UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATKY, FYZIKY A INFORMATIKY

ALGORITMUS VEDENIA HRÁN PRE NEDISJUNKTNÉ VRCHOLY

Diplomová práca

Bratislava, 2013

Bc.Martin Sarvaš

UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATKY, FYZIKY A INFORMATIKY

ALGORITMUS VEDENIA HRÁN PRE NEDISJUNKTNÉ VRCHOLY

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra Informatiky
Školiteľ:	RNDr. Jana Katreniaková, PhD.

Bratislava, 2013

Bc.Martin Sarvaš

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Martin Sarvaš
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský

Názov: Algoritmus vedenia hrán pre nedisjunktné vrcholy

Cieľ: Algoritmus postupného vedenia hrán, ktorý sa používa v súčasnosti sa dá využiť pre ľubovoľné rozloženie obdĺžnikových vrcholov a preto je zaujímavé jeho zovšeobecnenie aj na vrcholy, ktoré sa môžu prekrývať. Cieľom je vytvoriť algoritmus vedenia hrán v prípade, že vrcholy sú síce obdĺžniky, ale nie nutne sa medzi nimi nachádza dostatok priestoru na vedenie hrany. Snažíme sa optimalizovať estetické kritériá bežné pri kreslení grafov.

Vedúci: RNDr. Jana Katreniaková, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: doc. RNDr. Daniel Olejár, PhD.

Dátum zadania: 26.10.2010

Dátum schválenia: 28.10.2010

prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestne prehlasujem, že som diplomovú prácu vypracoval samostatne s použitím uvedenej literatúry.

Ďakujem všetkým, ktorí ma akýmkoľvek spôsobom podporovali pri mojej práci. Prvenstvo patrí Bohu, lebo iba vďaka nemu som tým, čím som, potom mojím rodičom, ktorí ma sprevádzali životom a maximálne podporovali v dobrom. V neposlednom rade patrí moja vďaka mojej školiteľke, ktorá mi pomáhala a viedla ma k odbornosti a svojou trpezlivosťou mi dávala nádej až do konca.

Abstrakt:
SARVAŠ, Martin: Algoritmus vedenia hrán pre nedisjunktné vrcholy [diplomová práca]/Martin Sarvaš. -Univerzita Komenského v Bratislave. Fakulta Matematiky, Fyziky a Informatiky; Katedra Informatiky. Školiteľ: RNDr. Jana Katreniaková, PhD. Bratislava FMFI UK, 2013

Cieľom tejto práce bolo navrhnutie a implementácia algoritmu na vedenie hrán v grafe, v ktorom nemožno meniť súradnice vrcholov a zároveň sa môžu vrcholy prekrývať. Algoritmus má byť jednoduchý, efektívny a tiež výsledné hrany má viesť podľa klasických estetických kritérií.

Kľúčove slová: Algoritmus, vizualizácia grafov, vedenie hrán, graf, fixné vrcholy, nedisjunktné vrcholy

Abstract:
SARVAŠ, Martin: Algorithm for rerouting edges in graphs with intersect vertices [masters's thesis]/Martin Sarvaš. -Komenius Univerzity in Bratislava. Faculty of Mathematics, Physics and Informatics; Department of Computer Science. Advisor: RNDr. Jana Katreniaková, PhD. Bratislava FMFI UK, 2013

The aim of this thesis is to create and implement a new algorithm for routing edges in graph with fixed-positioned vertices. We discuss also vertices that have some intersection. Algorithm should be simple, effective and preserves classical aesthetic criteria on routing edges in graphs.

Keywords: Algorithm, graph visualization, routing edges, rerouting, graph, fixed vertices, intersecting vertices

Obsah

Kapitola 1

Úvod

V súčastnosti sa pri vizualizácii grafov rieši hlavne problém vykresľovania vrcholov do najmenšej plochy, planárne zobrazenie grafu alebo rozmiestnenie vrcholov a hrán s najmenšou celkovou dĺžkou hrán. Avšak relatívne málo pozornosti sa venuje samotnému vedeniu hrán. Možno je to zapríčinené tým, že v grafe s dobre usporiadanými vrcholmi sa ľahko vedú hrany, no v každom prípade je oblasť vedenia hrán nezanedbateľná. Napríklad, keď si používateľ zadá vlastné súradnice vrcholov a nepraje si, aby mu ich vykresľovací algoritmus premiestnil. Alebo keď zobrazujeme graf, ktorý symbolizuje mapu - potrebujeme zachovať presné súradnice vrcholov.

Estetických kritérií na vykresľovanie je pomerne veľa a navzájom sa vylučujú. V takomto prípade je na mieste optimalizácia na zvolené estetické kritériá s istými váhami, lebo spokojnosť všetkých pravidiel nie je dosiahnuteľná vo všeobecnom prípade. Nami zvolené kritériá pre vedenie hrán sú spomenuté v 2. kapitole spolu so základnými definíciami o grafe, jeho reprezentácii a estetike.

V 3. kapitole nasleduje prehľad o typoch vizualizácie grafu. Medzi najdôle-

žitejšie usporiadania patrií ortogonálne a silové usporiadanie, ktoré sme použili pri návrhu heuristiky.

Keďže existuje algoritmus pre vedenie hrán v grafoch, v ktorých sú vrcholy zobrazené ako obdĺžniky, vychádzali sme z tohoto algoritmu a rozšírili sme ho na grafy s prekrývajúcimi sa vrcholmi. Detaily nášho návrhu sú spomenuté v kapitole 4. Návrh je jedna z najdôležitejších častí práce, kde sa nám podarilo vymyslieť dve nové heuristiky. Prvá je jednoduchšia a je založená na obchádzaní vrchola v tvare polygónu pomocou ťažiska. Druhá používa Dijkstrov algoritmus, v ktorom na ohodnocovanie ohybov pre nájdenie najkratšej cesty v grafe sme vymysleli penalizačnú funkciu na dosahovanie nami zvolených estetických kritérií.

V 5. kapitole sú zmienené implementačné detaily heuristík ako aj ich praktické porovnanie na rôznych typoch grafov. Táto práca obsahuje aj CD - prílohu, kde sa nachádzajú implementované heuristiky spolu s diplomovou prácou v elektronickej podobe.

Kapitola 2

Základné definície a estetické kritériá

V tejto kapitole sa budeme venovať základným pojmom z teórie grafov, vizualizácie a vymedzeniu oblasti záujmu, ktorej sa venujeme. Potom si pribížime všeobecné estetické kritériá kladené na zobrazený graf a tiež kritériá, ktoré sme si zvolili my.

2.1 Grafy

Definícia 2.1.1 *Graf je usporiadaná dvojica $G = (V, E)$, kde V je neprázdna množina všetkých vrcholov grafu G a E je množina dvojíc z V všetkých hrán grafu G. Ak sú dvojice z E usporiadané, hovoríme o orientovaných grafoch.*

Definícia 2.1.2 *Podgraf $G' = (V', E')$ grafu $G = (V, E)$, $G' \subseteq G$ je graf, ktorého vrcholy sú z množiny $V' \subseteq V$ a hrany z množiny $E' \subseteq E$.*

Definícia 2.1.3 *Vrcholy $v_1, v_2 \in V$ sú incidentné s hranou $e \in E$ ak hrana e tieto vrcholy spája v množine E. Označujeme $e = (v_1, v_2)$ alebo, $e = (v_2, v_1)$ v neorientovaných grafoch.*

Definícia 2.1.4 *Incidenčná matica $Im\,(n, n)$ pre graf $G = (V,\, E)$ je matica, kde n je počet vrcholov grafu G a $Im(i, j) = 1$ ak existuje hrana $e \in E$ v grafe G. Inak $Im(i, j) = 0$ pre $i, j \leq n$. Ak sa na pozíciách $Im(i, i)$ dá počet hrán incidentných s daným vrcholom, teda stupeň vrchola, môžme maticu Im nazvať aj Laplaceová matica.*

Definícia 2.1.5 *Pre graf $G = (V,\, E)$ existuje u-v sled $v_0, l_0, v_1, l_2, v_2, \ldots v_{n-1}, l_{n-1}, v_n$, kde $u = v_0, v = v_n, v_i \in V$ sú vrcholy a $l_i \in E$ sú hrany ak $\forall i \in (0..n-1)\,(v_i v_{i+1}) = l_{i+1} \in E$ medzi dvojicami vrcholov $(v_i v_{i+1})$ je hrana.*

Definícia 2.1.6 *Cesta v_0, v_n je v_0-v_n sled, v ktorom sa neopakujú vrcholy ani hrany. Ak sú hrany orientované, tak môžu byť v tomto slede iba ak sú incidentné v správnom smere.* $\forall i, j \in (0..n-1), v_i, v_j \in V\ v_i = v_j \Leftrightarrow i = j \wedge$
$\forall i, j \in (0..n-1)\, v_i v_{i+1}, v_j v_{j+1} \in E\ v_i v_{i+1} = v_j v_{j+1} \Leftrightarrow i = j$

Definícia 2.1.7 *Acyklický graf je taký graf $G = (V,\, E)$, v ktorom pre každý vrchol $x \in V$ existuje práve jedna cesta x, y pre $\forall y \in V, y \neq x$.*

Definícia 2.1.8 *Strom $T = (V,\, E)$ je taký acyklický orientovaný graf, v ktorom existuje jeden vrchol x, do ktorého nevedie žiadna cesta a z neho existuje cesta do každého vrchola grafu. Vrchol x sa nazýva tiež koreň a strom T možno nazvať aj Zakorenený strom.*

2.2 Reprezentácia grafov

Definícia 2.2.1 *Akékoľvek zakreslenie, vizualizáciu grafu $G = (V, E)$ do roviny (v našom prípade $\mathbb{Z} \times \mathbb{Z}$), kde všetky vrcholy $v \in V$ sú reprezentované geometrickým objektom(bod, polygón, kruh...) a hrany incidentné medzi dvoma vrcholmi $v_1, v_2 \in V$ sú krivky medzi týmito objektami nazývame reprezentácia grafu G.*

Definícia 2.2.2 *Planárna reprezentácia grafu $G = (V, E)$ je taká reprezentácia, v ktorej všetky vrcholy z množiny V sú reprezentované ako body v rovine a hrany incidentné medzi dvoma vrcholmi $v_1, v_2 \in V$ sú krivky medzi týmito bodmi. Zároveň musí platiť, že žiadne dve krivky sa nepretínajú(okrem prípadu, keď sú incidentné s tým istým vcholom) a krivky hrán pretínajú iba dva vrcholy, s ktorými sú incidentné.*

Definícia 2.2.3 *Planárny graf je taký graf, pre ktorý existuje planárna reprezentácia.*

Definícia 2.2.4 *Graf viditeľnosti $G_{vis} = (V, E_{vis})$ je podgraf úplného grafu $G = (V, E)$, ktorého vrcholy V predstavujú body jednoduchých polygónov P v euklidovskej rovine a $\forall e\,(v_1, v_2) \in E$ platí, že $e \in E_{vis}$ práve vtedy, keď sa v_1 a v_2 vidia, čiže, keď na úsečke z v_1 do v_2 nie je žiadny vrchol z V (okrem incidentných s e) a nepretína ju žiadna úsečka polygónov P.*

Pre lepšie pochopenie definícií odporúčam [18][12][10].

