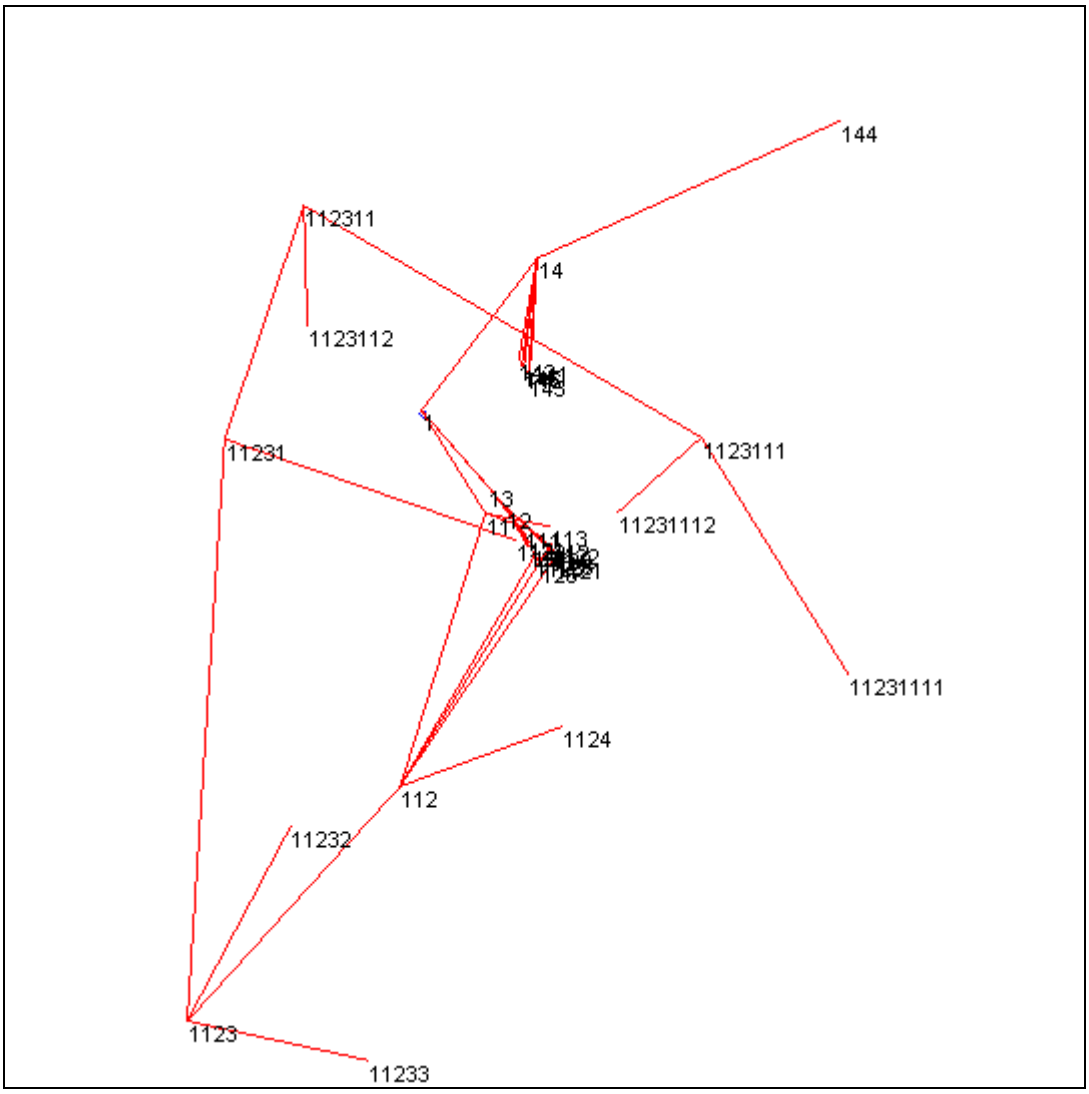
Vrcholy predstavujú neuróny v Kohonenovej vrstve. Zakreslené sú na základne súradníc x a y, v tomto prípade súradnice sú totožné so synaptickými váhami k vstupným dvom neurónom. Zakreslené hrany v tomto prípade predstavujú skutočné hrany v grafe. Na začiatku výpočtu sa súradnice x a y, teda synaptické váhy k vstupným neurónom, inicializujú do malej oblasti v strede kresliaceho plátna. Výpočet, učenie NN, spočíva v privádzaní náhodných hodnôt na vstupné neuróny, tieto hodnoty majú pokrývať celé kresliace plátno, kde chceme diagram grafu rozprestrieť. Výpočet končí po dosiahnutí požadovaného počtu iterácií.

Parametre výpočtu kohonenovej NN siete sme nastavili na bežne používané hodnoty: počet iterácií = 1000, $\gamma = <1.0; 0{,}0001>$, $r = <1.0; 0.0>$.

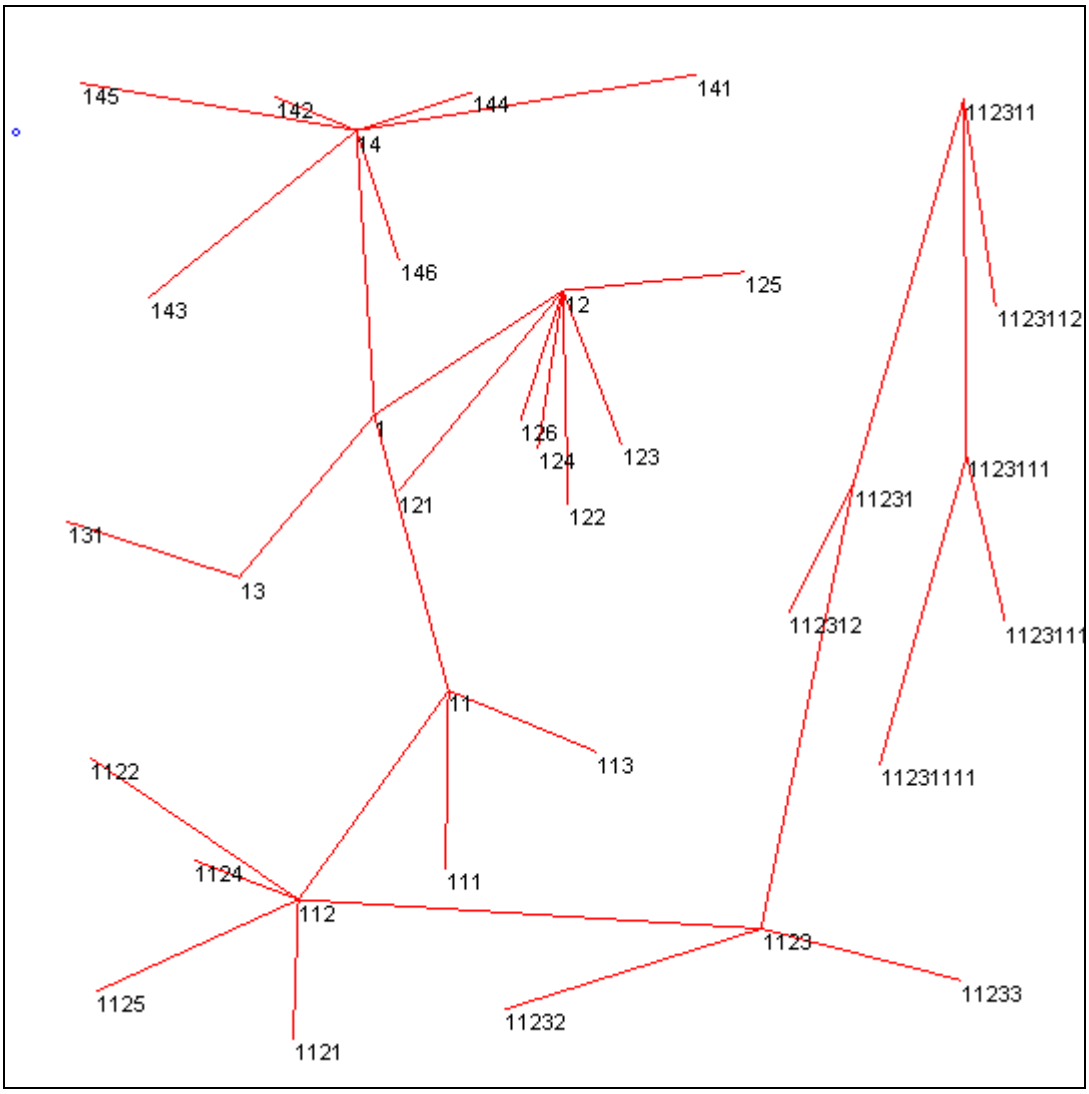
Opakovanými experimentmi sa nám nepodarilo potvrdiť, že iné hodnoty by mohli dosiahnuť lepšie výsledky.



Obr. 6 - 1 Kohonenova rozvrhovacia metóda. Kresliace plátno po inicializácii. (Prvý zo série Obr. 6-1 až 6-3). Vrcholy na obrázku predstavujú neuróny v Kohonenovej vrstve. Zakreslené sú na základne súradníc *x* a *y*, v tomto prípade súradnice sú totožné so synaptickými váhami k vstupným dvom neurónom. Zakreslené hrany v tomto obrázku prestavujú skutočné hrany v grafe. Na začiatku výpočtu sa súradnice x a y, teda synaptické váhy k vstupným neurónom, inicializujú do malej oblasti v strede kresliaceho plátna. Tým by sa malo dosiahnuť plynulejšie rozprestretie vrcholov na celé plátno.



Obr. 6 - 2 Kohonenova rozvrhovacia metóda. Kresliace plátno po 20 iteráciách. (Druhý zo série Obr. 6-1 až 6-3).



Obr. 6 - 3 Kohonenova rozvrhovacia metóda. Kresliace plátno po dokončení výpočtu. (Tretí zo série Obr. 6-1 až 6-3).

Experimenty potvrdili, že metóda rozmiestňuje vrcholy na celé plátno. Vrcholy ktoré majú spoločné hrany sú bližšie k sebe a križovanie hrán je relatívne minimálne. Metóda nedokáže úplne odstrániť zbytočné križovania hrán, zbytočnú blízkosť vrcholov pri hranách a zbytočne ostré uhly.

6.2 Porovnanie výsledkov navrhnutej metódy oproti vybraným metódam Graphviz-u

Druhá časť experimentov mala za cieľ porovnať výsledky rozvrhovacích metód vizuálne, z hľadiska estetiky a čitateľnosti. Experimenty by tiež mali ukázať, ktoré grafy sú vhodnejšie na vizualizáciu Kohonenovou rozvrhovacou metódou. Očakávame že, výsledky budú porovnateľné s DOT, CIRCO, FDP, NEATO a TWOPI metódami.

Vybrali sme päť grafov na porovnávanie.

