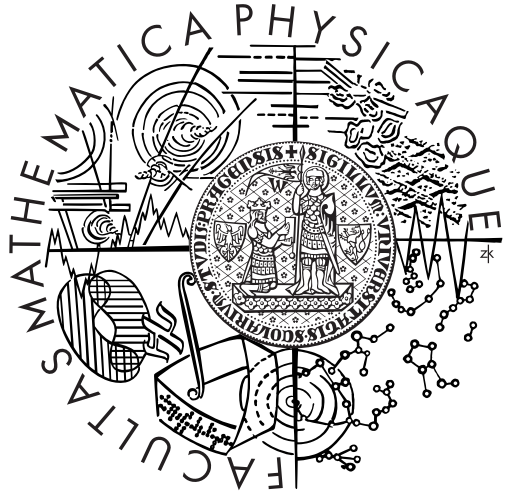
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Pavel Charvát

VÝPOČET TRIANGULACE S MINIMÁLNÍ VÁHOU

Kabinet software a výuky informatiky
Vedoucí diplomové práce: Doc. Dr. Ing. Ivana Kolingerová
Studijní program: Informatika
Studijní obor: Softwarové systémy

Praha 2005

Rád bych na tomto místě poděkoval Doc. Dr. Ing. Ivaně Kolingerové za odborné vedení, cenné rady a veškeré poskytnuté materiály, které mi umožnily vypracovat tuto diplomovou práci.

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

V Praze dne: 12. 12. 2005 .....

Pavel Charvát

Obsah

Název práce: Výpočet triangulace s minimální váhou
Autor: Pavel Charvát
Katedra: Kabinet software a výuky informatiky
Vedoucí diplomové práce: Doc. Dr. Ing. Ivana Kolingerová
e-mail vedoucího: kolinger@kiv.zcu.cz

Abstrakt: Pro výpočet MWT už dlouhou dobu není znám žádný polynomiální algoritmus a ani se neví, zda je NP. Tento stav zůstává podle našich zdrojů stále neznámý. V práci uvádíme přehled možných přístupů k problému, jako jsou modifikace zadání se známou složitostí nebo hledání nejrůznějších heuristik a aproximací, umožňujících v rozumném čase najít přesné nebo alespoň přibližné řešení, a porovnáváme jejich kvalitu v konkrétních situacích. Hlavní částí je popis a implementace efektivní heuristiky s (téměř?) lineární očekávanou složitostí pro rovnoměrně rozložené body v konvexní oblasti. Algoritmus modifikuje Drysdalův algoritmus hledání kandidátních hran GT a Beuroutiho výpočet modifikovaného LMT-skeletonu, u kterého navíc doplňujeme důkazy správnosti. MWT dokončíme z grafu kandidátních hran v $O(n \cdot d^3 + n \cdot d^{2+k})$, kde d je maximální stupeň vrcholu a k je největší počet vnitřních komponent nějaké stěny skeletonu. Dále navrhujeme novou aproximaci s polynomiální složitostí a (téměř?) lineární očekávanou složitostí, která se jen zřídka liší od optimální triangulace, a lze dokázat její nejhorší možný aproximační faktor $O(1)$. Aproximace kombinuje heuristiku modifikovaného LMT-skeletonu s omezenou quasi-greedy triangulací a s triangulací minimální kostry. Minimální triangulace nakonec aplikujeme v praktickém problému výpočtu vrstevnic a stručně porovnáme s nejčastěji používanou Delaunayovou triangulací.

Klíčová slova: triangulace s minimální váhou; výpočetní geometrie; výpočetní složitost

Title: Computation of the Minimum Weight Triangulation
Author: Pavel Charvát
Department: Department of Software and Computer Science Education
Supervisor: Doc. Dr. Ing. Ivana Kolingerová
Supervisor's e-mail address: kolinger@kiv.zcu.cz

Abstract: For a long time, it has been neither known whether MWT is solvable in a polynomial time nor whether it belongs to NP. As we now, its status still remain unknown. We present several known approaches to MWT such as modifications of the problem with known time complexity or various heuristics and approximations which allow us to find an exact or at least an approximate solution in a reasonable time. We compare the approximations in some particular situations. The main part of the work is devoted to a description and implementation of an efficient heuristic with (almost?) linear expected complexity for points uniformly distributed in some convex shape. The algorithm is a modification of Drysdale's algorithm for finding GT candidates and Beurouti's computation of the modified LMT-skeleton, where we add some proofs of the correctness. We are able to complete MWT from the graph of candidate edges in $O(n \cdot d^3 + n \cdot d^{2+k})$, where d is the the maximum degree and k is the maximum number of inner components of some skeleton face. Further, we suggest a new approximation of MWT with polynomial complexity in the worst case and (almost?) linear expected complexity, which only rarely differs from the optimal triangulation and has $O(1)$ approximation factor in the worst case. This approximation combines the LMT-skeleton heuristic with constrained quasi-greedy triangulation and with the triangulation of the minimum spanning tree. Finally, we apply MWT in a practical problem of computation of contour lines and compare with the Delaunay triangulations, most frequently used for this problem.

Keywords: minimum weight triangulation; computational geometry; computational complexity

Kapitola 1

Úvod

Tématem této diplomové práce je studium rovinných triangulací, které se používají nejen v matematice a kombinatorice, například pro interpolaci funkcí nebo jako aproximace problému obchodního cestujícího, ale mají i široké praktické uplatnění při rekonstrukci povrchu z množin bodů, vizualizaci dat v počítačové grafice nebo při simulaci dějů ve fyzice.

Existují mnohá optimalizační kritéria, kterými můžeme odlišit kvalitu triangulací v různých konkrétních situacích. V této práci se zaměříme na triangulace s co nejmenší celkovou délkou hran, tzv. minimální triangulace nebo triangulace s minimální váhou. Ačkoli vypadá toto kritérium velmi jednoduše, problém minimální triangulace pro obecnou množinu bodů odolává už od roku 1964, kdy byl v souvislosti s metodou konečných prvků poprvé uveden [9], snaze nalézt polynomiální algoritmus a ani se neví, zda patří do NP.

Úkolem pro diplomovou práci bylo prostudovat současný stav problematiky minimálních triangulací, navrhnout vhodný postup pro přibližné nebo přesné řešení a porovnat jej na vhodných datech a v některé konkrétní úloze s jinými známými přístupy.

V podstatě existují tři základní metody, jak k problému přistupovat. Prvním je složitost v nejhorším možném případě, která, jak již bylo zmíněno, zůstává otevřená. V kapitole 2 se podíváme na mírné modifikace problému s prokázanou polynomiální složitostí nebo naopak úzce související NP-úplné problémy.

Dalším možným přístupem je hledání triangulací, které nemusí být nutně optimální, ale většinou se příliš neliší. Nazýváme je aproximacemi minimální triangulace a mezi nejznámější patří například dobře prozkoumaná Delaunayova triangulace nebo greedy triangulace. Podrobněji se na aproximace zaměříme v kapitole 3. Jejich „kvalitu“ pak ověříme při praktických testech na náhodně generovaných i reálných množinách bodů v kapitole 6.

