Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Automatický layouting entitno-relačných modelov v Databázovom portáli

Bakalárska práca

2023
Filip Horváth

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Automatický layouting entitno-relačných modelov v Databázovom portáli

Bakalárska práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	Ing. Alexander Šimko, PhD.

Bratislava, 2023
Filip Horváth

20692346

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Filip Horváth
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Automatické rozloženie UML diagramov tried v DB Portáli
Automatic layouting of UML class diagrams in DB Portal

Anotácia: Databázový portál, ktorý sa používa na predmete Databázy (1) obsahuje modul na zobrazovanie entitno-relačných modelov pomocou UML diagramov. Diagramy sa zadávajú textovou formou v špeciálnej syntaxi, v ktorej sa definujú jednotlivé elementy diagramu. Z tejto definície sa následne generuje svg obrázok. Nevýhodou súčasného riešenia je, že definícia každého elementu musí obrahovať aj jeho pozíciu vo výslednom obrázku. To výrazne predlžuje proces vytvárania diagramov.

Cieľ: Cieľom práce je modifikovať súčasný modul tak, aby používateľ nemusel zadávať pozície jednotlivých elementov, ale aby pri generovaní svg obrázka boli vypočítané automaticky. Automatické pozície majú byť vypočítané tak, aby bol výsledný obrázok dobre čitateľný a aby sa v ňom čiary pretínali minimálne.

Vedúci: Ing. Alexander Šimko, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Dátum zadania: 01.10.2023

Dátum schválenia: 04.10.2023 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Abstrakt

Entitno-relačné diagramy databáz sú jednoduchá a intuitívna metóda ako komunikovať a reprezentovať vzťahy v rámci databázy. V rámci tejto prace predstavíme niektoré algoritmy na automatické vykresľovanie diagramov. Oboznámime sa s tým ako fungujú a aké vlastnosti majú. Taktiež identifikujem problémy ktoré by mohli nastáť pri ich implementácii na databázové entitno- relačné modeli.

Kľúčové slová: diagram, algoritmus, graf, layout

Abstract

Entity-relationship diagrams of databases are simple and intuitive method how to communicate and represent relationships inside of database. Throughout this work we will introduce some of the algorithms for automatic drawing of diagrams. We will understand with how they operate and what kind of attributes they posses. Also we will identify problems which might arise during their implementation for database entity-relationship models.

Keywords: diagram, algorithm, graph, layout

Obsah

Úvod

Cieľom tejto prace je oboznámiť sa s rôznymi algoritmami na vytváranie diagramov a zistiť, ktoré z nich by boli vhodné na automatické vytváranie diagramov reprezentujúcich entitno-relačné modely databáz, prípadne identifikovať problémy, ktoré by bolo treba vyriešiť pri ich implementácii.

Predstavíme tri typy algoritmov podrobnejšie a to Sugiyama algoritmy, algoritmy vkladajúce do štvorcovej siete a algoritmy vytvárajúce diagram podľa zoznamu pravidiel.

Kapitola 1

Sugiyama algoritmy

Skupina algoritmov založených na Sugiyamovom hierarchickom layout algoritme. [4] Tu konkrétne predstavíme Seemannou algoritmus na automatický layout objektovo-orientovaných software diagramov. [3]

Tento algoritmus dostane graf G, ktorého vrcholy predstavujú triedy a hrany predstavujú vzťahy medzi jednotlivými triedami. Samotný algoritmus sa skladá z viacerých fáz:

1.1 Fáza 1 - Predpríprava

Odstráni priame cykly a uloží ich ako vlastnosti jednotlivých vrcholov. Potom vytvorí pod graf I z G, ktorý obsahuje iba vrcholy reprezentujúce triedy, ktoré sú navzájom prepojene dedičnosťou a hrany reprezentujúce dedenie medzi triedami, ak výsledný I nie je súvislý graf pridá sa fiktívny vrchol, ktorý bude slúžiť ako spoločný koreň všetkých komponentov grafu.

1.2 Fáza 2 - Sugiyama algoritmus

Všetkým vrcholom I sú určené prvotne úrovne kde sa umiestnia podľa ich hĺbky v strome dedičnosti.

Následne preusporiadava vrcholy v rámci jednej úrovne tak aby zminimalizoval počet prienikov hrán, pričom berie do úvahy všetky hrany G medzi všetkými vrcholmi grafu I.