RS - je rozhodovací strom, Obr. 6-4 až 6-6,

NN - je graf neurónovej siete s dvoma skrytými vrstvami, Obr. 6-7 až 6-9,

Graf1 - testovací graf č. 1, má 18 vrcholov a 59 hrán, Obr. 6-10 až 6-14,

Graf2 - testovací graf č. 2, má 32 vrcholov a 54 hrán, Obr. 6-15 až 6-17,

Graf3 - testovací graf č. 3, má 129 vrcholov a 213 hrán, Obr. 6-18 až 6-20.

Prvé dva grafy sme zadefinovali ručne, napísaním popisných súborov. Posledné tri grafy reprezentujú zdrojové súbory v programovacom jazyku C, ich definície (súbory dot) boli vygenerované softvérom Doxygen. Tieto grafy obsahujú aj formátovacie vlastnosti vrcholov a hrán. Prvý z nich je najjednoduchší, posledný najzložitejší.

Testovanie prebiehalo v prostredí Graphviz-u, kde je implementovaná aj nová navrhnutá metóda. Programový projekt sme skompilovali v prostredí Linux (Ubuntu 8) a používali sme tiež binárne súbory pre MS Windows uverejnené na domovskej stránke Graphviz-u. Vygenerované diagramy grafov sú vo formáte gif.

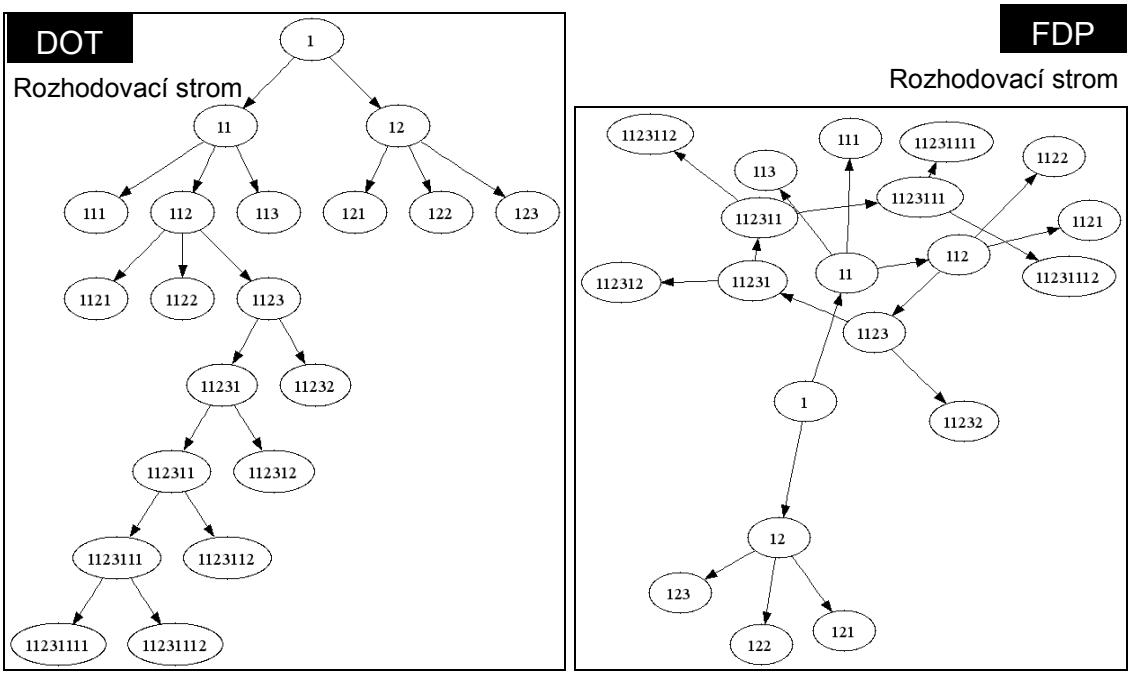
Z Kohonenovej metódy vyplýva, že každé spustenie algoritmu dosiahne iný výsledok. Preto pre každý graf sme vytvorili minimálne tri diagramy. Pre každú metódu Graphviz-u sme vygenerovali po jednom diagrame, hoci v nasledujúcich podkapitolách nie vždy všetky uvádzame.

6.2.1 Testovacie dáta: rozhodovací strom

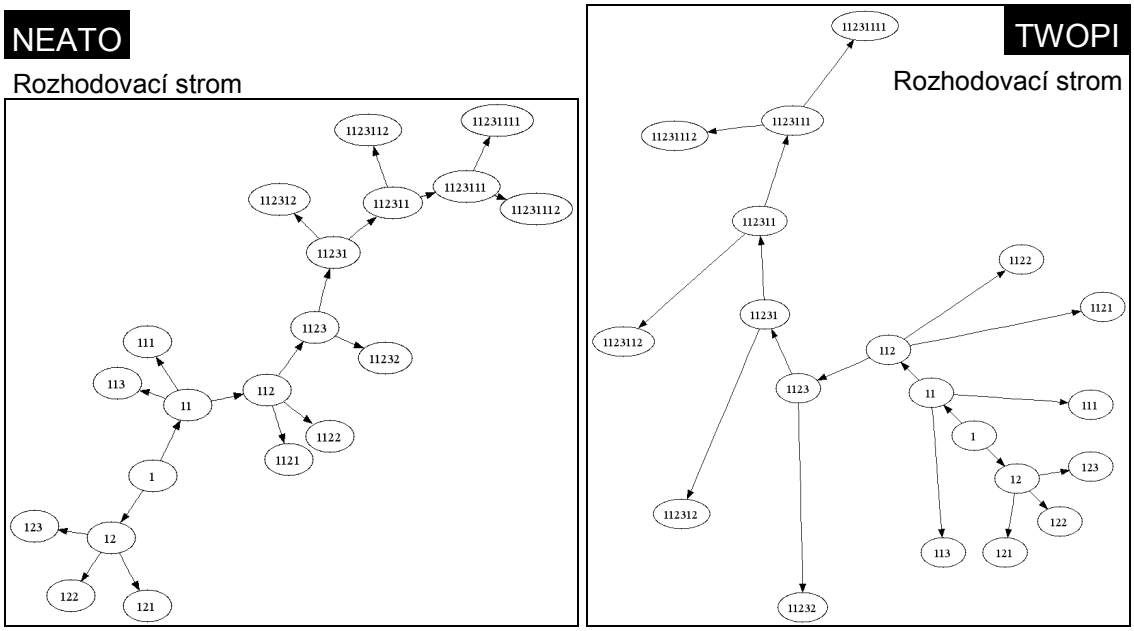
Výsledky vizualizácie sú na Obr. 6-4 až 6-6. V prvom experimente v tejto časti sme použili rozhodovací strom ako značne jednoduchý graf. Metóda CIRCO túto úlohu nezvládol, vygeneroval obrázok o veľkosti 24 000 x 16 000 pixlov. Metóda FDP (Obr. 6-4) vytvoril päť zbytočných križovaní.

KOHONEN (Obr. 6-6) v treťom prípade vytvoril dve zbytočné križovania a v prvých dvoch prípadoch vygeneroval po jednom veľmi ostrom uhle.

Kohonenova rozvrhovacia metóda je založená na náhodných vstupných hodnotách výpočtu, preto každý výpočet vygeneruje iný výsledok. Z toho dôvodu v tejto práci uvádzame niekedy viacero výsledkov na porovnanie.

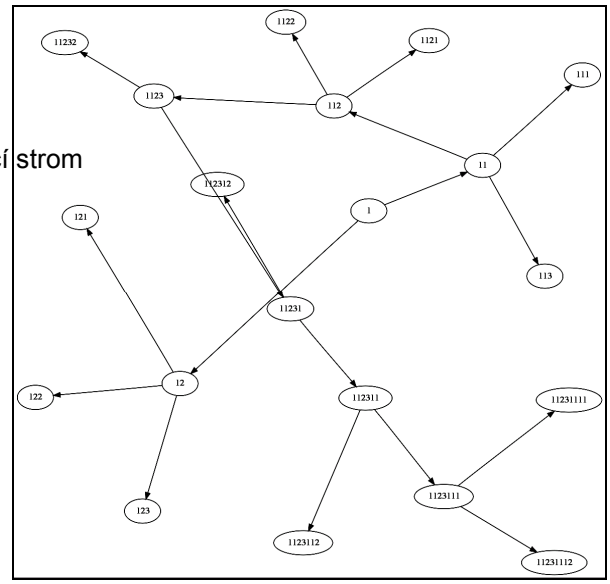


Obr. 6 - 4 Výsledok metód DOT a FDP na dátach rozhodovací strom.

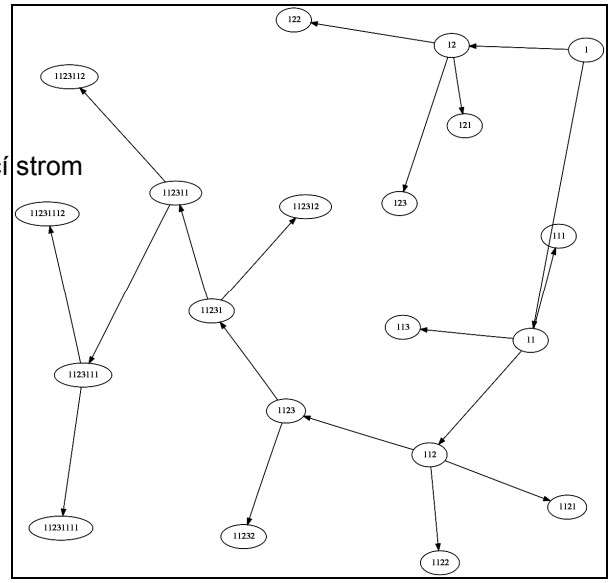


Obr. 6 - 5 Výsledok metód NEATO a TWOPI na dátach rozhodovací strom.