V oblasti, v ktorej sa budeme ďalej pohybovať nebudeme pracovať s grafom ako abstraktnou šktruktúrou definovanou v definícii 2.1.1, ale jeho reprezentáciou v rovine $\mathbb{Z} \times \mathbb{Z}$ karteziánskej sústavy. Vrcholy $v\,(x, y) \in V$ budú zobrazené ako štvorce so stredom v bode so súradnicami x, y. Hrany $e =$

$(v_1(x_1, y_1), v_2(x_2, y_2))$ budú reprezentované lomenými čiarami s koncovými bodmi v bodoch (x_1, y_1) a (x_2, y_2).

Definícia 2.2.5 *Okraj rozlíšiteľnosti je používateľom zvolená konštanta, ktorá určuje minimálnu vzdialenosť od objektov(vrcholov a hrán) v reprezentácii grafu.*

Definícia 2.2.6 *Nedisjunktné vrcholy budeme považovať tie reprezentácie vrcholov, ktorých tvary(štvorce) sa prekrývajú. Ak je definovaný okraj rozlíšiteľnosti, tak sú nedisjunktné vrcholy tie, ktorých okraje majú vzájomný prienik.*

V našej práci sa budeme venovať takým reprezentáciám neorientovaného grafu, ktoré majú nedisjunktné vrcholy s fixnými pozíciami v euklidovskej rovine. Hrany budeme zobrazovať ako lomené čiary.

2.3 Estetické kritériá

V oblasti vizualizácie grafov je viacero známych estetických kritérií.

- Planarita grafu alebo minimálny počet pretínaní hrán
- Symetria grafu - čo najväčší počet rôznych symetrií
- Minimálny počet zlomových bodov hrany, keď je reprezentovaná lomenou čiarou(celkový, priemerný)
- Pomer minimálnej plochy, do ktorej je možné vykresliť graf a najmenšej vzdialenosti medzi dvoma vrcholmi
- Minimálna dĺžka hrán(celková, priemerná, pevná)

- Maximalizovaný vzájomný uhol hrán vychádzajúcich z toho istého vrchola (minimálne ohraničenie)
- Rovnomerné rozmiestnenie vrcholov
- Ortogonalita
- Konzistenté označovanie - rovnaké tvary vrcholov

Pri rôznych typoch kreslenia grafov sa vyberajú rôzne estetické kritéria a ich kombinácie s rôznou prioritou. Estetika grafu je dosť subjektívna, preto vznikajú stále nové typy zobrazenia. Hlavným kritériom je, aby sa dalo z reprezentácie grafu dobre pochopiť vzťahy a vnútornú štruktúru abstratného grafu. V minulosti bolo pár pokusov sformalizovať estetiku do všeobecných pravidiel[19] a metrík[16], ale vzhľadom na definíciu estetiky sa asi ani nikdy nepodarí dokázať dokonalosť jednej množiny estetických pravidiel.
Ak by sme chceli zachovať všetky existujúce kritériá, tak to sa nám nepodarí, lebo v konkrétnych prípadoch grafu sa požiadavky na jeho zobrazenie vylučujú. Preto je potrebné na dosiahnutie výsledku nastaviť váhy kritériám. Niektoré pravidlá majú objektívne väčšiu váhu ako iné(napr. pretínanie vrcholov vs. najmenšia dĺžka hrany), pri tých by sme azda aj vedeli nastaviť váhu s akou sa má kritérium aplikovať. Avšak sú kombinácie pravidiel, pre ktoré všeobecne váhy nastaviť nevieme. Táto možnosť upraviť váhu estetickým kritériám by mala ostať na používateľovi, no okrem [19] som sa nestretol s reálnou aplikáciou.
Najväčším nepriateľom estetiky je predovšetkým zložitosť optimalizačných kritérií. Väčšina týchto pravidiel má exponenciálnu zložitosť [13][20][15], preto pre reálne použitie je múdre použiť heuristiky a aproximácie problému.

Pre nás sú niektoré estetické pravidlá nepoužiteľné, nakoľko používame vrcholy s fixnými pozíciami(napr. rovnomerné rozmiestnenie). Preto sme si zvolili vlastnú množinu kritérií, podľa ktorej sme optimalizovali heuristiku na vizualizáciu grafu.

Definícia 2.3.1 *Naše estetické kritériá sú:*

1. *Minimálny počet zlomových bodov hrany*
2. *Minimálna dĺžka jednotlivých hrán*
3. *Maximalizovaný vzájomný uhol hrán vychádzajúcich z toho istého vrchola*
4. *Maximalizovaný uhol dvoch pretínajúcich sa hrán*
5. *Maximálne zachovanie okraju rozlíšiteľnosti*

Pre jednoduchosť sme si preformulovali zadanie z vedenia hrán v grafe s nedisjunktnými vrcholmi, na všeobecnejšiu verziu: Vedenie hrán v grafe, kde sú vrcholy reprezentované ako jednoduché polygóny s fixnými pozíciami, podľa vyššie definovaných estetických kritérií .

Kapitola 3

Prehľad typov a štýlov vizualizácie grafu

Graf ako abstraktná štruktúra má široké uplatnenie v takmer každej oblasti života, kde existujú objekty a vzťahy medzi nimi. Pre pochopenie vzťahov je stále viac aktuálna téma vizualizácie abstraktnej štruktúry do reálneho nákresu. V mnohých odvetviach vznikli aj presné normy ako graf nakresliť(UML, Databázy, Stavbárstvo...), ktoré sú často odlišné a prispôsobené na oblasť záujmu. V tomto krátkom prehľade sa budeme venovať len základným zobrazovacím štýlom a typom zobrazenia, ktoré sa používajú hlavne v informatike.

3.1 Štýl a typ

Zobrazovanie grafu je vo všeobecnosti náročná úloha vzhľadom na dobré pochopenie súvislostí v grafe. Štýl vizualizácie objektu môžme definovať, že je to jeho grafická reprezentácia v rovine.

Základné štýly vrchola poznáme:

- Bod
- Kruh
- Štvorec(n-uholník)
- Polygón
- Horizontálny úsek, čiara, obdĺžnik
- Iný útvar(napr. obrázok)

Základné štýly hrany sú:

- Priama úsečka
- Lomená čiara
- Bézierová krivka
- Prerušovaná čiara
- Vertikálny prázdny priestor, obdĺžnik

Pri orientovaných grafoch sú hrany šípky. Okrem štýlu objektov poznáme aj typ zobrazenia, ktorý definujeme ako usporiadanie objektov grafu v rovine. Typy zobrazení sa delia na tie, ktoré usporadúvajú iba vrcholy(Straight-line), iba hrany alebo všetky objekty.
Prvá množina je možná vďaka tomu, že hrany sú reprezentované priamymi úsečkami a tak je jednoduché ich vykresliť po usporiadaní vrcholov. Tieto usporiadania sa najčastejšie používajú a výskum bol rozsiahly vďaka pomeru zložitosti a výslednej kvality grafu.

1. **Stromové usporiadanie** - je použiteľné len ak je samotný graf acyklický alebo zakorenený strom. Vrcholy rovnakej hĺbky od koreňa sa umiestňujú na rovnakú horizontálnu priamku.

2. **Kruhové usporiadanie** - vrcholy rozmiestni do jedného alebo viacerých kruhov tak, aby sa minimalizovali kríženia hrán.

3. **Vrstvové(hierarchické, Sugiyama-style) usporiadanie** - je podobné stromovému, ale funguje aj na skoro acyklických orientovaných grafoch. Vrcholy sa najprv usporiadajú, aby hrany smerovali zostupne, potom sa vrcholy zarovnajú na jednotlivé úrovne a nakoniec sa medzi sebou poposúvajú aby zminimalizovali kríženia hrán.

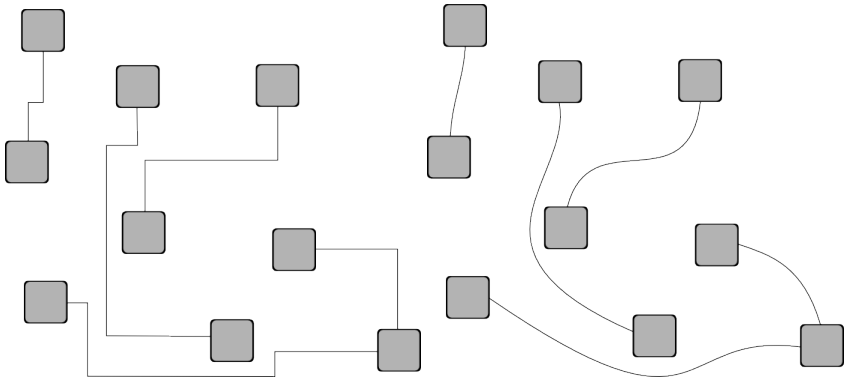
4. **Dominančné usporiadanie** - sa používa v acyklickom orientovanom grafe, v ktorom cesta z vrchola $v_1\,(x_1, y_1)$ do $v_2\,(x_2, y_2)$ existuje práve vtedy, keď $x_1 \leq x_2 \wedge y_1 \leq y_2$.

5. **Spektrálne usporiadanie** - používa vlastné vektory incidenčnej matice (Laplaceovej) ako súradnice vrcholov. [21]

6. **Silové usporiadanie** - nastavením odpudivých a príťažlivých síl medzi objektami grafu je možné postupne meniť súradnice vrcholov, kým nenastane rovnovážny stav. [14]

7. **Vzostupné usporiadanie** - v orientovanom grafe rozmiestni vrcholy tak, aby hrany neklesali v smere osi y.

Nasledujúca množina je charakterizovaná hlavne tým, že usporadúva aj vrcholy aj hrany. Prvé 3 sa však dajú použiť aj v prípade, že usporadúvame iba hrany do roviny a vrcholy už usporiadané máme.



(a) Ortogonálne (b) Organické

Obrázok 3.1: Príklad usporiadania

1. **Všeobecné usporiadanie s krivkami** - kreslí hrany ako lomené čiary a preto sa musí zvlášť venovať aj hranám.