1.3 Fáza 3 - Rozširovanie

Postupne rozšíri graf I nasledovne:

1. Vybranie vrcholu X z G, ktorý ešte nebol vložený

2. Ak má vrchol X spojenie s iba jedným vrcholom už v grafe I, tak sa pridá nasledovne:

   ak vrchol Y, s ktorým je X spojený ma menej ako dve spojenia v svojej úrovni, tak sa X pridá na lavo alebo na pravo od Y podľa toho ci už ma spojenie na úrovni a kde sa nachádza vzhľadom na Y

   inak, ak ešte neexistuje, sa vytvorí nová úroveň, ktorá sa vloží priamo pod úroveň kde sa nachádza Y a X sa vloží do tejto úrovne.

3. Ak je vrchol X spojený s viac ako jedným už vloženým, tak sa vyberie už vložený vrchol Y, s ktorým je spojený a je taký že ma najmenej spojení s už vloženými vrcholmi a následne sa vrchol X pridá rovnako ako v 2. a ešte sa pridajú hrany zo všetkými vrcholmi, s ktorými je X spojený.

Po pridaní všetkých vrcholov vykoná optimalizáciu pozícii vrcholov v rámci jednej vrstvy, tak aby zminimalizoval kríženie hrán.

1.4 Fáza 4 - Vykreslenie

Nakoniec tento algoritmus vykreslí výsledný diagram. Toto zahŕňa vypátranie veľkosti vrcholu na základe atribútov jednotlivých tried, doladenie pozícii vrcholov v rámci vrstvy, určenie ako vzdialené majú byť jednotlivé vrcholy a ďalšie vizuálne upraví, ktoré ale môžu záležať na konkrétnej implementácii.

1.5 Zhodnotenie spôsobilosti

Tento algoritmus predpokladá prítomnosť výrazných hierarchických vzťahov medzi aspoň podmnožinou vstupných vrcholov, to ale nemusí byť pravda v diagrame reprezentujúci databázový systém, čo by mohlo spôsobiť že by algoritmus nemal ako začať.

Kapitola 2

Vkladanie do štorcovej siete

Algoritmy vkladajúce graf do štorcovej siete (Grid embedding) majú využitie vo viacerých oblastiach najme ale rozloženie obvodu (circuit layout).

Takéto algoritmy zväčša predpokladajú že vrcholy sú body a majú maximálne štyri hrany, tie sú potom umiestnene na nejaké súradnice v jednotkovej sieti a hrany sú tvorene iba z vertikálnych a horizontálnych priamok, ktoré zvierajú priami uhol.

Tu ako príklad predstavíme jeden takýto plenárny algoritmus.[5]

2.1 Algoritmus

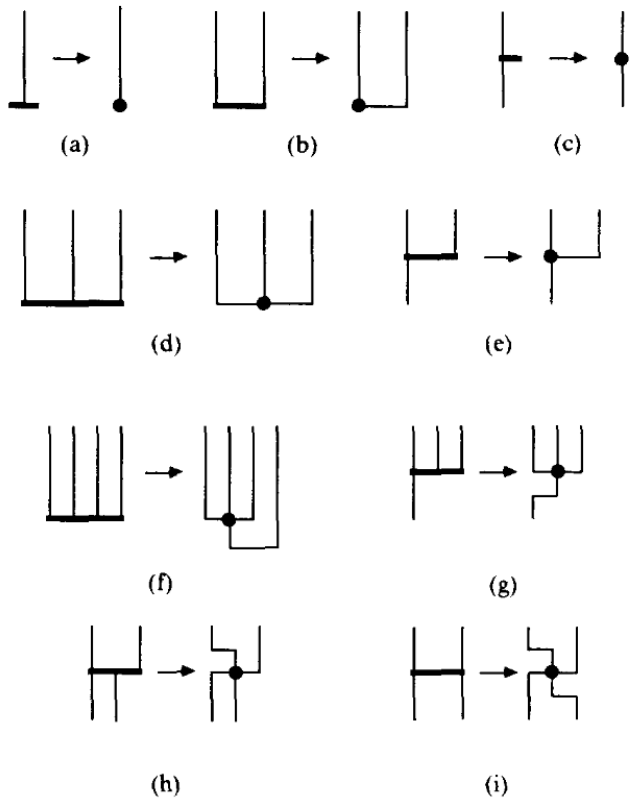
1. Na začiatku dostane algoritmus graf G s n vrcholmi, s ktorých majú všetky najviac štyri hrany. Štandardná vizualizácia G by mohla vyzerať ako na 2.2(a)

2. Na základe G vytvorí reprezentáciu H tak že každý vrchol premení na horizontálnu priamku, na 2.1 na lavo od šípky sú niektoré štandardne prípady, a hrany na vertikálne priamky tak že hrana zdieľa bod v reprezentácii iba s jej dvomi vrcholmi pripadne s inou hranou. Výsledne H by malo vyzerať podobne 2.2(b).

