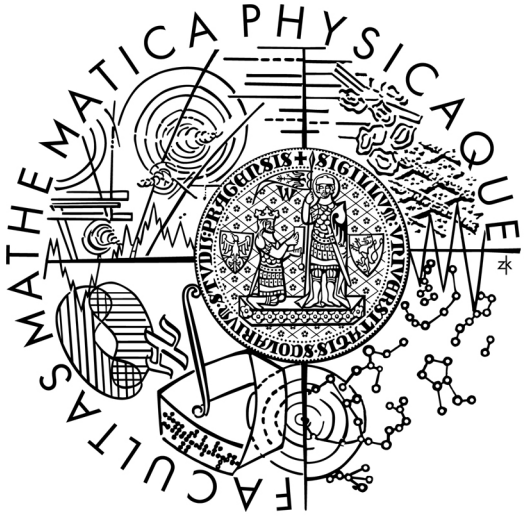
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Jan Macošek

Rovinná nakreslení grafů

Katedra aplikované matematiky

Vedoucí bakalářské práce: RNDr. Ondřej Pangrác, Ph.D.

Studijní program: Informatika, Obecná informatika

2007

Rád bych poděkoval všem, kteří mě jakkoli podpořili při psaní této práce. Zvláštní dík patří především jejímu vedoucímu RNDr. Ondřeji Pangrácovi, Ph.D za pomoc s výběrem tématu, trpělivé vysvětlování nejasností a inspirativní návrhy.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne Jan Macošek

Obsah

Název Práce: Rovinná nakreslení grafů

Autor: Jan Macošek

Katedra (ústav): Katedra aplikované matematiky

Vedoucí bakalářské práce: RNDr. Ondřej Pangrác, Ph.D.

e-mail vedoucího: pangrac@kam.mff.cuni.cz

Abstrakt: Tento text se společně s přiloženým programem soustřeďuje na problematiku rovinných nakreslení grafů: Nejprve poskytuje souhrn základních vědomostí tohoto oboru, poté se zabývá testováním rovinnosti, hledáním rovinného vnoření a následným rovinným nakreslením grafu pomocí úseček Schnyderovým algoritmem. U toho je navíc zkoumán vliv různých strategií potřebné triangulace grafu a také dopad klíčového výběru vnější stěny na výsledné nakreslení z hlediska různých kritérií. Vše je doplněno obrázky jednotlivých nakreslení grafů získanými pomocí přiloženého programu.

Klíčová slova: Rovinná nakreslení grafů, Schnyderův algoritmus

Title: Planar drawings of graphs

Author: Jan Macošek

Department: Department of Applied Mathematics

Supervisor: RNDr. Ondřej Pangrác, Ph.D.

Supervisor’s e-mail adress: pangrac@kam.mff.cuni.cz

Abstract: This text and the enclosed program focuses on the planar drawings of graphs: It first sums up the basics of this domain, then describes the planarity testing, finding of the planar embedding and the straight-line planar drawing of graph using Schnyder´s grid embedding algorithm. It also explores the effect of different triangulation strategies and the impact of the selection of the outer face on the planar drawing in several criteria. It also uses pictures made with the enclosed program to show the planar drawings.

Keywords: Planar drawings of graphs, Schnyder´s algorithm

1 Úvod

Nakreslení jsou pro člověka asi nejpřirozenějším způsobem reprezentace grafů. Bohužel ne všechna nakreslení jsou z vizuálního hlediska přijatelná, a proto jsou v této oblasti neustále vyvíjeny nové a nové postupy, jak vstupní graf zpracovávat. Mnoho z nich se týká právě oblasti rovinných nakreslení grafů, tedy nakreslení, v nichž se žádné dvě hrany vzájemně nekříží.

A právě na rovinná nakreslení grafů se zaměřuje tato práce – v jejím rámci byl vytvořen program, který pro libovolný rovinný graf najde jeho rovinné vnoření, podle něhož následně Schnyderovým algoritmem zkonstruuje rovinné nakreslení pomocí úseček. Program samozřejmě umožňuje také editaci grafu, testování rovinnosti a vykreslení nerovinných grafů alespoň na kružnici.

Tento text pak shrnuje jak celou tématiku rovinných nakreslení grafů, tak konkrétní poznatky získané při implementaci a následném používání programu. Nejprve je tedy nutné definovat několik základních pojmů z oblasti teorie grafů, poté se zaměříme na nakreslení grafů jako taková a s nimi spojená kritéria kvality, podle nichž lze jednotlivá nakreslení objektivně porovnávat. Následuje několik teoretických pojmů a tvrzení týkajících se přímo rovinných nakreslení grafů, z nichž některá byla využita přímo při implementaci, jiná zase sloužila jako teoretický základ použitých algoritmů. Mezi ty patří i algoritmy testování rovinnosti a hledání rovinného vnoření, ať už v podobě seznamu stěn rovinného nakreslení nebo uspořádání hran kolem jednotlivých vrcholů podle směru hodinových ručiček tak, jak mají být vykresleny. Seznamy stěn jsou následně využity pro nalezení velmi jednoduchých rovinných nakreslení lomenými čarami, uspořádání hran podle hodinových ručiček pak přímo vyžaduje Schnyderův algoritmus sloužící k nalezení rovinného nakreslení pomocí úseček. U Schnyderova algoritmu je dále zkoumán vliv možných voleb v jednotlivých krocích jeho implementace, konkrétně výběr jedné ze tří triangulací a volba vhodné vnější stěny v rámci jednoho ze tří vybraných kritérií kvality. Výsledky těchto možných alternativ jsou nadále posuzovány na příkladech nalezených právě pomocí vytvořeného programu, kterému je v závěru věnována samostatná kapitola shrnující veškeré ovládací prvky.

2 Grafy

V této kapitole si připomeneme několik zásadních pojmů z teorie grafů, zejména pak ty, které budeme hojně využívat v kapitolách následujících.

2.1. Grafy a jejich reprezentace

Graf je uspořádaná dvojice (V,E), kde V je neprázdná množina vrcholů a E je množina některých dvojic těchto vrcholů (těmto dvojicím se říká hrany). Z této definice lze odvodit základní způsob reprezentace grafu pomocí výpisu daných množin:

Graf $G_{1.1}$:
$G_{1.1} = (V,E)$

$V = \{0,1,2,3,4\}$

$E = \{\{0,1\}, \{1,2\}, \{2,3\}, \{3,4\}, \{4,0\}, \{0,3\}, \{0,2\}\}$

Samozřejmě ale existují i další způsoby reprezentace grafu lišící se nejen svým principem, ale především oblastí svého využití. Většina z nich je založena na maticovém zápisu – např. *matice sousednosti* pro graf G na *n* vrcholech je definována následovně:

$A_G = (a_{ij})$ $\quad i,j = 1...n$

$a_{ij} = 1$ pro $\{i,j\} \in E$

$a_{ij} = 0$ pro $\{i,j\} \notin E$

Jiným maticovým zápisem je *matice incidence*, která je pro graf G na *n* vrcholech a *m* hranách definována takto:

$G = (V,E)$

$|V| = n$

$E = \{e_1, e_2, ..., e_n,\}$

$I_G = (e_{ij})$ $\quad i = 1...n; j = 1...m$

$e_{ij} = 1$ pro $i \in e_j$

$e_{ij} = 0$ pro $i \notin e_j$

Například pro $G_{1.1}$ by tyto matice vypadaly následovně:

Matice sousednosti:

0 1 1 1 1
1 0 1 0 0
1 1 0 1 0
1 0 1 0 1
1 0 0 1 0

Matice incidence:

1 0 0 0 1 1 1
1 1 0 0 0 0 0
0 1 1 0 0 1 0
0 0 1 1 0 0 1
0 0 0 1 1 0 0

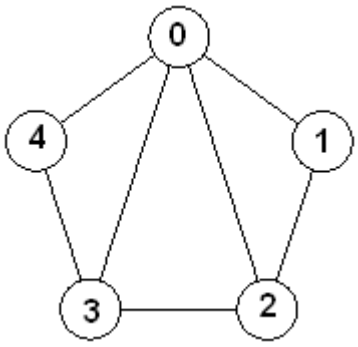
Dalšími maticovými zápisy grafu jsou *matice stupňů* a *Laplaceova matice*, ale pro ilustraci maticového zápisu by dvě výše uvedené možnosti maticového zápisu měly postačit. Místo toho si představíme zcela jiný, nematicový, způsob reprezentace grafu, který se využívá především v informatice, a to reprezentaci grafu *seznamem vrcholů*:

Je-li |V| = n, uspořádáme vrcholy grafu do pole velikosti n a v i-tém prvku tohoto pole bude ukazatel na spojový seznam vrcholů, které s vrcholem *i* sousedí.

Graf $G_{1.1}$ bychom pomocí seznamu vrcholů mohli zapsat například takto:

0: 1, 2, 3, 4
1: 0, 2
2: 0, 1, 3
3: 0, 2, 4
4: 0, 3

Dosud zmíněné způsoby reprezentace jsou vysoce použitelné v případě algoritmického zpracování, ale zároveň disponují pro běžného člověka dosti zásadním nedostatkem: Strukturu grafu z nich totiž lidský mozek dokáže určit jen velmi omezeně a většinou pouze po vynaložení nadměrného úsilí – toto úsilí je naštěstí zbytečné, neboť máme k dispozici ještě jednu možnost reprezentace grafu, a to reprezentaci *nakreslením*. Tomu se budeme věnovat prakticky v celé této práci, takže jej zde nebudu blíže popisovat a pro tuto chvíli raději dám přednost názornému příkladu možného nakreslení grafu $G_{2.1}$:



Obrázek 2.1.1 Graf $\mathbf{G_{2.1}}$ reprezentovaný nakreslením.

Už u takto malého grafu můžeme vytušit, že z nakreslení lze strukturu grafu nahlédnout mnohem rychleji a snadněji než z předchozích zápisů. To samozřejmě platí pouze v případě člověka, strojové zpracování nakreslení by naopak představovalo naprosto neúměrně a hlavně zbytečně složitou alternativu ke zpracování předchozích zápisů grafu. Proto se reprezentace grafů nakreslením nejčastěji využívá v čistě lidské rovině (načrtnutí grafu na přednáškách, zápis grafu na papír, ...) a také jako počítačový výstup čitelný pro člověka. A právě tomu se budeme věnovat na následujících stranách. Nejprve si ale definujme několik užitečných pojmů z oblasti teorie grafů:

2.2. Grafové pojmy

Nejprve si zavedeme několik základních typů grafů, které budeme v následujícím textu hojně potřebovat:

Graf $E_n = (V_n, \emptyset)$ nazveme *prázdným grafem* na n vrcholech.

Graf $K_n = (V_n, \{ \{u,v\} \mid u,v \in V_n, u \neq v \})$ nazveme úplným grafem na n vrcholech.

Graf $K_{m,n} = (V,E)$, kde

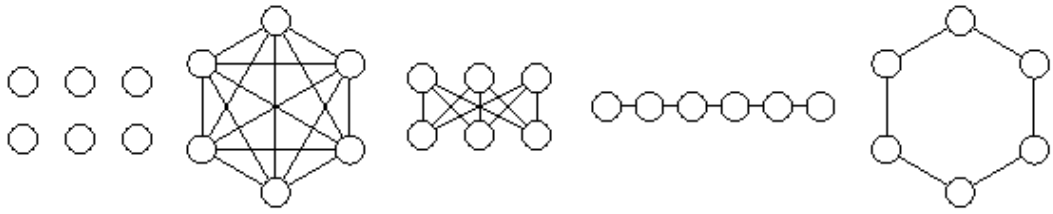
$V = V_a \cup V_b$; $V_a \cap V_b = \emptyset$; $|V_a| = m$, $|V_b| = n$

$E = \{ \{a,b\} \mid a \in V_a; b \in V_b \}$

nazveme *úplným bipartitním grafem*.

Graf $P_n = \{ [n], \{\{i, i+1\} \mid i \in 1...n\} \}$ nazveme *cestou* na n vrcholech.

Graf $C_n = \{ [n], \{\{i, i+1\} \mid i \in 1...n\} \cup \{\{n,1\}\} \}$ nazveme *kružnicí* na n vrcholech.



Obrázek 2.2.1 Příklady uvedených grafů, vždy na 6 vrcholech: E_6, K_6, $K_{3,3}$, P_6, C_6

Dále potřebujeme znát ještě pojem podgrafu, souvislosti a k-souvislosti:

Graf G je *podgrafem* grafu H, pokud $V(G) \subseteq V(H)$ a $E(G) \subseteq E(H)$.
Značíme $G \subseteq H$.

Graf G = (V,E) je *souvislý*, pokud $\forall u,v \in V$ existuje v G cesta z u do v.

Pro $k \in N$, $k \geq 2$ je graf G = (V,E) *(vrcholově) k-souvislý*, pokud:

$|V| \geq k+1$

$\forall U \subseteq V$ takové, že $|U| < k$ je $G \setminus U$ souvislý.

Pro $k \in N$, $k \geq 2$ je graf G = (V,E) *hranově k-souvislý*, pokud:

$|V| \geq k+1$

$\forall F \subseteq E$ takové, že $|F| < k$ je $G \setminus F$ souvislý.

Zde je třeba říci, že pojem hranové k-souvislosti pravděpodobně nebudeme v dalších kapitolách potřebovat – proto pokud uvedeme pojem k-souvislost, bude tím vždy myšlena vrcholová k-souvislost. A pokud přeci jen budeme potřebovat i hranovou k-souvislost, explicitně to uvedeme.

Nyní si ještě zavedeme několik základních operací na grafech:

Operace na grafech

Mějme graf G = (V,E). Potom na něm můžeme provést následující operace:

Odebrání vrcholu: $v \in V$ $\quad G \setminus v = (V \setminus \{v\}, E \setminus \{\{u_1,u_2\} \in E \mid u_1 = v \vee u_2 = v \})$

Odebrání hrany: $e \in E$ $\quad G \setminus e = (V, E \setminus \{e\})$

Přidání hrany: $e \notin E$ $G+e = (V, E \cup \{e\})$

Kontrakce hrany:

$\{u,v\} = e \in E$

$G.e = (V', E')$, kde

$V' = (V \setminus \{u,v\}) \cup \{w\}$

$E' = \{e \in E; e \cap \{u,v\} = \emptyset\} \cup \{\{w,x\}; x \in V \setminus \{u,v\}, \{w,u\} \in E$ nebo $\{w,v\} \in E\}$

Dělení hrany: $\{u,v\} = e \in E$ $G \div e = (V \cup \{w\}, E \setminus e \cup \{\{u,w\},\{v,w\}\})$

Zde se navíc hodí definovat rovnou dělení celého grafu:

Dělení grafu G je graf vzniklý z G opakovaným dělením hrany.

V teorii grafů samozřejmě existuje i mnoho dalších důležitých as užitečných pojmů, ale pro naše účely by měly postačit ty výše zmíněné. V dalším textu pak obvykle následují definice pojmů vyžadovaných přímo konkrétní problematikou, proto je uvádíme na vhodnějších místech.

3 Nakreslení grafů

Tato kapitola shrnuje základní pojmy z oblasti vizuální reprezentace grafu: Nejprve se podíváme na obecnou problematiku nakreslení grafů a následně se seznámíme s nakresleními rovinnými. Pro definici většiny pojmů byly využity *Kapitoly z diskrétní matematiky* [2], v nichž lze rovněž nalézt důkazy uvedených vět a tvrzení.

3.1. Obecně

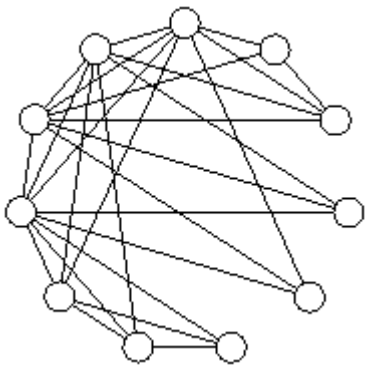
Jak již bylo naznačeno v úvodní části předchozí kapitoly, nakreslení představuje pro člověka asi nejpochopitelnější způsob reprezentace grafu. A přitom stačí na první pohled tak málo: Vrcholům grafu přiřadíme navzájem různé body roviny (případně nějakého tělesa, ale pro účely této práce zůstaneme pouze u roviny), označíme je nějakou vybranou značkou (kolečko, čtvereček, trojúhelníček,...) a vždy dva vrcholy spojíme přímou čarou nebo křivkou v případě, že mezi nimi je hrana. Tato „definice“ nakreslení grafu ovšem může působit dosti neformálně, a tak raději uvedeme definici správnou - začneme pojmem oblouku:

Oblouk je podmnožina roviny tvaru $o = \gamma([0,1]) = \{\gamma(x) \mid x \in [0,1]\}$, kde $\gamma : [0,1] \rightarrow \mathbf{R}^2$ je nějaké prosté spojité zobrazení uzavřeného intervalu [0,1] do roviny. Bodům $\gamma(0)$ a $\gamma(1)$ říkáme koncové body oblouku *o*.