2. **Ortogonálne usporiadanie** - je definované ortogonálnym grafickým štandardom a s obľubou používané pri návrhoch elektronických obvodov. Ortogonálne vedenie hrán je založené na vykreslení hrany ako lomenej čiary, pričom uhol, pod ktorým sa láme je vždy pravý, teda 90°. [3]

3. **Organické usporiadanie** - je použité v editore yEd, kde sa hrany vykresľujú ako krivky a spájajú sa do jednej, keď idú tým istým smerom. Medzi vrcholmi však musí byť dostatočný priestor na vedenie hrany. [2]

4. **Usporiadanie na základe viditeľnosti** - je reprezentácia planárneho grafu, v ktorom je každý vrchol reprezenotvaný ako horizontálny úsek a hrana medzi dvoma vrcholmi je znázornená ako vertikálna čiara spájajúca tieto dva horizontálne úseky, pričom nesmie pretínať vrchol, s ktorým nie je incidentná.[23] (Nie je to graf viditeľnosti ako v definícii 2.2.4)

5. **Planárne usporiadanie** - zobrazuje planárne grafy do roviny

6. **Dlaždicové usporiadanie** - vrcholy, hrany aj vnútorné plochy planárnej reprezentácie grafu zobrazuje ako obdĺžniky, ktoré sú susedné vzhľadom na geometrickú susednosť v pôvodnom grafe. [22]

Pre náš cieľ je zaujímavé hlavne Ortogonálne, Organické a Všeobecné usporiadanie. Organické usporiadanie spĺňa veľa z našich estetických kritérií, nespĺňa však napríklad maximálny uhol hrán vychádzajúcich z toho istého vrchola a práve naopak minimalizuje tento uhol.
Ortogonálne usporiadanie je na tom podobne, lebo spĺňa tiež väčšinu kritérií až na najkratšiu dĺžku hrany. Navyše majú jednu negatívnu vlastnosť, vďaka ktorej sme sa rozhodli o výskum v tejto oblasti a to, že takéto grafy sa relatívne ťažko čítajú. Keď je hrán veľa pri sebe splývajú a je ťažké sledovať jednotlivé spojenia. Tento typ vedenia hrán najčastejšie využívajú v oblasti VLSI, kde sa rieši problém ako zapojiť elektrický obvod na plochu, teda planárne vykresliť hrany. Ortogonálne usporiadanie nie je síce vhodné pre naše estetické kritériá, ale budeme ho ešte používať pri návrhu. Našou úlohou bude teda rozšíriť Všeobecné usporiadanie s lomenými čiarami.

3.2 Algoritmy

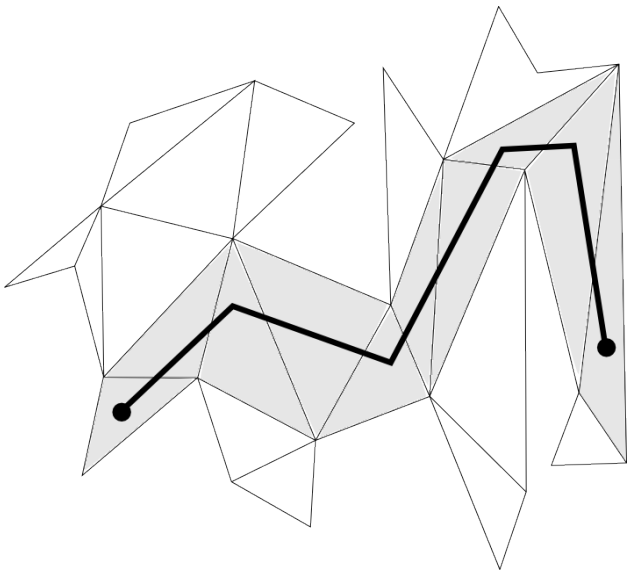
Poznáme 3 základné typy režimov ako môžu algoritmy vykresľovať hrany. Statický režim funguje tak, že vypočíta rozmiestnenie vrcholov a automaticky vykreslí hrany algoritmom alebo heuristikou. Prístup tohto riešenia, je síce jednoduchý, no použiteľnosť sa znižuje práve tým, že pri vytváraní nových hrán a vrcholov nevidíme výsledok a keď sa k tomu pridá premiestnenie vrcholov po

vykreslení, tak používateľ nedokáže dobre sledovať zmeny v grafe a tiež sémantický význam grafu.
Dynamický režim postupne pridáva hrany a vrcholy do už existujúceho grafu. Pri tom môžeme celkom dobre pozorovať, ako bude vyzerať graf po pridaní vrchola, alebo hrany, pretože výpočet môže byť aplikovaný počas vytvárania grafu. Problém môže nastať v tom, že pri istých prípadoch sa môže celý graf úplne zmeniť, aby sa zachovala planarita alebo iné estetické kritériá a preto aj v tomto riešení je niekedy ťažké sledovať sémantický význam grafu.
V inkrementálnom režime[4] postupne pri pridávaní hrán a vrcholov možno dobre pozorovať ako sa graf vytvára podobne ako v dynamickom. Rozdiel je v tom, že vrcholy sa nepremiestňujú, teda zachováva topológiu a zároveň nemusí spĺňať podmienku planarity grafu. Tým dostaneme graf, v ktorom sa môžu pretínať hrany. Pre nás sú vhodné dva režimy: Statický a Inkrementálny, lebo vrcholy v našom grafe sa nikdy nepresúvajú.

Jedným z algoritmov použiteľných pre vedenie hrán je Ortogonálne vedenie, ktoré vytvára Ortogonálne usporiadanie. Najznámejší algoritmus, ktorý počíta najkratšiu vzdialenosť je od Leeho[5]. Je to vlastne implementácia Dijkstrového algoritmu, s tým že plocha sa rozdelí na štvorcovú mriežku a postupne sa posúvajú vlny v tvare kosoštvorca. V praxi pre jednoduché aplikácie je však tento prístup úplne nevhodný, vzhľadom na veľkú časovú aj pamäťovú zložitosť. Soukop [6] vylepšil tento algoritmus tak, že až keď narazí na prekážku po ceste, tak začne prehľadávať Dijkstovým algoritmom ako obísť prekážku. Tým sa zložitosť znížila síce iba o konštantu, ale v praxi to má viditeľný význam. Tento algoritmus sa tiež podobá na heuristiku A* [7] [8]. Rozdiel je hlavne vo výpočte, avšak priebeh algoritmu je veľmi podobný. Jedno z ďalších vylepšení od Reinhar-

da [9] zlepšuje algoritmus tak, že nepracuje na rovnomernej mriežke, ale urobí štruktúru na základe vrcholov, ktoré treba obchádzať, tým sa zníži počet krokov algoritmu ale je potrebné zložitejšie počítať presnú trasu hrany.

Medzi najpoužiteľnejší algoritmus, ktorý dokáže usporiadať hrany podľa našich estetických kritérií by patril algoritmus pre Silové usporiadanie, ak by sme mu zafixovali pozície vrcholov. Problém môže nastať pri veľkých grafoch, pre ktoré by trval veľmi dlho a jeho časová zložitosť je pravdepodobne $O\left(n^3\right)$. Tento algoritmus bol pre nás vzorom ako by mali byť vedené hrany v grafe s tým, že sme sa snažili o jeho aproximáciu, aby sme znížili časovú náročnosť pôvodného algoritmu.



Obrázok 3.2: Graf viditeľnosti

Za zmienku stojí aj prístup z robotiky, kde sa snažia podobne ako vedenie

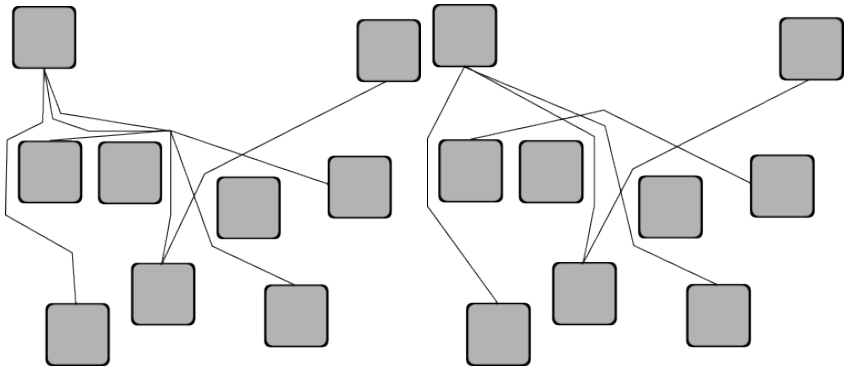
hrán v grafe nájsť trasu pre koordináciu robota,aby sa vyhol prekážkam [10] [11] [12]. Poväčšine sú tieto algoritmy založené na počítaní najkratšej cesty v Euklidovskom priestore. V niektorých sa vypočíta najprv triangulácia priestoru a vytvorí sa graf viditeľnosti(Visibility graph). Potom sa nájde najlepšia postupnosť trojuholníkov a nakoniec najlepšia cesta cez tieto trojuholníky. Táto trasa nie je najkratšia, lebo pre účely robotiky je bezpečnejšie, keď robot neprechádza tesne popri prekážke.

Existuje ešte veľa ad-hoc algoritmov a heuristík, ktoré vedú hrany. Väčšinou ide o snahu vykresliť hranu čo najpriamejšie, alebo planárne.

Pre naše účely si rozoberieme už len jednu heuristiku, ktorá má najbližšie k nášmu zadaniu a to je heuristika od Jiřího Dokulila a Jany Katreniakovej, ktorá má rovnaké estetické kritériá. Teda hrany sú lomené čiary, pri ktorých optimalizujeme:

1. uhol zlomu, aby bola čiara, čo najpriamejšia
2. počet zlomov bol čo najmenší
3. aby hrany nesplývali, keď napríklad veľa hrán vchádza do toho istého vrchola, alebo keď sa veľa hrán ohýba okolo toho istého vrchola
4. nemusí spĺňať planaritu a pri pretínaní dvoch hrán sa snaží byť čo najbližšie k vzájomnému uhlu 90°.

Táto heuristika spočíva z 3 krokov. Najprv pre všetky hrany posunie a ohne hrany okolo vrcholov, ktoré križujú dané vrcholy. Potom na ohyboch, kde je viacero hrán veľmi blízko pri sebe vypočíta poradie a posunie ich ďalej od seba. Nakoniec povyrovnáva hrany, ktoré sú zbytočne poohýbané. Táto heuristika



(a) pred (b) po

Obrázok 3.3: Aplikovanie estetických kritérií podľa Dokulila a Katreniakovej

je celkom efektívna a dosahuje pomerne dobré výsledky pri zobrazovaní, preto sme sa rozhodli na nej ďalej stavať. Bola navrhnutá pre jednoduché grafy, kde sú disjunktné vrcholy, preto ju bude treba upraviť.

Kapitola 4

Návrh

Našou úlohou bolo navrhnúť algoritmus na vedenie hrán v grafe, kde sa vrcholy(štvorce) alebo ich minimálne okraje rozlíšiteľnosti môžu prekrývať. Hrany by mali byť vykreslené podľa estetických kritérií, ktoré sme si zvolili v definícii 2.3.1. Okrem toho sme si túto úlohu preformulovali všeobecnejšie, teda vrcholy grafu môžu byť reprezentované jednoduchými polygónmi. Najprv navrhneme vytvorenie polygónov z nedisjunktných štvorcov, potom vytvorenie minimálneho okraju rozlíšiteľnosti, heuristiku na nájdenie odhadu najkratšej cesty medzi dvoma vrcholmi a heuristiku na vedenie hrán pomocou penalizačnej funkcie a Dijkstrovho algoritmu.