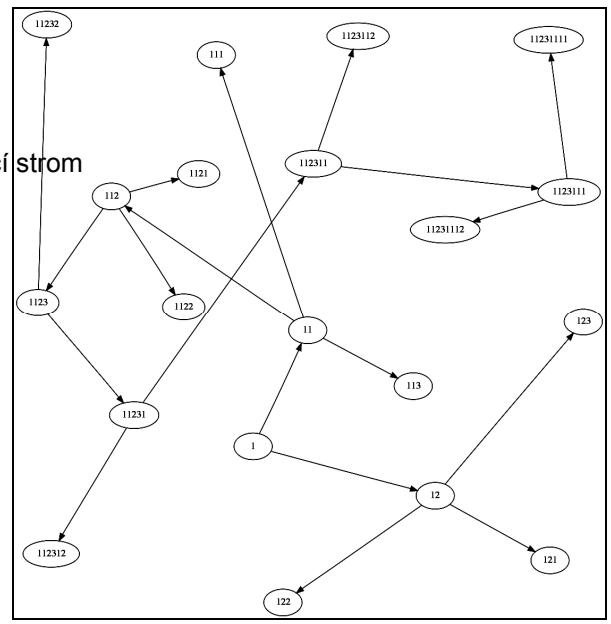
KOH
Rozhodovací strom



KOH
Rozhodovací strom



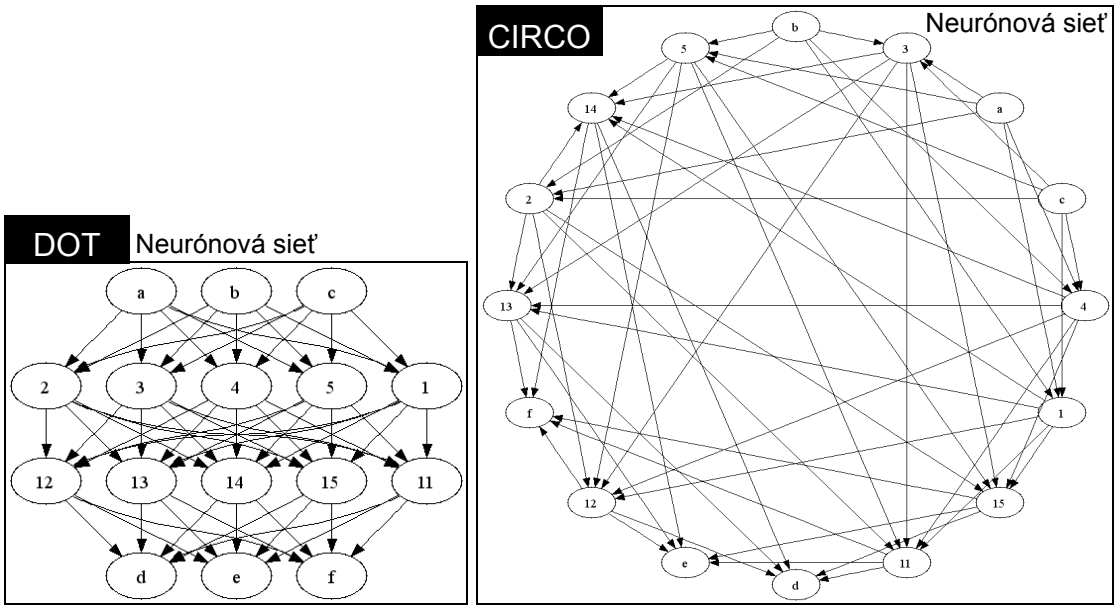
KOH
Rozhodovací strom



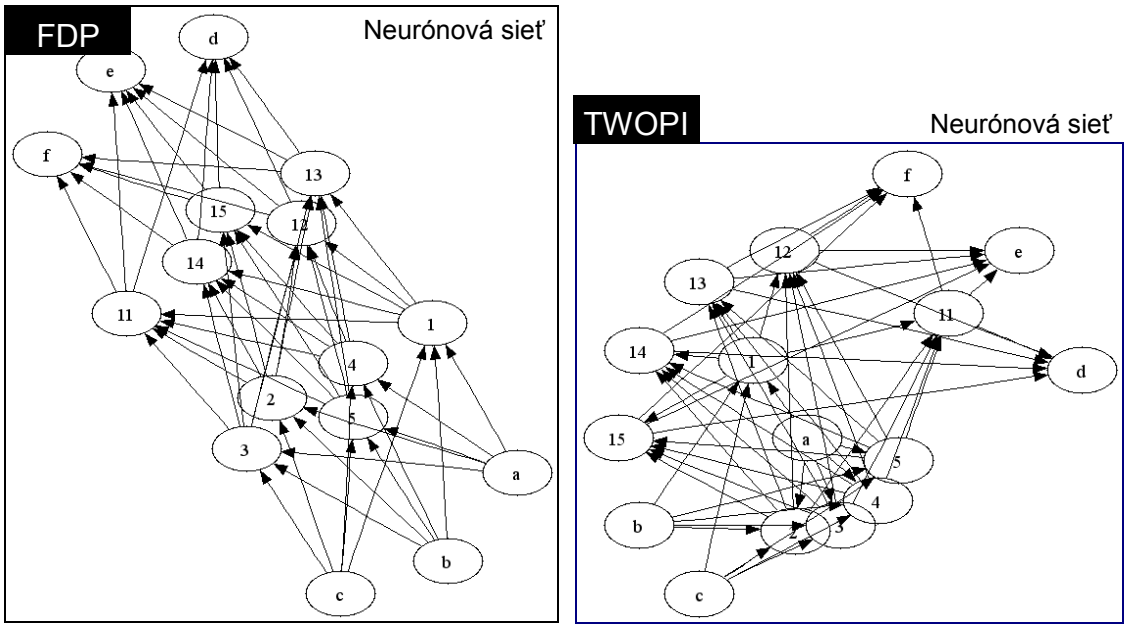
Obr. 6 - 6 Výsledok metódy KOHONEN na dátach rozhodovací strom. Tri rozdielne výsledky metódy. Vstupy do algoritmu sú náhodné hodnoty. Preto výsledok každého výpočtu je rôzny.

6.2.2 Testovacie dáta: neurónová sieť

Výsledky vizualizácie sú na Obr. 6-7 až 6-9. Graf neurónovej siete, hoci nie je rozsiahly, ale má veľa hrán, spôsobil problémy aj pružinovému FDP a hviezdicovitému TWOPI (Obr. 6-8). Hoci sú náznaky, že KOHONEN (Obr. 6-9) sa snaží jednotlivé vrstvy NN lokalizovať na jedno miesto, výsledok nemá takmer žiadnu alebo slabú vypovedaciu schopnosť o NN. Z uvedeného môžeme konštatovať, že KOHONEN nie je celkom vhodný pre grafy s veľkým počtom hrán a pre grafy, ktoré majú vyjadrovať hierarchiu.

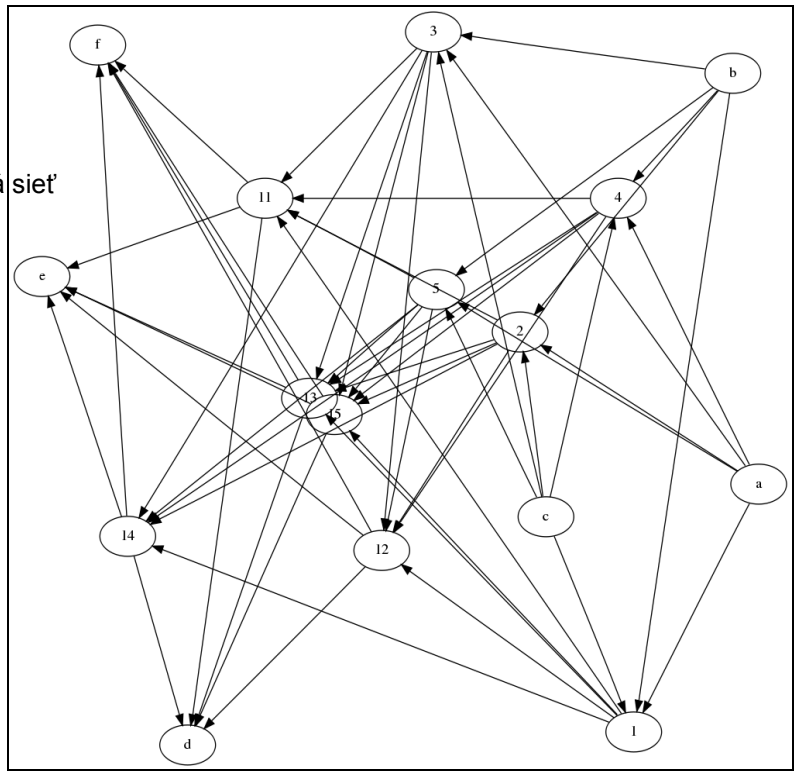


Obr. 6 - 7 Výsledok metód DOT a CIRCO na dátach neurónová sieť.

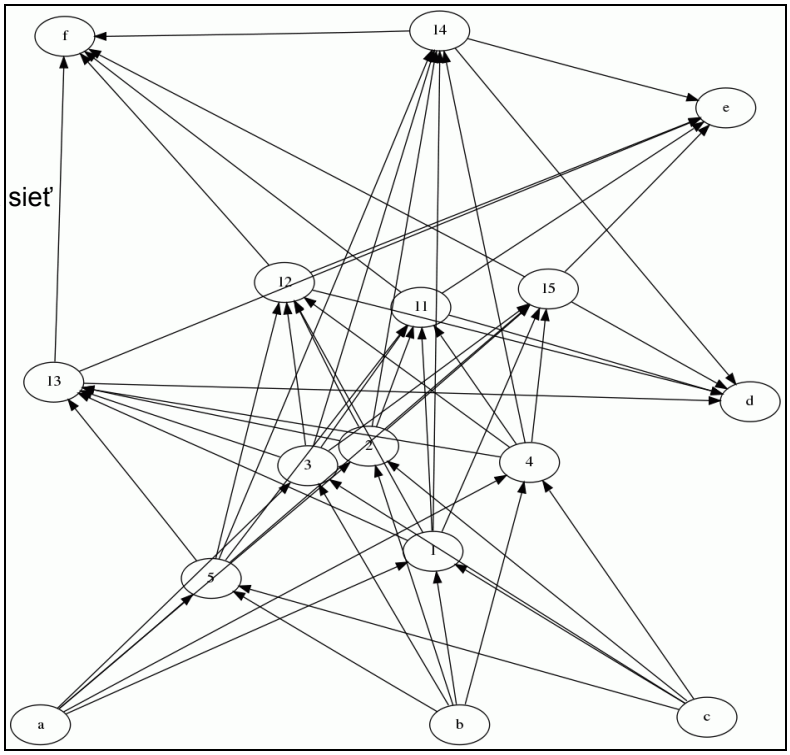


Obr. 6 - 8 Výsledok metód FDP a TWOPI na dátach neurónová sieť.