Do poslední skupiny algoritmů patří různé heuristiky, které vždy najdou minimální triangulaci a ve většině případů zrychlí jinak exponenciální výpočet na rozumnou dobu. Tomuto přístupu se věnuje kapitola 4.

Kapitola 5 podrobně, po jednotlivých krocích, popisuje kompletní algoritmus pro výpočet triangulace s minimální váhou optimalizovaný zejména pro rovnoměrné roz-

ložení bodů v konvexní oblasti, který umožňuje počítat přesnou minimální triangulaci až do statisíců bodů. Téměř kompletní implementace toto algoritmu je součástí programu přiloženého k diplomové práci, společně s algoritmy pro výpočet testovaných aproximací.

V další části textu (kapitola 6) porovnáme, jak už bylo zmíněno, kvalitu různých aproximací na několika pravděpodobnostních rozděleních vstupních množin bodů a také na datech pocházejících ze skutečných zdrojů. Jako kombinaci známých přístupů navrhneme novou aproximaci, která se ve většině případů shoduje s minimální triangulací, a i v nejhorším případě lze dokázat $O(1)$ aproximační faktor. Téměř lineární očekávanou složitost minimální triangulace i navržené aproximace experimentálně ověříme na rovnoměrně rozložených množinách bodů ve čtverci.

Minimální triangulace zkusíme v kapitole 7 použít jako vstup pro úlohu výpočet vrstevnic a stručně porovnáme rozdíl oproti tradičně používané Delaunayově triangulaci.

Na závěr shrneme výsledky vyplývající z textu a experimentů.

1.1 Základní značení

Poznámka 1.1. Pokud v textu není přímo u některého algoritmu nebo důkazu věty uvedený zdroj, pak byl odvozený samostatně, přestože základní myšlenka může být inspirovaná z jinak uvedených pramenů. Z důvodu hlubšího pochopení problému jsem se ve většině případů snažil upřednostnit tento postup. Podobně je samostatně formulována i většina uvedených definic.

Následující definice z teorie grafů vychází z přednášek na Matematicko-fyzikální fakultě university Karlovy.

Definice 1.1 (Grafy)**.** Neorientovaný graf G je každá uspořádaná dvojice $G = (V, E)$, kde V je nějaká konečná množina a $E \subseteq \binom{V}{2}$. Prvky množiny V nazýváme vrcholy a prvky množiny E hrany. Je-li $e \in E$ hrana grafu, pak $e = \{u, v\}$, kde $u, v \in V \wedge u \neq v$. Dále značíme $V(G) = V$ a $E(G) = E$.

Tahem v grafu nazveme posloupnost vrcholů $(v_0, \ldots, v_n)$, kde každé dva sousední jsou spojeny hranou, tj. $v_i \in V$ a $\{v_i, v_{i+1}\} \in E$, cestou pak tah s navzájem různými vrcholy. Řekneme, že G je souvislý, jestliže mezi každými dvěma vrcholy vede cesta. Komponentou grafu je každý maximální souvislý podgraf.

Nakreslením grafu $G = (V, E)$ rozumíme dvojici zobrazení (b, o), kde b přiřadí vrcholům různé body v rovině a o přiřadí každé hraně $e = \{u, v\} \in E$ oblouk spojující body $b(u)$ a $b(v)$, který neprochází žádným jiným vrcholem $b(w)$.

Nakreslení (b, o) grafu $G = (V, E)$ je rovinné, jestliže žádná dvojice hran nemá společný bod kromě společného krajního vrcholu, tj. $\forall e_1, e_2 \in E, e_1 \neq e_2 : o(e_1) \cap o(e_2) = b(e_1 \cap e_2)$.

Topologický graf je čtveřice $G = (V, E, b, o)$, kde (b, o) je nakreslením grafu (V, E). Nechť $G = (V, E, b, o)$ je topologický graf s rovinným nakreslením a $H = \bigcup_{e \in E} o(e)$.

Stěnou grafu G nazveme každou maximální souvislou podmnožinu $\mathbb{R}^2 \setminus H$. Topologický graf má vždy právě jednu nekonečnou stěnu, kterou nazveme vnější stěnou.

Triangulace raději zavedeme formálně, zejména proto, aby byly jasně definované situace, kdy se více bodů nachází na jedné přímce.

Definice 1.2 (SLG, PSLG)**.** Zobrazení, které každé neuspořádané dvojici vrcholů a, b přiřadí úsečku $\{(1-t)\cdot a+t\cdot b; t \in [0,1]\}$, označíme seg. Nechť $G = (V, E)$ je uspořádaná dvojice a $V \subseteq \mathbb{R}^2$. Jestliže $(V, E, id|_V, seg|_E)$[1] je topologický graf, pak G nazveme grafem s rovnými hranami – SLG. Jestliže je navíc nakreslení $(id|_V, seg|_E)$ grafu G rovinné, graf nazveme PSLG (planar straight line graph).

Definice 1.3 (Diagonála)**.** Diagonálou v PSLG $G = (V, E)$ je každá dvojice různých vrcholů $e \in \binom{V}{2} \setminus E$ taková, že $(V, E \cup \{e\})$ je PSLG. Množinu všech diagonál na G značíme $D(G)$, množinu všech diagonál na $(V, \emptyset)$ označíme $D(V)$. Diagonály také někdy nazýváme prázdnými hranami.

Definice 1.4 (Triangulace)**.** Nechť $V \subseteq \mathbb{R}^2$ je konečná množina, $G_G = (V, C_G)$ je SLG a $G_C = (V, C_C)$ je PSLG. Pak značíme:

$$T(V) = \{(V, E); (V, E) \text{ je PSLG bez diagonály}\}, \tag{1.1}$$

$$T_G(V, E_G) = \{(V, E) \in T(V); E \subseteq E_G\}, \tag{1.2}$$

$$T_C(V, E_C) = \{(V, E) \in T(V); E_C \subseteq E\}. \tag{1.3}$$

Prvky množiny $T(V)$ nazýváme triangulacemi na V a prvky $T_C(V)$ omezenými triangulacemi na G_C. Třídu všech triangulací značíme $\mathcal{T} = \bigcup_V T(V)$.

Z definic snadno vyplývají následující vlastnosti triangulací.

Věta 1.1 (Vlastnosti triangulací)**.**

(i) Každá vnitřní stěna triangulace má tvar trojúhelníku a na hranici obsahuje právě tři vrcholy.

(ii) Každý PSLG, který není triangulací, má alespoň jednu diagonálu.

(iii) Každý PSLG je podgrafem nějaké triangulace.

(iv) Pro každou neprázdnou konečnou množinu bodů V je $T(V) \neq \emptyset$.

Definice 1.5 (Váha)**.** Pro Euklidovskou váhu hran a grafů s rovnými hranami zavedeme následující značení:

$$\|(x, y)\| = \sqrt{x^2 + y^2}, \qquad x, y \in \mathbb{R}, \tag{1.4}$$

$$\|\{u, v\}\| = \|u - v\|, \qquad u, v \in \mathbb{R}^2, \tag{1.5}$$

$$\|E\| = \sum_{e \in E} \|e\|, \qquad E \subseteq 2^{\mathbb{R}^2}, \tag{1.6}$$

$$\|G\| = \|E(G)\|, \qquad G \text{ je SLG}, \tag{1.7}$$

$$\|M\|_{\min} = \min\{\|G\|; G \in M\}, \quad M = \{G_1, \ldots, G_n\}, G_i \text{ je SLG}. \tag{1.8}$$

[1] Výraz $f|_A$ značí zúžení zobrazení f na množinou A.