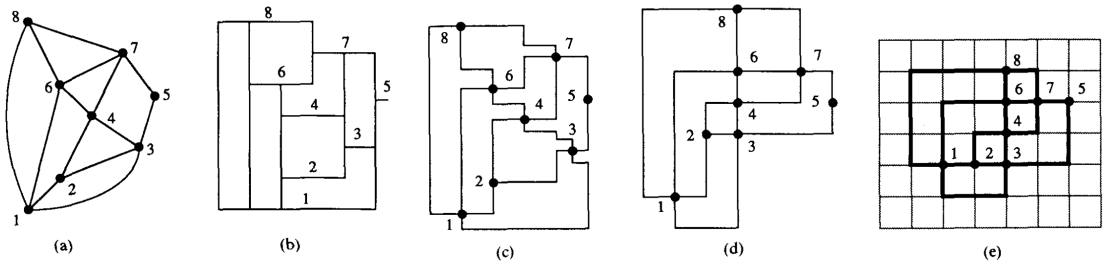
3. Premena H na ortogonálne vloženie (embedding) O pomocou premeny každého vrcholu na niektorú štruktúru z 2.1(premena z lava do prava), po skončení tohto kroku O vyzerá ako 2.2(c)

4. V tomto kroku algoritmu[5] aplikuje postupne viacero transformácii, ktoré ale nebudeme bližšie popisovať. Zjednodušene algoritmus presunie vrcholy a upraví hrany tak aby zminimalizoval počet ohnutí (bends) na všetkých hranách. 2.2(d)

5. Ďalej sa aplikuje podalgoritmus, ktorý zminimalizuje priestor, ktorý upravený graf O zaberá.2.2(e)



Obr. 2.1: Nahradzanie vrcholov pri druhom kroku vkladania do stvorcovejsiete[5]



Obr. 2.2: Vizuálny príklad krokov vkladania do štvorcovej siete[5]

2.2 Zhodnotenie spôsobilosti

Tento algoritmus ako sme ho predstavili potrebuje ako vstup plenárny graf, ale v ďalších prácach[1] už boli vytvorené heuristiky, ktoré sa na túto vlastnosť nespoliehajú. Ďalší problém ale je že každý vrchol môže mat iba štyri hrany. K tomuto problému existuje navrhnuté riešenie[1] a to že každý vrchol zo vstupného grafu, ktorý ma viac ako štyri hrany, je vo výstupnom grafe reprezentovaný skupinou nad sebou umiestnených prepojených vrcholov, ktoré by sa mali, podľa implementácie, vykresliť ako jeden objekt. Toto by nám ale mohlo byť problematické, lebo vzdialenosť krajných dvoch bodov z tejto skupiny by mohla byť väčšia ako zobrazenie vrcholu databázy, čo by spôsobilo veľké množstvo výnimiek pri implementácii.

Kapitola 3

Vytvaranie diagramu podla zoznamu pravidiel

Jeden z unikátnejších algoritmov, ktorý sme študovali bol určený na vytváranie layoutu pre sieťové diagramy[2]. Tento algoritmus interpretuje vytvorenie layoutu ako optimalizačný problém obmedzený výberom zo zoznamu pravidiel (ktoré sú určené pri implementácii).