KOH
Neurónová sieť



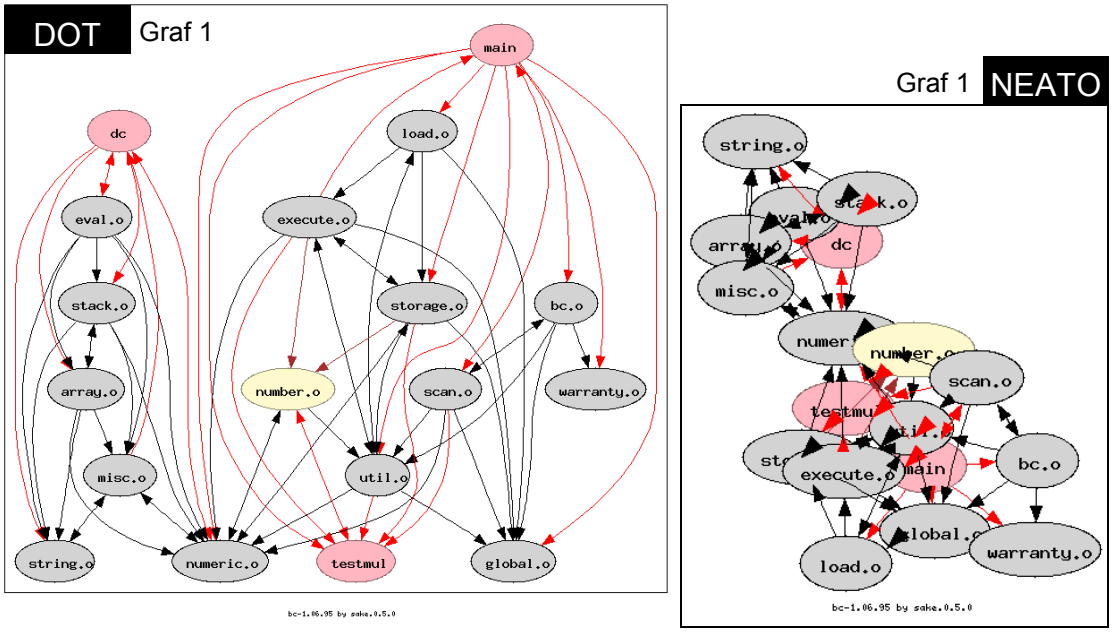
KOH
Neurónová sieť



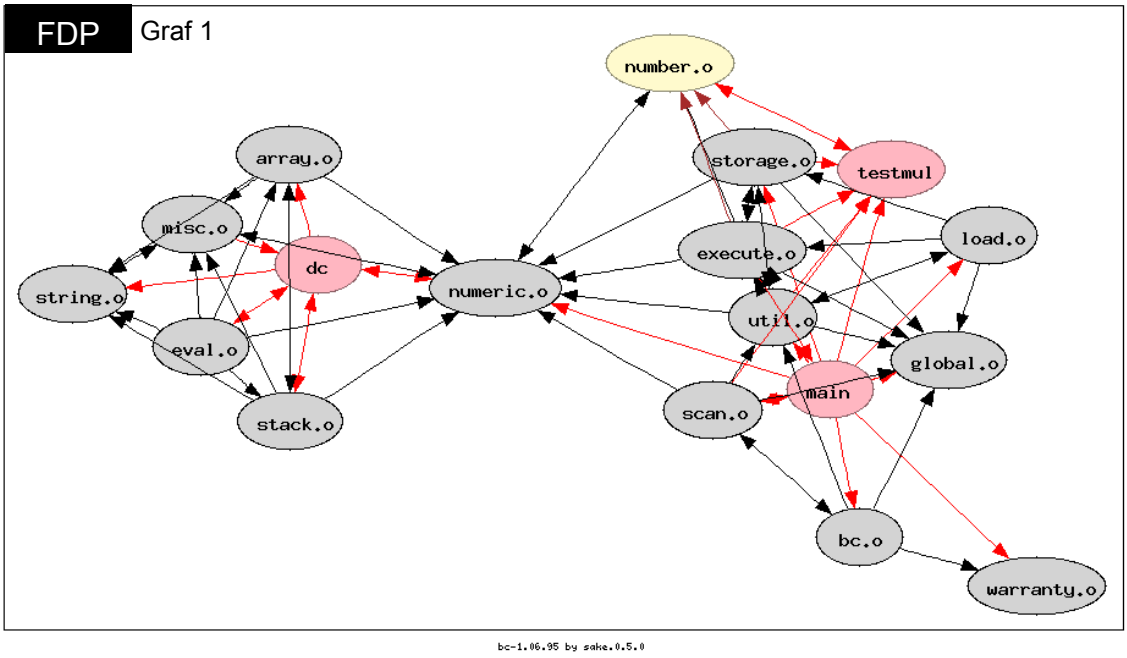
Obr. 6 - 9 Výsledok metódy KOHONEN na dátach neurónová sieť.

6.2.3 Testovacie dáta: graf 1

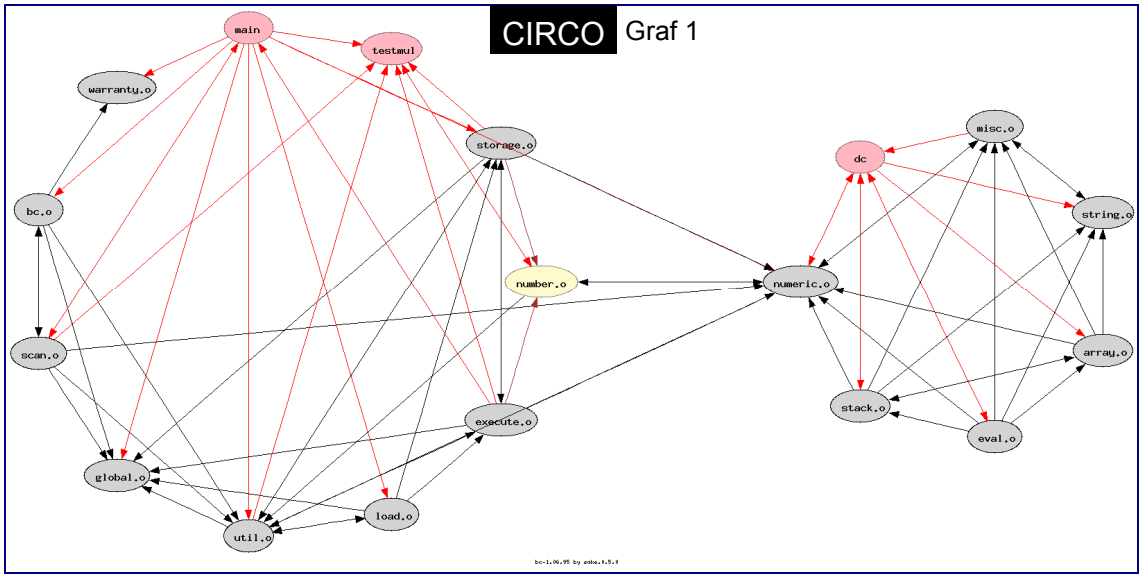
Výsledky vizualizácie sú na Obr. 6-10 až 6-14. Graf1 je pomerne jednoduchý, napriek tomu však má veľa hrán a vyhnúť sa križovaniu hrán nie je možné. Tak isto ostré uhly susedných hrán nie sú rušivým prvkom. Križovanie hrán a ostré uhly sú slabou stránkou KOHONEN-ovej metódy a preto naša metóda (Obr. 6-13 a 6-14) dosahuje dobrý výstup porovnateľný s ostatnými metódami, ak nie lepší. Najlepší výsledok z hľadiska čitateľnosti a prehľadnosti dosiahla metóda CIRCO (Obr. 6-12).



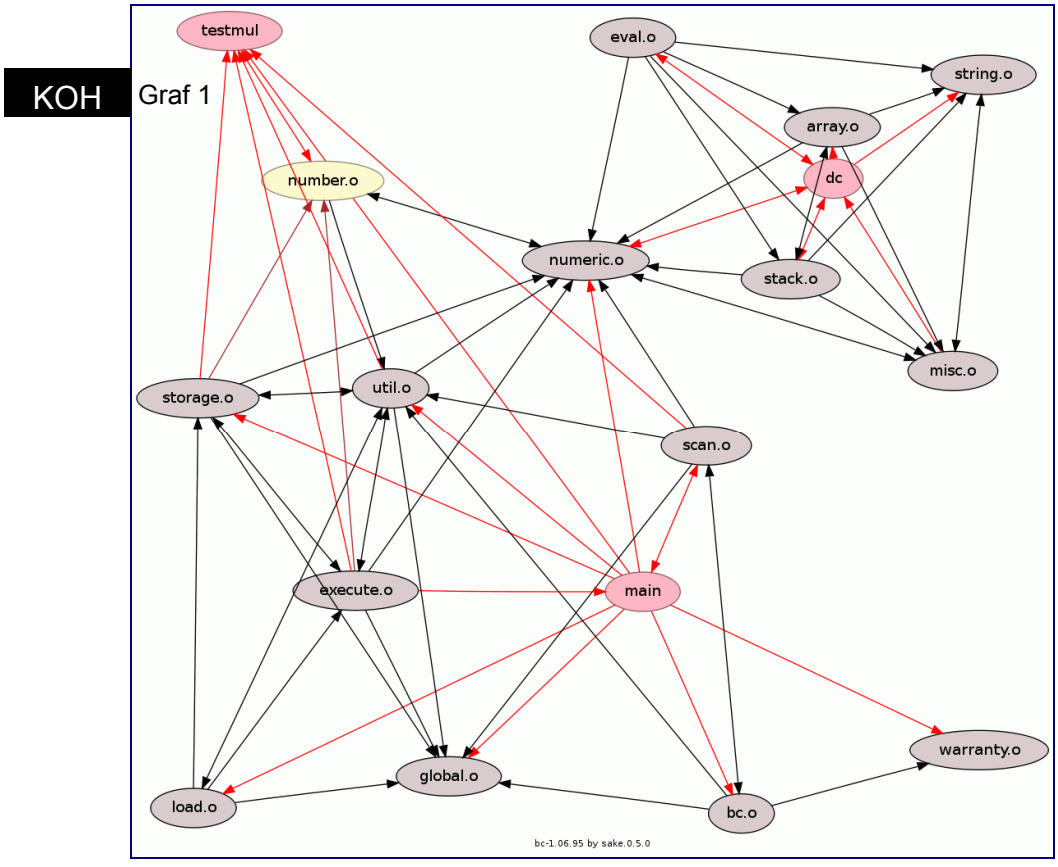
Obr. 6 - 10 Výsledok metód DOT a NEATO na dátach graf 1.



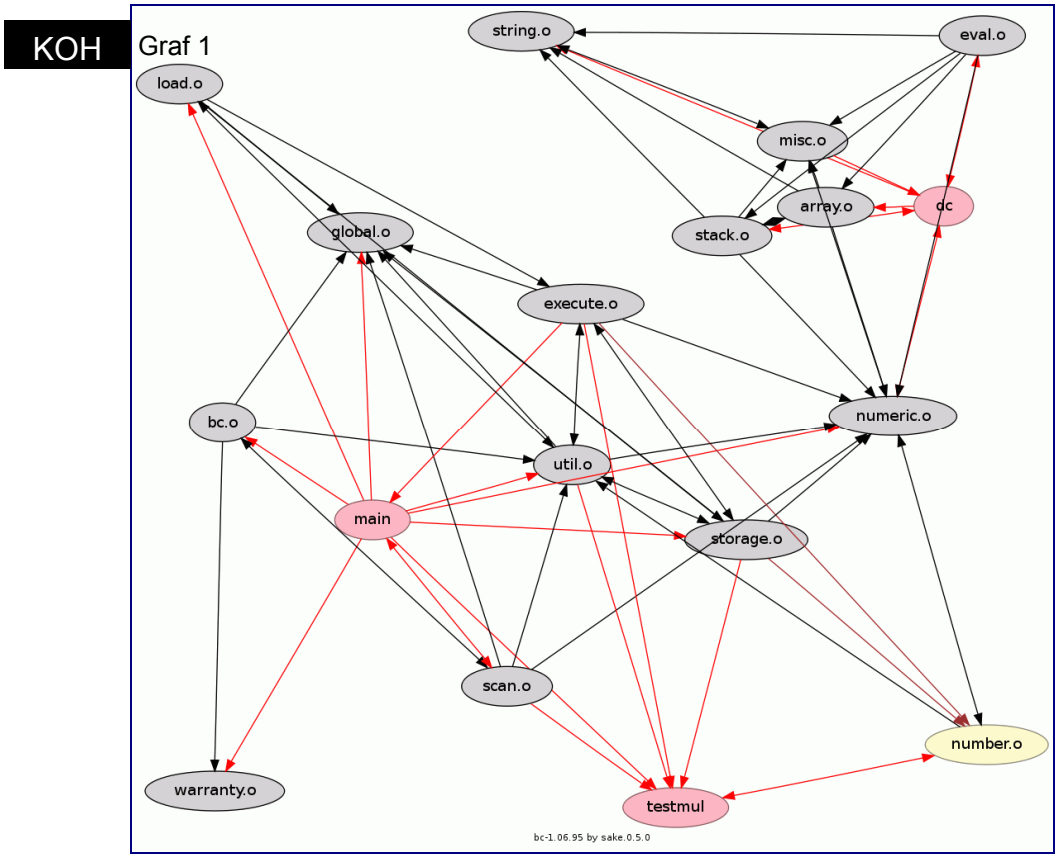
Obr. 6 - 11 Výsledok metódy FDP na dátach graf 1.