4.1 Zhlukovanie vrcholov

Ako prvé si treba graf vytvoriť, to by nemal byť problém, lebo pre náš algoritmus potrebujeme iba vrcholy s fixnými pozíciami a hrany, čo sú vlastne iba dve množiny. Ak máme graf potrebujeme zistiť, či tam nejaké dva vrcholy nemajú

prienik. Štruktúra, v ktorej si budeme pamätať, že vrcholy majú prienik nazývame zhluk.

Definícia 4.1.1 *Zhluk je tranzitívny uzáver vrcholov vzhľadom na prienik.*

Problémov tam vzniká hneď niekoľko. Napríklad, ak vyrobíme zhluk z dvoch vrcholov a potom zistíme, že sa pretína s iným vrcholom, tak pri naivnej implementácii vytvoríme ďalší zhluk namiesto toho, aby sme ho pridali do už existujúceho zhluku.

Rozdelíme si to teda na 4 prípady:

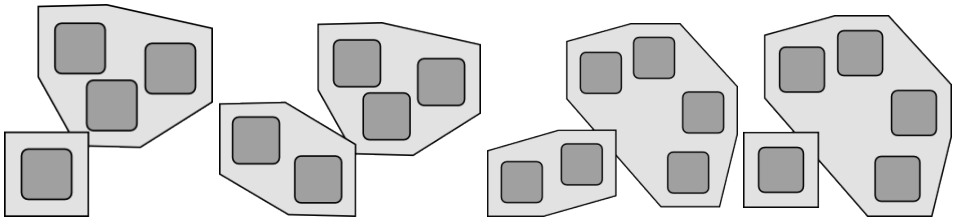
1. Keď sa pretínajú dva vrcholy, ktoré nie sú v žiadnom zhluku - jednoducho vytvoríme nový zhluk

2. Keď sa pretína vrchol v zhluku Z s vrcholom, ktorý nie je v žiadnom zhluku - jednoducho pridáme tento vrchol do zhluku Z.

3. Ak sa pretína vrchol v zhluku Z_1 s vrcholom v zhluku Z_2 - zmažeme zhluk Z_2 a všetky vrcholy premiestnime do zhluku Z_1.

4. Keď sa pretína zhluk Z_1 so zhlukom Z_2, ale žiadne z ich vrcholov sa nepretínajú. Alebo ak sa pretína vrchol so zhlukom Z_1, ale nepretína sa so žiadnym vrcholom zo zhluku Z_1. V tomto prípade neurobíme nič, lebo v skutočnosti je tento prienik zanedbateľný, lebo konvexný obal zhluku je redundantný a treba ho zoptimalizovať.

Zložitosť tejto časti je minimálne $O\left(n^2\right)$, lebo musíme zistiť každú možnú hranu a tých je $n * (n-1)/2$. Okrem toho, ak by sme to takto implementovali, tak pri každom rozhodovaní musíme skontrolovať všetky zhluky a v nich všetky

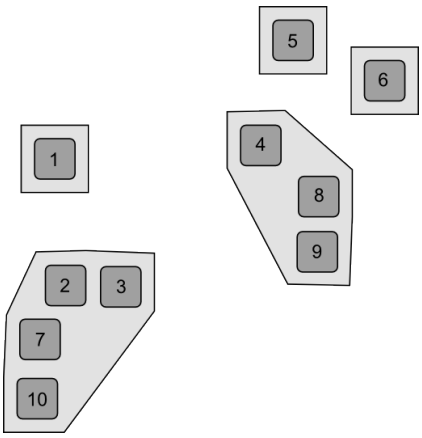


(a) Prípad 2 (b) Prípad 3 (c) Prípad 4 (d) Prípad 4

Obrázok 4.1: Prípady pretínania vrcholov

vrcholy. Kvôli lepšej zložitosti si vytvoríme incidenčnú maticu, do ktorej zapíšeme všetky pretínania vrcholov. Následne iba vytvoríme zhluky, ktoré treba pretraverzovaním incidenčnej matice.

	1	2	3	4	5	6	7	8	9	10
1	0	x	x	x	x	x	x	x	x	x
2	0	0	x	x	x	x	x	x	x	x
3	0	1	0	x	x	x	x	x	x	x
4	0	0	0	0	x	x	x	x	x	x
5	0	0	0	0	0	x	x	x	x	x
6	0	0	0	0	0	0	x	x	x	x
7	0	1	0	0	0	0	0	x	x	x
8	0	0	0	1	0	0	0	0	x	x
9	0	0	0	0	0	0	0	1	0	x
10	0	0	0	0	0	0	1	0	0	0



(a) Incidenčná Matica (b) Graf

Obrázok 4.2: Príklad zhlukovania vrcholov

4.2 Konvexný obal a okraj rozlíšiteľnosti

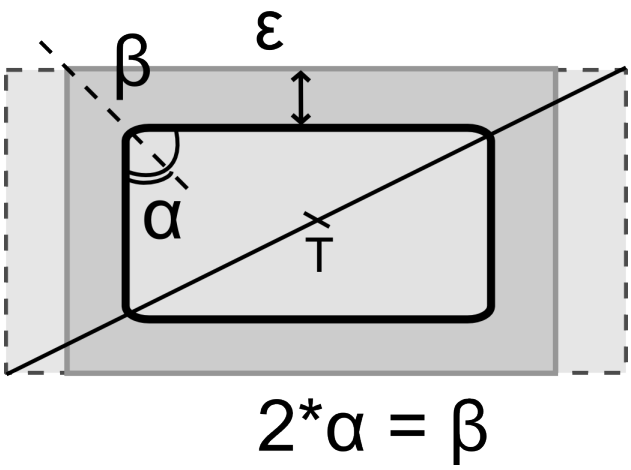
Po tom ako vytvoríme zhluky potrebujeme vytvoriť ich konvexné obaly, ktoré budú naše polygónové vrcholy. Konvexný obal z množiny bodov dokážeme urobiť pomocou Chanového algoritmu[25] v čase $O\left(n * log\left(h\right)\right)$, kde n je počet všetkých bodov a h je počet bodov na výslednom konvexnom obale. Okrem toho poznáme efektívne algoritmy na tvorbu konvexného obalu ako Grahamov prechod[26] a Jarvisov pochod(balenie balíčka)[27], ktoré majú trochu horšiu zložitosť $O\left(n * log\left(n\right)\right)$, ale nám to stačí, lebo nepresiahneme $O\left(n^2\right)$, lebo každý z vrcholov je iba v jednom zhluku. Vybrali sme si balenie balíčka, pre jeho jednoduchosť.

Po vytvorení konvexného obalu môže nastať situácia, že veľa priestoru zostane nepokrytého vrcholami zo zhluku. Zistíme to na základe pomeru voľnej a pokrytej plochy v zhluku. Vieme to vylepšiť vytvorením štruktúry **strom konvexných obalov**, kde rozdelíme zhluk na viacero konvexných podzhlukov tak, aby bola pokrytá plocha pod vrcholmi väčšia ako nepokrytá.

Konvexný obal tvorí jednoduchý polygón zhluku. Pre ďalšie použitie nás ale bude zaujímať aj jeho okraj rozlíšiteľnosti.

Definícia 4.2.1 *Minimálny okraj rozlíšiteľnosti je vzdialenosť od objektu, v ktorej sa nesmie nachádzať iný objekt, aby sa dali objekty ľahko identifikovať.*

Jeden z nápadov ako implementovať okraj je použiť dilatáciu pomocou ťažiska. Tento prístup, ale nevytvorí okraj s rovnakou vzdialenosťou od objektu. Dobre sa tento rozdiel ilustruje na obdĺžniku. Dilatácia zväčší rovnomerne obdĺžnik, ale vzdialenosť od hrán je rozdielna. Ak body okraju vypočítame pomocou osi uhla pri rohu obdĺžnika, dostaneme rovnakú vzdialenosť.



Obrázok 4.3: Minimálny okraj rozlíšiteľnosti

4.3 Heuristika 1

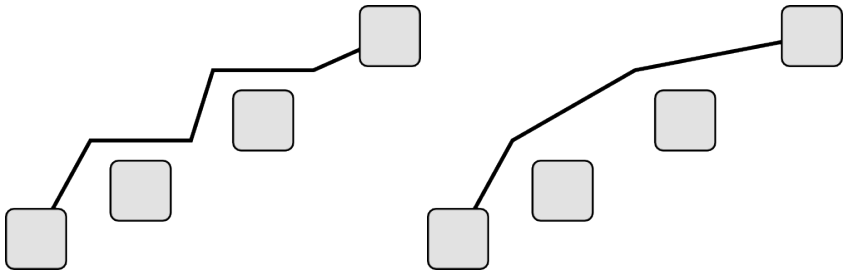
Pri analýze problému a hľadaní spôsobu ako ľahko vypočítať prienik priamky s polygónom sme našli istú súvislosť s ťažiskom polygónu.

Veta 4.3.1 *Ak priamka pretína jednoduchý konvexný polygón, tak určite pretne aspoň jednu zo spojníc bodov polygónu a jeho ťažiska.*

Dôkaz tejto vety môžeme oprieť o definíciu jednoduchého konvexného polygónu. Akokoľvek by sme rozdelili polygón priamkou p, na každej strane musí byť aspoň jeden bod polygónu. A keďže je ťažisko len jedno, musí byť na jednej strane priamky(alebo na nej), minimálne jedna spojnica bude teda pretínať priamku p.

Vďaka tomuto faktu môžeme potom pri jednoduchom testovaní, či hrana pretína polygón, ľahko zistiť aj to ktoré sú potencionálne vrcholy na obchádzanie, kade môže ísť hrana. Ťažisko je teda dobrý rozdeľovací bod pre vedenie hrán.

Heuristika založená na tejto podmienke je veľmi rýchla a dosahuje celkom dobré výsledky vzhľadom na estetické kritérium dĺžky hrán. Objavujú sa tam však aj menšie nepresnosti, lebo nie všetky vrcholy, ktoré sú na spojnici s ťažiskom a pretne ich najkratšia vzdialenosť medzi vrcholmi, je dobré obchádzať. To sa dá ľahko odstrániť skontrolovaním a zjednodušením hrany. Ak sa nachádzajú dva body na hrane, ktorých priama úsečka nepretína žiaden vrchol, tak úsek medzi nimi možno odstrániť.



(a) Naivné (b) Zjednodušené

Obrázok 4.4: Príklad zjednodušenia hrany

Pre lepšie pochopenie si predstavíme jednotlivé fázy heuristiky a neskôr objasníme detaily.