Definice 1.6 (Minimální triangulace)**.** Nechť $V \subseteq \mathbb{R}^2$ je konečná množina, $G_G = (V, E_G)$ SLG a $G_C = (V, E_C)$ PSLG. Pak značíme:

$$MWT(V) = \{T \in T(V); \|T\| = \|T(V)\|_{\min}\}, \tag{1.9}$$
$$GMWT(V, E_G) = \{T \in T_G(V, E_G); \|T\| = \|T_G(V, E_G)\|_{\min}\}, \tag{1.10}$$
$$CMWT(V, E_C) = \{T \in T_C(V, E_C); \|T\| = \|T_C(V, E_C)\|_{\min}\}. \tag{1.11}$$

Prvky množiny $MWT(V)$ nazýváme triangulacemi s minimální váhou (MWT) nebo minimálními triangulacemi na V, prvky $GMWT(V, E_G)$ zobecněnými triangulacemi s minimální váhou ($GMWT$) na G_G a prvky $CMWT(V, E_C)$ omezenými triangulacemi s minimální váhou ($CMWT$) na G_C.

Souvislosti mezi definovanými třídami minimálních triangulací shrnuje následující věta.

Věta 1.2 (Vlastnosti tříd minimálních triangulací)**.**

(i) $MWT(V) \neq \emptyset$,

(ii) $MWT(V) = CMWT(V, \emptyset) = GMWT(V, D(V))$,

(iii) $CMWT(V, E_C) = GMWT(V, E_C \cup D((V, E_C)))$,

(iv) $MWT(V) \subseteq \bigcup_{E_C} CMWT(V, E_C) \subseteq \bigcup_{E_G} GMWT(V, E_G)$.

Definice 1.7 (Konvexní obal)**.** Nechť $V = (v_1, \ldots, v_n) \subseteq \mathbb{R}^2$ a

$$H = \partial \left\{ \sum_{i=1}^{n} c_i v_i; c_i \in [0, 1] \wedge \sum_{i=1}^{n} c_i = 1 \right\} \tag{1.12}$$

je hranice konvexního uzávěru V. Potom konvexním obalem množiny V budeme nazývat PSLG, který je definován následovně:

$$CH(V) = \left(V \bigcap H, \{e \in D(V); seg(e) \subseteq H\} \right). \tag{1.13}$$

Věta 1.3 (Vlastnosti konvexního obalu)**.**

(i) Jestliže všechny body $V \neq \emptyset$ leží na společné přímce, potom $V(CH(V)) = V$, $|E(CH(V))| = |V| - 1$, $CH(V)$ obsahuje jedinou stěnu a platí $T(V) = \{CH(V)\}$.

(ii) Jestliže v množině V existuje trojice bodů, které neleží na společné přímce, potom $|V(CH(V))| = |E(CH(V))|$ a konvexní obal má právě dvě stěny, přičemž vnitřní stěna obsahuje všechny vrcholy V. Pro každou triangulaci $T \in T(V)$ je konvexní obal podgrafem T a $|E(T)| = 3|V| - |V(CH(V))| - 3 = O(|V|)$.

Důkaz: Snadno jde dokázat indukcí. □

Kapitola 2

Složitost problému

Problém nalezení $T \in MWT(V)$ pro množinu bodů $V = \{v_1, v_2, \ldots, v_n\}$ v polynomiálním čase, zůstává už od svého uvedení roku 1964 stále otevřený. Existuje řada heuristik pro hledání optimální triangulace, které dobře fungují pro speciálně rozložené body, ale nikdy ne pro všechny možné vstupy.

Pokud nám stačí jen triangulace dostatečně blízko minimální, nejlepšího asymptotického výsledku dosahují tzv. quasi-greedy triangulace (viz. kapitola 3.3). Tuto aproximaci MWT jde spočítat v polynomiálním čase $O(n \log n)$ pro libovolný vstup a má váhu $O(\|MWT(V)\|)$.

Složitost výpočtu triangulace z libovolné neprázdné třídy lze, podobně jako hledání konvexního obalu množiny bodů, zdola omezit výrazem $\Omega(n \log n)$, protože konvexním obalem či triangulací snadno setřídíme posloupnost čísel.

2.1 Obecný případ

Následuje formální definice několika rozhodovacích problémů, které včetně důkazů složitosti uvádí L. S. Heath a S. V. Pammaraju [13], a které úzce souvisí s minimální triangulací.

Problém 2.1 (MWT [13])**.**

Instance: *Konečná množina* $V \subset \mathbb{Q}^2$ *a kladné* $k \in \mathbb{Q}$*.*

Dotaz: *Existuje triangulace* $T \in T(V)$ *taková, že* $\|T\| \leqq k$*?*

Jedná se o rozhodovací variantu výpočtu MWT. Složitost zůstává otevřená. Zřejmě námi řešený problém nalezení triangulace $T \in MWT(V)$ pro konkrétní konečnou množinu $V \subseteq \mathbb{R}^2$ nemůže mít nižší složitost, protože rozhodovací problém MWT jde řešit výpočtem minimální triangulace.

Na tomto místě je ještě vhodné zmínit rozdíl mezi reálnými a racionálními množinami bodů. Přestože reálná čísla nejde obecně reprezentovat do konečné paměti, budeme u zápisu algoritmů uvádět reálné body a budeme předpokládat konstantní paměťovou

a časovou složitost všech operací, včetně netriviálního porovnávání sumy odmocnin. V konkrétní implementaci typicky na vstupu povolíme jen nějakou podmnožinu $\mathbb{R}^2$, například $\mathbb{Q}^2$.

Problém 2.2 (TRI [13])**.**

Instance: *Konečná množina* $V \subset \mathbb{Q}^2$ *a* $E \subseteq \binom{V}{2}$.

Dotaz: *Existuje* $E' \subseteq E$*, že* (V, E') *je triangulace?*

Problém existence triangulace v dané množině hran řeší SAT a je NP-úplný.

Definice 2.1. Nechť $V \subset \mathbb{R}^2$ je konečná množina bodů v rovině $E = E(V)$. Grafem průsečíků na V nazveme graf

$$G = \left(E, \{e_i, e_j\} \in \binom{E}{2}; e_i \text{ protíná } e_j\right). \tag{2.1}$$

Problém 2.3 (RMMI [13])**.**

Instance: *Graf průsečíků* $G = (E, C)$ *na* V*, pro každou hranu* $e \in E$ *váha* $w(e) \in \mathbb{Q}$ *a kladné* $k \in \mathbb{Q}$.

Dotaz: *Obsahuje* G *maximální nezávislou množinu* I *takovou, že* $\sum_{e \in I} w(e) \leqq q$*?*

Tento problém zobecňující MWT na libovolné ohodnocení hran řeší TRI a je NP-úplný.