Obr. 6 - 12 Výsledok metódy CIRCO na dátach graf 1.



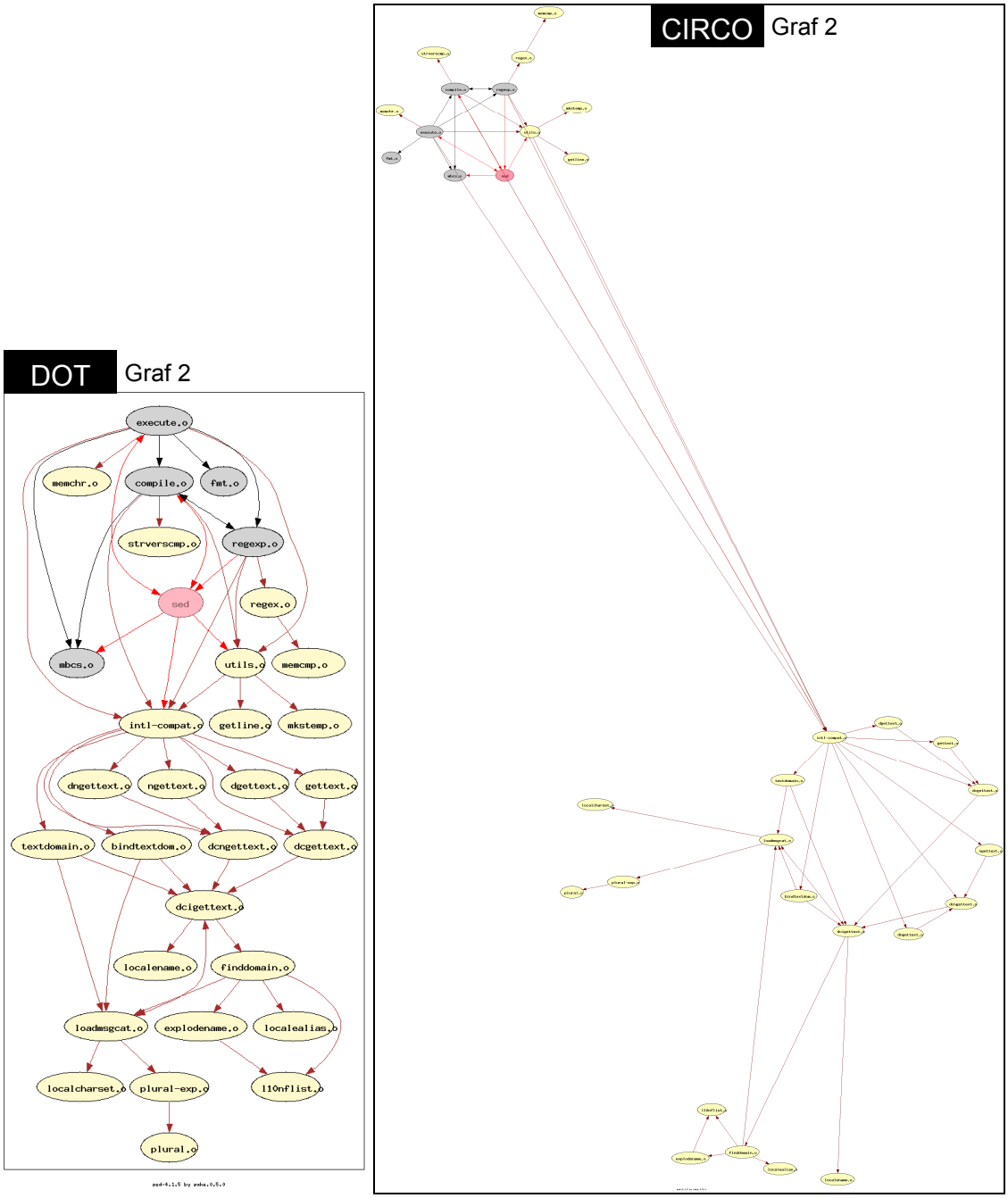
Obr. 6 - 13 Výsledok metódy KOHONEN na dátach graf 1.



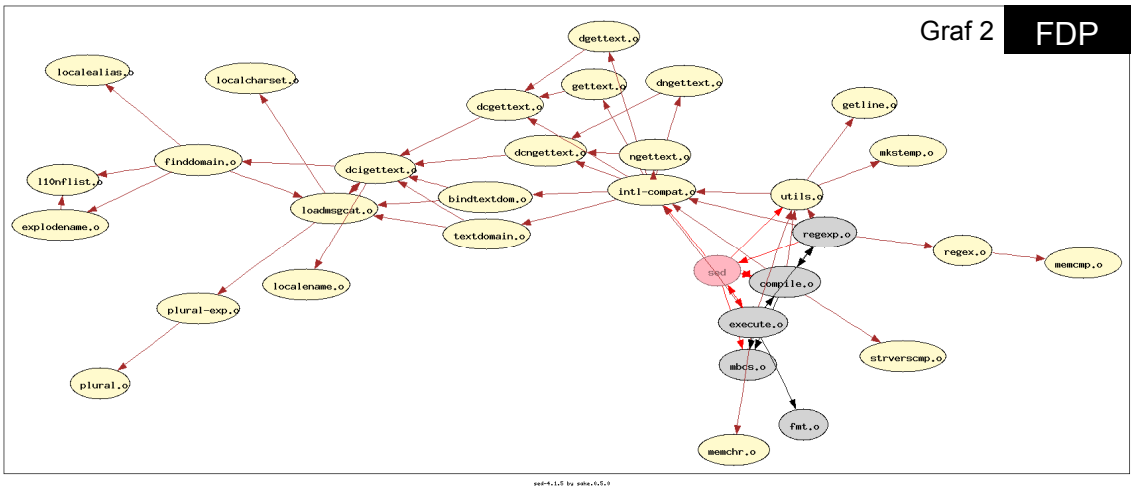
Obr. 6 - 14 Výsledok metódy KOHONEN na dátach graf 1.

6.2.4 Testovacie dáta: graf 2

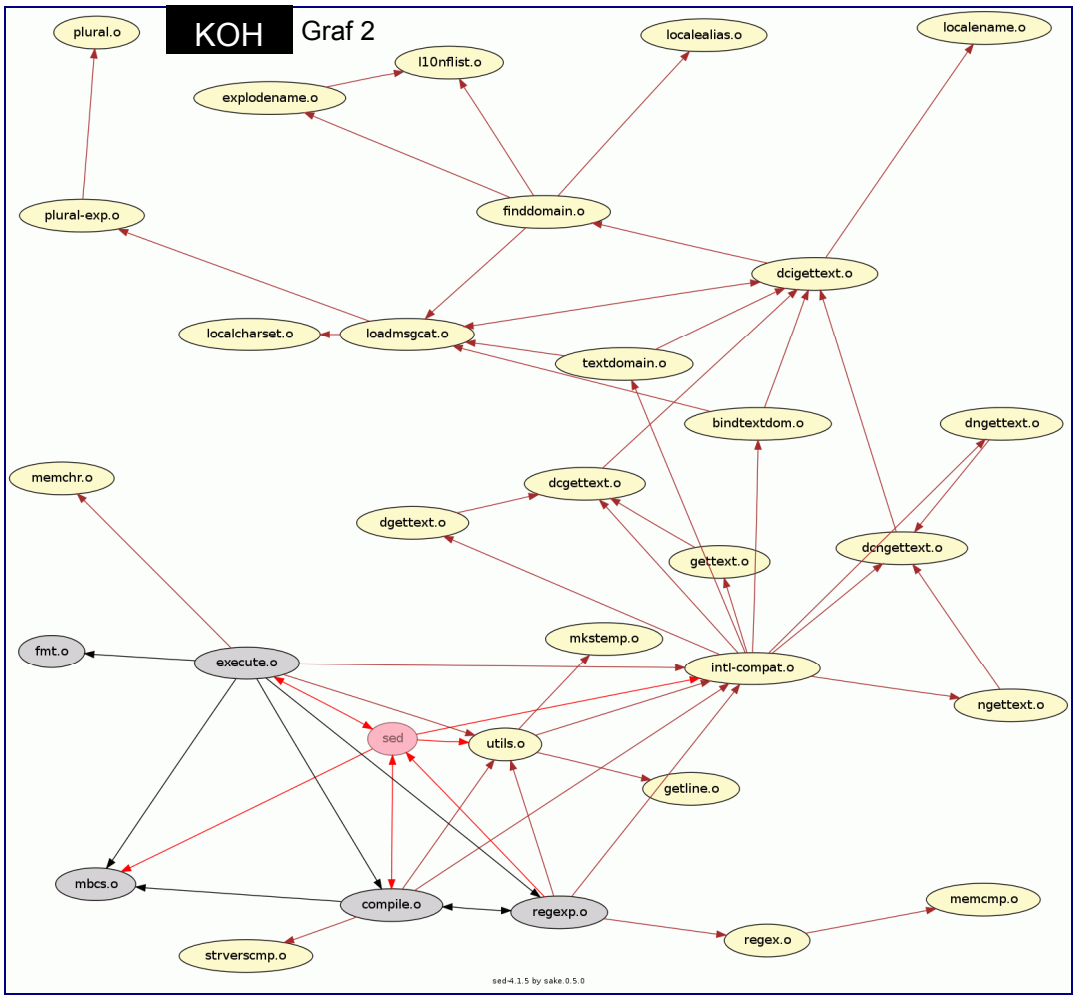
Výsledky vizualizácie sú na Obr. 6-15 až 6-17. Graf2 je pomerne jednoduchý má však viac vrcholov a hrán. Opäť platí to isté čo v predchádzajúcom prípade. Križovania a ostré uhly nie sú rušivým prvkom a metóda KOHONEN dosahuje dobré výsledky (Obr. 6-17) v porovnaní s ostatnými metódami. Metóda CIRCO (Obr. 6-15) by bola znovu najprehľadnejšou, pokiaľ by dokázala eliminovať zbytočne dlhé hrany.