1. Fáza - Utriedi hrany podľa dĺžky

   Ostatné fázy platia pre jednu hranu z utriedeného zoznamu:

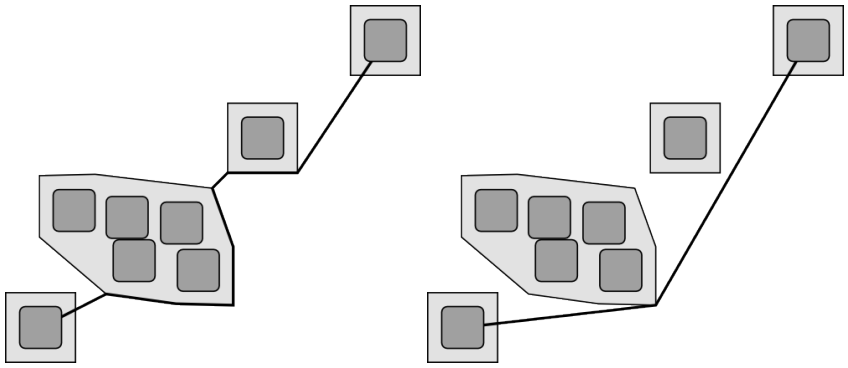
2. Fáza - Nájde zhluky a body, ktoré pretína najkratšia úsečka medzi vrcholmi

3. Fáza - Nájde vrcholy a body, ktoré pretína najkratšia úsečka

4. Fáza - Nájde najlepšie body von zo zhlukov

5. Fáza - Zjednoduší hranu a použije minimálny počet bodov

6. Fáza - Lokálne zoptimalizuje poradie hrán pri vrcholoch a zhlukoch



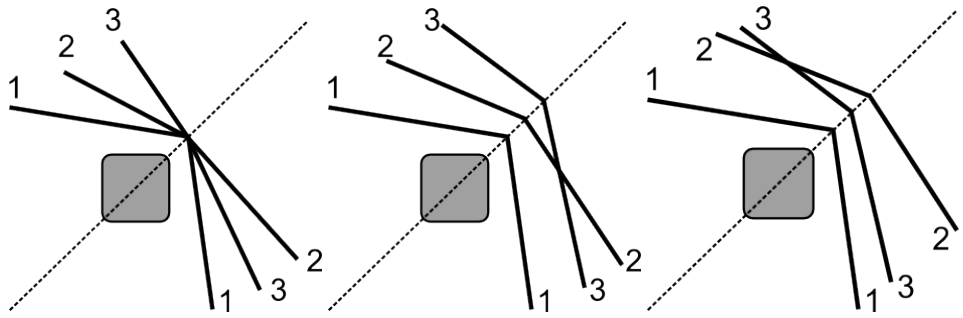
(a) Po 4. Fáze (b) Po 6. Fáze

Obrázok 4.5: Postup Heuristiky 1

Prvá fáza je dôležitá kvôli správnemu poradiu hrán pri vrchole, keď vrchol obchádza okolo toho istého bodu viac hrán. Tento problém sa rieši neskôr v 2. a 3. fáze. Vychádzame z predpokladu, že keď ako prvé budeme viesť hrany, ktoré majú veľkú penalizáciu, dostaneme poradie pri vrchole podobné optimálnemu usporiadaniu hrán a vyhneme sa veľkým zmenám usporiadania, ktoré by mohli nastať v okrajových prípadoch.

Tento predpoklad vychádza z pozorovania v heuristike od Dokulila a Katreniakovej[1], kde sa pri vrchole hľadala správna permutácia hrán, aby sa dosiahlo, čo možno najmenej pretínaní hrán. Výsledok bol väčšinou taký, že ostrejšie uhly sa snažili posúvať bližšie k vrcholu z každej strany obchádzania, lebo potom sa tak veľmi nepretínali s ostatnými hranami. Problematické boli také uhly, ktoré boli skoro rovnaké, ale mali mať z každej strany iné poradie pri vrchole. Ich

poradie sa nedalo usporiadať bez krížení a dobré riešenia boli viaceré, kde stačilo vymeniť krížiace sa hrany.



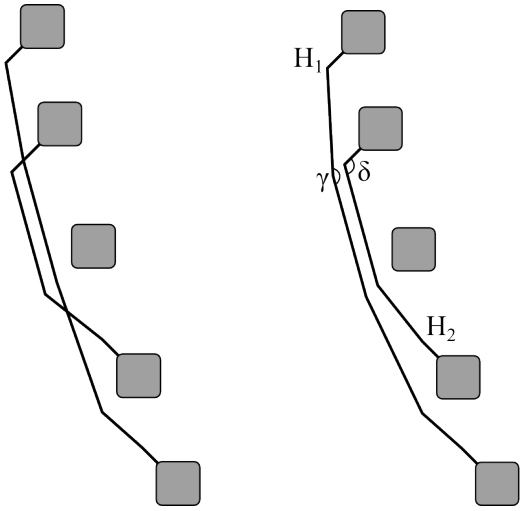
(a) Pred použitím heuristiky (b) Riešenie 1 (c) Riešenie 2

Obrázok 4.6: Heuristika Dokulila a Katreniakovej

Pre zjednodušenie tohto pravidla budeme vzájomne porovnávať celkové veľkosti uhlov obchádzajúcich hrán. Ak vrchol obchádzajú dve hrany H_1 a H_2, uhol $\delta \in H_1$ a uhol $\gamma \in H_2$, zároveň $\delta \leq \gamma$, tak z veľkou pravdepodobnosťou bude optimálne usporiadanie pri vrchole $H_1 \geq H_2$. Keď sa totiž nachádza pri vrchole ostrý uhol a podarí sa nám ho zmenšiť alebo zmenšujeme dĺžku hrany, znižujeme penalizáciu.

V druhej fáze na základe už spomínanej vety 4.3 o ťažisku polygónu nájdeme tie polygóny a vrcholy, ktoré prichádzajú v úvahu. Pomocou pretínania najkratšej hrany nájdeme prvotný zoznam. Tento sa môže neskôr rozšíriť, ak niektorý úsek novej hrany pretína ďalší vrchol alebo zhluk.

Tretia fáza dostane na vstupe zoznam pretínajúcich zhlukov a vrcholov, ktoré sme dostali z 2. fázy. Cieľom tejto fázy je nájsť presné body hrany, ktoré majú obchádzať vrchol alebo zhluk. Znovu použijeme vetu o ťažisku 4.3 na nájdenie konkrétnych bodov. Ak sa cez jeden roh polygónu už jedna hrana vedie, ďalšia



(a) Neoptimálne (b) Optimálne

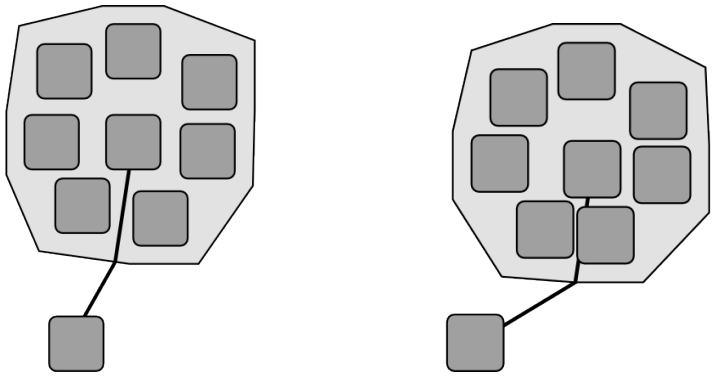
Obrázok 4.7: Obchádzanie dvoch hrán

hrana môže ísť až o kúsok ďalej, kvôli dobrej rozlíšiteľnosti. Toto poradie nie je však konečné a dokončí sa v 6. fáze po lokálnej optimalizácii.

Druhá a tretia fáza sa musia opakovať, lebo pri vytvorení nového úseku sme mohli zasiahnuť do iného vrchola alebo zhluku. Bude sa preto opakovať pokým nejaký úsek hrany pretína ľubovoľný vrchol alebo zhluk.

V štvrtej fáze nájdeme najlepší bod na okraji zhluku, ktorým danú hranu povedieme. Tento bod bude buď v rohu, alebo v päte kolmici na okrajovú hranu zo stredu vrchola. Nemalo by sa však porušiť pravidlo, že hrana pretína iný vrchol zo zhluku. Môže nastať prípad, keď sú okolo vrchola, z ktorého vychádza hrana tak blízko ostatné vrcholy, že neexistuje priama cesta, bez porušenia pretínania iného vrchola. Mohli by sme hľadať najlepšiu cestu zo zhluku aj zložitejšie - obchádzaním vnútorných vrcholov. Túto myšlienku sme sa rozhodli neimplementovať vzhľadom na to, že je to veľmi okrajová záležitosť a náročne by sa musela

realizovať. Ak má hrana koncové body vo vrcholoch, ktoré nie sú v zhluku, tak túto fázu nevykonávame.



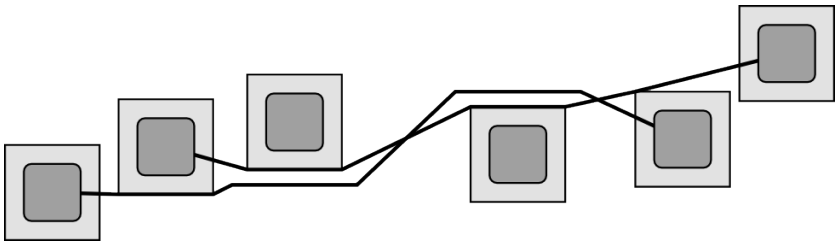
(a) Existuje cesta (b) Neexistuje cesta

Obrázok 4.8: Príklad vychádzania hrany zo zhluku

V piatej fáze odstránime všetky nadbytočné úseky hrany tak, že skontroluje každý úsek, či sa nedá skrátiť. Táto fáza je problematická vzhľadom k tomu, že môže existovať viacero rôznych skrátení a je dosť ťažké určiť, ktoré je optimálne.

Vo všeobecnosti neplatí, že stačí hrany usporiadať globálne podľa penalizácie a budú dobre obchádzať okolo každého vrchola. Preto potrebujeme 6. fázu, v ktorej v každom okrajovom bode vrchola a zhluku musíme hrany správne lokálne usporiadať. Usporiadanie je na základe veľkosti uhla pri vrchole a celkového zníženia penalizácie tak, aby najostrejšie uhly boli čo najviac pri vrchole, a aby sa znížila penalizácia zmenených hrán.

Táto heuristika by sa dala nazvať vylepšením vedenia hrán od Dokulila a Katreniakovej.