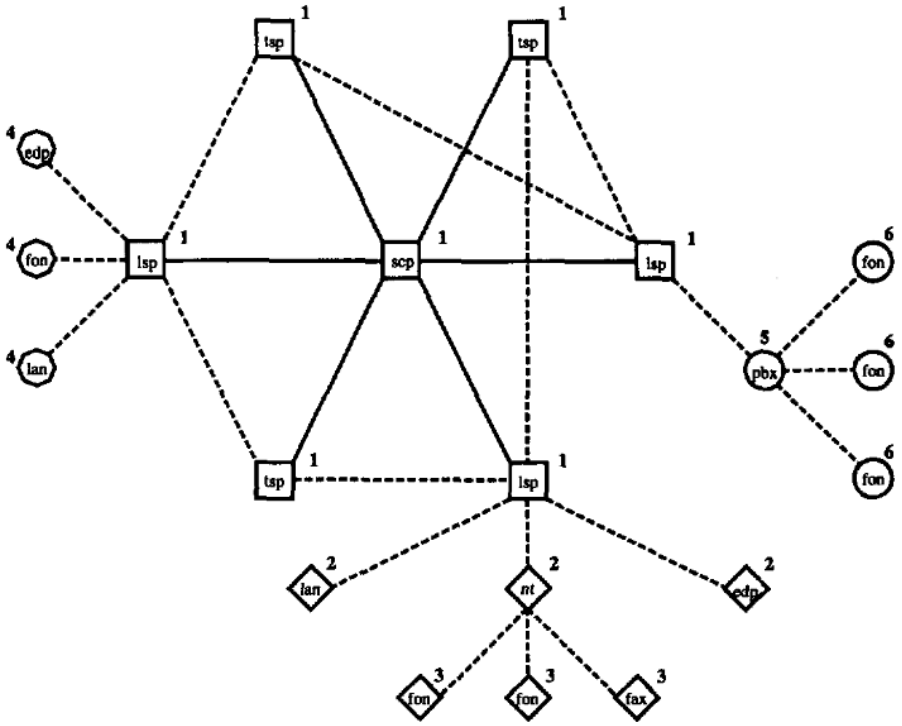
Tu sa pod pravidlom myslí spôsob ako rozšíriť už existujúci graf a za akých podmienok je toto pravidlo možné aplikovať. Ako príklad použijeme pravidlo z 3.2. Toto HUB pravidlo predpokladá že existuje vrchol $n0$, ktorý je spojený s viacerými vrcholmi, (tu by mohlo byť určené minimum) na obrázku znázornené prerušovanými čiarami. Aplikáciou tohto pravidla sa spojené vrcholy vložia do grafu v pravidelnom rozložení okolo $n0$ a vložia sa aj hrany.

Pravidla majú aj ďalšie podmienky ako napr. novy vrchol nemôže prekrývať už predtým vložený vrchol alebo novo vzniknutá hrana nemôže pretínať už predtým vložený vrchol. Tieto podmienky sú ale závisle na implementácii.

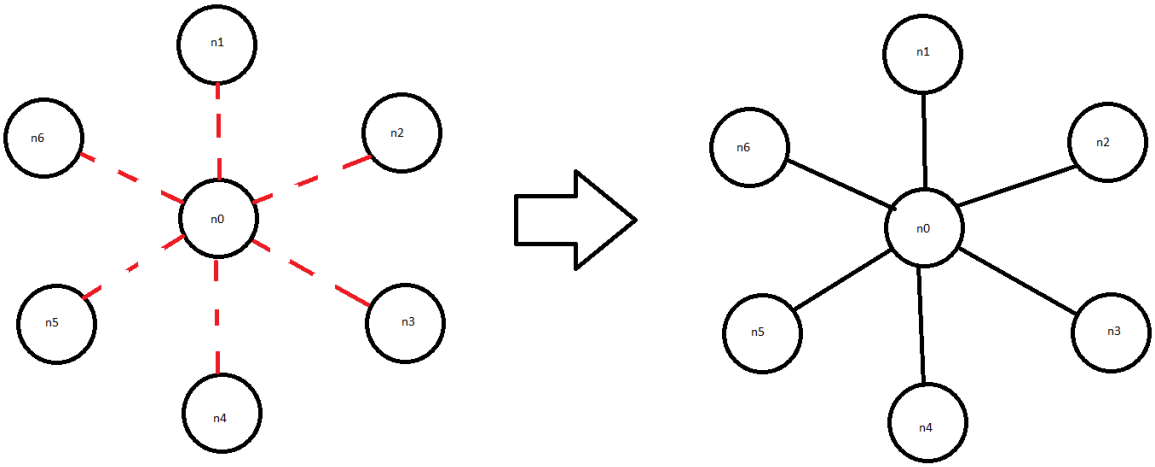
3.1 Implementácie

K tomuto spôsobu vytvárania diagramu boli navrhnuté dva typy implementácie prvý je backtrackingový algoritmus, ktorý má svoj zoznam pravidiel zoradený od najviac preferovaných po najmenej a keď vytvára diagram tak najprv skúsi aplikovať viac preferovane. Vždy pridá jedno pravidlo a zapamätá si, ktoré pridal a cyklus opakuje s novým diagramom. Ak do diagramu sú už pridané všetky vrcholy a hrany alebo ak už nemá aké pravidlo aplikovať tak backtackuje. Počas behu si pamätá najlepší celý diagram, ktorý zatiaľ našiel.