Nyní nám již nic nebrání ve formálním popisu nakreslení grafu:

Nakreslením grafu G = (V,E) rozumíme přiřazení, které každému vrcholu *v* grafu V přiřazuje bod *b*(*v*) roviny, a každé hraně e = {*u*,*v*} přiřazuje oblouk *o*(*e*) v rovině s koncovými body *b*(*u*) a *b*(*v*). Přitom předpokládáme, že zobrazení *b* je prosté (různým vrcholům odpovídají různé body), a žádný z bodů tvaru *b*(*v*) není nekoncovým bodem žádného z oblouků *o*(*e*). Graf spolu s nějakým nakreslením nazýváme *topologický graf*.

I formální definice nám ale popisuje ten samý postup: Vrcholy se vykreslí do roviny a následně spojí oblouky. Až tak jednoduché to ale nebude, tedy pokud chceme získat vizuálně uspokojivé nakreslení. Proto se při kreslení grafů využívá všemožných strategií rozvržení vrcholů do roviny i jejich následného spojení hranami tak, aby výsledek vypadal co možná nejlépe. Mezi základní strategie patří vykreslení vrcholů na kružnici a jejich následné spojení úsečkami tak, jak je to možné vidět v přiloženém programu. Tato strategie vyniká především jednoduchou implementací, ovšem kvalita výsledku zejména při větším počtu hran dosti pokulhává – například už u grafu na obrázku 3.1.1 není úplně jednoduché rozeznat jeho strukturu, přestože má pouze 11 vrcholů (k „rozumnějšímu“ vykreslení tohoto grafu se dostaneme později).



Obrázek 3.1.1 Graf $G_{3.1}$ na 11 vrcholech vykreslený na kružnici.

V předchozím odstavci jsem uvedl, že se při vykreslování grafů klade důraz především na to, aby graf vypadal co nejlépe. Ovšem otázkou je, co přesně si představit pod slovem nejlépe. Pokud totiž náhodně zvolenému lidskému jedinci předložíme dvě různá nakreslení stejného grafu, velmi často nám dokáže říci, které z nich považuje za hezčí. Ovšem pravděpodobně nám nedokáže odpovědět na otázku, na jakém základě si vybral právě toto nakreslení a proč nedal přednost raději tomu druhému. A i kdyby nám dokázal dát rozumnou a fakty podloženou odpověď, u dalšího náhodně zvoleného jedince bychom se mohli setkat se zcela jiným názorem. Proto je poměrně obtížné formálně popsat kvalitu nakreslení. I přesto ale lze při posuzování jednotlivých nakreslení vzít v úvahu mnoho různých kritérií:

- Vizuální komplexnost – aneb za jak dlouho z nakreslení rozpoznáme nejdůležitější prvky struktury grafu
- Pravidelnost – jakým způsobem jsou kresleny opakující se prvky
- Symetrie – jsou-li možné symetrie skutečně vykresleny symetricky
- Tvar, velikost a úměrnost (vzhledem k nějakému reálnému podkladu)
- Podobnost vykreslování používaném v daném oboru (ER-diagramy, struktura chemických sloučenin,...)
- A další

Pokud se pak tyto poměrně obecná kritéria pokusíme nějakým způsobem formalizovat, dostáváme mnoho různých matematicky zapsatelných a porovnatelných vlastností, která u nakreslení můžeme posuzovat:

- Symetrie – geometrická (osová, středová, otočení), grafová, ...
- Geometrická kritéria – délka hran, rozložení úhlů mezi nimi, kreslení do mřížky, ...
- Diskrétní kritéria – počet lomů a křížení hran, ...

A právě minimalizováním, nebo spíš naprostým anulováním počtu křížení hran se zabývá speciální oblast kreslení grafů, tedy rovinná nakreslení. Ale to už je námět pro další podkapitolu.

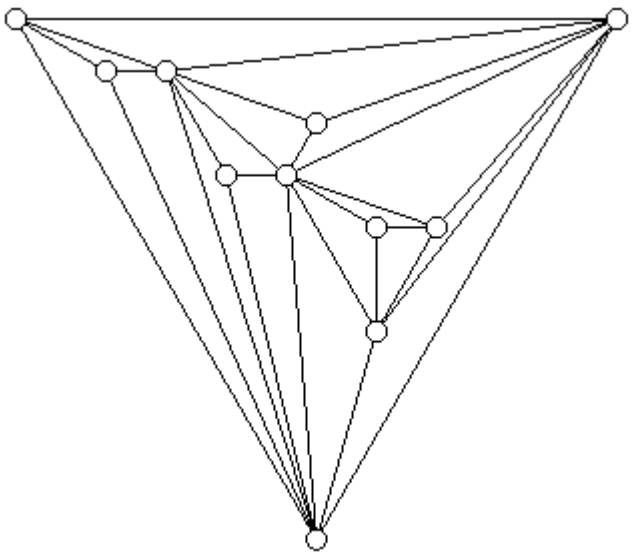
3.2. Rovinná nakreslení

Definici rovinného nakreslení grafu lze získat poměrně jednoduchým rozšířením definice nakreslení grafu z předchozí kapitoly:

Nakreslení grafu G = (V,E), v němž oblouky odpovídající různým hranám mají společné nanejvýš koncové body, se nazývá *rovinné nakreslení*. Graf G je *rovinný*, má-li aspoň jedno rovinné nakreslení.

Rovinné nakreslení grafu je tedy takové nakreslení, v němž se žádné dvě hrany neprotínají jinde než ve vrcholech grafu, čímž dochází k optimalizaci jednoho z dříve zmíněných kritérií kvality grafových nakreslení. Toho se využívá nejen pro co nejlepší vizuální reprezentaci grafu (v nerovinném nakreslení může křížení hran vést ke zmatení cílové osoby – např. nemusí být vždy jasné, kudy hrana z daného překřížení pokračuje; při velké hustotě křížení hran na malém prostoru pak může dojít i k úplnému zkreslení struktury grafu), ale také pro důvody žádané praxí v různých oborech lidské činnosti (například při návrhu jednovrstevných integrovaných obvodů je zapotřebí křížení oblouků zcela vyloučit).

Jako příklad rovinného nakreslení si předvedeme rovinné nakreslení grafu $\mathbf{G_{3.1}}$ z Obrázku 3.1.1:



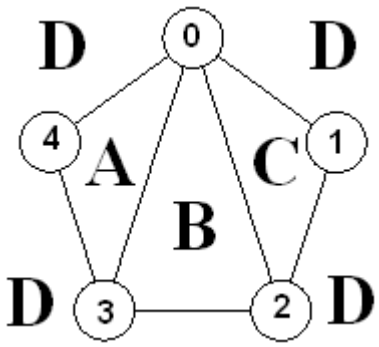
Obrázek 3.2.1 Rovinné nakreslení grafu $G_{3.1}$.

Troufám si tvrdit, že toto nakreslení působí esteticky mnohem lépe než nakreslení na Obrázku 3.1.1. Ovšem nejde jen o estetiku – kromě lehce sporného výsledku v oblasti rovnoměrnějšího rozložení vrcholů (kde toto rovinné nakreslení sice dosahuje lepšího výsledku, ale zase ztrácí na pravidelnosti tohoto rozložení) totiž kvalitu nakreslení na Obrázku 3.2.1 můžeme podložit jasným faktem v podobě počtu křížení hran. K tomu zde totiž nedochází, což oproti Obrázku 3.1.1 představuje skutečně výrazné zlepšení. A právě díky absenci křížení hran se v rovinném nakreslení grafu velmi dobře orientuje. Na druhou stranu obvykle není triviální rovinné nakreslení grafu najít, a proto byly pro tento účel vyvinuty speciální algoritmy. Těm se ovšem budeme věnovat až v následujících kapitolách. Ještě

předtím si ale shrneme několik poznatků o rovinných grafech – a začneme definicí stěny:

Stěnami rovinného nakreslení grafu G rozumíme maximální souvislé oblasti $\mathbf{R}^2 \backslash X$, kde X jsou všechny body daného rovinného nakreslení (tedy sjednocení všech oblouků).

Např. na následujícím obrázku jsme označili písmeny A,B,C,D jednotlivé stěny grafu $G_{2.1}$ (čtyřmi písmeny D je označena jedna speciální, tzv. vnější, stěna):



Obrázek 3.2.2 Vyznačení stěn grafu $\mathbf{G_{2.1}}$

A právě o stěnách, resp. jejich struktuře v 2-souvislých grafech, pojednává následující tvrzení:

Tvrzení 3.2.1
Nechť G je 2-souvislý rovinný graf. Potom hranice každé stěny v libovolném rovinném nakreslení grafu G je kružnice v grafu G.

To samozřejmě lze využít při hledání rovinného nakreslení, kdy se například ze souvislého grafu přidáním hran vytvoří 2-souvislý proto, aby se později s jeho stěnami lépe pracovalo.

Důležitou informaci o rovinných nakresleních poskytuje také základní kvantitativní vztah pro rovinné grafy, Eulerův vzorec:

Tvrzení 3.2.2 (Eulerův vzorec)
Nechť G = (V,E) je souvislý rovinný graf a s je počet stěn nějakého rovinného nakreslení G.
Pak platí:
$|V|-|E|+s = 2$

Z toho plyne, že počet stěn nezávisí na způsobu rovinného nakreslení grafu G, ale na počtu jeho vrcholů a hran. Obecně tedy lze nalézt prakticky libovolný počet rovinných nakreslení jednoho rovinného grafu, ale počet stěn bude u každého z těchto nakreslení stejný.

Následující tvrzení už sice přímo nepoužívá stěny grafu, ale zase nám dovoluje vyloučit rovinnost některých grafů už na základě jednoduchého součtu hran a

vrcholů, čímž se často můžeme vyhnout zbytečné aplikaci rovinnost testujících algoritmů:

Tvrzení 3.2.3
Nechť G = (V,E) je rovinný graf, |V|>3.
Pak platí:
$|E| \leq 3|V|-6$

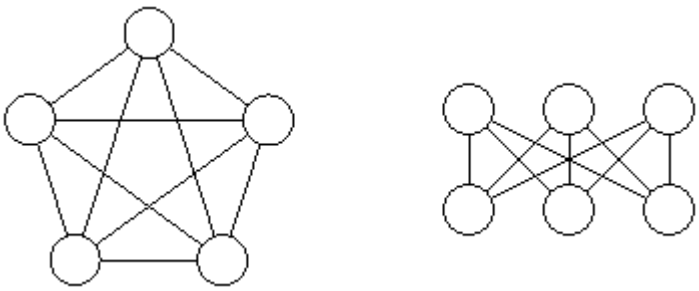
Pokud navíc G neobsahuje K_3 jako podgraf, platí dokonce:

$$E| \leq 2|V|-4$$

Zejména u první části tohoto tvrzení je dobré si povšimnout, že nastane-li rovnost, nelze už do grafu přidat žádnou hranu bez porušení rovinnosti. V takovém případě se jedná o maximální rovinný graf, jehož všechny stěny jsou trojúhelníky, tedy kružnice na třech vrcholech. Tohoto faktu využívá mnoho algoritmů rovinného kreslení grafů (včetně námi používaného Schnyderova algoritmu, k němuž se ale blíže dostaneme v kapitole 4).

Tvrzení 3.2.3 nám ukázalo zápornou stránku rovinných nakreslení, tedy fakt, že ne každý graf lze rovinně nakreslit. Nyní si ukážeme dva základní příklady nerovinných grafů:

Tvrzení 3.2.4
K_5 a $K_{3,3}$ nejsou rovinné grafy.



Obrázek 3.2.3 K_5 a $K_{3,3}$

Ano, tyto dva grafy ani člověk, ani sebelepší algoritmus nedokáží nakreslit rovinně (s nakreslením třeba na torus nebo Möbiův list už je to lepší, ale tím se zde bohužel zabývat nebudeme) a přitom jde o grafy poměrně jednoduché. Ovšem tím tato stinná stránka rovinného kreslení zdaleka nekončí – nakreslit totiž nejde ani grafy, které obsahují K_5 nebo $K_{3,3}$ jako podgraf. A platí dokonce silnější tvrzení, které shrnuje pro rovinné kreslení grafů zcela zásadní Kuratowského věta:

Věta 3.2.5 (Kuratowského)

Graf G je rovinný, právě když G neobsahuje dělení K_5 ani $K_{3,3}$ jako podgraf.

Z toho plyne, že velké množství grafů nebude rovinné, a tudíž pro ně ani nepůjde nalézt rovinné nakreslení. Nicméně metoda testování rovinnosti grafu založená na hledání dělení K_5 nebo $K_{3,3}$ jako podgrafu by sice jistě šla použít, ovšem její časová

složitost by přesahovala únosnou míru. Proto se pro testování rovinnosti používají speciální algoritmy založené přímo na hledání rovinného nakreslení, které obvykle kromě otestování samotné rovinnosti poskytnou také základ pro samotné rovinné nakreslení. Ale o tom si více povíme v další kapitole.

Na závěr této kapitoly ještě poznamenejme, že kromě rozdělení samotných nakreslení na rovinná a nerovinná můžeme rozdělit i samotná rovinná nakreslení, a to zejména podle způsobu vykreslování hran – podle něj lze jednotlivá nakreslení (a také algoritmy pro jejich hledání) rozdělit na tři základní skupiny:

1) Lomenými čarami – jednotlivé oblouky nakreslení grafu se skládají vždy z konečného počtu úseček.
2) Beziérovými křivkami - oblouky nakreslení grafu jsou tvořeny Beziérovými křivkami.
3) Úsečkami – každý oblouk je vždy vykreslen jako úsečka.

Zde je potřeba si uvědomit, že toto rozdělení platí právě pro rovinná nakreslení, nikoli přímo pro rovinné grafy. Pro ty totiž obecně platí následující tvrzení:

Tvrzení 3.2.6
Nechť G je rovinný graf. Potom existuje rovinné nakreslení grafu G pomocí úseček.

Libovolný lomený graf lze tedy nakreslit jak lomenými čarami, tak Beziérovými křivkami či úsečkami. A záleží pak především na účelu samotného nakreslení, jaký typ zvolíme – např.z úsečkového nakreslení se díky eliminaci lomů hran dá lépe vyčíst struktura grafu, ale zase často může zabrat větší prostor než nakreslení lomenými čarami. Beziérovy křivky pak poskytují „pěkné“ nakreslení jakoby bez lomů, ale zase dosahují podstatně vyšší časové složitosti než předchozí způsoby nakreslení. I proto se jim v této práci nevěnujeme.

4 Testování rovinnosti a nalezení stěn

V této kapitole se zaměříme na algoritmy testování rovinnosti a s ním spojeného nalezení stěn grafu. Velká část poznatků zde uvedených pochází z Kučery [1].

4.1. Myšlenka

Již u Kuratowského věty (3.2.5) jsme naznačili, že testování rovinnosti hledáním dělení K_5 nebo $K_{3,3}$ v testovaném grafu by nebylo zrovna rozumnou volbou: Zaprvé bychom se nevyhnuli zbytečné časové složitosti, zadruhé bychom tím dostali pouze odpověď na test rovinnosti a samotné rovinné nakreslení bychom v případě odpovědi kladné museli hledat dodatečně. Proto je vhodné využít speciálních algoritmů pracujících nejhůře v kvadratickém čase, které nám kromě odpovědi rovinný/nerovinný poskytnou také základ pro rovinné nakreslení v podobě seznamu jednotlivých stěn nebo seřazení hran u každého vrcholu podle směru hodinových ručiček tak, jak mají být rovinně vykresleny. Obvykle totiž testují rovinnost konstruktivně tak, že se pokouší rovnou hledat základ pro rovinné nakreslení (tento základ budeme nadále nazývat *rovinné vnoření*, jde buď o seznam stěn nebo seřazení hran v jednotlivých vrcholech podle směru hodinových ručiček) – pokud se jim to povede, je součástí výsledků právě základ rovinného nakreslení, pokud ne, graf není rovinný a na základě dosud nalezeného rovinného podgrafu a poslední provedené části algoritmu se v něm může hledat dělení K_5 nebo $K_{3,3}$.

Algoritmů podobným způsobem testujících rovinnost existuje poměrně dost, často se však jedná pouze o různé verze podobného postupu. Všechny strategie se samozřejmě shodují v počátečním využití Tvrzení 3.2.3 o maximálním počtu hran v rovinném grafu, čímž dokáží okamžitě eliminovat velkou část nerovinných grafů. Následně je celý graf rozdělen do komponent souvislosti (případně rovnou 2-souvislosti), na které se aplikuje konkrétní algoritmus testování rovinnosti a hledání rovinného vnoření. Asi nejznámější a zároveň historicky první pracující v lineárním čase je Hopcroft-Tarjanův algoritmus založený na původní myšlence L. Auslandera, S.V. Partera a A.J. Goldsteina, který pracuje na poměrně jednoduchém principu, k jehož popsání ovšem potřebujeme zavést pojem části a chordy v grafu:

Definice 4.1.1
Nechť G je graf a H jeho podgraf.
Nechť G\H je graf vzniklý z G vynecháním vrcholů grafu H a hran s nimi incidentních.
Chorda v G vzhledem k H je cesta v grafu G, jejíž koncové vrcholy patří do H a žádný jiný vrchol ani žádná hrana nepatří do H.
Částí grafu G vzhledem k H nazveme buď hranu, která do H nepatří, ale jejíž konce jsou vrcholy grafu H, nebo komponentu I grafu G\H, ke které jsou přidány všechny hrany mající jeden koncový vrchol v I a jeden v H a koncové vrcholy těchto hran ležící v H.