Problém 2.4 (GMWT [13])**.**

Instance: *Konečná množina* $V \subset \mathbb{Q}^2$*,* $E \subseteq \binom{V}{2}$ *obsahující alespoň jednu triangulaci a kladné* $k \in \mathbb{Q}$

Dotaz: *Existuje* $E' \subseteq E$*, že* $T = (V, E')$ *je triangulace a* $\|T\| \leqq k$*?*

Problém MWT zobecněný na danou podmnožinu povolených hran je NP-úplný, jak lze dokázat redukcí TRI na GMWT.

Z TRI a GMWT přímo vyplývá, že v námi zavedeném značení je pro SLG (V, E_G) výpočet $T \in GMWT(V, E_G)$ v NP, a to i když (V, E_G) obsahuje alespoň jednu triangulaci.

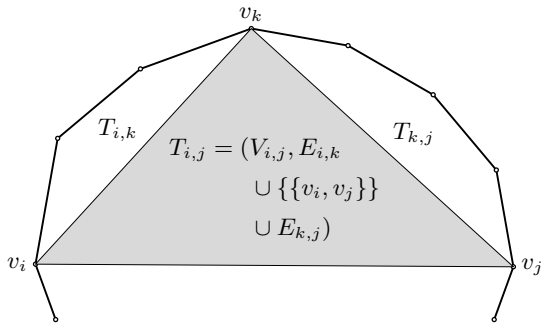
2.2 Konvexní polygon

Naštěstí ne všechny případy jsou pro hledání MWT tak těžké jako obecná množina bodů. Jednou ze speciálních vstupních množin jsou body rozložené na svém vlastním konvexním obalu (tzv. konvexní polygon). Pro tento případ lze řešit $T \in MWT(V)$ v polynomiálním čase $O(n^3)$, jak ukážeme na následujícím algoritmu.

Algoritmus 1 MWT konvexního polygonu

Vstup: $V \subseteq \mathbb{R}^2$ konečná množina, $V = V(CH(V))$.
Výstup: $(V, E) \in MWT(V)$

1: $(V, E), (V, E) \leftarrow CH(V)$
2: **if** všechny body V leží na jedné přímce **then**
3: **return** ▷ $CH(V) \in MWT(V)$
4: **end if**
5: $(v_1, v_2, \ldots, v_n) \leftarrow$ vrcholy $CH(V)$ po směru hodinových ručiček
6: **for** $j = 2, 3, \ldots, n$ **do**
7: $w_{j-1,j} \leftarrow \|v_{j-1} - v_j\|$ ▷ Inicializace hrany na konvexním obalu
8: **for** $i = j-2, j-3, \ldots, 1$ **do** ▷ Průchod ostatními hranami
9: $w_{i,j} \leftarrow \infty$
10: **if** $\{v_i, v_j\} \in D(V)$ **then**
11: **for** $k = i+1, i+2, \ldots, j-1$ **do**
▷ Průchod trojúhelníky nalevo od (v_i, v_j)
12: **if** $(w_{i,k} + w_{k,j} < \infty) \wedge (w_{i,j} > w_{i,k} + w_{k,j})$ **then**
▷ Jestliže je tvořen diagonálami
▷ a je nejlepší dosud nalezený,
13: $w_{i,j} \leftarrow w_{i,k} + w_{k,j}$ ▷ uložíme váhu
14: $k_{i,j} \leftarrow k$ ▷ a vrchol pro konečnou rekonstrukci
15: **end if**
16: **end for**
17: **end if**
18: $w_{i,j} \leftarrow w_{i,j} + \|v_i - v_j\|$
19: **end for**
20: **end for**
21: $L \leftarrow \{(1, n)\}$
22: **while** $(i, j) \in L$ **do** ▷ Rekonstrukce nejmenší nalezené triangulace
23: $L \leftarrow L \setminus \{(i, j)\}$
24: **if** $(w_{i,j} < \infty) \wedge (i + 1 < j)$ **then**
25: $E \leftarrow E \cup \{\{v_i, v_j\}\}$
26: $L \leftarrow L \cup \{(i, k_{i,j}), (k_{i,j}, j)\}$
27: **end if**
28: **end while**



Obrázek 2.1: Hledání MWT v konvexním polygonu.

Poznámka 2.1. Algoritmus jsem odvodil samostatně, přestože se pravděpodobně podobá některému z obecně známých algoritmů. V další části kapitoly výpočet i důkazy zobecníme na obecnější případ jednoduchého polygonu.

Věta 2.1. *Algoritmus 1 najde pro každou vstupní množinu bodů $V = V(CH(V))$ triangulaci s minimální váhou.*

Důkaz: Bez újmy na obecnosti můžeme předpokládat, že body V neleží na jedné přímce, jinak $MWT(V) = \{CH(V)\}$. Tuto variantu ošetřuje řádek (2). Jestliže body neleží na přímce, snadno očíslujeme vstupní body průchodem po hranách $CH(V)$. Označme $V_{i,j} = \{v_i, v_{i+1}, \ldots, v_j\}$, $i < j$.

Program funguje na principu dynamického programování. Na řádcích (6)–(20) se pro všechna $i < j$ s využitím už dříve spočítaných intervalů najde, pokud existuje,

$$T_{i,j} \in GMWT\left(V_{i,j}, D(V) \bigcap \binom{V_{i,j}}{2}\right). \tag{2.2}$$

Kdybychom věděli, že žádná trojice bodů neleží na přímce, šel by algoritmus zjednodušit a vynechat testování hran na diagonály $D(V)$.

Aby se v paměti nemusely udržovat celé triangulace $T_{i,j}$, zaznamená se pro každé $i + 1 < j$ jen index $i < k_{i,j} < j$ vrcholu $v_{k_{i,j}}$, který v $T_{i,j}$ tvoří trojúhelník $(v_i, v_j, v_{k_{i,j}})$, a váha $w_{i,j} = \|T_{i,j}\|$. Když pro $i < j$ neexistuje $T_{i,j}$, bude $w_{i,j} = \infty$.

Krajní hodnoty $w_{i,i+1}$ pro dvoubodové triangulace se nastaví v (7) ($k_{i,i+1}$ není potřeba). Zřejmě hrany $CH(V)$ jsou v $D(V)$ a proto $w_{i,i+1} < \infty$.

Nechť je program na řádku (9) a $i + 1 < j$. Díky pořadí a směru zpracování cyklů pro i a j můžeme předpokládat, že jsme už dříve spočítali všechny hodnoty $w_{i,k}$, $w_{k,j}$, $k_{i,k}$ a $k_{k,j}$ pro $i < k < j$. Je-li $\{v_i, v_j\} \notin D(V)$, zřejmě $w_{i,j} = \infty$. V opačném případě $D(V_{i,j}) = D(V) \bigcap \binom{V_{i,j}}{2}$, tedy musí existovat nějaká $T'_{i,j} \in GMWT\left(V_{i,j}, D(V) \bigcap \binom{V_{i,j}}{2}\right)$ s hranou $\{v_i, v_j\}$, a jako každá jiná triangulace obsahuje i $T'_{i,j}$ trojúhelník (v_i, v_j, v_k), $i < k < j$ u hrany $\{v_i, v_j\}$. V cyklu (11) tento trojúhelník prozkoumáme, a algoritmus proto najde $T_{i,j}$, $\|T_{i,j}\| = \|T'_{i,j}\|$. Pro úplnost důkazu je potřeba zmínit, že podmínkou (12) projdou jen trojúhelníky z diagonál.