Druhý typ implementácie je paralelný genetický algoritmus. Pre tento typ algoritmu je nevyhnutná spôsobilostná (fitness) funkcia, ktorá by doterajšiemu diagramu priradila



Obr. 3.1: Priklad diagramu vytvoreneho podla zoznmu pravydie



Obr. 3.2: Príklad *HUB* pravidla

číselnú hodnotu reprezentujúca ako veľa je daný diagram vhodný. Takýto algoritmus by nazačiatku spustil implementáciou určený počet vlákien, ktoré by mali medzi sebou určenú susednosť.

Všetky vlákna by si náhodne vybrali niektoré pravidlo, ktoré aplikujú na doterajší diagram. Následne si vlákna môžu náhodne vybrať jedného suseda, od ktorého odkopírujú jedno alebo viac pravidiel alebo umiestnenie vrcholu. Tento cyklus sa opakuje kým nie je splnená podmienka napr. zmena spôsobilosti najlepšieho za posledných par cyklov bola menšia ako určené minimum alebo prestúpený maximálny počet cyklov.

3.2 Zhodnotenie spôsobilosti

Pri tomto type algoritmov má veľkú váhu zoznam pravidiel a spôsobilostná funkcia, ktoré závisia od konkrétnej implementácie, čiže teoreticky neexistuje diagram, ktorý by sa nedal vytvoriť týmto spôsobom, ale pri niektorých prípadoch by takýto spôsob mohol byť menej efektívny, v porovnaní so štandardnejšími algoritmami, alebo by mohlo byť príliš náročné vytvoriť vhodný zoznam pravidiel. Na druhej stane takéto algoritmy nepredpokladajú o vstupnom grafe nič.

Záver

Uviedli sme niekoľko typov algoritmov na vykresľovanie diagramov a opísali ich bližšie fungovanie. Taktiež sme identifikovali niektoré problematické aspekty o týchto typoch vzhľadom na našu tému a to vykreslenie entitno-relačných modelov databáz.

Ako prvý sme rozobrali typ, ktorý sme my nazvali Sugiyama algoritmy. Tieto algoritmy sú primárne určené na vykresľovanie objektovo-orientovaných software diagramov. Výsledné diagramy pripomínajú stromy.

Problematické ale je že Sugiyama algoritmy sa spoliehajú na to že vstupné grafy obsahujú hierarchiu medzi podmnožinou vstupných vrcholov.

Nasledovali algoritmy, ktoré vkladali graf do štorcovej siete. Tento typ algoritmov sa najčastejšie využíva na vytváranie diagramu rozloženia obvodnou. Takto vytvorené diagramy majú vrcholy umiestnene na kríženiach priamok jednotkovej štvorcovej siete a hrany sa skladajú z vertikálnych a horizontálnych segmentov.

Takéto algoritmy zväčša vyžadujú aby všetky vrcholy mali najviac štyri hrany, čo je problém lebo pri databázach nevieme tuto vlastnosť zaručiť. Tento problém bol ale vyriešený nahradením vrcholu skupinou prepojených nad sebou umiestnených vrcholov. Toto riešenie ale prináša ďalší potenciálny problém a to že takto umiestnené vrcholy sú vždy navzájom od seba vzdialené podľa jednotkovej siete, táto vzdialenosť by ale mohla byť väčšia ako reprezentácia samotného vrcholu, čiže by sa mohlo stať že niektoré hrany nesmerujú do vrcholu.

Posledný typ algoritmov bol postupné vytváranie diagramu podľa zoznamu pravidiel. Tento typ je špeciálny v tom že výsledný diagram záleží primárne na tom aký konkrétny zoznam pravidiel použijeme a teda nemá žiadny štandardný tvar. Štandardný algoritmus tohto typu funguje na princípe backtracku a to postupným pridávaním vrcholov podľa pravidiel, ktoré sú zoradené od viac preferovaných po menej.

Najväčší problém pri implementácii tohto typu algoritmov je vytvorenie vhodného zoznamu pravidiel.

Literatúra

[1] Therese Biedl and Goos Kant. A better heuristic for orthogonal graph drawings. *Computational Geometry*, 9(3):159–180, 1998.

[2] Corey Kosak, Joe Marks, and Stuart Shieber. Automating the layout of network diagrams with specified visual organization. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(3):440–454, 1994.

[3] Jochen Seemann. Extending the sugiyama algorithm for drawing uml class diagrams: Towards automatic layout of object-oriented software diagrams. In *Graph Drawing: 5th International Symposium, GD'97 Rome, Italy, September 18–20, 1997 Proceedings*, pages 415–424. Springer, 2005.

[4] Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.

[5] Roberto Tamassia and Ioannis G Tollis. Planar grid embedding in linear time. *IEEE Transactions on circuits and systems*, 36(9):1230–1234, 1989.