Obrázok 4.9: Problém pre lokálnu optimalizáciu

4.4 Heuristika 2

Pri návrhu druhej heuristiky sme vychádzali z Dijkstrovho algoritmu a heuristiky A*. Ten potrebuje pre svoje fungovanie ohodnocovaciu funkciu, podľa ktorej sa rozhoduje, ktoré vrcholy rozvinie. Navrhli sme penalizačnú funkciu f_p tak, aby odrážala estetické kritériá. $f_p = \frac{d}{D}W_d + W_u \sum_{i=1}^{v} \frac{360}{H_i \alpha_i} + W_\varepsilon r$

- d - celková dĺžka jednotlivých úsekov navrhovanej hrany
- D - dĺžka najkratšej hrany medzi dvoma vrcholmi
- W_d - váha, akú má kritérium dĺžky hrany
- W_u - váha, akú má kritérium uhlu hrán
- v - počet bodov, kde sa hrana láme alebo pretína inú hranu
- H_i - počet vychádzajúcich hrán z bodu. Keď sa láme $H_i = 2$
- α_i - najmenší uhol medzi vychádzajúcimi hranami z bodu
- W_ε - váha kritéria porušenia minimálneho okraja rozlíšiteľnosti
- r - počet porušení okraja

V literatúre sú už pre ortogonalitu a iné estetické kritériá metriky spísané [16]. Pre náš problém nie sú celkom vhodné, nakoľko potrebujeme jednu funkciu pre Dijkstrov algoritmus. Heuristika 2 bude postupovať podobne ako prvá heuristika, teda pripraví si pomocné štruktúry a po jednotlivých hranách usporiada.

1. Fáza - Utriedenie hrán
2. Fáza - Visibility graf Ďalšie fázy platia pre jednu hranu:
3. Fáza - Vchádzanie a vychádzanie zo zhluku
4. Fáza - Upravený Dijkstrov algoritmus
5. Fáza - Lokálne úpravy

V prvej fáze je potrebné stanoviť čo najlepšie poradie, aby boli lokálne úpravy čo najmenšie. Ak utriedime hrany iba podľa ich dĺžky nemusíme dostať dobré globálne poradie. Ak by sme však použili heuristiku 1 na zistenie dobrého poradia, dostaneme poradie, ktoré sa od optimálneho veľa nelíši a lokálne úpravy budú v priemernom prípade minimálne.

V druhej fáze vytvoríme dôležitú štruktúru graf Viditeľnosti - tzv. Visibility graf. Je to štruktúra, v ktorej sú spojené práve tie body, ktoré sa ”vidia”, teda nepretínajú vrchol ani zhluk. Táto štruktúra nám bude slúžiť na hľadanie správnych bodov na obchádzanie vrcholov a zhlukov. V Dijkstrovom algoritme potrebujeme poznať susedov daného bodu, a aby sme nemuseli pri každej hrane znova týchto susedov zisťovať, urobíme to naraz, na začiatku a pre všetky body. Problém ale nastáva hneď na začiatku, lebo nevieme presne, ktorý bod je ten najlepší, kade môžeme danú hranu viesť. Pri ortogonálnom vedení to je celý

priestor, čo môže byť veľmi veľa bodov. To je ale nereálne a nie celkom potrebné. Ak hľadáme podľa našich estetických kritérií také vedenie hrán, kde sú hrany v prvom rade čo najkratšie, podstatné sú pre nás body nachádzajúce sa v tesnom okolí vrcholov (štvorcov) a zhlukov (polygónov). Konkrétne sú to body v rohoch vrcholov a zhlukov [29]. Ak by sme dávali dôraz na iné kritérium museli by sme brať do úvahy omnoho viac bodov.

Tretia fáza je prípravná, aby sme mohli začať Dijkstrov algoritmus. Predpoklady na to sú, že každý vrchol musí poznať suseda, musí vedieť vypočítať v každom bode penalizačnú funkciu a musí byť zadaný počiatočný a koncový bod. Pri vrcholoch by to nemal byť problém, nakoľko z vrchola môžeme hranu viesť ľubovoľným smerom. Zhluky, keďže sú zložené z viacerých vrcholov, potrebujeme vyriešiť samostatne. Podobne ako v prvej heuristike nájdeme na okraji polygónu body, ku ktorým môžeme viesť hranu z vrchola bez pretínania ostatných vrcholov v zhluku. Ak hrana začína zhlukom, tak tieto body na začiatku hrany, na okraji zhluku ohodnotíme a dáme do zoznamu otvorených bodov. Ak sa hrana končí zhlukom, tak musíme zabezpečiť, že algoritmus skončí. Do zoznamu otvorených bodov preto pridáme podobne body z okraja zhluku, kde hrana končí. Penalizáciu nastavíme na (takmer) nekonečno a keď sa tam v Dijkstrovom algoritme dostaneme, tak sme na konci.

V štvrtej fáze prebieha samotný Dijkstrov algoritmus, kde na začiatku vloží prvý bod do zoznamu otvorených bodov. Potom tento zoznam utriedi tak, aby bol na začiatku zoznamu bod s minimálnou penalizáciou. Vyberieme prvý bod zo zoznamu otvorených bodov, pre všetkých jeho susedov vypočítame penalizáciu, vložíme ich do zoznamu otvorených bodov a tento prvý bod presunieme do zoznamu zatvorených bodov. Tento postup opakujeme, pokým nie je na za-

čiatku zoznamu otvorených bodov koncový vrchol hrany. Penalizácia je trochu upravená a podobá sa preto na heuristiku $A*$. Pri počítaní pomeru dĺžky hrany s dĺžkou najkratšej hrany sa nám môže stať, že bude algoritmus neefektívny. Ak budú v grafe Viditeľnosti body v inom smere ako koncový bod hrany priamejšie ako správna cesta, tak sa nimi bude zaoberať algoritmus, aj keď je to zlá cesta. Pridaním heuristickej predpovede ako v heuristike $A*$ môžeme tieto neželané cesty podstatne zredukovať. Heuristická predpoveď d_h bude pre nás úsečka z daného bodu do koncového bodu hrany. Výsledná penalizácia bude teda: $\frac{d+d_h}{D}$ kde D je dĺžka najkratšej hrany od začiatku po koniec hrany a d je aktuálna vzdialenosť od začiatku po spracovávaný otvorený bod.

V piatej fáze nasledujú úpravy podobne ako na konci prvej heurstiky. Je to usporiadanie hrán pri vrchole podľa veľkosti uhla a zníženia penalizácie. Okrem toho je potrebné vymazať nadbytočné body, ktoré boli pomocné pre dijkstrov algoritmus na začiatku a na konci hrán.

Kapitola 5

Implementačné detaily

V tejto kapitole si predstavíme pár detailov z implementácie, ktoré sú vhodné spomenúť, nakoľko ponúkajú zaujímavý spôsob riešenia daného problému.

Implementoval som program v jazyku C++ v prostredí CodeBlocks - 10.05 a s grafickou knižnicou wxWidgets - 2.8.12.

5.1 Pripodobnenie silovému usporiadaniu

Pri návrhu bola nosná myšlienka efektívne naprogramovať usporiadanie podobné silovému, ktoré by sa limitne blížilo k optimálnemu riešeniu. V silovom usporiadaní sa mnohonásobne opakuje algoritmus, až pokým sa nenájde rovnovážny stav všetkých síl. Výsledok je esteticky dobre vykreslený, ale zložitosť je veľmi veľká a pri väčších grafoch je toto riešenie nepoužiteľné. Zvolili sme metódu aproximácie optimálneho riešenia. Heuristika 1 používa veľmi jednoduchú funkciu na zistenie približnej cesty hrany, preto výsledok nám dá dobrý odhad výslednej penalizácie obchádzajúcej hrany. Tento odhad potom použijeme na usporiadanie

hrán pred začatím Heuristiky 2. Môže sa stať, že keby sme opakovali Heuristiku 2, tak by sa riešenia našli viaceré a alternovali by na základe počiatočného usporiadania hrán. Tento problém by mala vyriešiť posledná fáza algoritmu - Lokálne úpravy.

5.2 Štruktúry a reprezentácia dát

Na začiatku si spravíme krátky prehľad pojmov a štruktúr, ktoré som navrhol a implementoval.

Bod - je reprezentovaný v 2D priestore ($\mathbb{Z} \times \mathbb{Z}$) s presnosťou na desatiny. Je to potrebné kvôli priblíženiu, aby nám nevznikali nepresnosti. V knižnici wxWidgets som použil na vykresľovanie dcbuffer.h a ten používa ako základnú jednotku pixel. Keďže som potreboval lepšiu presnosť, uchovávam súradnice bodov prenásobené 10.

Priamka - je uložená v smernicovej rovnici priamky, kde je k - smernica a q - posun. Pri tejto rovnici priamky dochádza k istým nepresnostiam a skresleniu keď je priamka skoro vertikálna a vertikálna. Smernica vtedy dosahuje hodnoty (takmer) nekonečna.

Hrana - si pamätá priamku, body najkratšej úsečky medzi dvoma bodmi, celkovú penalizáciu hrany a jednotlivé body po vedení hrany v utriedenom zozname.

Vrchol - je vykreslený v tvare štvorca, konkrétne štvorca so zaoblenými rohami. Výpočty pracujú pre jednoduchosť s klasickým štvorcom. Pamätá si, či je súčasťou zhluku a jeho identifikátor.

Zhluk - má množinu vrcholov, ktoré do neho patria. Pamätá si konvexný

obal, jeho okraj a ťažisko.

Nástenka - je štruktúra, ktorá sa nachádza v každom rohu vrchola a zhluku. Všade tam, kde sa môže lámať hrana, teda aj pri vychádzaní zo zhluku - na okraji zhluku. Obsahuje počiatočný a verejný bod, ktorý je posledný voľný a dá sa tade viesť ľubovoľná hrana. Pamätá si aj medzeru ako ďaleko má byť ďalší bod, lebo v polygónoch, kde nie sú pravé uhly sú tieto vzdialenosti rozdielne.

Bod hrany - je špecifický bod. Pamätá si navyše či a do ktorého patrí vrchola alebo zhluku, identifikátor Nástenky, penalizáciu pri Heuristike 2 a predchádzajúci bod.

Graf viditeľnosti - nevytvára sa medzi vrcholmi grafu ako takými, ale medzi okrajovými bodmi polygónov, kade je možné viesť hranu podľa estetických kritérií, teda medzi Nástenkami. Vytvára sa na začiatku algoritmu a potom sa iba aktualizujú potrebné hrany. Uchováva vzdialenosti medzi bodmi, aby sme vedeli, či je ešte stále dosť priestoru medzi objektami. Keď už nie je, nemôžeme cez tento priestor viesť ďalšie hrany. Pamätá si susedov, medzi ktorými môže byť úsečka, ktorá nepretína ani vrchol ani zhluk. Okrem toho vie rozlišovať, či je na okraji objektu alebo medzi vrcholmi a zhlukmi.