Tím jsme dokázali, že na řádku (20) jsou správně spočítané všechny hodnoty $w_{i,j}$ pro $1 \leqq i < j \leqq n$ a $k_{i,j}$ pro $1 < i+1 < j \leqq n$. Zbytek výpočtu je snadný. Triangulace $T_{1,n} \in GMWT(V, D(V)) = MWT(V)$ a dá se rekonstruovat průchodem binárního stromu, který vznikne postupným dělením intervalu $(1, n)$ indexy $k_{i,j}$, a to přesně dělá část programu (21)–(28). □

Věta 2.2. *Algoritmus 1 jde implementovat v čase $O(n^3)$ a s paměťovou složitostí $O(n^2)$.*

Důkaz: Složitost výpočtu konvexního obalu je $O(n \log n)$, (2)–(5) a (21)–(28) jde snadno implementovat v $O(n)$. Každý z cyklů (6), (8) a (11) se provede nejvýše n krát. Test dvojice bodů na diagonálu V jde provést v čase $O(n)$, čímž získáváme celkovou složitost algoritmu $O(n^3)$.

Kromě grafu a pomocných proměnných se v paměti se udržují jen pole $w_{i,j}$, $k_{i,j}$ a spojový seznam L o celkové velikosti $O(n^2)$. □

2.3 Jednoduchý polygon

Polynomiální algoritmus 1 pro počítání MWT konvexního polygonu je možné rozšířit na obecnější případ tzv. jednoduchého polygonu. Nejprve zavedeme několik pojmů.

Definice 2.2 (Orientovaná hrana)**.** Pro SLG $G = (V, E)$ značíme

$$S(E) = \{(u, v); \{u, v\} \in E\} \tag{2.3}$$

a prvky této množiny nazýváme orientované hrany v G. Pro jednodušší zápis algoritmů na G dále zavedeme značení:

$$(u, v)^* = (v, u), \quad u, v \in V, u \neq v, \tag{2.4}$$

$$(u, v).e = \{u, v\}, \quad u, v \in V, u \neq v, \tag{2.5}$$

$$(u, v).v = v, \quad u, v \in V, u \neq v, \tag{2.6}$$

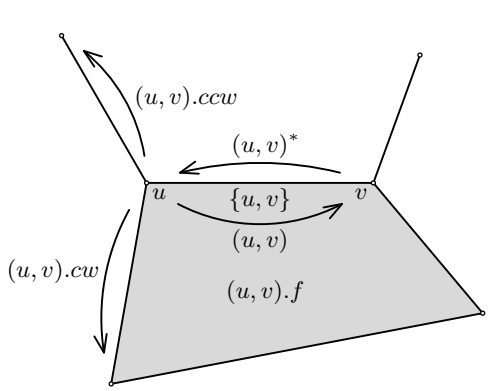
$$(u, v).cw = (u, w), \quad \{u, v\} \in E \text{ a } \{u, w\} \text{ je první hranou ve směru hodinových ručiček kolem vrcholu } u \text{ v } G, \tag{2.7}$$

$$(u, v).ccw = (u, w), \quad \{u, v\} \in E \text{ a } \{u, w\} \text{ je první hranou proti směru hodinových ručiček kolem vrcholu } u \text{ v } G. \tag{2.8}$$

Jestliže je navíc G PSLG:

$$(u, v).f = f, \quad \{u, v\} \in E \text{ a } f \text{ je stěna } G \text{ napravo od orientované hrany } (u, v). \tag{2.9}$$

Značení graficky znázorňuje obrázek 2.2.



Obrázek 2.2: Definice orientované hrany v PSLG.

Poznámka 2.2. U PSLG se většinou uvažují odkazy na sousední hrany ve spojových seznamech kolem hranic stěn. Aby šla definice použít i pro obecnější SLG, zavedl jsem *cw* a *ccw* nezávisle na rovinnosti grafu. Definice *cw* a *ccw* je jednoznačná, protože SLG může obsahovat jedině diagonály $D(V)$, tedy dvojice hran vychází z krajního bodu vždy pod rozdílným úhlem.

Definice 2.3 (Triangulace stěny)**.** Nechť f je stěna nějakého PSLG $G = (V, E)$. Triangulací stěny f v G rozumíme každý PSLG vzniklý z G přidáním diagonál z f, a který v f nemá žádnou další diagonálu. Množinu všech triangulací stěny f v G značíme $T(G, f)$ a množinu všech diagonál G procházejících danou stěnou $D(G, f)$. Podobně jako v definicích pro obyčejné triangulace zavedeme $T_G(G, f, E_G)$, $T_C(G, f, E_C)$:

$$T_G(G, f, E_G) = \{T \in T(G, f); E(T) \subseteq E(G) \cup E_G\}, \quad E_G \subseteq D(G, f), \tag{2.10}$$

$$T_C(G, f, E_C) = \{T \in T(G, f); E_C \subseteq E(T)\}, \quad E_C \subseteq D(G, f). \tag{2.11}$$

Definice 2.4 (Jednoduchý polygon)**.** Jednoduchým polygonem v PSLG $G = (V, E)$ nazveme každou omezenou stěnu G se spojitou hranicí. Obvodem jednoduchého polygonu f v G nazveme libovolný uzavřený tah $(v_0, v_1, \ldots, v_m = v_0)$ po vrcholech $V \cap \overline{f}$, pro který $s_i = (v_i, v_{i+1}) \in S(E)$, $s_{i+1} = s_i^*.ccw$, $s_0.f = f$ cyklicky pro všechna i a každou orientovanou hranu obsahuje nejvýše jednou. (tj. tah po směru hodinových ručiček kolem f).

Věta 2.3. *Pro* $(v_0, v_1, \ldots, v_m = v_0)$ *obvod jednoduchého polygonu* f *platí, že* $|f \cap V| \leqq m \leqq 2|f \cap V|$.

Důkaz: Obvod polygonu musí obsahovat všechny vrcholy na hranici f, tedy $|f \cap V| \leqq m$. Druhá nerovnost vyplývá z toho, že se orientované hrany na obvodu nesmí opakovat a že mezi každou dvojicí vrcholů existují nejvýše dvě orientované hrany. □

Nyní již máme všechny potřebné definice pro popis výpočtu minimální triangulace jednoduchého polygonu. Vstupem bude nějaký PSLG G a jednoduchý polygon f v G,

výstupem pak minimální triangulace f v G. Historicky tuto úlohu vyřešili v $O(n^3)$ nezávisle na sobě P. D. Gilbert [12] a G. T. Klincsek [18].

Místo toho, abychom popsali jen výpočet minimální $T \in T(G, f)$, vyřešíme v polynomiálním čase rovnou obecnější problém, totiž nalezení minimální $T \in T_G(G, f, E_G)$, pokud existuje. O tomto problému z kapitoly 2.1 víme, že je pro obecnou vstupní množinu bodů (netvořících jednoduchý polygon) NP-úplný.