Hopcroft-Tarjanův algoritmus je pak pro nějaký 2-souvislý graf G založený na následujícím postupu:

1) Nalezení kružnice K v G a její zakreslení do roviny (čímž získáme vnější a vnitřní stěnu nakreslení K)

2) Určení všech částí v grafu G vzhledem ke kružnici K. Pro každou takto nalezenou část C pak provedeme následující kroky:

    a. Rekurzivním použitím algoritmu zjistíme, zda je spojení C a K rovinný graf. Pokud ne, tak je podle Kuratowského věty (3.2.5) i celý G nerovinný, protože podgraf G obsahuje dělení K_5 nebo $K_{3,3}$ jako podgraf, a tak i G nutně obsahuje dělení K_5 nebo $K_{3,3}$. Jinak postoupíme k dalšímu kroku.

    b. Pomocí nakreslení nalezeného v kroku 2a se pokusíme přidat C do celkového nakreslení K a již dříve nakreslených částí. To buď jde okamžitě, nebo po převrácení některých dříve vykreslených částí oproti kružnici K tak, aby se dostali buď z vnější stěny K do vnitřní nebo naopak. V takových případech je dosavadní graf rovinný a pokračuje se další částí. Ovšem může nastat i situace, kdy nelze převrátit již zakreslené části tak, aby do již nalezeného nakreslení šlo rovinně přidat C. V takovém případě graf G není rovinný a výpočet končí.

3) Podařilo-li se rovinně zakreslit všechny části, je graf rovinný a máme i jeho rovinné vnoření.

Tento popis Hopcroft-Tarjanova algoritmu je poměrně hrubý, bližší informace lze ale nalézt
v Kučerovi [1]. Více jej tu nerozvádím proto, že v softwarovém projektu jsem využil sice podobného, ale přeci jen odlišného postupu, který popsali G. Demoucron, Y. Malgrange a R. Peruiset a jehož popis lze rovněž nalézt v [1]. Vzhledem k jeho využití v projektu si ho ale dovolím probrat v samostatné kapitole.

4.2. Demoucron-Malgrange-Pertuisetův algoritmus

Tento algoritmus pracuje na podobném principu jako algoritmus Hopcroft-Tarjanův, jen místo přidávání celých částí vždy přidává pouze chordy. Konkrétní odlišnosti ale budou vidět na následujícím popisu algoritmu, který jsem (oproti popisu v Kučerovi [1]) upravil o vlastní kroky sloužící k usnadnění následujícího rovinného nakreslení:

Algoritmus 4.2.1 (Demoucron-Malgrange-Pertuisetův)
Na vstupu dostáváme souvislý graf G, na nějž aplikujeme následující kroky:

1) Nalezení kružnice: V G určíme kružnici, označíme ji H a vytvoříme podle ní první dvě stěny S_1 a S_2 ($S_1 = S_2$, ale S_1 označíme jako vnější), které přidáme do Seznamu stěn. Pokud kružnici nenalezneme, samotný algoritmus končí, graf G je rovinný a následně se vykreslí jako strom.

2) Test ukončení: Je-li G = H, výpočet končí a G je rovinný graf. Nalezené stěny pak můžeme použít pro konstrukci rovinného nakreslení.
3) Určení stěn pro zakreslení částí: Nechť C_1, ..., C_m jsou části grafu G vzhledem k H a S_1, ..., S_k jsou dosud nalezené stěny grafu H. Část C_i je možné umístit do stěny S_j, pokud tato stěna obsahuje všechny výstupní vrcholy C_i (tedy všechny vrcholy C_i s alespoň jedním sousedem mimo C_i). Nastává jedna z následujících možností:
    a. Důkaz nerovinnosti: Některou část C_i nelze umístit do žádné stěny. Pak graf není rovinný a výpočet končí.
    b. Jednoznačně umístitelná část: Existuje část C_i, kterou je možné umístit jen do jediné stěny S_j. Položíme $C = C_i$, $S = S_j$ a pokračujeme krokem 4.
    c. Vícenásobně umístitelné části: Každou část C_i lze umístit do alespoň dvou různých stěn. Vybereme libovolnou část C a za S zvolíme jednu ze stěn, do níž je možné právě zvolenou část C umístit. Přejdeme do bodu 4.
4) Přidání chordy: Vybereme chordu P v grafu G vzhledem k H, která je obsažena v části C. Přidáme tuto chordu do grafu H, do seznamu stěn vložíme obě stěny vzniklé rozdělením původní stěny S chordou P a stěnu S ze seznamu smažeme. Přejdeme do bodu 2.

Časová složitost tohoto algoritmu se oproti algoritmu Hopcroft-Tarjanovu zhoršila z lineární na kvadratickou, což ovšem díky předpokládanému nízkému počtu vrcholů zase tolik nevadí – spíš naopak: Díky o několik řádů nižší konstantě se Demoucron-Malgrange-Pertuiset u grafů s nízkým počtem vrcholů ukazuje jako velmi užitečný algoritmus. Navíc s ním získáváme snadněji (nikoli snadno) implementovatelný postup, jehož lehkou úpravou lze místo seznamu stěn rovinného nakreslení zadaného grafu získat rovnou určitý postup, jak toto nakreslení pomocí lomených čar realizovat – stačí si v datové struktuře stěny zavést ukazatel UzavíracíChorda na nějakou chordu, zavést kromě seznamu stěn také seznam těchto chord a následujícím způsobem upravit algoritmus 4.2.1:

Algoritmus 4.2.2 (Demoucron-Malgrange-Pertuisetův s postupným seznamem chord)
Na vstupu dostáváme souvislý graf G, na nějž aplikujeme následující kroky:

1) Nalezení kružnice: V G určíme kružnici, označíme ji H a vytvoříme podle ní první dvě stěny S_1 a S_2 ($S_1 = S_2$, ale S1 označíme jako vnější), které přidáme do Seznamu stěn. Oběma těmto stěnám nastavíme jako uzavírací chordu celou kružnici. Pokud kružnici nenalezneme, samotný algoritmus končí, graf G je rovinný a následně se vykreslí jako strom.
2) Stejný jako v 4.2.1
3) Stejný jako v 4.2.1
4) Přidání chordy: Vybereme chordu P v grafu G vzhledem k H, která je obsažena v části C. Přidáme tuto chordu do grafu H a do seznamu stěn vložíme obě stěny vzniklé rozdělením stěny S chordou P (označme je S_a a S_b). Navíc je třeba upravit seznam chord a také nově vzniklým stěnám přiřadit uzavírací chordy – zde může nastat několik případů v závislosti na průniku chordy P s uzavírací chordou původní stěny S (označme ji P_S):

a. P a P_S nemají společné vrcholy nebo jsou tyto vrcholy na okrajích obou chord: Potom je celá chorda P_S podgrafem právě jedné z S_a a S_b. Nechť je tedy P_S podgrafem S_a. Stěně S_a přiřadíme jako uzavírací chordu P_S, stěně S_b chordu P. Chordu P navíc zařadíme do seznamu chord těsně před P_S.

b. P a P_S mají právě jeden společný vrchol *v*, který ovšem není krajním vrcholem chordy P_S (a tedy je krajním vrcholem chordy P): Chordu P_S rozdělíme ve vrcholu *v* na dvě cesty, označme je P_1 a P_2. Položme P rovno grafu vzniklému spojením P_1 a P ve vrcholu *v*. Stěnám S_a a S_b přiřaďme jako uzavírací chordu tu z dvojice P_2 a P, která je celá podgrafem dané stěny. Ze seznamu chord smažeme P_S a na její místo vložíme nově vzniklé chordy v pořadí P, P_2.

c. P a P_S mají dva společné vrcholy *u* a *v*, které jsou krajními vrcholy chordy P: Chordu P_S rozdělíme ve vrcholech *u* a *v* na tři cesty, označme je P_1, P_2 a P_3. Spojíme P s P_1 v bodě u a s P_2 v bodě *v*, vzniklou cestu dosaďme do P. Stěnám S_a a S_b přiřaďme jako uzavírací chordu tu z dvojice P_2 a P, která je celá podgrafem dané stěny. Ze seznamu chord smažeme P_S a na její místo vložíme nově vzniklé chordy v pořadí P, P_2.

Po upravení seznamu chord smažeme stěnu S ze seznamu stěn a přejdeme do bodu 2.

Tento algoritmus nám pro každý 2-souvislý rovinný graf vygeneruje seznam chord v pořadí, jak je lze postupně přidávat do vnější stěny rovinného nakreslení tak, aby rovinnost nebyla porušena. V případě ne2-souvislého (ovšem stále alespoň souvislého) grafu je třeba lehce upravit krok 3 algoritmu 4.2.2 v případě, že vybraná část C_i má pouze jeden výstupní vrchol *v* (byť by do něj vedlo libovolně mnoho hran). V takovém případě se celý algoritmus rekurzivně zavolá na C_i , čímž se převede na 2-souvislý a najde se jeho vnější stěna. Následně se do celého grafu G přidá hrana vedoucí z libovolného vrcholu stěny S odlišného od v do libovolného vrcholu vnější stěny části C_i odlišného od v a pokračuje se dalšími kroky algoritmu. Po nalezení rovinného nakreslení se pak přidané hrany z grafu i z nakreslení smažou.

4.3. Aplikace Demoucron-Malgrange-Pertuiseta

Nyní si ukážeme, jak lze algoritmus 4.2.1 (resp. 4.2.2) využít k nalezení základního rovinného nakreslení. Následně také uvedeme, jakým způsobem lze výsledky tohoto algoritmu převést na uspořádání sousedů podle hodinových ručiček, které vyžaduje většina algoritmů hledajících rovinné nakreslení pomocí úseček (včetně námi použitého algoritmu Schnyderova, popsaného v kapitole 5).

Základní rovinné nakreslení grafu lomenými čarami lze nalézt pomocí seznamu chord a seznamu stěn generovaných algoritmem 4.2.2 – stačí na tyto seznamy aplikovat následující postup:

Algoritmus 4.3.1 (rovinné nakreslení lomenými čarami)

Vstup: Graf G, seznam jeho chord a seznam jeho stěn vygenerované algoritmem 4.2.2.

1) Každé chordě přidělíme stěnu, kterou uzavírá (Tu vždy musí mít, protože přidáním chordy do grafu se vždy rozpadne jedna stěna na dvě).

2) První chordu vykreslíme jako kružnici do roviny. Střed této kružnice označíme *s*. Zároveň tuto kružnici vezmeme jako počáteční vnější stěnu rovinného nakreslení.

3) Pro každou chordu P ze seznamu chord aplikujeme následující postup:

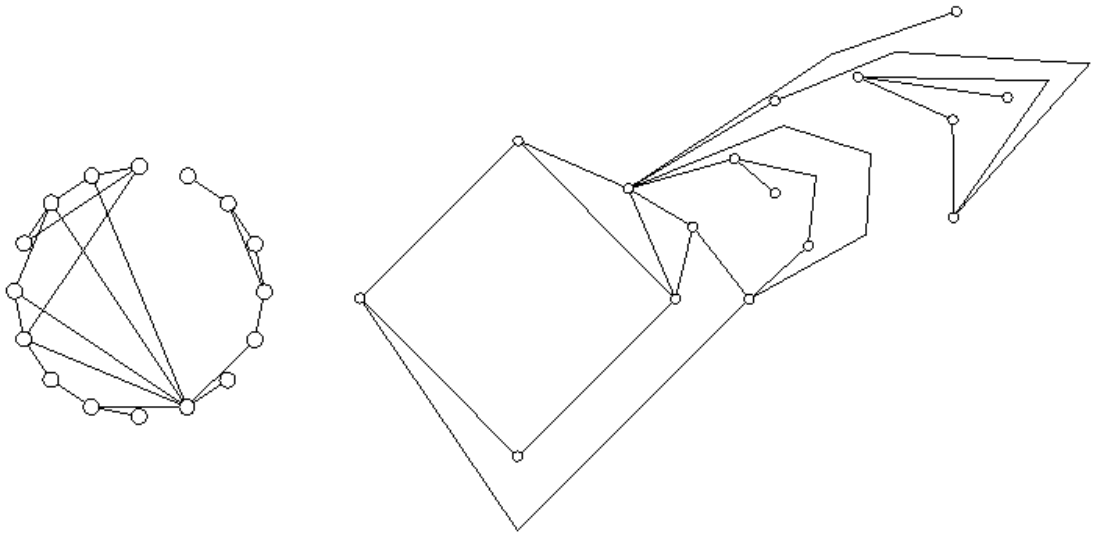
    a. Vezmeme stěnu S, kterou chorda P uzavírá a najdeme doplněk chordy P v S(označme jej D), tedy graf vzniklý z S odebráním všech vrcholů i hran v P. V D ovšem ponecháme krajní vrcholy P. D je vždy podgrafem vnější stěny dosud nalezeného rovinného nakreslení – známe tedy rovinné souřadnice jeho vrcholů.

    b. Nyní do P nebo do D umístíme pomocné vrcholy tak, aby byl počet vrcholů
    v P a D stejný. Toho dosáhneme například dělením hran v té části, která má méně vrcholů (Je ovšem záhodno ošetřit rovnoměrné rozdělení přidaných vrcholů mezi vrcholy původní).

    c. Nyní můžeme každému vrcholu v P jednoznačně určit jeho vzor v D: Začneme stejným krajním vrcholem v P i D a po obou těchto cestách zároveň postupujeme až do druhého krajního vrcholu. Přitom vždy aktuálnímu vrcholu v_P z P nastavíme souřadnice v rovině jako součet souřadnic jeho vzoru v_D z D a nějakého násobku vektoru $v_D - s$.

    d. Nyní již můžeme vykreslit vrcholy i hrany chordy P. Zároveň upravíme vnější stěnu dosavadního rovinného nakreslení – jednoduše v ní D nahradíme chordou P.

4) Z G (nikoli z jeho právě nalezeného nakreslení) smažeme všechny vrcholy přidané
    v kroku 3b a do grafu vrátíme všechny původní hrany, které byly odebrané dělením. Pokud navíc graf G nebyl 2-souvislý (a my jsme tak museli přidávat hrany k dosažení 2-souvislosti), smažeme také všechny hrany přidané upraveným krokem 3 algoritmu 4.2.2. Nalezli jsme rovinné nakreslení G lomenými čarami.



Obrázek 4.3.1 Nakreslení téhož grafu na kružnici a rovinně lomenými čarami algoritmem 4.2.3.

Nutno ovšem poznamenat, že tento algoritmus ani zdaleka nedosahuje optimálního počtu lomů hran (tato optimalita samozřejmě záleží i na jiném kritériu, protože podle tvrzení 3.2.6 je optimální počet lomů hran 0) a v některých případech se v něm dokonce orientuje hůř než kdybychom stejný graf jednoduše nerovinně vykreslili na kružnici. I tak ale lze algoritmus 4.3.1 využít k vizuálnímu důkazu rovinnosti grafu a k určité demonstraci fungování algoritmu 4.2.2 – a přesně k tomuto účelu jej využívá náš softwarový projekt, v němž k nalezení pořádného rovinného nakreslení používáme Schnyderův algoritmus. K jeho fungování ovšem nepotřebujeme seznamy stěn ani chord, ale uspořádání seznamu sousedů v každém vrcholu podle hodinových ručiček tak, jak budou kolem daného vrcholu rovinným nakreslením vykresleny. Proto se na závěr celé kapitoly pojďme podívat, jak lze toto uspořádání z dosud získaných seznamů získat.

Můžeme postupovat hned dvěma způsoby. V prvním z nich přidáme vrcholům v grafu speciální seznam sousedů podle ručiček a poté upravíme krok 4 algoritmu 4.2.1 následujícím způsobem:

4) Přidání chordy: Vybereme chordu P v grafu G vzhledem k H, která je obsažena v části C. Přidáme tuto chordu do grafu H a do seznamu stěn vložíme obě stěny vzniklé rozdělením původní stěny S chordou P. *Oběma krajním vrcholům chordy P navíc vložíme jejich souseda v P do seznamu sousedů podle ručiček přímo mezi sousedy v původní stěně S, kterou následně smažeme ze seznamu stěn.* Přejdeme do bodu 2.