5.3 Rozhranie

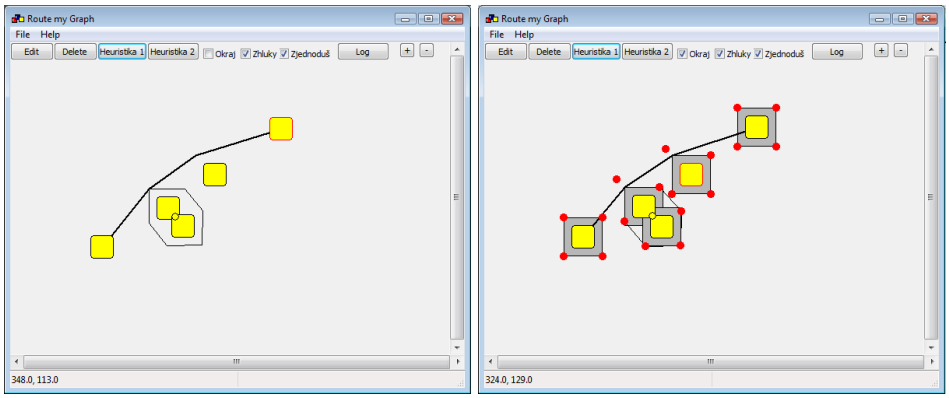
Používateľské rozhranie je intuitívne a podobné k ostatným grafovým softvérom. Ako prvé je nutné vytvoriť graf. Program pracuje v 3 módoch.

1. Vkladací - tzv. Insert mode, v tomto móde sa dajú vkladať vrcholy a hrany.

2. Upravovací - tzv. Edit mode, v tomto móde sa dajú presúvať vrcholy a mazať vrcholy a hrany

3. Zobrazovací - tzv. View mode, slúži na zobrazovanie a nedajú sa v ňom robiť žiadne zmeny

Vo vkladacom móde vložíme vrchol kliknutím ľavého tlačidla myši. Ak ľavé tlačidlo necháme stlačené, presunieme sa na druhý vrchol a pustíme ľavé tlačidlo vytvoríme hranu. V upravovacom móde držaním pravého tlačidla nad vrcholom meníme pozíciu tohto vrchola spolu s hranami, ktoré sú s ním spojené. Ak klikneme ľavým tlačidlom myši na vrchol, zmení farbu na červenú a môžme ho vymazať stlačením tlačidla Delete. Hranu označíme tým istým spôsobom ako sme ju vytvárali a potom ju môžeme vymazať.



(a) So zhlukmi (b) S okrajmi a nástenkami

Obrázok 5.1: Používateľské rozhranie

Priblíženie a oddialenie môžeme vykonať stlačením tlačidla + a -. Pri týchto operáciách sa nemenia vnútorné štruktúry, iba zobrazenie, takže možno ľubovoľne veľa krát opakovať túto činnosť. Ak by sme vnútorné štruktúry menili, čoskoro by nám vzniklo netriviálne skreslenie.

Na spustenie Heuristiky 1 jednoducho stlačíme tlačidlo s príslušným názvom. Po vykonaní sa graf vykreslí s novým vedením hrán. Podobne aj Heuristika 2.

Zaškrtnutím rámčeka Okraj dovolíme programu zobraziť Okraje jednotlivých objektov a nástenky. Zaškrtnutím rámčeka Zhluky zobrazí ohraničenie zhlukov a ich ťažisko. Zaškrtnutím rámčeka Zjednoduš zahrnie do algoritmu zjednodušovanie hrán. Stlačením tlačidla Log môžeme sledovať časy a penalizáciu jednotlivých spustení heuristík.

Výsledný graf sa dá exportovať do vektorovej grafiky vo formáte .svg.

5.4 Testovanie

Výsledné heristiky majú rôzne vlastnosti, preto je dôležité ich vzájomné porovnanie.

Heuristika 1	**Heuristika 2**
rýchla, menšia zložitosť	pomalá, väčšia zložitosť
nenájde optimálnu hranu	vždy nájde optimálnu hranu
nevie predísť prehusteniu hrán	dokáže zatarasiť cestu
nepotrebuje špeciálne štruktúry	potreba visibility grafu
potreba zjednodušovania hrany	nie je nutné zjednodušenie
vie zahrnúť iba dĺžku hrany	zahŕňa všetky estetické kritériá

Tabuľka 5.1: Porovnanie heuristík

Rýchlosť heuristík môžeme merať dvoma spôsobmi. Jeden je teoretický - zložitosť programu na základe počtu cyklov a druhý je praktický - koľko sekúnd trvá program na danej počítačovej zostave. Zložitosť heuristiky 1 sa dá celkom ľahko teoreticky odhadnúť, lebo je jednoduchá. Počet vrcholov budeme označovať N, počet všetkých hrán E a počet zhlukov Z.

Prípravnú fázu nezarátavame do porovnania, lebo je rovnaká u obidvoch heuristík. Je potrebné vytvoriť z vrcholov zhluky. Nájsť incidentné vrcholy a vytvoriť konvexný obal každého zhluku a to je $O\left(N^2 + Z\right)$.

- Na začiatku sa nachádza cyklus, ktorý prechádza všekými hranami E a utriedi ich - $O\left(E * log\left(E\right)\right)$
- Potom pre každý úsek hrany skontroluj, či ho nepretína vrchol alebo zhluk - $O\left(\left(4 * N\right)^2 * E\right)$ 4N - maximálny počet Nástenok na jeden vrchol.
- Ak úsek hrany niečo pretína, tak nájde cestu ako ho obísť - to je počet bodov zhluku alebo vrchola. $O\left(\left(4 * N\right)^2 * E\right)$
- Potom nájde cestu von zo zhlukov na začiatku a na konci hrany - to je opäť počet bodov zhluku
- Na záver zjednoduší každú hranu tak, že opäť skontroluje každý úsek, či sa nedá skrátiť - cez všetky zhluky a vrcholy $O\left(\left(4 * N\right)^3 * E\right)$

Celková časová zložitosť Heuristiky 1 je teda maximálne $O\left(\left(4 * N\right)^3 * E\right)$. Heuristika 2 je zložitejšia. Už len na začiatku máme na výber, ako utriedime hrany. Buď podľa dĺžky, alebo spustením Heuristiky 1 môžeme dostať dobrý odhad penalizácie jednotlivých hrán.

- Utriedenie - $O\left(E * log\left(E\right)\right)$ alebo $O\left(\left(4 * N\right)^3 * E\right)$.
- Prípravná štruktúra graf Viditeľnosti má veľkú zložitosť nakoľko musí zistiť, s kým má každý bod na okraji zhluku a vrchola voľnú cestu - $O\left(N^3 * Z + Z^2\right)$.
- Vychádzanie zo zhlukov je podobné ako v Heuristike 1 a to je zanedbateľné.

- Dijkstorv algoritmus má zložitosť $O\left(N^2\right)$, v našom prípade sú to všetky okrajové body, ktoré sú na okraji vrcholov a zhlukov. Nesmieme zabudnúť aj na zložitosť penalizačnej funkcie, ktorá musí skontrolovať pretínanie každej hrany s inými hranami. - $O\left(\left(N+Z\right)^2 * E^2\right)$.

Celková časová zložitosť Heuristiky 2 je $O\left(N^3 * Z + Z^2 + \left(N+Z\right)^2 * E^2\right)$.

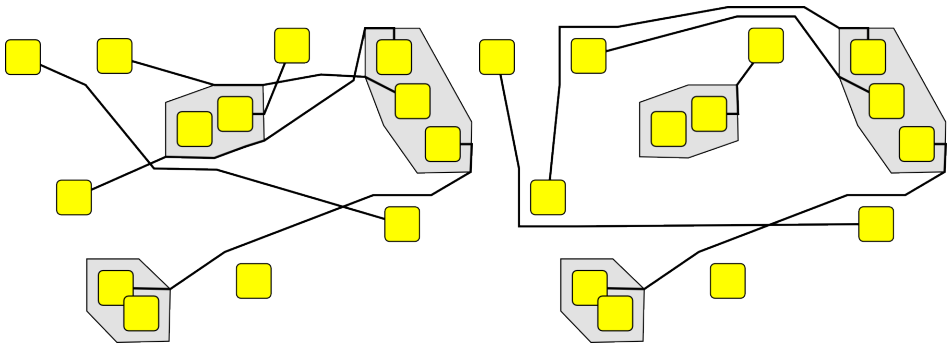
Praktickú rýchlosť heuristík som testoval na notebooku s procesorom 2.0 GHz Intel Core 2 Duo T7300 a pamäťou 2 GB, 667 MHz DDR2. Grafická karta bola použitá 128 MB Nvidia GeForce Go 8400 a operačný systém pod ktorým boli spustené bol Windows Vista Premium. Testoval som ich na náhodných grafoch s istým počtom vrcholov N, zhlukov Z a hránE.

Graf	**Heuristika 1**		**Heuristika 2**	
	sekundy	**kvalita**	**sekundy**	**kvalita**
5N, 4E	0.001	0.945160	0.013	1.036716
10N, 2Z, 7E	0.006	1.450872	0.063	1.267663
20N, 4Z, 15E	0.0032	1.037538	0.458	0.997230
30N, 10Z, 15E	0.004	1.062958	1.33	1.023399
60N, 15Z, 30E	0.117	1.009267	7.495	0.988424
25N, 4Z, 12E	0.013	0.955204	0.705	0.918595
25N, 4Z, 24E	0.016	0.955827	0.819	0.958933
25N, 4Z, 50E	0.023	0.944596	1.113	0.955929
10N, 45E - K_{10}	0.003	0.931962	0.171	0.942405
8N, 16E - $K_{4,4}$	0.002	0.946256	0.052	0.947336

Tabuľka 5.2: Časové porovnanie heuristík v sekundách a Kvalita grafov

Kvalitu výsledného grafu Q_g budeme merať pomocou penalizačnej funkcie, ktorá platí pre jednu hranu $f_{p_j} = \frac{d+H_d}{D} * W_d + W_{zlom} * \frac{\sum_{i=1}^{zlom_j} \frac{180}{\beta_i}}{zlom_j} + W_u * \frac{\sum_{i=1}^{v} \frac{90}{\alpha_i}}{vl} + W_\varepsilon r$. Čím je hodnota kvality nižšia, tým je výsledný graf krajší podľa našich kritérií. Pôvodnú penalizačnú funkciu sme museli trochu zmeniť, aby sme mohli lepšie ohodnocovať pretínania hrán W_u a zlomy hrany W_z. Počet bodov, kde sa hrana láme označíme vl a tam kde sa hrany pretínajú v. Uhol zlomu lomenej čiary hrany označíme β_i a najmenší uhol medzi pretínajúcimi sa hranami α_i. Počet zlomov hrany j označíme $zlom_j$ a váhu zlomov hrany ako W_{zlom}. H_d je heuristický odhad ako ďaleko je koniec hrany. Celková dĺžka jednotlivých úsekov navrhovanej hrany zostáva d, D - dĺžka najkratšej hrany medzi dvoma vrcholmi a W_d - váha, akú má kritérium dĺžky hrany. Počet porušení okraja - r a váha tohto kritéria bude W_ε.