Obr. 6 - 15 Výsledok metód DOT a CIRCO na dátach graf 2.



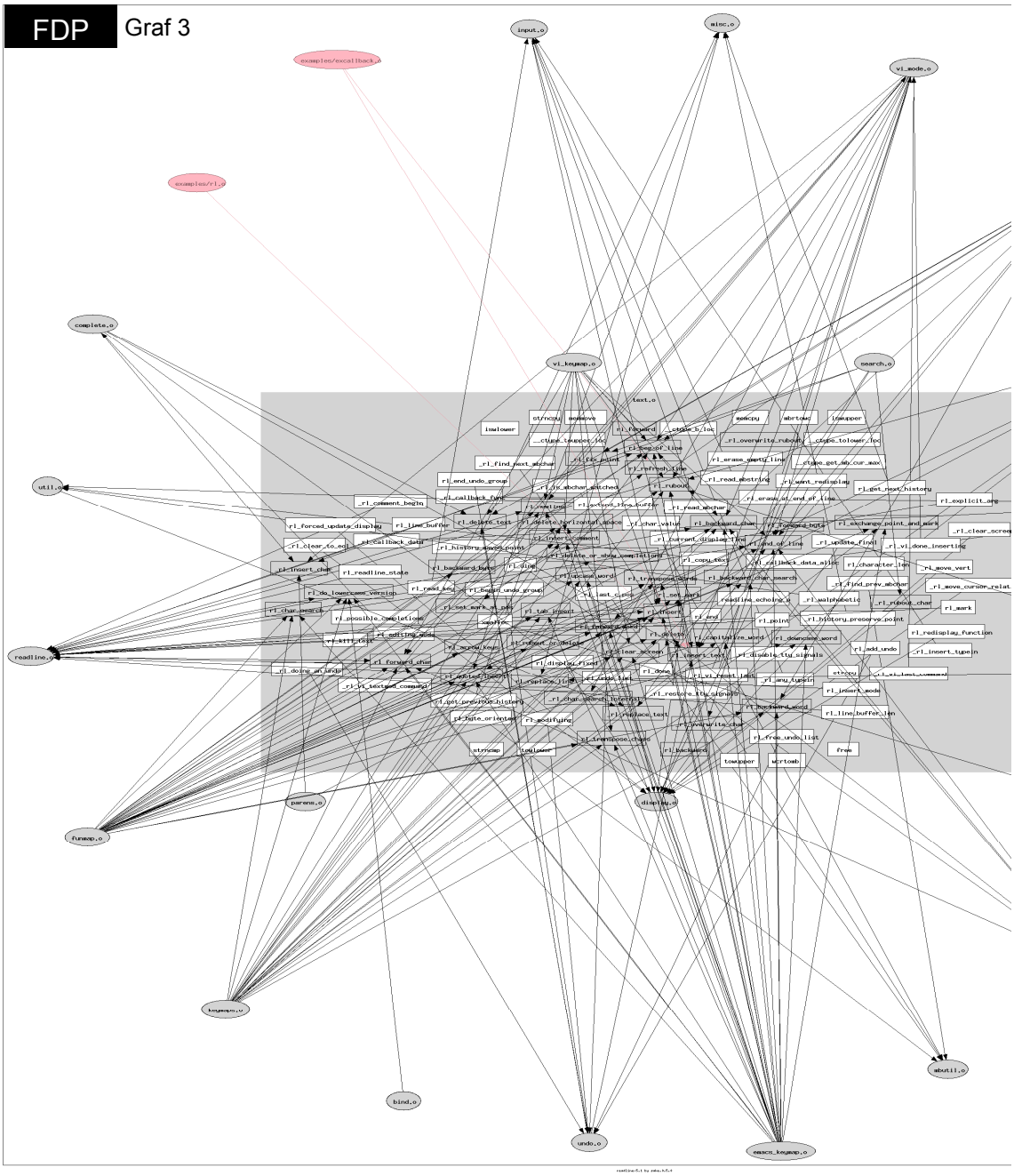
Obr. 6 - 16 Výsledok metódy FDP na dátach graf 2



Obr. 6 - 17 Výsledok metódy KOHONEN na dátach graf 2.

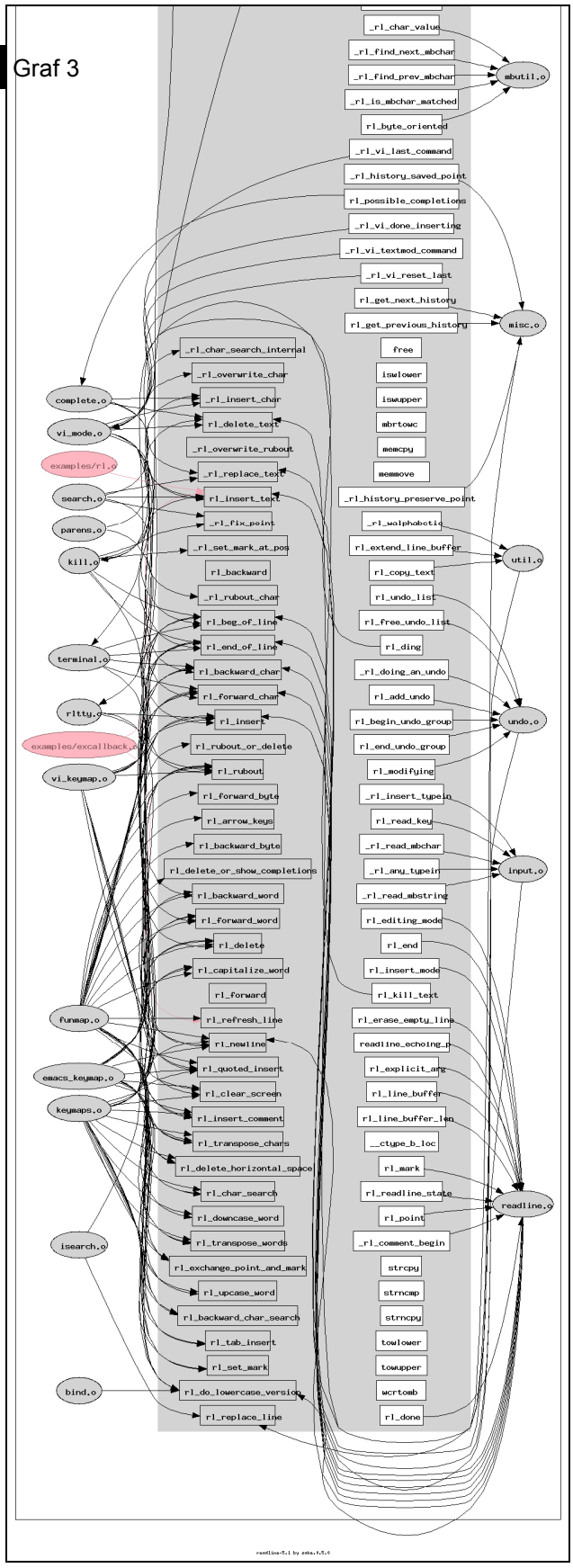
6.2.5 Testovacie dáta: graf 3

Výsledky vizualizácie sú na Obr. 6-18 až 6-20. Tento graf je podstatne zložitejší, môžeme ho označiť ako veľký graf. Len metóda DOT (Obr. 6-19) zaručuje 100% čitateľnosť údajov. Ostatné metódy vrátane KOHONEN-a (Obr. 6-20), poskytujú svoju zaujímavú štruktúru diagramu. Sú oblasti, kde sú dobre čitateľné údaje a sú oblasti, kde sú údaje nečitateľné. KOHONEN je v tomto prípade plne porovnateľný s metódami TWOPI a NEATO. FDP (Obr. 6-18) má ešte menšiu vypovedaciu hodnotu a výpočet CIRCO metódy bežal príliš dlho a preto sme ho zastavili.

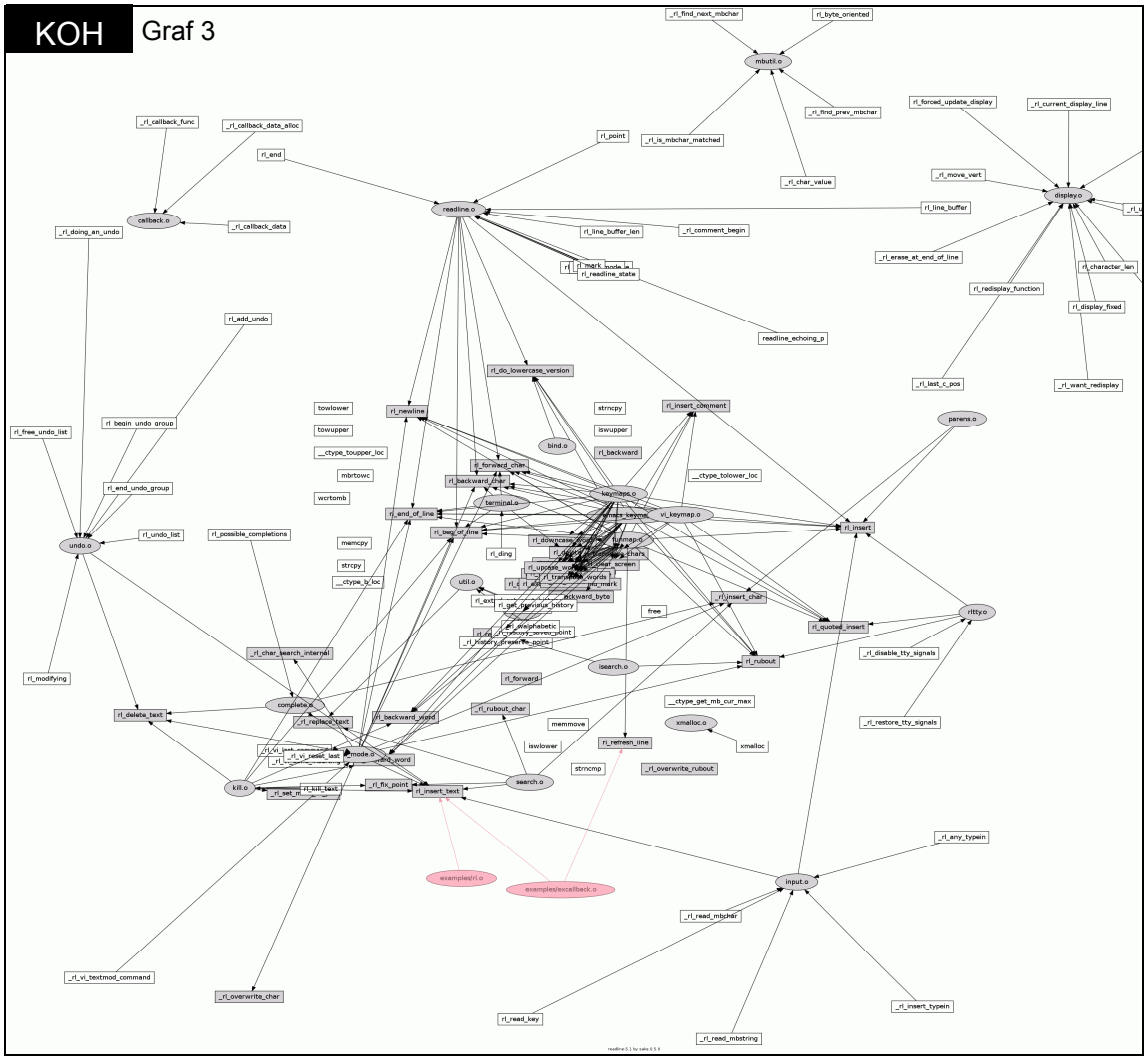


Obr. 6 - 18 Výsledky metódy FDP na dátach graf 3. (Obrázok je orezaný zprava.)