Navíc je ale nutné ošetřit možnost přidání souseda *u* vrcholu *v*, který má v seznamu sousedů podle ručiček zatím pouze dva sousedy (*a,b*). Takový vrchol i oba jeho sousedi totiž leží ve dvou různých stěnách, přičemž přidání *u* mezi *a,b* v jedné z nich by mělo změnit seznam sousedů vrcholu *v* podle ručiček z *a,b* na *a,u,b*, zatímco v druhé na *a,b,u*. To lze vyřešit seřazením vrcholů v každé stěně podle směru hodinových ručiček vůči pomyslnému středu této stěny. V jedné stěně pak budou vrcholy seřazeny jako *a,v,b*, zatímco v druhé jako *b,v,a*. Pak už stačí vzít sousedy *v* v pořadí dané stěny a v seznamu sousedů vrcholu *v* na místo mezi nimi zařadit souseda *u*. Tedy v případě počátečního seznamu sousedů *v* podle ručiček *a,b* a

seřazení vrcholů v dané stěně *a,v,b* bude seznam sousedů vrcholu *v* podle ručiček po přidání *u* vypadat *a,u,b*, při seřazení *b,v,a* pak *a,b,u*.

Druhá možnost získání uspořádání sousedů vrcholů podle hodinových ručiček spočívá ve využití seznamu stěn a seznamu chord získaného aplikací algoritmu 4.2.2 podobně, jako jsme je využili v algoritmu 4.3.1 – jen místo vykreslování chord vždy jejím krajním vrcholům pozměníme seznam sousedů podle ručiček. Následuje takto pozměněná verze algoritmu 4.3.1:

Algoritmus 4.3.4 (Nalezení uspořádání sousedů podle hodinových ručiček)
Vstup: Graf G a seznam jeho chord vygenerovaný algoritmem 4.2.2.

1) Každé chordě přidělíme stěnu, kterou uzavírá (Tu vždy musí mít, protože přidáním chordy do grafu se vždy rozpadne jedna stěna na dvě).

2) První chordu vykreslíme jako kružnici do roviny. Zároveň tuto kružnici vezmeme jako počáteční vnější stěnu rovinného nakreslení a všem jejím vrcholům vložíme do seznamu sousedů podle ručiček oba sousedy na této kružnici ve stejném pořadí, v jakém byly zapsány v původní chordě.

3) Pro každou chordu P ze seznamu chord aplikujeme následující postup:

    a. Nechť *u* a *v* jsou krajní vrcholy chordy P. Vezměme sousedy *a,b* vrcholu *u* v dosavadní vnější stěně vykresleného grafu – nechť jsou v této stěně v pořadí *a,u,b*. Vrcholu *u* vložme do seznamu sousedů podle ručiček jeho souseda v P (označme jej *s*) mezi sousedy *a,b* tak, aby výsledné pořadí těchto vrcholů bylo *a,u,b*. Stejný postup aplikujme i pro vrchol *v*.

    b. Nyní upravíme vnější stěnu rovinného nakreslení – jednoduše v ní D nahradíme chordou P tak, aby vrcholy dosavadní vnější stěny zůstaly ve stejném pořadí, v jakém byly před tímto nahrazením.

Tento způsob sice zbytečně generuje a následně prochází seznam chord, ale zase v něm nemusíme řadit vrcholy jednotlivých stěn podle hodinových ručiček – chordu totiž vždy přidáváme do vnější stěny, jejíž pořadí vrcholů zůstává stále seřazeno ve směru ručiček. Proto nelze vybrat jiné pořadí sousedů vrcholu *u* (resp. *v*) v této stěně než právě podle hodinových ručiček.

Dokážeme-li pro každý graf najít uspořádání sousedů jednotlivých vrcholů podle směru hodinových ručiček, nic nám nebrání v použití Schnyderova algoritmu rovinného nakreslení grafu pomocí úseček.

5 Rovinná nakreslení pomocí úseček – Schnyderův algoritmus

V této kapitole se budeme zabývat Schnyderovým algoritmem, který slouží k nalezení rovinného nakreslení grafu pomocí úseček takového, že pozice vrcholů v tomto nakreslení leží na trojúhelníkové mřížce. Kompletní popis tohoto algoritmu včetně důkazů uvedených vět lze nalézt v původním článku Waltera Schnydera [3].

5.1. Realizátor

Vstupem pro Schnyderův algoritmus je rovinný triangulovaný graf se sousedy jednotlivých vrcholů seřazenými podle směru hodinových ručiček. K otestování rovinnosti lze použít algoritmus 4.2.2, k seřazení sousedů podle směru hodinových ručiček zase algoritmus 4.3.4 (případně lze oba algoritmy skombinovat), ovšem ještě jsme si nedefinovali pojem triangulovaného grafu, což nyní napravíme:

Definice 5.1.1
Řekneme, že rovinný graf G je *triangulovaný*, jestliže každá jeho stěna v libovolném rovinném nakreslení je trojúhelník (tedy $K_3 = C_3$).

Zde je dobré si uvědomit, že triangulovaný graf je zároveň maximální rovinný, protože do něj již nelze přidat žádnou hranu bez přidání rovinnosti.

Nyní je možné se pozastavit nad faktem, že ne každý rovinný graf je triangulovaný, pročež Schnyderův algoritmus půjde použít pouze na malou část všech rovinných grafů. To je sice pravda, ovšem naštěstí platí následující tvrzení:

Tvrzení 5.1.2
Z každého rovinného grafu lze postupným přidáváním hran vytvořit graf triangulovaný.

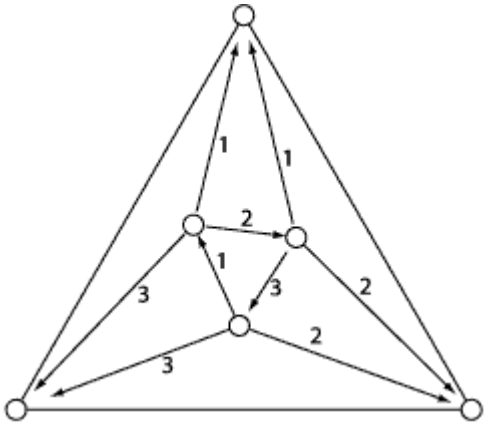
Proto můžeme z každého vstupního grafu ještě před samotnou aplikací Schnyderova algoritmu vytvořit triangulovaný graf (tomuto procesu budeme nadále říkat triangulace). Zde samozřejmě lze namítnout, že triangulací přidané hrany pozmění strukturu grafu a budou vidět i ve výsledném nakreslení, ovšem tuto námitku lze velmi jednoduše zamítnout – přidané hrany se po nalezení pozic vrcholů v rovině jednoduše smažou a nalezené rovinné nakreslení je tak obsahovat nebude. Více se problematice triangulace a vykreslování přidaných hran pomocí našeho programu budeme věnovat v kapitole 6, nyní se ale spokojíme s tím, že Schnyderův algoritmus pracuje pouze s triangulovanými grafy a že libovolný rovinný graf lze pro tento algoritmus předem triangulovat.

Klíčovým pro celý algoritmus pak je následující pojem realizátoru. Před jeho definicí ještě poznamenejme, že vnější hranou myslíme libovolnou hranu ve vnější stěně (triangulovaný graf tedy má právě tři vnější hrany) a vnitřní jsou všechny hrany grafu, které neleží v jeho vnější stěně. Podobně lze definovat i vnější a vnitřní vrcholy. Nyní již můžeme definovat realizátor:

Definice 5.1.3
Realizátorem triangulovaného grafu G rozumíme orientaci vnitřních hran grafu G a jejich rozdělení do tří množin T_1, T_2, T_3 hran tak, že pro každý vnitřní vrchol *v* grafu G platí:

1) *v* má výstupní stupeň jedna v každé z T_1, T_2, T_3.
2) Seřadíme-li hrany vycházející z *v* proti směru hodinových ručiček, budou seřazeny
   v následujícím pořadí: Výstupní v T_1, vstupní v T_3, výstupní v T_2, vstupní v T_1, výstupní v T_3, vstupní v T_2.

Pro ilustraci uvedeme klasický příklad realizátoru:



Obrázek 5.1.1 Ukázka realizátoru triangulovaného grafu $G_{5.1}$.

Zde je nutné poznamenat, že existuje i *normální označení* triangulovaného grafu použitelné prakticky stejně jako realizátor. Normální označení a realizátor nějakého grafu jsou ale na sebe vzájemně převoditelné, takže se spokojíme pouze se zmíněním alternativního pojmu a nadále budeme používat výhradně pojem realizátoru.

Tvrzení 5.1.4
Ke každému triangulovanému grafu lze nalézt realizátor.

Toto tvrzení se ukazuje jako zcela zásadní, protože bez realizátoru vstupního grafu by Schnyderův algoritmus nebyl schopen pracovat. Kromě potvrzení existence realizátoru pro všechny triangulované grafy ovšem potřebujeme umět pro nějaký konkrétní graf realizátor najít. A právě k tomu slouží následující algoritmus:

Algoritmus 5.1.5 (Nalezení realizátoru)
Vstup: Triangulovaný graf G s hranami uspořádanými kolem každého vrcholu podle hodinových ručiček.

1) Vybereme jednu stěnu grafu G a zvolíme ji za vnější stěnu. Její vrcholy označíme v_1, v_2, v_3.
2) Dokud v G existuje nějaký vnitřní vrchol, opakujeme následující kroky:
   a. Ze sousedů vrcholu v_1 vybereme takový vrchol *u*, který má s v_1 právě dva společné sousedy. Musí ale platit $v_2 \neq u \neq v_3$.
   b. Vrchol *u* vložíme do zásobníku odebraných vrcholů.

c. Pro vrchol u si vytvoříme seznam S_u všech jeho sousedů, kteří nesousedí s v_1. Naopak jeho sousedy sousedící s v_1 si označíme jako x_u a y_u tak, aby v uspořádání sousedů u podle směru hodinových ručiček byli v pořadí v_1, x_u, y_u.
d. Provedu kontrakci hrany $\{u, v_1\}$.

3) Ze zásobníku beru postupně vrcholy (poslední odebraný opět vždy označím u) a provádím jejich „odkontrakci“ v grafu G:
   a. Pro každý vrchol w z S_u smažu z grafu G hranu (w, v_1).
   b. Vrchol u přidám do grafu G.
   c. Do grafu přidám hranu (u, v_1) a ohodnotím ji 1.
   d. Do grafu přidám hranu (u, y_u) a ohodnotím ji 2.
   e. Do grafu přidám hranu (u, x_u) a ohodnotím ji 3.
   f. Pro každý vrchol w z S_u přidám do grafu hranu (w, u) a ohodnotím ji 1.
4) Nyní definujeme T_i jako sjednocení všech hran ohodnocených i. T_1, T_2 a T_3 pak tvoří realizátor grafu G.

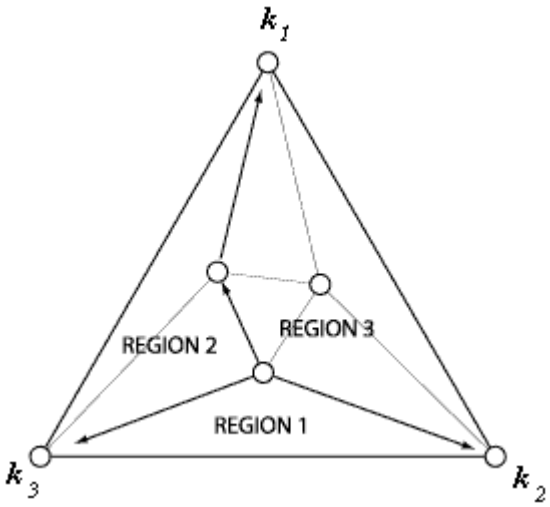
V této podkapitole jsme ukázali, jak pro triangulovaný graf G najít jeho realizátor. V následující podkapitole pak uvedeme popis využití realizátoru ke konstrukci rovinného nakreslení pomocí úseček.

5.2. Nalezení rovinného nakreslení

Schnyderův algoritmus hledá pozice vrcholů triangulovaného grafu G pro rovinné nakreslení na základě vzdálenosti od jednotlivých hran vnější stěny – tato vzdálenost je určována podle počtu vrcholů (případně trojúhelníků) v částech grafu daných aktuálně zpracovávaným vrcholem a množinami T_1, T_2, T_3 definovanými v 5.1.3. Následující tvrzení naznačuje, jak tyto části najít:

Tvrzení 5.2.1
Nechť G je triangulovaný graf na nejméně čtyřech vrcholech a nechť T_1, T_2, T_3 tvoří realizátor grafu G. Potom každé T_i tvoří strom na všech vnitřních vrcholech a právě jednom vnějším vrcholu grafu G. Navíc jsou všechny hrany v T_i orientovány směrem k jedinému vnějšímu vrcholu v T_i. Vnější vrcholy patřící do T_1, T_2, T_3 jsou vzájemně různé a ve vnější stěně jsou seřazeny proti směru hodinových ručiček.

Nazveme-li jediný vnější vrchol v T_i kořenem T_i a označíme-li jej k_i, můžeme díky předchozímu tvrzení pro každý vnitřní vrchol v grafu G nalézt orientovanou cestu z v do k_i skládající se výhradně ze hran ohodnocených číslem i. Dostáváme tedy cesty z v do všech vrcholů vnější stěny G, které rozdělují G na tři regiony. Region mezi cestami z v do k_2 a z v do k_3 označíme Region 1, region mezi cestami z v do k_3 a z v do k_1 pak Region 2 a Regionem 3 nazveme region mezi cestami z v do k_1 a z v do k_2. Pro ilustraci raději použijeme následující obrázek:



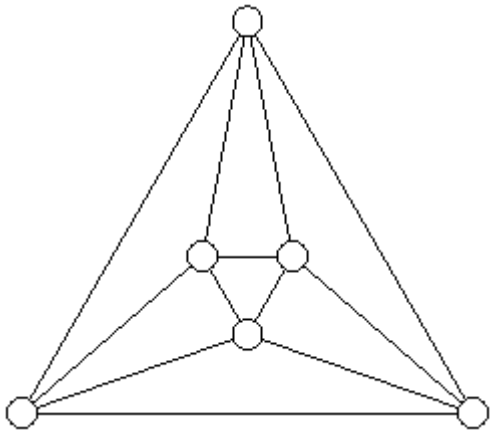
Obrázek 5.2.1 Rozdělení grafu $G_{4.1}$ realizátorem z obrázku 4.1.1 na 3 regiony.

Nyní již stačí pro každý vnitřní vrchol *v* spočítat tzv. barycentrické souřadnice x,y,z, v nichž x vyjadřuje vzdálenost *v* od vnější hrany $\{k_2, k_3\}$, y vzdálenost *v* od $\{k_1, k_3\}$ a z od $\{k_1, k_2\}$. K tomu ale potřebujeme určit, do kterých regionů spadají vrcholy ležící na jednotlivých cestách – to provedeme jednoduchým pravidlem:

Vrcholy na cestě z vrcholu *v* do k_i spadají vždy do regionu, který je sevřen cestou z *v* do k_i a cestou z *v* do vrcholu následujícího za k_i v uspořádání vrcholů vnější stěny podle hodinových ručiček. Samotný vrchol v pak nezapočítáváme nikam.

Tím dosáhneme toho, aby vrcholy z jedné cesty nebyly započítávány do více regionů zároveň.

Nyní již stačí každému vrcholu za x dosadit počet vrcholů v Regionu 1, za y počet vrcholů
v regionu 2 a za z počet vrcholů v Regionu 3, čímž dostáváme barycentrické souřadnice daného vrcholu, které lze využít buď přímo pro vykreslení grafu do trojúhelníkové mřížky nebo odstraněním z-ové souřadnice a vykreslením grafu do roviny s dvourozměrnými souřadnicemi x a y. Ve druhém případě ovšem dochází k lehké vizuální deformaci výsledného nakreslení, a proto v mém programu využívám první možnosti. Nalezené vrcholy pak již jen stačí spojit (úsečkovými) hranami a graf je rovinně vykreslen.



Obrázek 5.2.2 Rovinné nakreslení grafu $G_{5.1}$ pomocí úseček

Zde si ještě dovolím malou poznámku: Místo vrcholů v regionech lze počítat jednotlivé trojúhelníkové stěny, které jsou v daném regionu obsaženy, čímž také nalezneme rovinné nakreslení daného grafu pomocí úseček. Ovšem sám Schnyder [3] dává přednost počítání vrcholů, a tak jsem se zachoval podle jeho úsudku a rovněž upřednostnil právě vrcholy před stěnami.