Věta 2.4. *Pro libovolný vstupní PSLG $G_{IN} = (V, E_{IN})$, $s_f \in S(E_{in})$, $f = s_f.f$ jednoduchý polygon v G_{IN} a množinu povolených diagonál $E_G \subseteq D(G_{IN}, f)$, vrátí algoritmus 2 false právě když $T_G(G_{IN}, f, E_G) = \emptyset$, jinak skončí s návratovou hodnotou true a výsledná triangulace polygonu (V, E_{OUT}) je minimální v $T_G(G_{IN}, f, E_G)$.*

Důkaz: Když porovnáme algoritmus 2 s algoritmem 1, zjistíme, že jsou až na několik řádků stejné, a i důkaz bude podobný.

Na řádku (1) lze najít obrys jednoduchého polygonu f průchodem po orientovaných hranách G_{IN}, počínaje hranou e_f. Zřejmě počet orientovaných hran na obvodu je $n \geqq 3$.

Označme $e_{i,j} = \{v_i, v_j\}$ a pro $e_{i,j} \in E_G, i < j$ (a navíc pro $(i, j) = (1, n)$) PSLG $G_{i,j} = (V, E_{IN} \cup e_{i,j})$ se stěnou $f_{i,j} = (v_j, v_i).f$. Podobně jako v algoritmu 2 generujeme (pokud existují) pro všechny definované $G_{i,j}$ s využitím už dříve spočítaných intervalů minimální

$$T_{i,j} \in M_{i,j} = T_G\left(G_{i,j}, f_{i,j}, E_G \bigcap D(G_{i,j}, f_{i,j}))\right). \tag{2.12}$$

Význam $k_{i,j}$ je analogický jako v minulém algoritmu, tj. index vrcholu $v_{k_{i,j}}$ trojúhelníku $(v_i, v_j, v_{k_{i,j}})$ nalevo od (v_i, v_j) v $T_{i,j}$.

$w_{i,j}$ představuje součet váhy všech hran $T_{i,j}$ přidaných do $G_{i,j}$. Pro $e_{i,j}$ hrany na obvodu f kromě poslední definujeme $w_{i,j} = 0$ a zbylé $w_{i,j}$ (s nedefinovaným $T_{i,j}$) budou mít hodnotu ∞.

Krajní hodnoty $w_{i,i+1}$ pro sousední dvojice vrcholů na obvodu se nastaví v (3).

Nechť je program na řádku (5) a $i + 1 < j$. Díky pořadí a směru zpracování cyklů pro i a j můžeme předpokládat, že jsme už dříve spočítali všechny hodnoty $w_{i,k}$, $w_{k,j}$, $k_{i,k}$ a $k_{k,j}$ pro $i < k < j$.

Nejprve předpokládejme $(i, j) \neq (1, n)$. Potom pro $e_{i,j} \notin E_G$ je $M_{i,j} = \emptyset$ a algoritmus nastaví správně $w_{i,j} = \infty$. Uvažujme libovolnou $T'_{i,j} \in M_{i,j}$. Tato triangulace jistě obsahuje nějaký trojúhelník $(v_i, v_j, v_k), i < k < j$ vlevo od orientované hrany (v_i, v_j) a $w_{i,k}, w_{k,j} < \infty$. Prvními dvěmi podmínkami na řádku (8) ale projdou všechny takové trojúhelníky a proto po skončení cyklu na (12) má $w_{i,j}$ rovnou nebo menší, než jak jsme ji definovali. Dále je zřejmé, že pro každý nalezený trojúhelník v (8) lze sestavit triangulaci $f_{i,j}$ v $G_{i,j}$ a proto $w_{i,j}$ a $k_{i,j}$ nabudou po provedení (14) právě námi definovaných hodnot pro nějaký minimální $T_{i,j}$.

Pro $(i, j) = (1, n)$ lze uvažovat analogicky, z předchozího odstavce jsme tuto možnost vyloučili jen kvůli jednoduššímu značení.

Když program na (16) ukončí hlavní cyklus, je v $w_{1,n}$ a $k_{1,n}$ uložena nějaká minimální triangulace $T_{1,n} \in M_{1,n} = T_G(G_{IN}, f, E_G)$, a když neexistuje, platí $w_{1,n} = \infty$. Zbylá část algoritmu rekonstruuje $T_{1,n}$ z hodnot $k_{i,j}$ stejně jako v algoritmu 1. □

Algoritmus 2 *GMWT* jednoduchého polygonu

Vstup: $G = (V, E)$ je PSLG, $s_f \in S(E)$,
$f = s_f.f$ jednoduchý polygon v G, $E_G \subseteq D(G, f)$.
Výstup: Vrátí false pro $T_G(G, f) = \emptyset$, jinak minimální $(V, E) \in T_G(G, f, E_G)$.

1: $(v_0, v_1, v_2, \ldots, v_n) \leftarrow$ obvod jednoduchého polygonu f v G
2: **for** $j = 2, 3, \ldots, n$ **do**
3: $w_{j-1,j} \leftarrow 0$ ▷ Inicializace hrany na obvodu
4: **for** $i = j - 2, j - 3, \ldots, 1$ **do** ▷ Průchod ostatními hranami
5: $w_{i,j} \leftarrow \infty$
6: **if** $\{v_i, v_j\} \in E_G \vee (i, j) = (1, n)$ **then**
7: **for** $k = i + 1, i + 2, \ldots, j - 1$ **do**
▷ Průchod trojúhelníky u (v_i, v_j)
8: **if** $(w_{i,k} + w_{k,j} < w_{i,k} + w_{k,j}) \wedge (v_k$ je nalevo od $(v_i, v_j))$ **then**
▷ Jestliže trojúhelník tvoří
▷ triangulaci (jinak vyjde ∞)
▷ a je nejlepší dosud nalezená,
9: $w_{i,j} \leftarrow w_{i,k} + w_{k,j}$ ▷ uložíme váhu
10: $k_{i,j} \leftarrow k$ ▷ a vrchol pro konečnou rekonstrukci
11: **end if**
12: **end for**
13: **end if**
14: $w_{i,j} \leftarrow w_{i,j} + \|v_i - v_j\|$
15: **end for**
16: **end for**
17: $L \leftarrow \{(1, n)\}$ ▷ Rekonstrukce nejmenší nalezené triangulace
18: **while** $(i, j) \in L$ **do**
19: $L \leftarrow L \setminus \{(i, j)\}$
20: **if** $(w_{i,j} < \infty) \wedge (i + 1 < j)$ **then**
21: $E \leftarrow E \cup \{\{v_i, v_j\}\}$
22: $L \leftarrow L \cup \{(i, k_{i,j}), (k_{i,j}, j)\}$
23: **end if**
24: **end while**
25: **return** $(w_{1,n} < \infty)$

Věta 2.5. *Algoritmus 2 jde implementovat v čase* $O(k^3)+T(n,k) = O(n+k^3) = O(n^3)$ *a s paměťovou složitostí* $O(k^2)$*, kde* n *je počet vrcholů vstupního grafu,* k *počet vrcholů na hranici jednoduchého polygonu a* $T(n,k)$ *složitost výpočtu jeho obvodu.*

Důkaz: Obvod jednoduchého polygonu jde nalézt v čase $O(|V| + |E_{IN}|) = O(n)$. Na obvodu se každá orientovaná hrana vyskytne nejvýše jednou, tedy jeho délka je $O(k)$. Test $\{v_i, v_j\} \in E_G$ trvá s $O(k + |E_G|) = O(k^2)$ předpočítáním sousedních diagonál s v_i na obvodu polygonu $O(k)$. Složitost závěrečného přidání hran do grafu je $O(k)$.