DOT Graf 3



Obr. 6 - 19 Výsledky metódy DOT na dátach graf 3. (Obrázok je orezaný z hora.)



Obr. 6 - 20 Výsledky metódy KOHONEN na dátach graf 3. (Obrázok je orezaný zprava.)

6.2.6 Zhrňujúce vyhodnotenie výsledkov 6.2

V tejto kapitole uvádzame podrobnejšie výsledky porovnaní jednotlivých rozvrhovacích metód. Prvé päť metódy sú z Graphviz-u, posledná metóda je náš implementovaný nový návrh.

Porovnávali sme nasledujúce parametre:

Križovanie - počet križovaní hrán má byť minimálny. Ak počet prekročil hodnotu 20, použili sme zápis >20, ak ich bolo neporovnateľne viac použili sme zápis >100.

Čitateľnosť a prehľadnosť- maximálna čitateľnosť a prehľadnosť reprezentuje hodnota 100%. Prehľadnosť by mala zachytávať nejakú štruktúru grafu ak existuje. Ak je diagram čitateľný no menej prehľadný hodnota sa mierne znižuje. Prekrývanie sa vrcholov a veľká hustota hrán v jednom mieste, tak že texty sú nečitateľné alebo neviditeľné, potom túto hodnotu znižujú podstatne.

Veľkosť plátna - veľkosť plochy, na ktorej je graf vykreslený má byť minimálna. Veľkosť uvádzame v pixloch.

Dlhé hrany - počet zbytočne dlhých hrán má byť minimálny. Ak počet prekročil hodnotu 20, použili sme zápis >20, ak ich bolo neporovnateľne viac použili sme zápis >100.

Pokrytie plátna - v ideálnom prípade by malo byť 100%. Voľné plochy na kresliacom plátne potom znižujú pomerne túto hodnotu.

Prekrývanie vrcholov - žiadne vrcholy by sa nemali prekrývať, počítame prekrytia, ktoré znemožňujú čitateľnosť mien vrcholov. Ak počet prekročil hodnotu 20, použili sme zápis >20, ak ich bolo neporovnateľne viac použili sme zápis >100.

Vyhodnocovanie Kohonenovej rozvrhovacej metódy sme robili ako aritmetický priemer troch diagramov, nakoľko každé spustenie dá iný výsledok.

Tab. 6 - 1 Tabuľka porovnaní jednotlivých rozvrhovacích metód. Pod názvom testovaného grafu je uvedený počet vrcholov a hrán. Parameter križovanie má zmysel sledovať len pri jednoduchých grafoch ako RS, pri zložitejších grafoch sa križovaniu nedá vyhnúť. Pri dátach Graf3 a metóde CIRCO výpočet nebol dokončený kvôli výpočtovej náročnosti.

		križo-vanie	čitateľnosť prehľadnosť	veľkosť plátna	dlhé hrany	prekrytie vrcholov	pokrytie plátna
RS V: 20 H: 19	DOT	0	100%	622 x 730	0	0	60%
	CIRCO	0	10%	24 210 x 16 082	10	0	1%
	FDP	5	90%	625 x 653	0	0	60%
	NEATO	0	95%	837 x 774	0	0	50%
	TWOPI	0	90%	858 x 982	0	0	60%
	KOH	1,33	90%	1 200 x 1 150	0	0	90%
NN V: 16 H: 55	DOT	-	100%	466 x 346	0	0	90%
	CIRCO	-	60%	941 x 917	0	0	20%
	FDP	-	80%	566 x 641	0	0	60%
	NEATO	-	70%	294 x 332	0	10	70%
	TWOPI	-	50%	557 x 489	0	4	60%
	KOH	-	50%	950 x 950	0	0	90%
Graf1 V: 18 H: 59	DOT	-	90%	773 x 712	0	0	80%
	CIRCO	-	95%	1 634 x 818	0	0	60%
	FDP	-	80%	928 x 546	0	0	50%
	NEATO	-	40%	366 x 446	0	7	60%
	TWOPI	-	50%	561 x 610	0	3	55%
	KOH	-	80%	1 150 x 1 100	0	0	90%
Graf2 V: 32 H: 54	DOT	-	90%	601 x 1 288	0	0	90%
	CIRCO	-	30%	2 458 x 4 636	9	0	20%
	FDP	-	80%	1 712 x 744	0	0	60%
	NEATO	-	65%	740 x 754	0	3	50%
	TWOPI	-	70%	716 x 850	0	0	70%
	KOH	-	80%	1 600 x 1 500	0	0	95%
Graf3 V: 129 H: 213	DOT	-	85%	1 000 x 3 790	0	0	75%
	CIRCO	-	-	-	-	-	-
	FDP	-	20%	3 000 x 2 973	0	>100	50%
	NEATO	-	30%	1 240 x 1 180	0	>100	90%
	TWOPI	-	30%	1 272 x 1 032	0	>20	90%
	KOH	-	50%	3 500 x 3 300	0	>20	70%

Vhodnosť použitia metódy KOHONEN zhrnieme zvlášť pre malé, stredné a veľké grafy.

Malé grafy – počet hrán do 30 (graf RS): KOHONEN-ová metóda dosahuje dobre čitateľné výsledky (riadok: RS/KOH, stĺpec: čitateľnosť), avšak táto metóda vygenerovala aj niekoľko zbytočných križovaní hrán (riadok: RS/KOH, stĺpec: križovanie) a zbytočne ostré uhly susedných hrán (Obr. 6-6). V malých grafoch

a v grafoch, kde používateľ chce zachytiť nejakú hierarchiu je to rušivým prvkom. Preto najvhodnejšou metódou je DOT, ktorá presne zachytáva hierarchiu, napr. NN.

Stredné grafy - počet hrán 30-150 (graf1, graf2): Ak grafy nezachytávajú zjavnú štruktúru alebo hierarchiu, križovania hrán a ostré uhly nie sú rušivým prvkom. Metóda KOHONEN dosahuje dobré alebo lepšie výsledky v porovnaní s ostatnými (riadok: Graf1/KOH a Graf2/KOH, stĺpec: čitateľnosť prehľadnosť). Všetky vrcholy, ich popisy, všetky hrany sú 100% čitateľné (Obr. 6-13, 6-14, 6-17).

Veľké grafy – počet hrán nad 200 (graf3): Všetky metódy okrem DOT, nedokážu zabezpečiť čitateľnosť všetkých údajov. Ak toto nie je prekážkou použitia, potom metóda KOHONEN dosahuje uspokojivejšie výsledky (Obr. 6-20) ako CIRCO, NEATO, TWOPI a FDP (riadok: Graf3, stĺpec: čitateľnosť prehľadnosť).

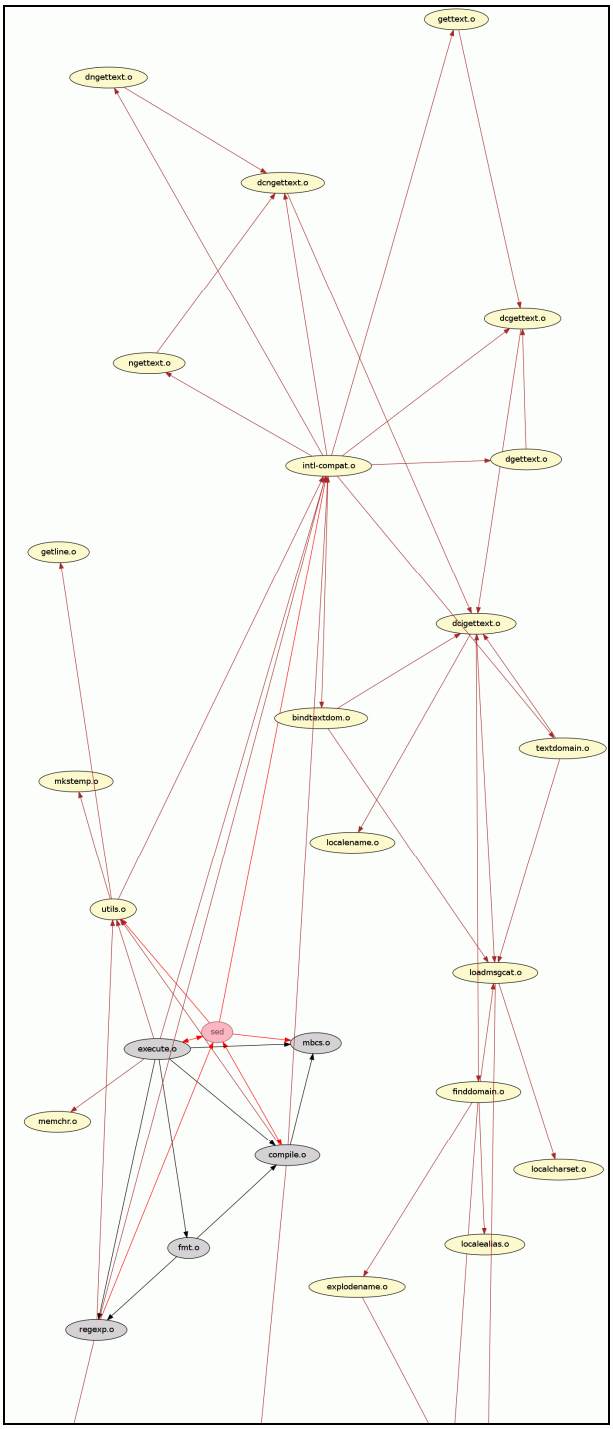
Jednou z vlastností KOHONEN-ovej metódy je rovnomerné pokrytie celého kresliaceho plátna (stĺpec: pokrytie plátna). V prípade veľkých grafov, nad 200 hrán, úplné pokrytie kresliaceho plátna klesá, ostávajú oblasti ktoré sú veľmi riedke a oblasti ktoré sú veľmi husté, rozdiel môže byť 10 až 100 násobný (Obr. 6-20).