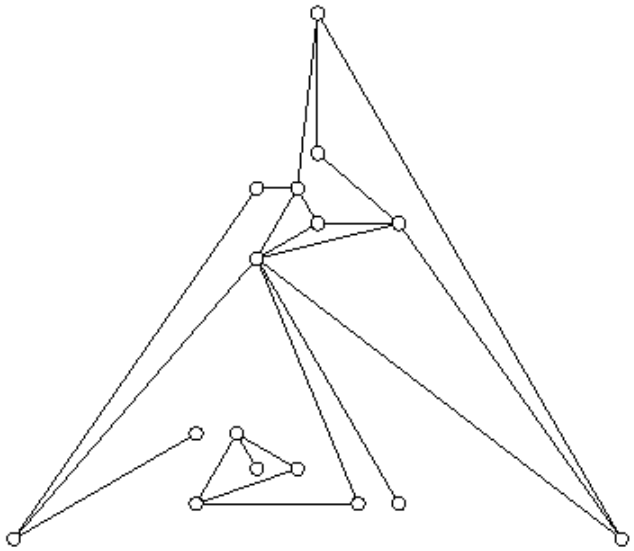
5.3. Kompletní algoritmus

V předchozích podkapitolách 5.1 a 5.2 jsme popsali jednotlivé části Schnyderova algoritmu, nyní se dostáváme k jeho celkovému zápisu, v němž dokonce ustoupíme nejen od požadavku na maximalitu vstupního grafu, ale i od rovinnosti tohoto grafu jako takové – půjde tedy o kompletní algoritmus se vším všudy. Jeho jednotlivé kroky ovšem popíšeme pouze velmi stručně – pro bližší informace pak doporučuji vyhledat daný krok v dřívějším textu. V případě kroku 2 pak prosím o shovívavost, neboť jeho konkrétní detaily jsou úzce spjaty s námětem následující kapitoly, a proto se mu věnujeme až tam.

Algoritmus 5.3.1 (Schnyderův, obohacený o test rovinnosti a triangulaci)
Vstup: Graf G.

1) Test rovinnosti spojený s nalezením planárního vnoření v podobě setřídění hran podle ručiček (například aplikací algoritmů 4.2.2 a 4.3.4)
2) Triangulace G.
3) Konstrukce realizátoru T_1, T_2, T_3 grafu G.
4) Nalezení barycentrických pozic všech vrcholů grafu G pomocí realizátoru T_1, T_2, T_3.
5) Rovinné nakreslení pomocí úseček.

Tak tedy vypadá kompletní algoritmus pro nalezení rovinného nakreslení grafu pomocí úseček – příklad jeho výsledku poskytuje následující obrázek:



Obrázek 5.3.1 Graf z obrázku 4.3.1 vykreslený algoritmem 5.3.1

Výsledky Schnyderova algoritmu ovšem záleží na konkrétních volbách, které je třeba provést v jednotlivých krocích – kromě na konci minulé kapitoly zmíněné možnosti počítání stěn v regionech místo vrcholů (kde jsem se ovšem striktně řídil poznatky samotného Waltera Schnydera a zvolil vrcholy) jde především o způsob triangulace a také o volbu vnější stěny. A právě tomu se budeme věnovat v následující kapitole.

6 Závislost Schnyderova algoritmu na triangulaci a výběru vnější stěny

V této kapitole se pokusíme zjistit, jaký vliv můžou mít volby mezi různými možnostmi
v průběhu Schnyderova algoritmu – konkrétně půjde o výběr jednoho ze tří typů triangulací, poté se podíváme ještě na vliv volby vnější stěny. Nejprve se ale podíváme na kritéria, podle nichž budeme kvalitu jednotlivých nakreslení posuzovat:

6.1. Kritéria vizuální kvality nakreslení

Již v kapitole 3.1 jsme se zmiňovali o kritériích kvality, podle nichž lze určitým způsobem fakticky zachytit kvalitu jednotlivých nakreslení grafů. A jelikož se v této práci rozhodně nechci spolehnout pouze na svůj objektivní úsudek o „kráse“ toho kterého nakreslení, rozhodl jsem se některá z estetických kritérií zahrnout i do samotného programu. Konkrétně půjde o součet délek hran, nejmenší úhel mezi hranami a nejvíce zaplněné okolí vrcholu, tedy určitou formu shluků vrcholů na jednom místě. Pojďme se podívat, co jednotlivými kritérii myslíme:

Součet délek hran

Toto kritérium jednoduše sečte délky všech hran v aktuálním nakreslení rovinného grafu. Samozřejmě se nezapočítávají délky hran přidaných triangulací, a to ani v případě zvolení jejich vykreslení na obrazovku uživatelem. Součet délek hran sice ne vždy vypovídá o estetické kráse, ale často podává poměrně kvalitní měřítko mezi jednotlivými vykresleními – zejména se díky němu dají eliminovat extrémní případy, kdy dojde k nadměrnému natažení hran v důsledku nešťastné volby vnější stěny. Toto kritérium pak má taky rozumný praktický význam, od ušetření inkoustu při tisku (k tomu ovšem nebylo v průběhu této práce využíváno) až po praktické aplikace, kdy je potřeba podle rovinného nakreslení skutečně něco zkonstruovat a docházelo by ke zbytečné spotřebě materiálu. Součet délek hran je tedy záhodné pokud možno minimalizovat.
Ještě je třeba dodat, že celkovou délku hran budeme měřit podle vzdálenosti souřadnic jednotlivých vrcholů po přepočítání výsledků Schnyderova algoritmu do trojrozměrné mřížky. V žádném případě tedy nejde o aktuální vzdálenosti v pixelech na obrazovce, ale vždy o stejné měřítko, v němž jeden bod vyjadřuje minimální možnou vzdálenost dvou vrcholů v nakreslení nalezeném Schnyderovým algoritmem.

Nejmenší úhel mezi hranami

Úhly mezi hranami jsou pro vizuální kvalitu nakreslení grafů velmi klíčovým kritériem – některé vykreslovací algoritmy jsou dokonce speciálně navrženy tak, aby se ve výsledném nakreslení úhly mezi hranami co nejvíce blížily hodnotě 360° dělené počtem hran kolem vrcholu. My se ale spokojíme s nalezením nejmenšího úhlu v daném nakreslení – už díky tomu totiž lze celkem pěkně rozlišit mezi některými grafy, protože jediný skutečně malý úhel dokáže celé nakreslení vizuálně naprosto deformovat: Může díky němu totiž dojít k přehlédnutí některé z hran, případně k mylné představě, že hrana vede do jiného vrcholu. Oba případy by mohly

mít kritické následky při použití v praxi, tudíž rozhodně nemusí být od věci pokusit se nalézt triangulaci, resp. stěnu, jejíž nejmenší úhel mezi hranami by byl v rámci všech možností maximální.
Nejmenší úhel budeme udávat ve stupních.

Nejvíce zaplněné okolí vrcholu

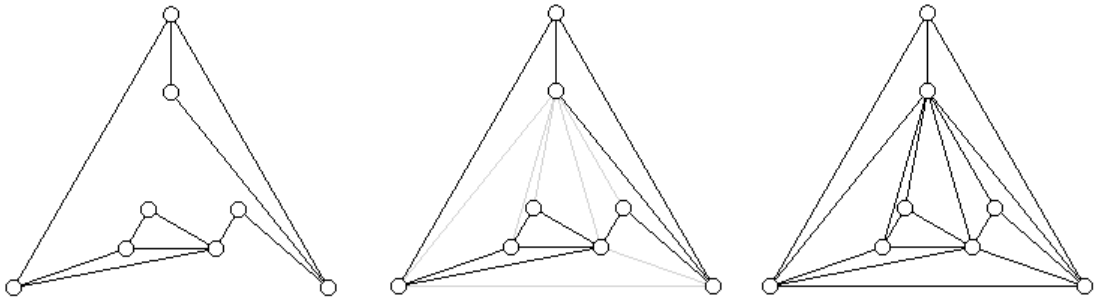
Také shluky vrcholů na malém prostoru můžou výrazně kazit estetický dojem celého nakreslení, protože soustředí velkou část nakreslení do jediného místa a zbytek grafu zůstává prázdný. V případě Schnyderova algoritmu pak sice nedochází k vyloženě extrémním shlukům, ale i tak se vyplatí jej v tomto ohledu podrobit řádnému zkoumání. Proto u každého rovinného nakreslení tímto algoritmem hledám vrchol s nejvyšším počtem ostatních vrcholů v určitém okolí – jeho poloměr jsem heuristicky zvolil jako odmocninu z celkového počtu vrcholů. Díky tomu lze určitým způsobem určit maximální shluk v grafu, který by ve většině nakreslení měl být pokud možno co nejmenší.
Zaplněnost okolí budeme udávat počtem vrcholů v tomto okolí ležících. Toto kritérium je potřeba pokud možno minimalizovat.

6.2. Triangulace

Už na začátku kapitoly 5.1 jsme se zmínili o tom, že Schnyderův algoritmus vyžaduje na vstupu triangulovaný graf, do algoritmu 5.3.1 jsme pak triangulaci přímo zařadili, ovšem stále jsme se nezmínili o jejím provedení. To proto, že jsme si tuto problematiku šetřili až pro aktuální kapitolu, kde se triangulacím budeme plně věnovat – nejprve uveďme, co přesně pojmem triangulace myslíme:

Triangulace grafu G je proces postupného přidávání hran do grafu G tak, aby ve výsledku byly všechny stěny rovinného nakreslení grafu G trojúhelníky.

Pro lepší představu uvedeme příklad triangulace na následujícím obrázku (vlevo původní graf, uprostřed šedivě znázorněné triangulací přidávané hrany, vpravo triangulovaný graf):



Obrázek 6.2.1 Triangulace grafu

Triangulaci jsme tedy označili za proces přidávání hran, ovšem stále jsme neuvedli, jak ji provést. A právě zde leží kámen úrazu: Samozřejmě nejde o nic složitého (jednoduše triangulovat prakticky jakýkoli rovinně nakreslený graf by pravděpodobně dokázalo i malé dítě), ale problém vězí spíš v tom, že existuje mnoho strategií, jak grafy triangulovat. Na následujících řádcích si tedy představíme tři mnou implementované strategie triangulace.

Triangulace z jednoho vrcholu

Tato strategie spočívá ve výběru jednoho z vrcholů každé stěny a následného přidání hran mezi vybraný vrchol a všechny ostatní vrcholy dané stěny s daným vrcholem nesousedící. Takový vrchol lze vybrat v každé netrojúhelníkové stěně, a to díky následujícímu tvrzení:

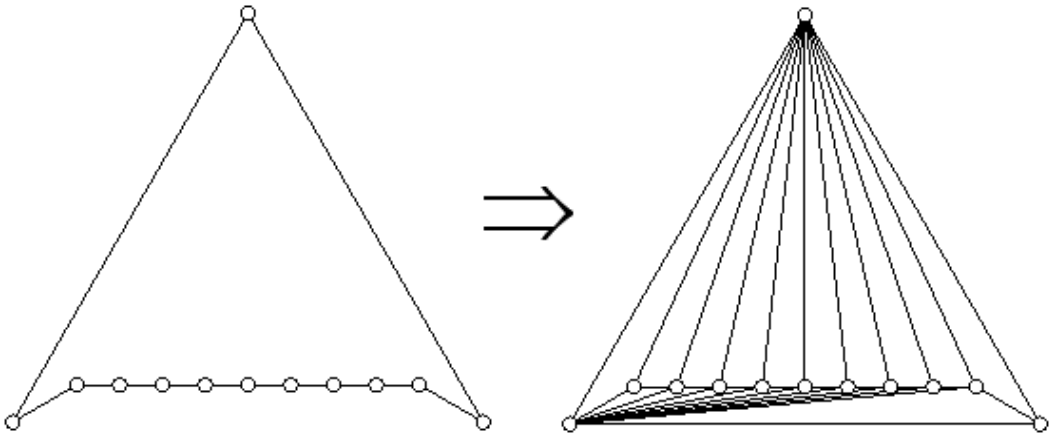
Tvrzení 6.2.1
Nechť G je vnějškově rovinný graf, v němž jedna stěna S obsahuje všechny vrcholy G (tedy G je vnějškově rovinný). Pak v G existuje vrchol, jehož jedinými sousedy jsou jeho dva sousedi ve stěně S.

Pokud tedy vezmeme graf G = (V,E), kde V jsou právě vrcholy aktuálně triangulované stěny S a E jsou právě hrany mezi vrcholy z V, tak díky tvrzení 6.2.1 zjišťujeme, že tento graf musí obsahovat vrchol, který má v G právě dva sousedy ležící v S. Z tohoto vrcholu tedy můžeme vést hranu do všech ostatních vrcholů v S kromě dvou zmíněných sousedů a získat tak triangulaci stěny S. Pokud pak uvedený postup aplikujeme na každou stěnu grafu G, dostaneme triangulaci celého G. Následující algoritmus shrnuje celý tento postup:

Algoritmus 6.2.2 (Triangulace z jednoho vrcholu)
Vstup: 2-souvislý rovinný graf G a seznam jeho stěn.

Pro každou netrojúhelníkovou stěnu S provedeme následující operace:

1) Nalezneme vrchol *v*, který má v G právě dva sousedy ležící v S (označme je s_1 a s_2).
2) Pro každý vrchol *u* z S, $u \neq v$, $u \neq s_1$, $u \neq s_2$, přidáme do grafu G hranu $\{u,v\}$.

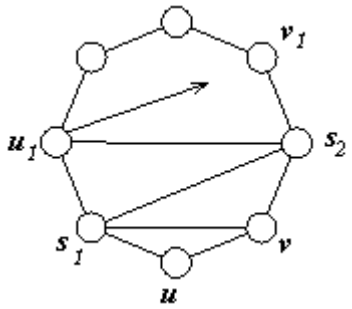


Obrázek 6.2.2 Triangulace C_{12} z jednoho vrcholu (oba grafy vykresleny Schnyderovým algoritmem).

Triangulace Cik-Cak

Jak už název této triangulační strategie napovídá, hrany se v ní přidávají na přeskáčku mezi pomyslnými polovinami dané stěny. Konkrétně i zde začneme od vrcholu s právě dvěma sousedy v G ležícími v S (tento vrchol označíme *v*, jeho sousedy v S pak s_1 a s_2). Následně najdeme souseda *u* vrcholu s_1 v S takového, že

$u \neq v$, a do grafu G přidáme hrany $\{v,u\},\{u,s_2\}$. Poté položíme u_1 rovné sousedovi u v S tak, aby $u_1 \neq s_1$ a přidáme do G hranu $\{s_2,u_1\}$. Nyní za v_1 vezmeme souseda s_2 různého od v a do grafu přidáme hranu$\{u_1,v_1\}$. Takto pokračujeme až do kompletní triangulace celé stěny S (již v předchozím postupu je třeba testovat, zda jsme už nedosáhli triangulace stěny – ale pro lepší nastínění práce algoritmu jsem to za každé přidání hrany neuváděl). Tento postup ilustruje následující obrázek:



Obrázek 6.2.3 Náčrt triangulace C_8 Cik-Cak způsobem

Během triangulace Cik-Cak je ale třeba si dát pozor na dřívější existenci přidávaných hran – ne pro každý vrchol ve stěně S totiž platí Tvrzení 6.2.1, a tak by se mohlo stát, že bychom do G přidali duplicitní hranu (protože taková již mezi zvolenými vrcholy leží mimo aktuální stěnu). To naštěstí lze ošetřit poměrně jednoduchým způsobem: Řekněme, že by $\{x,y\}$ měla být duplicitní hrana, přičemž do vrcholu x již vede hrana $\{x,z\}$ přidaná právě probíhající triangulací. V takovém případě postupujeme od vrcholu y podél okraje stěny S v opačném směru než v jakém leží soused z až do chvíle, kdy narazíme na vrchol y_1, z něhož nevede hrana do vrcholu x, případně do okamžiku, kdy narazíme na samotný vrchol x. Pokud jsme nedošli až do x, přidáme do grafu G hranu $\{x,y_1\}$ a pokračujeme v Cik-Cak triangulaci z y_1. Nesmíme ale zapomenout na přeskočené vrcholy, které společně s body x,z a případným y_1 vytvoří samostatnou stěnu, kterou triangulujeme odděleně. Tím se nám sice lehce poruší hledaná Cik-Cak struktura, ale zase půjde o korektní triangulaci bez duplicitních hran.

Dosavadní popis Cik-Cak triangulace možná mohl působit trochu nepřehledně, a tak ji raději opět zapíšeme jako algoritmus. Nejprve ale označme jako levého souseda libovolného vrcholu ve stěně S jeho následníka v této stěně po směru hodinových ručiček. Pravým sousedem pak je následník v dané stěně proti směru hodinových ručiček.

Algoritmus 6.2.2 (Triangulace Cik-Cak)
Vstup: 2-souvislý rovinný graf G a seznam jeho stěn.