Graf diagonál s vrcholy na obvodu a pole $w_{i,j}$, $k_{i,j}$ zabírají $O(k^2)$ prostoru. Celková velikost zbylých pomocných proměnných je $O(k)$. □

2.4 Spojitý podgraf

Podívejme se nyní na situaci, kdy potřebujeme vytvořit minimální triangulaci ze spojitého PSLG (V, E_C). Vnitřní stěny každého takového grafu tvoří podle definice jednoduché polygony, a ty už umíme triangulovat v čase $O(n^3)$. Vnější stěna se dá ošetřit přidáním konvexního obalu (je součástí každé triangulace), čímž vzniknou další jednoduché polygony a jejich doplněním dostaneme omezenou minimální triangulaci grafu (V, E_C).

Tuto myšlenku zobecňuje algoritmus 3, který uvažuje jen triangulace s danou podmnožinou povolených diagonál.

Algoritmus 3 *GMWT* na spojitém podgrafu

Vstup: (V, E_C) je spojitý PSLG a $E_G \subseteq D(V, E_C)$.
Výstup: $(V, E) \in GMWT(V, E_C \cup E_G)$, jinak false.

```
1: E ← E(CH(V))
2: if E ⊈ E_C ∪ E_G then
3:     return false
4: end if
5: E ← E ∪ E_C
6: F ← {f; f je vnitřní stěna (V, E)}
7: for ∀f ∈ F do
8:     (V, E) ← minimální T ∈ T_G((V, E), f, E_G ∩ D((V, E), f))
9:     if triangulace f neexistuje then
10:        return false
11:    end if
12: end for
13: return true
```

Věta 2.6. *Nechť* (V, E_C) *je spojitý PSLG a* $E_G \subseteq D(V, E_C)$*. Jestliže* $GMWT(V, E_C \cup E_G) = \emptyset$*, algoritmus 3 vrátí false, jinak skončí s návratovou hodnotou true a výsledný graf* $T_{OUT} = (V, E) \in GMWT(V, E_C \cup E_G)$.

Pokud navíc $E_C \subseteq E(T_{MIN}) \subseteq E_C \cup E_G$ *pro nějakou* $T_{MIN} \in MWT(V)$*, výsledná triangulace* $T_{OUT} \in MWT(V)$*.*

Důkaz: Označme $M = GMWT(V, E_C \cup E_G)$. Konvexní obal je obsažen v každé triangulaci. Proto když $E \not\subseteq E_C \cup E_G$, můžeme výpočet přerušit, jinak přidat $E(CH(V))$ a E_C do výsledného PSLG.

Na řádku (6) je (V, E) spojitý PSLG, tedy všechny jeho vnitřní stěny mají spojitou hranici a tvoří jednoduché polygony. Pokud nějakou vnitřní stěnu f nejde triangulovat, $M = \emptyset$, jinak by pro $T \in M$ šla f doplnit hranami T.

Vnější ani žádná vnitřní stěna neobsahuje diagonálu a k E_C jsme přidali jen hrany E_G. Proto $T_{OUT} \in T_G(V, E_C \cup E_G)$.

Nechť $T \in M$ a algoritmus skončí s PSLG, $\|T_{OUT}\| > \|T\|$. Potom pro nějakou $f \in F$ stěnu $(V, E_C \cup E(CH(V))$ je váha hran T v f menší než váha hran T_{OUT}, čímž dostáváme spor s řádkem (8).

Důkaz druhého tvrzení je snadný. Z předpokladů přímo plyne $T_{MIN} \in T_G(V, E_C \cup E_G)$ a odtud $\|T_{OUT}\| \leqq \|T_{MIN}\|$. □

Věta 2.7. *Složitost algoritmu 3 je* $O(n^3)$*, paměťová* $O(n^2)$*.*

Důkaz: Konvexní obal můžeme najít a sjednotit se vstupním grafem v čase $O(n \log n)$, spolu s doplněním informace *s.ccw* ke všem orientovaným hranám. S využitím *s.ccw* jde najít obvod každé stěny v lineárním čase vzhledem k počtu vrcholů na obvodu. Vnitřní stěny $f \in F$ můžeme identifikovat pomocí nějaké orientované hrany s_f, že $s_f.f = f$ a snadno v lineárním čase vytvoříme seznam F jedním průchodem grafu. Prvky E_G jde přidělit k příslušným stěnám v $O(|E_G| \log n) = O(n^2 \log n)$. Celková složitost triangulace oblastí je podle věty 2.5 $O(\sum_{f \in F} |V \cap \overline{f}|^3)$, což je nejhůře $O(n^3)$.

Paměťová složitost vyplývá opět z věty 2.5. □

Poznámka 2.3. Popsali jsme polynomiální algoritmy, které pro speciální typy vstupů řeší TRI, MWT, $CMWT$ i $GMWT$. Stačí, abychom znali libovolný PSLG, který společně s konvexním obalem tvoří souvislý podgraf nějaké triangulace z příslušné třídy.

Kapitola 3

Aproximace

V této kapitole stručně popíšeme některé třídy triangulací s polynomiální složitostí výpočtu, které se dají použít jako přibližné řešení minimální triangulace nebo omezené minimální triangulace. Zaměříme se na známé odhady jejich časové složitosti a na relativní rozdíl váhy proti optimální hodnotě, tzv. aproximační faktor. Experimentálně naměřené hodnoty faktorů některých zde zmíněných aproximací lze nalézt v kapitole 6.

3.1 Delaunayova triangulace

Delaunayova triangulace je v praxi jedna z nejpoužívanějších triangulací kvůli svým zajímavým vlastnostem a oproti ostatním triangulacím poměrně jednoduchému výpočtu v $O(n \cdot \log n)$.

Delaunayovu triangulaci lze definovat jako libovolné doplnění PSLG, který je duálem tzv. Voronoiova diagramu. Podrobnější definice a informace o vlastnostech Delaunayových triangulací lze nalézt například v [2, 10].

Definice 3.1. Nechť $V = \{v_1, v_2, \ldots, v_n\} \subset \mathbb{R}^2$ je konečná množina bodů v rovině. Body v_i budeme nazývat generátory. Voronoiův diagram $VD(V)$ množiny V rozděluje rovinu do uzavřených oblastí, jedné pro každý generátor. Tyto oblasti nazýváme Voronoiovy polygony a definujeme je jako

$$VS(v_i) = \{x \in \mathbb{R}^2; \|x - v_i\| \leqq \|x - v_j\| \forall j \neq i\}. \tag{3.1}$$

Definice 3.2 (Delaunayova triangulace)**.** Nechť $VD(V)$ je Voronoiův diagram množiny $V = \{v_1, v_2, \ldots, v_n\} \subset \mathbb{R}^2$ a $VS(v_i)$ jsou Voronoiovy polygony $VD(V)$. Pak definujeme PSLG

$$PDT(V) = (V, \{\{v_i, v_j\}; v_i \neq v_j \text{ a } VS(v_i), VS(v_j) \text{ mají společnou hranu}\}). \tag{3.2}$$

Delaunayovou triangulací (DT) množiny V nazveme každou triangulaci, která má $PDT(V)$ jako podgraf. Množinu všech Delaunaových triangulací na V značíme $DT(V)$.