Na základe prevedených experimentov sme zistili, že navrhnutá KOHONEN-ová metóda je najvhodnejšia na vykresľovanie grafov, ktoré majú od 20 do 150 hrán a neuchovávajú v sebe zjavnú hierarchiu. Čitateľnosť dát je 100%, križovanie hrán a ostré uhly susedných hrán v tomto prípade nie sú rušivým prvkom. Navrhnutú metódu je možné použiť aj na vykresľovanie väčších grafov v závislosti od subjektívnej požiadavky na čitateľnosť dát. Pri väčších grafoch navrhnutá metóda nedokáže zabezpečiť úplnú čitateľnosť údajov v diagrame grafu.

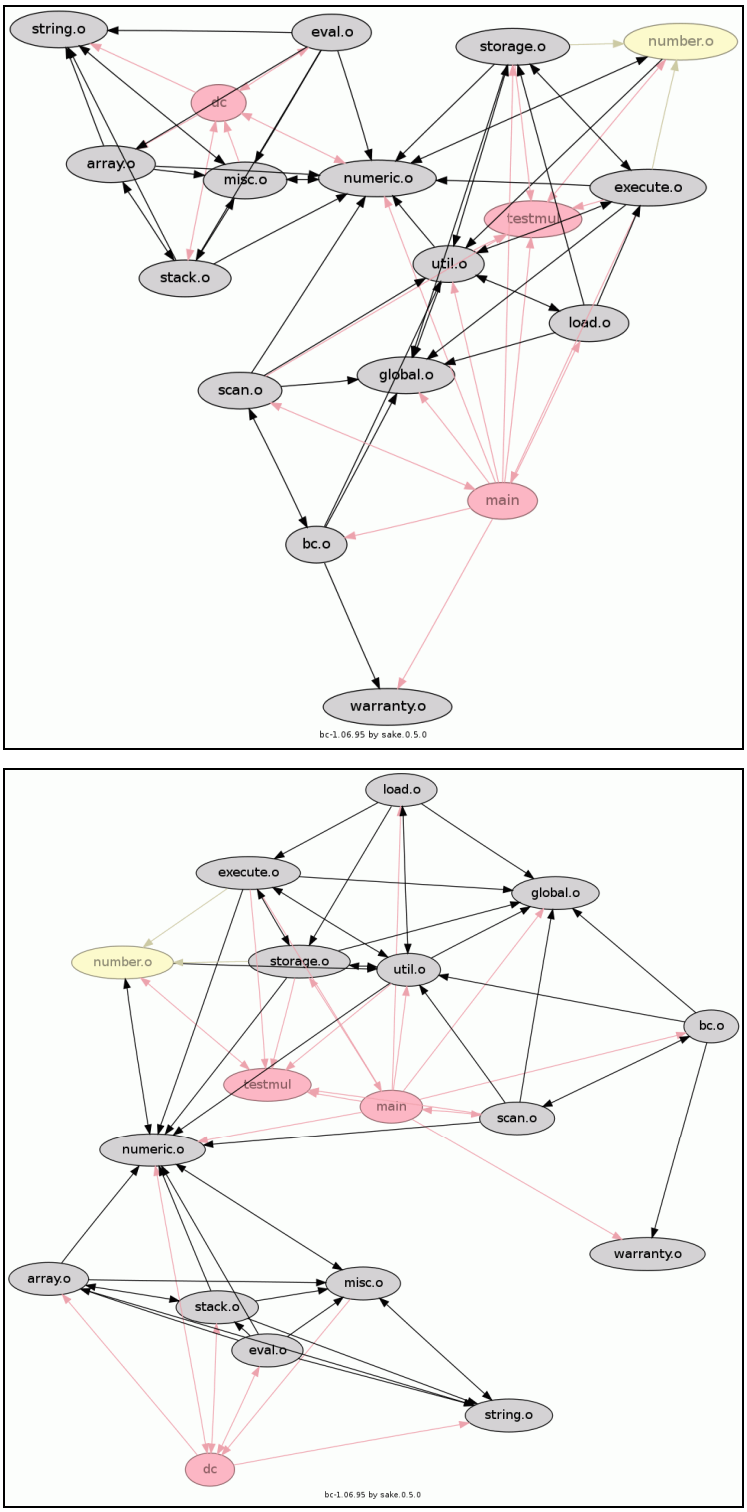
Výpočtovú náročnosť sme sledovali na základe trvania výpočtu. KOHONEN-ovú metódu môžeme zaradiť niekde do stredu. Časovo najnáročnejšia bola metóda CIRCO. Pri Grafe3 sme sa výsledku nedočkali vôbec. Metóda DOT spotrebovala 2 a viac násobok času oproti KOHONEN-ovi. Ostatné metódy boli rýchlejšie ako KOHONEN.

6.3 Experimentálne overenie vplyvu zmeny tvaru kresliaceho plátna

Posledná časť experimentov mala za cieľ overiť možnosť použitia navrhovanej metódy pre vykresľovanie grafov na rôzne tvary kresliaceho plátna. Použili sme tvar trojuholník, kruh a obdĺžnik a očakávali sme rozprestretie grafu do týchto tvarov.



Obr. 6 - 21 Použitie Kohonenovej metódy pre rozprestretie vrcholov grafu do tvaru obdĺžnika. (Obrázok je orezaný z dola.)



Obr. 6 - 22 Použitie Kohonenovej metódy pre rozprestretie vrcholov grafu do tvaru trojuholníka a kruhu.

Experimenty potvrdili, že algoritmus rozmiestňuje vrcholy grafu do požadovaného tvaru, pričom požadovaný tvar kresliaceho plátna nezhoršuje dosiahnuté výsledky oproti štvorcovému plátnu.

7 Záver

Táto diplomová práca ukazuje, že navrhnutú Kohonenovu rozvrhovaciu metódu je možné použiť na vykresľovanie grafov.

Navrhnutá Kohonenova metóda je extrémne jednoduchá oproti bežne používaným sofistikovaným metódam. Má malé výpočtové nároky. Je silno prispôsobivá pre rôzne typy grafov a tvary kresliaceho plátna.

Na základe prevedených experimentov sme zistili, že navrhnutá Kohonenova metóda je najvhodnejšia na vykresľovanie grafov, ktoré majú od 20 do 150 hrán a neuchovávajú v sebe zjavnú hierarchiu. V tomto prípade sú výsledky najlepšie zo všetkých porovnávaných metód. Čitateľnosť dát je 100%, križovanie hrán a ostré uhly susedných hrán v tomto prípade nie sú rušivým prvkom. Navrhnutú metódu je možné použiť aj na vykresľovanie väčších grafov v závislosti od subjektívnej požiadavky na čitateľnosť dát. Pri väčších grafoch navrhnutá metóda nedokáže zabezpečiť úplnú čitateľnosť údajov v diagrame grafu.

Návrh novej metódy podrobne rozoberá kapitola 3.

Pre overenie praktického použitia novej metódy na vykresľovanie grafov sme použili najrozšírenejší softvér v tejto oblasti – Graphviz. Praktická implementácia novej metódy do tohto prostredia je detailne popísaná v kapitole 5.

Experimentálna časť tejto práce overuje možnosť a vhodnosť používania novej metódy. Samotnú funkčnosť metódy sme overovali v kapitole 6.1. Vhodnosť použitia v kapitole 6.2, ktorá je hlavnou časťou experimentov. Podrobné zhrnutie výsledkov je v kapitole 6.2.6.

Navrhnutú metódu je možné ďalej modifikovať a upravovať. Priestor na overenie lepších výsledkov je v nasledujúcich oblastiach:

1) V prípade orientovaných grafov výpočet matice vzdialeností vrcholov upraviť tak, aby zohľadňovala orientáciu hrán. Matica potom nebude symetrická podľa diagonály. Vzdialenosť za vrcholu A do B nebude rovnaká ako z B do A. V procese učenia NN by sa to malo prejaviť tak, že víťaz v každej iterácií bude k sebe priťahovať len vrcholy v smere orientácie. Toto by mohlo pomôcť lepšie zachytiť hierarchickú štruktúru grafu.

2) Proces učenia Kohonenovej neurónovej siete je v navrhnutej metóde ovplyvňovaný hlavne vzdialenosťou vrcholov medzi sebou. Bolo by vhodné doplniť ďalšie kritéria pre učiaci algoritmus Kohonenovej siete. Ďalším kritériom by mohli byť križovania hrán vychádzajúcich z daného vrcholu, alebo ostré uhly susedných hrán z daného vrcholu. Uvedené kritéria predstavujú značné zvýšenie náročnosti výpočtu.

Zoznam použitej literatúry

[1] ELLSON, John – GANSNER, Emden – YIFAN, Hu: Graphviz – Graph Visualization Software, AT&T Research, (1994-2009). Dostupné na internete: <http://www.graphviz.org/>.

[2] KLEŠČ, M. : Diskrétna matematika, ISBN 80-8073-698-7, Elfa, 2006

[3] CRUZ, Isabel F. – TAMASSIA, Roberto: Graph Drawing Tutorial, 1998. Dostupné na internete: < http://www.cs.brown.edu/people/rt/papers/gd-tutorial/gd-constraints.pdf >.

[4] GANSNER , Emden R. – KOUTSOFIOS, Eleftherios – NORTH, Stephen C. – VO, Kiem-Phong: A Technique for Drawing Directed Graphs, AT&T Bell Laboratories, 1993. Dostupné na internete: < http://www.graphviz.org/Documentation/TSE93.pdf >.

[5] SINČÁK, P. – ANDREJKOVÁ, G.: Neurónové siete Inžiniersky prístup, ELFA Press, 1996.

[6] ROJAS, Raul: Neural Networks - A Systematic Introduction, Springer-Verlag, Berlin, 1996. s. 391-412. Dostupné na internete: <http://www.inf.fu-berlin. de/inst/ag-ki/rojas_home/pmwiki/pmwiki.php?n=Books.NeuralNetworksBook>.

[7] MEYER, Bernd: Self-Organizing Graphs A Neural Network Perspective of Graph Layout. Department of Computer Science, University of Munich. In: Graph Drawing (GD‘98), Montreal, Canada, August 1998.

[8] BONABEAU, Eric - HÉNAUX, Florian: Self-Organizing Maps for Drawing Large Graphs. Information Processing Letters 67 (1998) 177-184.

[9] KOVÁČ, Július: Vyhľadávanie v obrazovej databáze s využitím samoorganizujúcich sa máp a multiscale reprezentácie, Diplomová práca, KKUI FEI TU, Košice, 2007.

[10] UŽÁK, Matúš: Vizualizácia a interakcia v procese učenia neurónových sietí, Diplomová práca, KKUI FEI TU, Košice, 2005.

[11] ABAS, Marcel – HÍC, Pavol: Diskrétna matematika, Slovenská Technická Univerzita, Trnava, 2005. Dostupné na internete: < http://pdfweb.truni.sk/hic/abas_diskretna_matematika.pdf>

Prílohy

Príloha A: CD médium – diplomová práca v elektronickej podobe, prílohy v elektronickej podobe.

Príloha B: Používateľská príručka

Príloha C: Systémová príručka