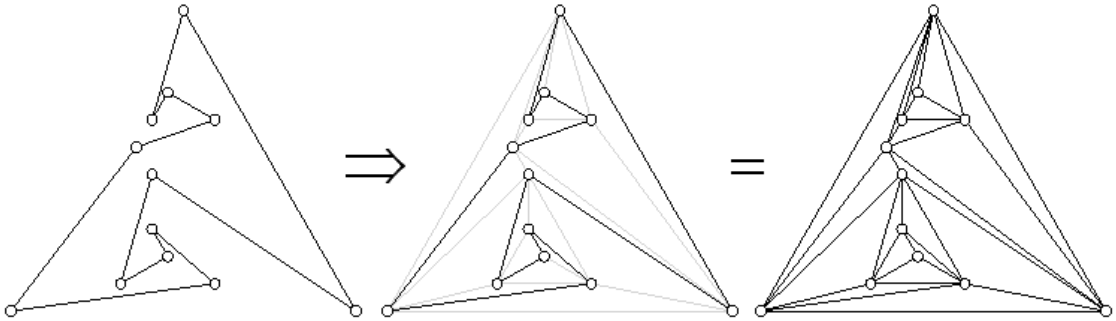
Pro každou netrojúhelníkovou stěnu S ze Seznamu stěn provedeme následující operace:

1) Nalezneme vrchol v, který má v S právě dva sousedy Nechť s_1 je jeho levý soused ve stěně S a s_2 jeho pravý soused ve stěně S.
2) Označme u levého souseda s_1 ve stěně S a do grafu G přidejme hranu $\{v,u\}$.
3) Je-li již stěna S triangulována, pokračujeme další stěnou ze Seznamu stěn.

4) Alokujeme novou stěnu S_1, vložíme do ní vrcholy *v*,*u* a opakujeme následující postup
    i. Vložíme *v* do S_1.
    ii. Položíme *v* rovno jeho pravému sousedovi ve stěně S

   až do chvíle, kdy $\{v,u\} \notin E(G)$ nebo *u* je soused *v*. Má-li S_1 více než tři vrcholy, vložíme S_1 na konec Seznamu stěn, jinak S_1 dealokujeme.
5) Je-li *u* soused *v*, pokračujeme další stěnou ze Seznamu stěn. Jinak do grafu G přidáme hranu $\{v,u\}$.
6) Je-li již stěna S triangulována, končíme.
7) Alokujeme novou stěnu S_1, vložíme do ní vrcholy *v*,*u* a opakujeme následující postup
    iii. Vložíme *u* do S_1.
    iv. Položíme *u* rovno jeho levému sousedovi ve stěně S

   až do chvíle, kdy $\{v,u\} \notin E(G)$ nebo *u* je soused *v*. Má-li S_1 více než tři vrcholy, vložíme S_1 na konec Seznamu stěn, jinak S_1 dealokujeme.
8) Je-li *u* soused *v*, pokračujeme další stěnou ze Seznamu stěn. Jinak do grafu G přidáme hranu $\{v,u\}$ a pokračujeme krokem c.



Obrázek 6.2.4 Triangulace C_{12} Cik-Cak způsobem (všechny grafy vykresleny Schnyderovým algoritmem).

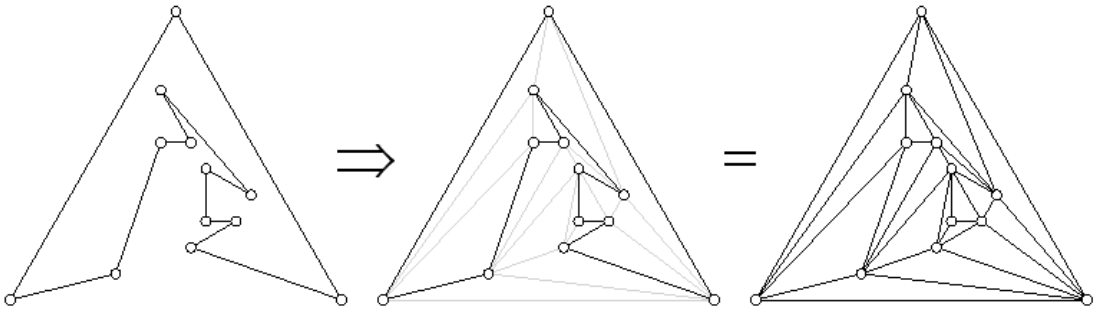
Náhodná triangulace

Tato triangulační strategie sází místo jakéhokoli systematického postupu na čistě náhodný výběr hran, které se do aktuálně zpracovávané stěny S přidávají – jednoduše náhodně vybere z S vrchol *v* a poté (samozřejmě opět náhodně) zvolí z ostatních vrcholů stěny S vrchol *u* tak, aby v G neležela hrana $\{u,v\}$. Následně přidá do grafu G hranu $\{u,v\}$ a samostatně zpracuje takto vzniklé podstěny S – zde je ovšem potřeba ošetřit fakt, že díky opětovné možnosti náhodného zvolení *u* nebo *v* v rámci triangulace nově vzniklých podstěn by se mohl výrazně zvýšit stupeň těchto vrcholů ve výsledném triangulovaném grafu (což by obecně nevadilo, ale výsledek by se často podobal výsledku triangulace z jednoho vrcholu, což pro naše účely není záhodné). Proto všem vrcholům stěny S počítáme, kolikrát již byli v rámci triangulace S náhodně vybrány, a následně vždy vybíráme pouze z vrcholů se společným nejmenším počtem výběrů. Pokud by takový vrchol byl jeden nebo dokonce žádný, zkusíme vybrat i z vrcholů s o 1 vyšším počtem použití, případně o 2, 3, atd. Počet použití si ale jednotlivé vrcholy nepřenášejí pouze do triangulace

podstěn, ale i do triangulace dalších stěn. Tím je docíleno relativně rovnoměrného rozdělení přidaných hran mezi vrcholy, a tak je rozptyl stupňů vrcholů v rámci triangulací přidaných hran velmi malý (zde samozřejmě existují výjimky v podobě vrcholů s velkým množstvím sousedů již v původním grafu). Následuje přesný popis algoritmu náhodné triangulace:

Algoritmus 6.2.2 (Náhodná triangulace)
Vstup: 2-souvislý rovinný graf G a seznam jeho stěn.

1) Všem vrcholům v G vynuluj PočetPoužití. Nastavíme VybranýPočet na 0.
2) Pro každou netrojúhelníkovou stěnu S provedeme následující operace:
    a. Z vrcholů v S s PočetPoužití=VybranýPočet vyberu náhodně vrchol *v*.
    b. Z vrcholů v S s PočetPoužití=VybranýPočet a nesousedících s *v* vyberu náhodně vrchol *u*.
    c. Do grafu přidám hranu $\{u,v\}$. A vrcholům *u* a *v* zvýším PočetPoužití o 1.
    d. Hranou $\{u,v\}$ rozdělím stěnu S na stěny S_1 a S_2. Na ty rekurzivně zavolám krok 3.

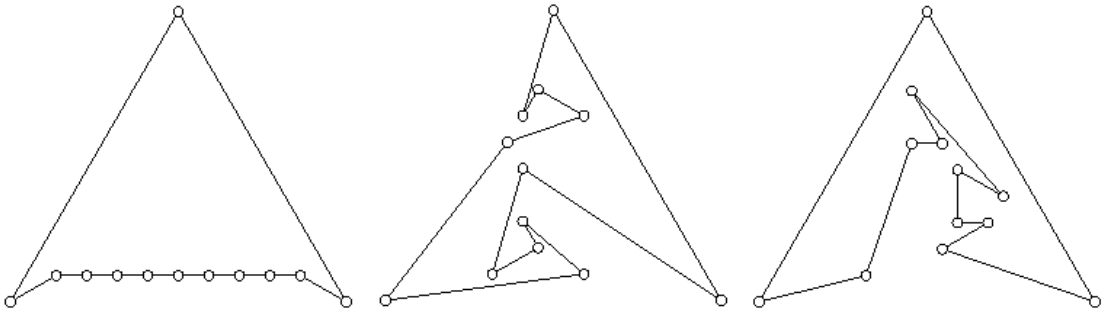


Obrázek 6.2.5 Náhodná triangulace C_{12} (všechny grafy vykresleny Schnyderovým algoritmem).

6.3. Dopad volby triangulace

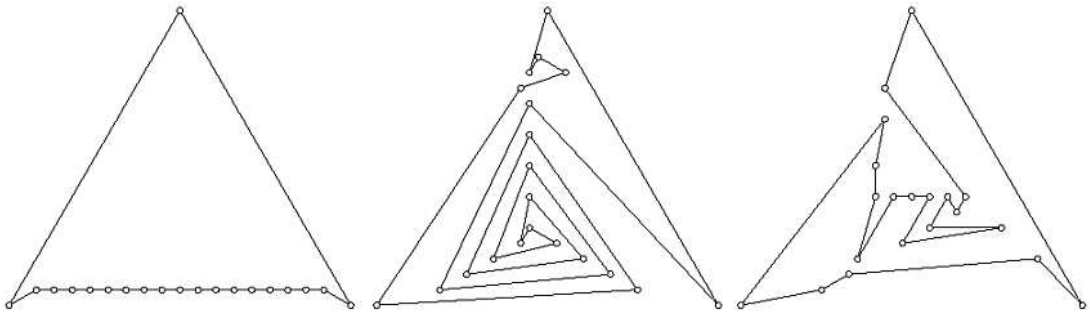
Po představení jednotlivých typů triangulací se můžeme začít zabývat jejich konkrétními výsledky v praxi – vliv typu triangulace na rovinné nakreslení Schnyderovým algoritmem je totiž už po několika pokusech jasně patrný, a tak se jej zde pokusíme ukázat na nejzajímavějších příkladech.

Již z obrázků 6.2.2, 6.2.4 a 6.2.5 je možné nahlédnout, že výsledek Schnyderova algoritmu může na použitém typu triangulace záviset poměrně zásadním způsobem – zatímco použití triangulace z jednoho vrcholu dá vzniknout nakreslení, které s trochou snahy skutečně připomíná kružnici, Cik-Cak a náhodná triangulace vyústí v dosti zamotaná nakreslení. Pro snadnější porovnání tohoto jevu je určen následující obrázek shrnující nakreslení $\mathbf{C}_{12}$ pomocí všech tří typů triangulace:



Obrázek 6.3.1 Nakreslení C_{12} pomocí tří různých triangulací: Z jednoho vrcholu, Cik-Cak a náhodné.

Z obrázku tedy (alespoň dle subjektivního mínění autora této práce) vítězně vychází nakreslení získané po triangulaci z jednoho vrcholu. Ovšem v tomto případě nezůstává pouze u subjektivního mínění – součet délek hran totiž vychází na 33,46, nejmenší úhel pak má celých 30,01° a v nejzaplněnějším okolí se nachází 6 dalších vrcholů, což až na třetí kritérium vyznívá jasně lépe než výsledky Cik-Cak triangulace (49,89; 13,9°; 6) a triangulace náhodné (49,25; 10,89°; 7). A jelikož ani z hlediska nejvyšší zaplněnosti okolí nebyla triangulace z jednoho vrcholu překonána, lze ji v tomto případě brát jako nejlepší možnost. Zbylé dvě triangulace pak ve srovnání s triangulací z jednoho vrcholu podávají víceméně podobný výsledek, a proto bych se jejich výsledkem dále nezabýval. Místo toho se podívejme, zda pro větší počet vrcholů dopadnou jednotlivá nakreslení podobně – vezměme např. $\mathbf{C}_{20}$:



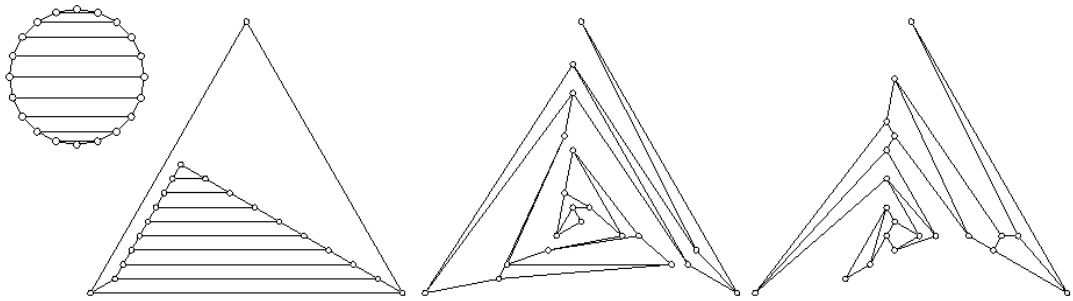
Obrázek 6.3.2 Nakreslení C_{20} pomocí tří různých triangulací: Z jednoho vrcholu, Cik-Cak a náhodné.

Dle očekávání získáváme v případě triangulace z jednoho vrcholu prakticky totožný výsledek – do výsledků kritérií se zvýšení počtu vrcholů promítlo víceméně úměrně počtu přidaných vrcholů, takže dostáváme (57,46; 30,01°; 8). Naproti tomu výsledné nakreslení po Cik-Cak triangulaci až na malou výchylku ve své horní části připomíná spirálu, čímž se sice podstatně zhorší součet délek hran (145,82) a prakticky nezmění nejmenší úhel (13,01°), ale zase se díky celkem rovnoměrnému rozdělení vrcholů dostáváme k lepšímu výsledku v oblasti nejzaplněnějšího okolí (v němž zde leží 7 vrcholů) než u triangulace z jednoho vrcholu. Náhodná triangulace (95,25; 8,95°; 11) pak podává zajímavý výsledek jedině v oblasti délek hran, ovšem i tím překonává pouze Cik-Cak triangulaci (zde je třeba poznamenat, že jedna z dalších náhodných triangulací téhož grafu poskytla překvapivý nejmenší úhel 30°, ale jelikož se podobný výsledek objevil jen u jedné z deseti vyzkoušených náhodných triangulací, rozhodl jsem se jej ignorovat).

Při dalším zvyšování stupně kružnice pak nedošlo prakticky k žádnému překvapení – triangulace z jednoho úhlu vypadá nadále prakticky stejně a poskytuje jasně nejlepší

výsledek v oblasti součtu délek hran a nejmenšího úhlu. U Cik-Cak triangulace pak dochází k neustálému rozšiřování spirály, díky čemuž nadále zůstává nepříznivý nejmenší úhel a naprosto přehnaný součet délek hran, ale zase se setkáváme s pokračující tendencí nejmenšího počtu vrcholů v nejzaplněnějším okolí ze všech tří triangulací. Náhodná triangulace pak (vcelku pochopitelně) pokračuje v zavedené chaotičnosti a překonává jedině Cik-Cak triangulaci v oblasti délek hran, kde zpravidla dosahuje výsledku pohybujícího se kolem průměru výsledků obou dalších triangulací.
Pro vykreslování kružnic (případně jen lehce pozměněných grafů) bych tedy doporučil triangulaci z jednoho vrcholu pro co nejmenší součet délek hran a maximalizaci nejmenšího úhlu, v případě požadavku na co nejméně zaplněná okolí vrcholů se pak dá úspěšně využít také Cik-Cak triangulace (u níž zároveň stojí za povšimnutí, že většina vrcholů díky spirálovitosti leží na třech přímkách).

Nicméně kružnice se dají s ještě mnohem lepším výsledkem vykreslovat přímo na kružnici, a proto se zkusíme podívat také na jiné grafy – nicméně od kružnic ještě přeci jenom neodběhneme a pokusíme se je trochu modifikovat. Jak třeba dopadne následující „poledníkový“ graf ?



Obrázek 6.3.3 Poledníkový graf: Na kružnici a Schnyderem po triangulazi z 1 vrcholu, Cik-Cak a náhodné.

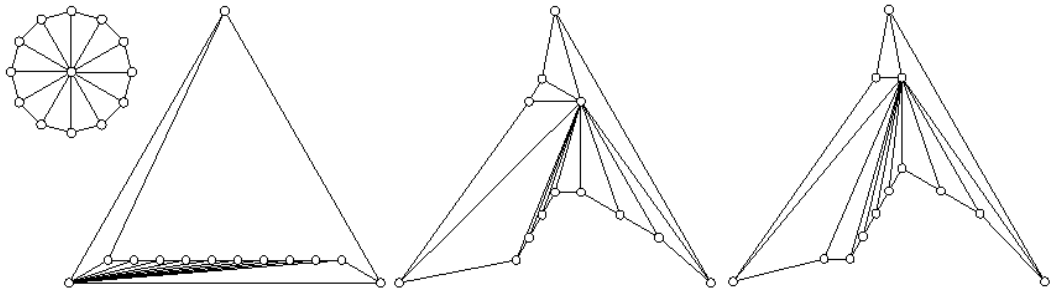
Z jednoho vrcholu: (154,31; 30,01°; 9)

Cik-Cak: (202,51; 0,6°; 9)

Náhodná: (152,81; 3,41°; 9)

Triangulace z jednoho vrcholu i zde působí velmi stabilním dojmem v oblasti délek hran i nejmenšího úhlu (navíc se jí asi jako jediné celkem daří zachytit „poledníkovou“ strukturu), zato triangulace Cik-Cak naprosto propadla – kromě o třetinu vyššího součtu délek hran oproti ostatním dvěma triangulacím se v ní nachází také opravdu malý úhel. Naopak nejlepším součtem délek hran překvapila náhodná triangulace (nutno ovšem říci, že zde jí velmi pomohla volba vnější stěny).

Mohlo by se tedy zdát, že triangulace z jednoho vrcholu dominuje kritériím součtu délek hran a nejmenšího úhlu. Ovšem to nemusí platit ve všech případech, jak ukazuje následující graf „kola“:



Obrázek 6.3.4 Kolo: Na kružnici a Schnyderem po triangulazi z 1 vrcholu, Cik-Cak a náhodné.