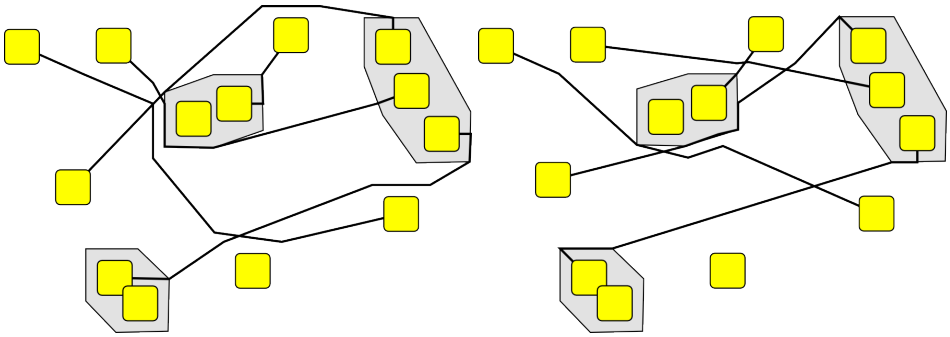
Celkovú kvalitu grafu Q_g vypočítame ako priemernú penalizáciu na hranu v grafe $Q_g = \frac{\sum_{j=1}^{E} f_{p_j}}{E}$.



(a) $W_u = 0.3, W_d = 2.0, W_{zlom} = 0.1$ (b) $W_u = 2.0, W_d = 2.0, W_{zlom} = 0.1$

Obrázok 5.2: Heuristika 2 s rôznymi váhami

Estetické hľadisko výsledných grafov môžeme zmerať pomocou zadaných



(a) $W_u = 0.3, W_d = 0.0, W_{zlom} = 3.0$ (b) Heuristika 1

Obrázok 5.3: Porovnanie Heuristík

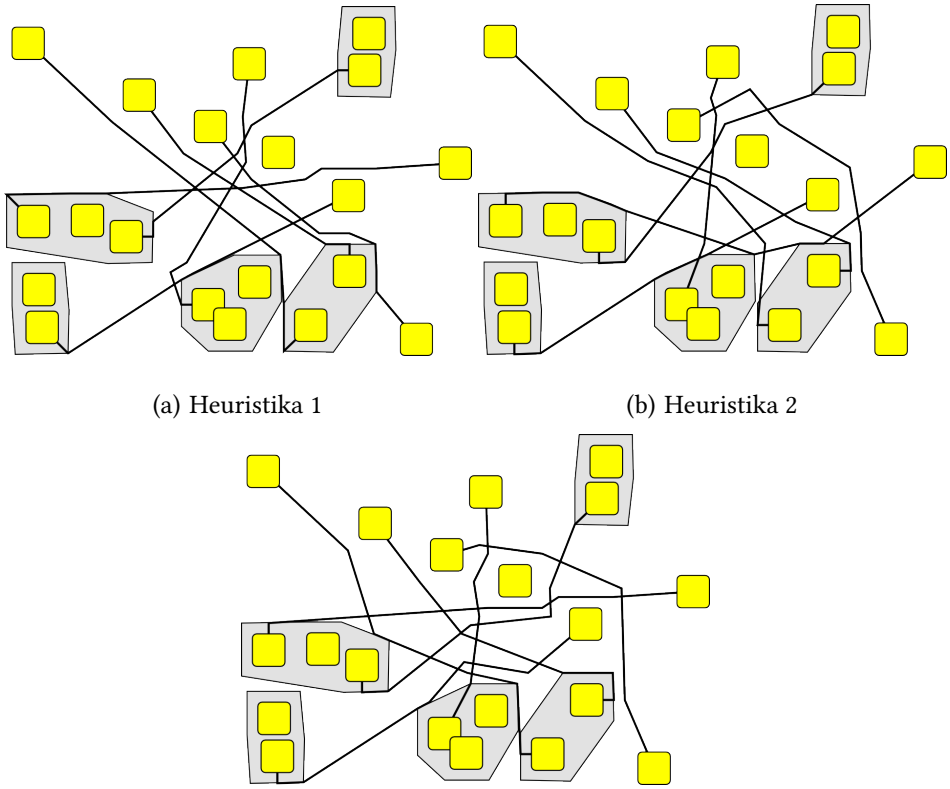
estetických kritérií, či sa dosahujú všetky a v akej miere. Veľmi dôležité je aj nastavenie jednotlivých konštánt v Heuristike 2, nakoľko zmena váhy zodpovednej za pretínanie hrán dokáže v špeciálnych prípadoch vytvoriť planárne vykreslenie grafu.

Výsledné grafy majú podobnú penalizáciu a veľmi rozdielny čas vytvorenia. Keďže je Heuristika 1 omnoho rýchlejšia ako Heuristika 2, môžeme ju použiť na odhad a utriedenie pre Heuristiku 2 na začiatku. Výsledný graf bude potom o niečo krajší a celková penalizácia grafu bude nižšia.

5.5 Rozšíriteľnosť

Program by sa dal rozšíriť na rôzne iné ďalšie estetické kritériá ako napríklad ortogonalita a podobne. Táto úloha nie je jednoduchá, pretože by bolo potrebné hľadať nové obchádzateľné body. Náš problém bol podstatne jednoduchší vďaka tomu, že obchádzateľné body sme ľahko našli.

Okrem toho je možnosť práce na zefektívnení Heuristík a porovnaní so si-



(a) Heuristika 1 (b) Heuristika 2

(c) Spojené heuristiky

Obrázok 5.4: Spojenie Heuristík

lovým vedením. Priestor na prácu je v strome konvexných obalov a lokálnej optimalizácii, ktoré sme nestihli dokončiť.

5.6 Licencovanie

Parser TinyXML je vydaný pod licenciou zlib. A keďže tá je kompatibilná s GPLv3, výsledný editor vydávam pod licenciou GNU General Public License (GPL) version 3 [28].

Kapitola 6

Záver

V práci sme uviedli prehľad rôznych typv, štýlov kreslenia grafov. Základné algoritmy a estetické kritériá, ktoré sú podstatné pre zrozumiteľnosť abstraktných vzťahov v grafe. Podarilo sa nám navrhnúť dve heuristiky na vedenie hrán a veríme, že budú aspoň malým prínosom do odbornej spoločnosti. Cieľ práce je splnený, lebo kombináciou heuristík dosiahneme esteticky priateľné vedenie hrán, avšak implementovaná heuristika 2 nedosahuje požadovanú efektivitu.

Literatúra

[1] Dokulil J., Katreniaková, J.: Edge Routing with Fixed Node Positions. 12th International Conference Information Visualisation, IEEE Computer Society, 2008.

[2] yWorks: yFiles - Java Graph Layout and Visualization Library. Web Page (2011) www.yworks.com/products/yfiles/.

[3] Wybrow M., Marriott K., Stuckey P.J.: Orthogonal connector routing In: GD 2005. Volume 3843 of LNCS., 446-457, Springer, 2009 .

[4] M. Wybrow, K. Marriott, and P.J. Stuckey. Incremental connector routing. In Proceedings of 13th International Symposium on Graph Drawing (GD '05), LNCS 3843, 446—457. Springer-Verlag, 2006.

[5] C. Y. Lee. An Algorithm for Path Connections and its Applications. IRE Transactions on Electronic Computers, EC- 10(3):346–365, September 1961.

[6] J. Soukup. Fast Maze Router. In Proceedings of the 15th Conference on Design Automation, 100–102, 1978.

[7] Hart, P., Nilsson, N., and Raphael, B., A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Trans. Syst. Science and Cybernetics, SSC-4(2):100-107, 1968.

[8] Hart, P., Nilsson, N., and Raphael, B., Correction to 'A Formal Basis for the Heuristic Determination of Minimum-Cost Paths', SIGART Newsletter, no. 37, 28-29, December, 1972.

[9] T. Reinhard, C. Seybold, S. Meier, M. Glinz, and N. Merlo-Schett. Human-Friendly Line Routing for Hierarchical Diagrams. In Proceedings of the 21st IEEE International Conference on Automated Software Engineering (ASE'06), 273–276, 2006.

[10] J-C Latombe, Robot Motion Planning, Kluwer Academic Publishers, Boston, 1991.

[11] T. Lozano-Perez and M. A. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, Communication of ACM, 22, 560-570, 1979.

[12] M. Overmars and E. Welzl, New methods for constructing visibility graphs, in Proc. 4th ACM Symposium on Computational Geometry, 164-171, 1988.

[13] M. R. Garey and D. S. Johnson, Crossing Number is NP-Complete, SIAM J. Algebraic and Discrete Methods, 4:3, 312-316, 1983.

[14] T. Dwyer, K. Marriott and M. Wybrow, Integrating Edge Routing into Force-Directed Layout, Graph Drawing, 8-19, 2006.

[15] Di Battista et al., Algorithms for Drawing Graphs: an Annotated Bibliography, Computational Geometry: Theory and Applications 4: 235–282, 1994.

[16] H. C. Purchase, Metrics for graph drawing aesthetics, Journal of Visual Languages and Computing, 13 (5) , 501-516, 2002.

[17] L. Čech, Kreslenie grafov [Diplomová práca], Univerzita Komenského, Fakulta matematiky, fyziky a informatiky, Katedra informatiky. Školiteľ: Prof. RNDr. Branislav Rovan PhD.,2005.

[18] R. Diestel, Graph Theory, Graduate Texts in Mathematics, Volume 173, Springer-Verlag, Heidelberg, July 2010 (2005, 2000, 1997). ISBN 978-3-642-14278-9

[19] Ch. Ding and P. Mateti,A Framework for the Automated Drawing of Data Structure Diagrams, IEEE Trans. Software Eng., Volume 16:5, 543-557, 1990.

[20] Z. Miller and J.B. Orlin, NP-Completeness for Minimizing Maximum Edge Length in Grid Embeddings, J.Algorithms, vol. 6, 10-16, 1985.

[21] Koren and Yehuda , Drawing graphs by eigenvectors: theory and practice, Computers & Mathematics with Applications 49 (11-12): 1867–1888, 2005.

[22] R. Tamassia and I.G. Tollis, Tessellation Representations of Planar Graphs, Proc. 27th Annual Allerton Conf., 48-57, 1989.

[23] H. Zhang and X. He, Improved visibility representation of plane graphs, Computational Geometry 30(1), 29–39, 2005.

[24] P. Eades et al., Dominance Drawings of Bipartite Graphs, manuscript 1993.

[25] T. M. Chan, Optimal output-sensitive convex hull algorithms in two and three dimensions, Discrete and Computational Geometry, Vol. 16, 361–368, 1996.

[26] R.L. Graham, An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set, Information Processing Letters 1, 132-133, 1972.

[27] R. A. Jarvis, On the identification of the convex hull of a finite set of points in the plane, Information Processing Letters 2: 18–21, 1973.

[28] GPLv3. Text licencie. http://www.gnu.org/licenses/gpl.html 2012

[29] Bednárek D., Dokulil J., Katreniaková J.: Shortest Path Approach to Edge Routing, to be published in Proc. of IV 2013