Z jednoho vrcholu: (118,12; 0,49°; 6)

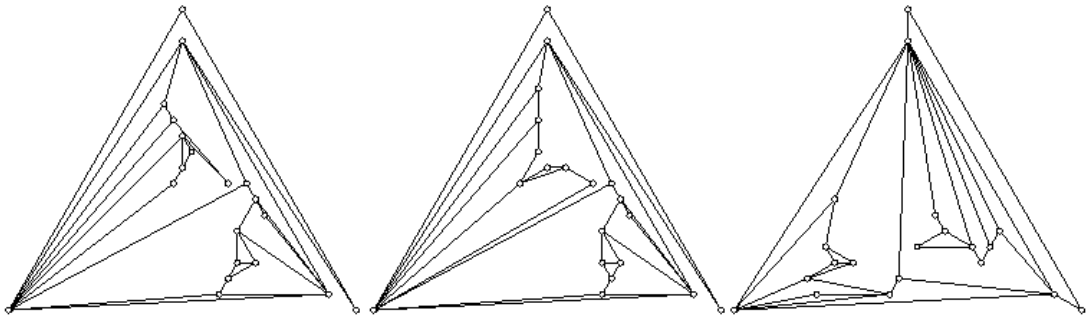
Cik-Cak: (98,97; 1,35°; 7)

Náhodná: (107,89; 2,2°; 7)

Jak je vidět, zde triangulace z jednoho vrcholu pokulhává v pro ni dosud příznivých kritériích – oba ostatní způsoby triangulace jsou totiž lepší jak v součtu délek hran, tak v nejmenším úhlu. První nakreslení pak sice disponuje nejméně zaplněným okolím, ovšem rozdíl rozhodně není nijak výrazný – navíc pokud bychom se zaměřili spíš na určitou formu pokrytí vnitřní plochy grafu, triangulace z jednoho vrcholu by se velkého úspěchu nedočkala.
Zbylé dvě triangulace v prvních dvou kritériích určitě poskytují lepší výsledky a je pouze na volbě mezi součtem délek hran a nejmenším úhlem, která z nich bude úspěšnější.

S podobným „selháním“ triangulace z jednoho vrcholu jsem se během testování setkal vícekrát a nutno říci, že většinou šlo o případ sdílející základní neduh výše znázorněného „kola“, tedy jeden vrchol vysokého stupně, z nějž vedou hrany do mnoha vrcholů nějaké větší stěny – ta je pak vytriangulována z jednoho vrcholu, čímž dostáváme dva vrcholy vysokého stupně, které sdílí velké množství sousedů. Díky tomu dochází ke špatnému vykreslení daného grafu podobně, jak je tomu na obrázku 6.3.4. Proto je v případě grafu se strukturou podobnou kolu (tedy s vysokými stupni vrcholů a zároveň několika velkými stěnami) třeba zvolit jiný způsob triangulace nebo se pokusit o správný výběr vnější stěny (čemuž se budeme věnovat v kapitole 6.4).

Právě jsme se přesvědčili, že ani triangulace z jednoho vrcholu nemusí vždy generovat dobré výsledky. Nyní se pro změnu pokusíme ukázat, že dosud víceméně neúspěšná náhodná triangulace naopak dokáže prokázat své kvality – zároveň tím opustíme oblast grafů založených převážně na velké kružnici (to ovšem neznamená že by se v následujícím grafu nedala nějaká velká kružnice najít):



Obrázek 6.3.5 Náhodný graf nakreslený Schnyderem po triangulaci z 1 vrcholu, Cik-Cak a náhodné.

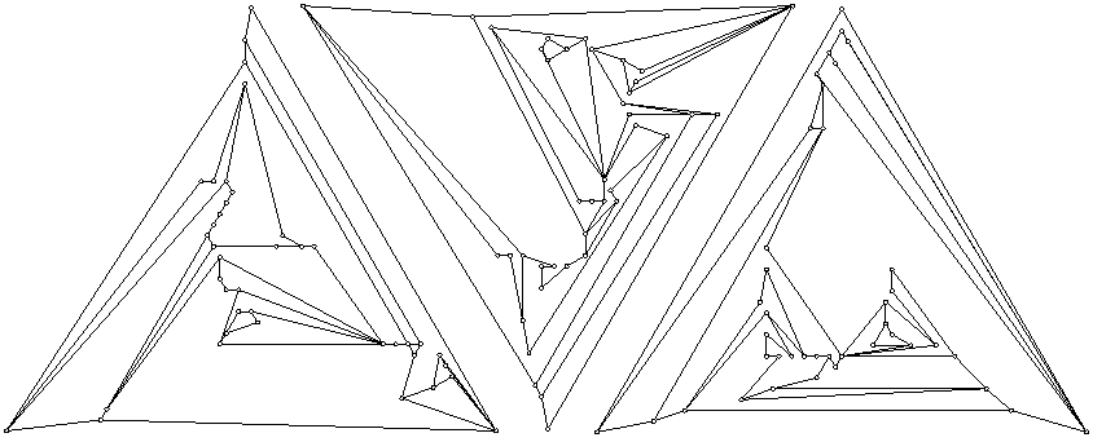
Z jednoho vrcholu: (248,23; 1,35°; 10)

Cik-Cak: (245,81; 1,13°; 9)

Náhodná: (210,73; 2,98°; 8)

Zde se náhodné triangulaci podařil vskutku husarský kousek – dokázala překonat druhé dvě triangulace ve všech třech kategoriích. Je sice třeba říci, že tak skvělého výsledku pro tento graf nedosáhnou všechny náhodné triangulace daného grafu, ale z mnou vyzkoušených vždy každá překonala jednovrcholovou i Cik-Cak triangulaci minimálně v jedné hodnocené kategorii. Navíc si (samozřejmě opět zcela subjektivně) dovolím náhodnou triangulaci na obrázku 6.3.5 označit i za vizuálně nejpodařenější. Cik-Cak a jednovrcholová triangulace pak v tomto případě přinášejí velmi podobné výsledky s tím, že až na nejmenší úhel je přeci jen o trochu lepší strategie Cik-Cak. To je další důkaz, že triangulace z jednoho vrcholu nemusí vždy přinášet ideální výsledky.

Nyní se na chvíli pozastavme nad tím, že triangulace z jednoho vrcholu a Cik-Cak triangulace přinesly dva dosti podobné výsledky. To je dáno především málo vrcholy v jednotlivých stěnách grafu na obrázku 6.3.5 – pokud totiž máme stěnu s malým počtem vrcholů a přidáme fakt, že první vrchol pro Cik-Cak triangulaci se vybírá stejným způsobem jako klíčový vrchol pro triangulaci z jednoho vrcholu, je jasné, že se výsledky nebudou lišit až tak zásadně, jako když jsme stejným způsobem triangulovali stěny o deseti a více vrcholech. K podobnému jevu samozřejmě dochází i u dalších grafů s malými stěnami, a proto je někdy pro takové grafy možné testovat jen jednu z těchto podobných triangulací a spolehnout se na podobnost výsledků. Ovšem zde je třeba si dávat pozor, protože pro větší stěny podobná heuristika rozhodně neplatí – o tom se ostatně můžeme přesvědčit na následujícím příkladu, na němž je také vidět již dříve naznačená vhodnost Cik-Cak triangulace pro minimalizaci počtu vrcholů v nejvíce zaplněném okolí (z důvodu rozumného rozložení byl prostřední graf překlopen, stále ale jde o stejné pořadí triangulací):



Obrázek 6.3.6 Náhodný graf vykreslený Schnyderem po triangulaci z 1 vrcholu, Cik-Cak a náhodné.

Z jednoho vrcholu: (436,77; 1°; 18)

Cik-Cak: (481,38; 2,18°; 13)

Náhodná: (518,32; 0,03°; 15)

Z jednotlivých výsledků lze vyčíst nejen poměrně výraznou odlišnost jednovrcholové a Cik-Cak triangulace v prvních dvou kritériích, ale především jejich výrazný rozdíl v oblasti zaplněnosti okolí – 5 vrcholů v okolí jediného vrcholu je totiž podstatný

rozdíl i u grafu na 40 vrcholech. Náhodná triangulace pak vykreslila jeden z úhlů mezi hranami tak malý, že tyto dvě hrany na obrázku zcela splývají (a to nejen v této zmenšené verzi), navíc i součet délek hran zaostává za předchozími triangulacemi, a tak nelze toto nakreslení přijmout i přesto, že nejzaplněnější okolí také celkem slušnou měrou překonává hodnotu dosaženou triangulací z jednoho vrcholu.

Pokud bych tedy měl vynést nějaké závěrečné zhodnocení jednotlivých způsobů triangulace, určitě bych doporučil použití triangulace z jednoho vrcholu pro optimalizaci součtu délek hran a nejmenšího úhlu u grafů s malými stupni vrcholů a stěnami s velkými počty vrcholů – tam totiž podává velmi dobré výsledky. Na ostatních grafech pak není od věci použít Cik-cak triangulaci, a to zejména v případě minimalizace počtu vrcholů v nejplnějším okolí, ve které tato triangulace nikdy nepodá vyloženě špatný výsledek. Pro grafy s malým počtem vrcholů v jednotlivých stěnách je pak možné použít triangulaci náhodnou, případně z ní udělat alternativu pro jednu z předchozích triangulačních strategií. Nutno ovšem říct, že náhodná triangulace dělá čest svému jménu – někdy vyloženě překvapí skvělým výsledkem, ale při opětovné aplikaci na ten samý graf může naprosto selhat.

U velkého množství grafů ovšem daleko víc než na způsobu triangulace závisí na správném výběru vnější stěny – a právě o tom pojednává následující podkapitola:

6.4. Výběr vnější stěny

Kromě různých strategií triangulací má na výsledek Schnyderova algoritmu vliv také počáteční volba vnější stěny pro konstrukci realizátoru. Už třeba pouhý fakt, jestli jsou hrany v této stěně původní nebo přidané triangulací má obrovský vliv na součet délek hran, do nějž se ty přidané triangulací samozřejmě nezapočítávají. Ovšem i kdybychom brali v potaz výhradně triangulované grafy a mohli tak zanedbat jakýkoli vliv procesu triangulace, zjistili bychom, že se nakreslení téhož grafu s různými vnějšími stěnami liší. Proto jsem do svého programu implementoval také možnost pustit Schnyderův algoritmus postupně pro každou stěnu jeho rovinného vnoření tak, že daná stěna se vždy bere jako vnější. Po vyzkoušení všech stěn se pak vybere ta, která dosáhla nejlepšího výsledku ve vybraném estetickém kritériu. Kombinací Eulerova vzorce (Tvrzení 3.2.2) s tvrzením 3.2.3 navíc dostáváme, že počet stěn triangulovaného grafu je $2|V|-4$, takže se nám celková složitost zvýší o jeden řád výše, takže se výsledku po krátké chvilce vždy dočkáme.

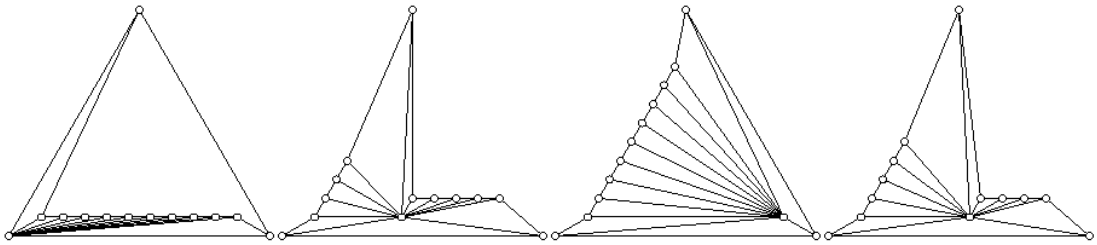
Na následujících řádcích se pokusím shrnout výsledky, které tento postup přinesl.

Nejprve ale doplním, že v implementaci Schnyderova algoritmu klasickým způsobem (tedy implementaci podle algoritmu 5.3.1 přes jedinou vnější stěnu), jsem za vnější vždy zvolil první stěnu ze seznamu stěn dodaného algoritmem rovinného vnoření– tento výběr je tedy pseudonáhodný, a proto jej využijeme pro porovnání

s výsledky hledání „nejlepší“ vnější stěny. Při něm jsem v následujících příkladech vždy využíval triangulace grafu z jednoho vrcholu.

Zde je ještě třeba poznamenat, že testováním výběru vnější stěny se nesnažíme najít žádný obecný způsob, jak tuto stěnu ze všech stěn grafu vybrat – to totiž skutečně záleží graf od grafu. A přestože nějaké obecné heuristiky by určitě nalézt šly (např. výběr vnější stěny s co nejvíce triangulací přidanými hranami pro minimalizaci součtu délek hran), my se zaměříme skutečně spíš na to, jak velký vliv z hlediska jednotlivých kritérií výběr vnější stěny může mít – tedy na porovnání nejlepšího výsledku se zmíněným pseudonáhodným výběrem vnější stěny u klasického algoritmu.

A začneme u grafu, jehož nakreslení po triangulaci z jednoho vrcholu jsme v podkapitole 6.3 tak trochu křivdili. Jde samozřejmě o „kolo“, tedy graf z obrázku 6.3.4, jehož výsledky se totiž s výběrem vhodné (resp. nejvhodnější) vnější stěny i pro tuto triangulaci z jednoho vrcholu výrazně zlepší:



Obrázek 6.4.1 Kolo triangulované z 1 vrcholu vykreslené Schnyderem a Schnyderem optimalizovaným pro délku hran, nejmenší úhel a zaplněnost okolí.

Původní výsledek: (118,12; 0,49°; 6)

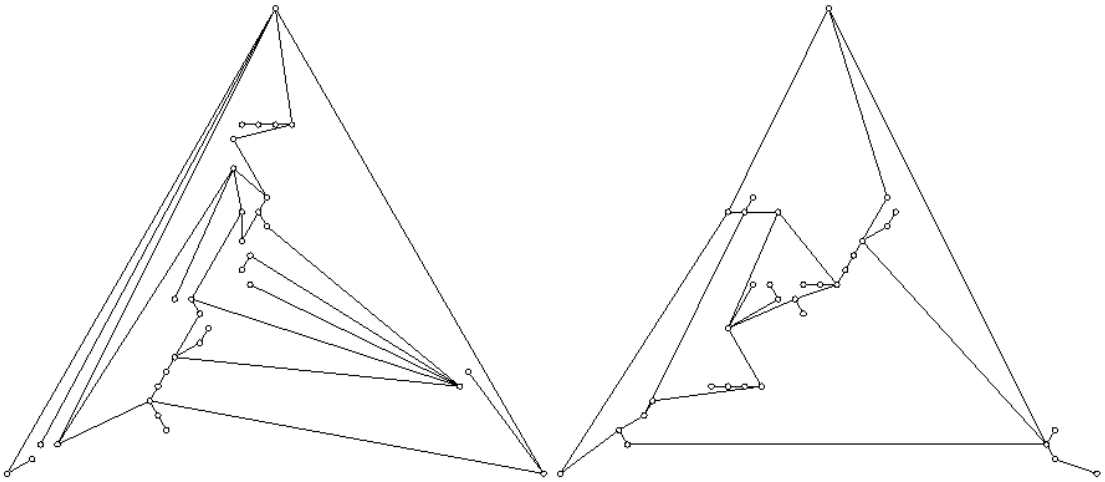
Minimalizace součtu délek hran: (89,49; 3°; 7)

Maximalizace nejmenšího úhlu: (133,13; 4,71°; 6)

Minimalizace počtu vrcholů v nejvíce zaplněném okolí: (92,18; 3,58°; 5)

Kolo tedy evidentně bylo „diskriminováno“ špatným výběrem vnější stěny (což se u základního Schnyderova algoritmu občas stává), kdy při výběru té nejlepší dojde k jasnému zlepšení všech tří kategorií: Součet délek hran se o čtvrtinu sníží, úhel o 4 stupně zvýší a z nejzaplněnějšího okolí také jeden vrchol ubude. Nejvíce se pak ale zlepší samotný vizuální pojem z celého grafu, kdy zejména díky zvýšení nejmenšího úhlu nedochází ke splývání vykreslených hran.

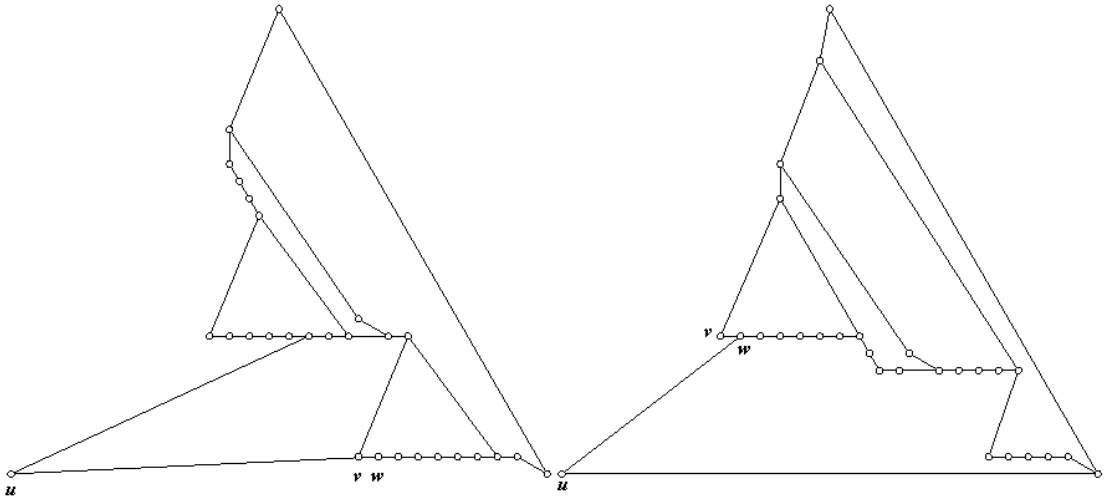
Kolo si tedy dokázalo správným výběrem vnější stěny pro každé kritérium výrazně polepšit – a nejinak je tomu i u ostatních grafů: Někdy lze výběrem správné stěny za vnější dosáhnout skutečně nečekaných výsledků, a to už v porovnání s pseudonáhodnou volbou klasického Schnydera, v porovnání s nejhorším případem bychom pravděpodobně dosáhli ještě vyšších rozdílů. Pro ilustraci možného rozdílu uvedeme pro každé kritérium jeden zvláštní případ, na němž je dané zlepšení skutečně markantní – a začneme u minimalizace součtu délek hran:



Obrázek 6.4.2 Nakreslení téhož grafu Schnyderovým algoritmem a tímtéž algoritmem s výběrem vnější stěny.

Zde si troufám tvrdit, že výrazné zlepšení je po výběru vnější stěny jasně vidět už pouhým okem, nicméně číselné vyjádření jej prokazuje zcela objektivně: Zatímco první nakreslení disponuje celkovou délkou hran 389,32, u nakreslení druhého jde o pouhých 197,39. Tím jsme se dostali téměř na polovinu předchozí hodnoty (a to bez výrazné změny dalších dvou kritérií), a tak o skutečně příznivém vlivu výběru vnější stěny na výsledné nakreslení v tomto případě nelze pochybovat.

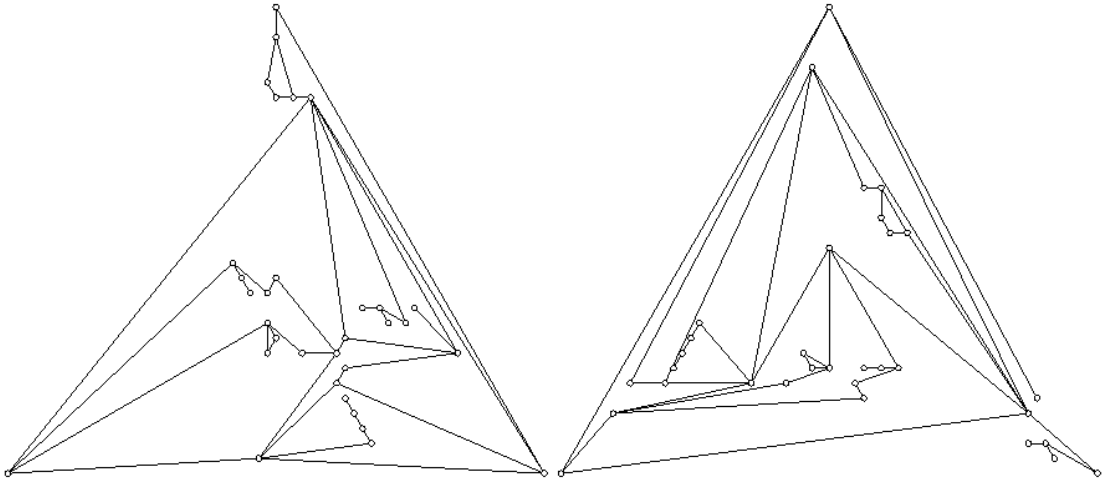
Na následujícím obrázku si pak ukážeme možný rozdíl mezi nejmenšími úhly:



Obrázek 6.4.3 Nakreslení téhož grafu Schnyderovým algoritmem a tímtéž algoritmem s výběrem vnější stěny.

V prvním nakreslení byl nejmenší úhel 2,68° (jde o úhel mezi hranami $\{u,w\}$ a $\{w,v\}$, které na prvním nakreslení splývají), ovšem v nakreslení druhém dosáhl nejmenší úhel velikosti celých 30°! Už sice nejde o úhel mezi zmíněnými hranami, ale to rozhodně nic nemění na tom, že jsme dosáhli změny od nakreslení s vysokým rizikem záměny hran k nakreslení jasně čitelnému (a nyní si troufám tvrdit, že toto nakreslení za čitelné považuji zcela objektivně). 30° totiž není pro člověka problém rozeznat, zatímco 2,68° v dosti malém úseku ano. Výhoda plynoucí z nalezení vhodné vnější stěny je tedy i zde zřejmá.

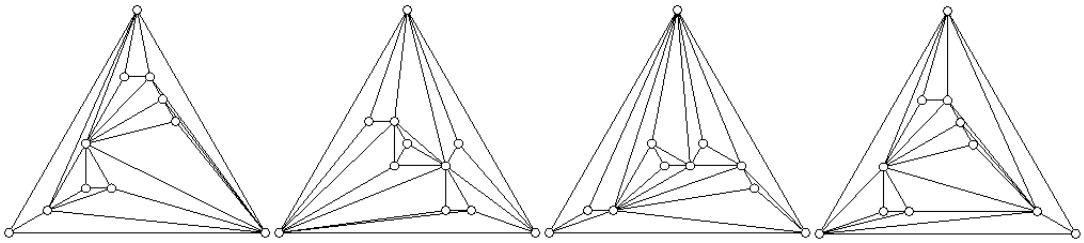
Zbývá ukázat potenciál výběru vnější stěny v oblasti minimalizace zaplněnosti okolí:



Obrázek 6.4.4 Nakreslení téhož grafu Schnyderovým algoritmem a tímtéž algoritmem s výběrem vnější stěny.

I zde získáváme výrazné zlepšení – místo 18 vrcholů v nejzaplněnějším okolí jich dostáváme pouhých devět, tedy pouhou polovinu. To vše navíc bez výrazné změny na součtu délek hran nebo nejmenším úhlu (což ovšem nelze brát jako zákonitost u všech grafů, někdy si s výběrem stěny pro optimalizaci jednoho kritéria ostatní položky výrazně pohorší, někdy naopak dokonce polepší). Proto i tento případ lze považovat za úspěšný.

Výběr vnější stěny nám také dovolí lehce upravit i nakreslení takových grafů, na něž různé druhy triangulace nemají žádný vliv – ano, řeč je o už triangulovaných grafech, na něž triangulace vůbec nemá cenu používat, natož abychom se snažili zvolit tu nejlepší. Ovšem výběr vnější stěny dokáže vylepšit i nakreslení triangulovaných grafů. Sice často jen velmi málo, ale i tak jde o vítanou možnost úpravy daného nakreslení.



Obrázek 6.4.5 Triangulovaný graf na 11 vrcholech vykreslený Schnyderem a Schnyderem optimalizovaným pro délku hran, nejmenší úhel a zaplněnost okolí**.**

Původní výsledek: (122,19; 1°; 5)

Minimalizace součtu délek hran: (117,29; 1°; 6)

Maximalizace nejmenšího úhlu: (118,56; 5,21°; 5)

Minimalizace počtu vrcholů v nejvíce zaplněném okolí: (118,71; 1,94°; 4)

Jak je vidět, pro každé kritérium se nám podařilo dosáhnout určitého zlepšení – samozřejmě ne až tak výrazného, ale to by se u triangulovaného grafu na více

vrcholech díky většímu počtu stěn jistě změnilo. V tomto případě pak za nejvýraznější považuji maximalizaci nejmenšího úhlu, která dokázala celé nakreslení skutečně zpřehlednit.

Výběr vnější stěny pro optimalizaci toho kterého kritéria se ukazuje jako ještě zásadnější úprava Schnyderova algoritmu než možnost výběru triangulace. Máme-li totiž zájem vylepšit nakreslení z hlediska jednoho kritéria, právě výběr vnější stěny často pomůže mnohem víc než pokusy s triangulacemi. To je sice vykoupeno zvýšením časové složitosti, ale ukázali jsme, že toto zvýšení není až tak zásadní, a tak se na dnešních počítačích dočkáme výsledku pro grafy na méně jak 100 vrcholech do jedné minuty, což za často velmi vysoké zlepšení celého nakreslení jistě stojí.

7 Závěr

Vyzkoušeli jsme Schnyderův algoritmus rovinného nakreslení grafů pomocí úseček podpořený algoritmem Demoucron-Malgrange-Pertuisetovým hledajícím rovinné vnoření grafu a zjistili jsme, že i když Schnyderův algoritmus vždy najde správné rovinné nakreslení, často je vhodné vyzkoušet jiný způsob triangulace daného grafu, případně zkusit najít Schnyderův realizátor a z něj vzešlé nakreslení pro každou stěnu rovinného nakreslení daného grafu zvlášť a následně vybrat pro skutečné vykreslení tu, která dává nejlepší výsledek z hlediska zvoleného kritéria. Teprve potom často dostáváme uspokojivé rovinné nakreslení daného grafu.

8 O programu...

Program byl vyvíjen v rámci Ročníkového projektu a Bakalářské práce v průběhu bakalářského studia na MFF UK, a to pod vedením RNDr. Ondřeje Pangráce, Ph.D. Slouží k vizuální prezentaci rovinných nakreslení grafů, zejména pak nakreslení získaných Schnyderovým algoritmem. Pro lepší užívání program obsahuje také jednoduchý grafový editor a možnost archivace grafu.

8.1. Systémové požadavky

Program ke svému spuštění vyžaduje operační systém Windows s nainstalovaným Microsoft .NET Framework 2.0. Byl testován na sestavě AMD Athlon XP 2000+, 768 MB, GeForce 4 Ti 4200, na níž běžel naprosto bezproblémově. Na pevném disku zabere přibližně 1 MB.

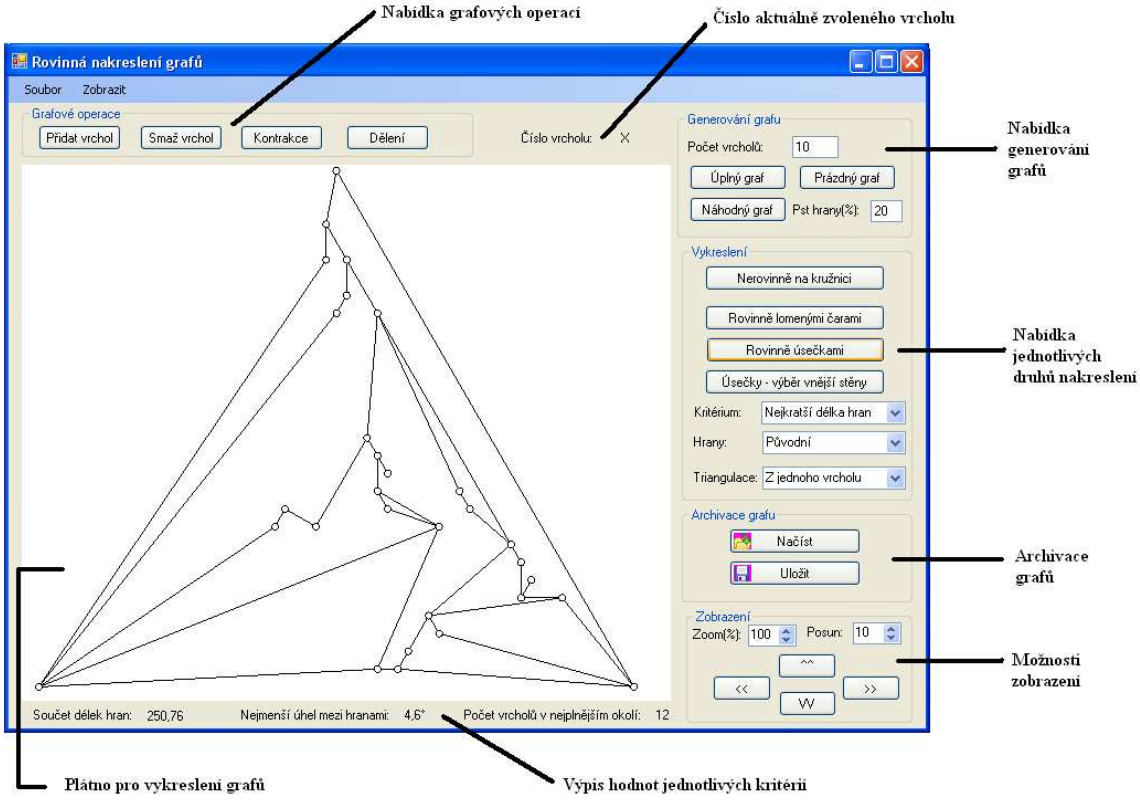
8.2. Instalace a spuštění

Postupujte následujícím způsobem:

1) Vložte CD do mechaniky.
2) Překopírujte složku RoNaGra na Váš pevný disk.
3) Otevřete překopírovanou složku a spusťte soubor RoNaGra.exe.

8.3. Uživatelské rozhraní

Spuštěný program vypadá následovně:



Obrázek 8.3.1 Spuštěný program

Jednotlivé položky pak slouží k následujícím možnostem:

Plátno pro vykreslení grafů: Klíčový prvek celého programu, na němž se zobrazují jednotlivá nakreslení. V jisté míře se díky němu dá vykreslený graf také editovat – kliknutím levým tlačítkem na libovolný vrchol se tento zobrazí červeně. Levým kliknutím na jiný vrchol se pak mezi tímto a červeným přidá hrana, případně odebere, pokud už existovala. Kliknutím levého tlačítka mimo jakýkoli vrchol se pak červený vrchol „odoznačí" – vykreslí se opět s bílou výplní a je možné červeně označit jiný vrchol. Kliknutím pravého tlačítka na libovolný vrchol se tento vrchol obarví žlutě, což využívá následující nabídka.

Nabídka grafových operací: Zde je možné provádět na grafu operace definované v 3.2: Přidání a smazání vrcholu se vždy aplikuje na červeně označený vrchol, kontrakce a dělení hrany pak na hranu mezi červeným a žlutým vrcholem.

Číslo aktuálně označeného vrcholu: Číslo vrcholu, na který uživatel naposledy kliknul.

Nabídka generování grafů: Zde se nachází několik tlačítek sloužících k vytvoření grafu se zadaným počtem vrcholů. Lze tedy vytvořit graf úplný, prázdný (popsané v 2.2) nebo graf náhodný, u nějž se ovšem musí zadat pravděpodobnost hrany – následně se vytvoří prázdný graf, u kterého se pro každou možnou dvojici vrcholů vytvoří hrana právě podle zadané pravděpodobnosti.

Nabídka jednotlivých druhů nakreslení: Odsud je možné vybrat způsob nakreslení, jakým se daný graf má vykreslit. Jakýkoli graf lze vykreslit rovinně na kružnici (resp. více kružnic, pro každou komponentu zvlášť), rovinné grafy pak lze nakreslit lomenými čarami (algoritmem 4.3.1), úsečkami Schnyderovým algoritmem (5.3.1) nebo tím samým algoritmem obohaceným o výběr stěny podle kritéria vybraného v příslušné nabídce. Položka hrany pak slouží k zobrazení triangulovaných hran – buď se zobrazí jen hrany původní, nebo se zobrazí triangulací přidané hrany šedivě, případně lze zobrazit celý triangulovaný graf. V poslední nabídce je pak možno zvolit způsob triangulace ve Schnyderově algoritmu.

Archivace grafů: Tato nabídka poskytuje možnost uložit graf do textového souboru na disk, případně jej z takového souboru načíst. Stačí vždy jen kliknout na danou položku a vybrat cestu pro uložení, resp. načtení.

Možnosti zobrazení: Nabízí možnost přibližovat, oddalovat a posouvat aktuální nakreslení. Položka posun slouží k určení, o kolik bude nakreslení posunuto po kliknutí na jedno z posouvacích tlačítek.

Výpis hodnot jednotlivých kritérií: Zde se vypisují hodnoty jednotlivých kritérií platné pro aktuální nakreslení Schnyderovým algoritmem.

9 Literatura

[1] Kučera, L. (1989): Kombinatorické algoritmy, 152-159. 2. vydání, SNTL, Praha.

[2] Matoušek J., Nešetřil J. (2000): Kapitoly z diskrétní matematiky. 2. vydání, Karolinum, Praha.

[3] Schnyder W. (1990): Embedding Planar Graphs on the Grid. *First ACM-SIAM Symposium on Discrete Algorithms*, 138-147.