Poznámka 3.1. Jestliže žádná čtveřice bodů neleží na společné kružnici, je definice jednoznačná. V opačném případě vznikne mezi body na kružnici prázdný polygon v $PDT(V)$ a ten lze doplnit libovolnými diagonálami.

Delaunayova triangulace jde definovat i konstruktivním algoritmem.

Definice 3.3 (Voronoiův diagram)**.** Nechť $T \in T(V)$ je triangulace. Dvojice trojúhelníků $T_1 = (u, v, w_1)$ a $T_2 = (u, v, w_2)$ v T kolem společné hrany je lokálně Delaunayova, jestliže w_1 neleží v otevřeném kruhu opsaném T_2 a w_2 neleží v otevřeném kruhu opsaném T_1. Triangulace je lokálně Delaunayova, jestliže všechny hrany jsou lokálně Delaunayovy.

Věta 3.1. *Triangulace je Delaunayova právě tehdy, když je lokálně Delaunayova.*

Podobně lze definovat i omezenou Delaunayovu triangulaci.

Definice 3.4 (Omezená Delaunayova triangulace)**.** Triangulace $T \in T(V)$ je omezenou Delaunayovou triangulací (CDT) grafu $G_C = (V, E_C)$, jestliže pro každou hranu $e = \{u, v\} \in E(T)$ existuje kružnice procházející vrcholy u a v, která neobsahuje žádný vrchol viditelný z hrany e v G_C. Vrchol w je viditelný z e v G_C, pokud v G_C existuje úsečka bez průsečíku spojující w s nějakým bodem úsečky uv. Množinu všech omezených Delaunayových triangulací na G_C značíme $CDT(V, E_C)$.

Vynecháme-li omezující hrany, dostaneme další možnou definici Delaunayových triangulací.

Mezi zajímavé vlastnosti DT patří například to, že vždy obsahuje Euklidovskou minimální kostru nebo že maximalizuje minimální úhel. V DT se tak oproti jiným triangulacím méně často vyskytují extrémně úzké trojúhelníky, což může být v některých aplikacích užitečné.

3.1.1 Složitost výpočtu DT

Definice 3.4 určuje přímý konstruktivní algoritmus pro (omezenou) Delaunayovu triangulaci, která ovšem není příliš efektivní. O něco lepší je algoritmus, který začne libovolnou triangulací a prohazováním diagonál postupně vylepšuje dvojice sousedících trojúhelníků, které nejsou lokálně Delaunayovy. Dostaneme tak algoritmus 4 pracující v čase $O(n^2)$. Definici lokálně Delaunayových dvojic trojúhelníků jde snadno rozšířit i na omezený případ.

Optimální časové složitosti $O(n \log n)$ lze dosáhnout například konstrukcí Voronoiova diagramu průchodem roviny nebo metodou rozděl a panuj [2].

V některých případech je dokonce možné sestavit (omezenou) Delaunayovu triangulaci v lepším čase než $O(n \log n)$. Chceme-li například doplnit CDT v jednoduchém polygonu, sníží se složitost na $O(n)$ [15]. Odtud také plyne lineární složitost, jestliže známe nějakou kostru triangulace, speciálně stačí Euklidovská minimální kostra, která je vždy podgrafem DT. Pro rovnoměrně rozložené množiny bodů dostaneme opět lineární očekávanou složitost [8].

Algoritmus 4 Výpočet DT

Vstup: (V, E_C) PSLG.
Výstup: (V, E), $(V, E) \in CDT(V, E_C)$.

1: $(V, E) \leftarrow$ libovolná triangulace, kde $E_C \subseteq E$.
2: $M \leftarrow E$
3: **while** $\exists e = (u, v) \in M$ **do**
4: $\quad M \leftarrow M \setminus \{e\}$
5: $\quad$ **if** trojúhelníky $(u, v, w_1), (u, v, w_2)$ nejsou lokálně Delaunayovy **then**
6: $\quad\quad E \leftarrow (E \setminus \{e\}) \cup \{\{w_1, w_2\}\}$
7: $\quad\quad M \leftarrow M \cup \{\{u, w_1\}, \{v, w_1\}, \{u, w_2\}, \{v, w_2\}\}$
8: $\quad$ **end if**
9: **end while**

3.1.2 Kvalita DT aproximace

Bohužel pro libovolné n existuje vstupní množinu n vrcholů, pro kterou je váha Delaunayovy triangulace $O(n)$-krát větší než minimální triangulace. Konstrukce takové množiny lze najít například v [16].

3.2 Greedy triangulace

Greedy triangulace se definuje procedurálně jako výsledek opakovaného přidávání nejkratší diagonály do PSLG. Výsledná množina hran nemusí být jednoznačná, protože neuvádíme pořadí přidávání stejně dlouhých diagonál.

Algoritmus 5 Výpočet omezené greedy triangulace

Vstup: (V, C) je PSLG.
Výstup: (V, E) je triangulace.

1: $E \leftarrow C$
2: **while** (V, E) není triangulace **do**
3: $\quad e \leftarrow$ nejkratší diagonála v PSLG (V, E)
4: $\quad E \leftarrow E \cup \{e\}$
5: **end while**

Definice 3.5. Nechť (V, C) je PSLG. Potom omezenou greedy triangulací (CGT) na grafu (V, C) nazveme každou triangulaci, která může vyjít z Algoritmu 5. Prvky množiny C jsou omezující hrany. Množinu všech omezených greedy triangulací na (V, C) značíme $CGT(V, C)$.

Definice 3.6. Nechť $V \subset \mathbb{R}^2$ je konečná množina bodů v rovině. Pak prvky množiny $GT(V) = CGT(V, \emptyset)$ nazýváme greedy triangulacemi (GT) na V.

Algoritmus je korektní, protože každý PSLG jde doplnit na úplnou triangulaci. Výsledek nemusí být jednoznačný, má-li více hran stejnou délku.

3.2.1 Složitost výpočtu GT

V algoritmu 5 přidáme do grafu celkem $O(n)$ hran. Nejkratší diagonálu v PSLG snadno najdeme v čase $O(n^3)$, a tedy celková složitost je $O(n^4)$.

Na úkor paměťové složitosti můžeme vylepšit časovou složitost na $O(n^3)$. Stačí před spuštěním cyklu vygenerovat všechny možné diagonály (V, C) a ty pak procházet v neklesajícím pořadí podle délky. Tento postup popisuje algoritmus 6.

Algoritmus 6 Výpočet omezené greedy triangulace v $O(n^3)$

Vstup: (V, C) je PSLG.
Výstup: (V, E) je omezená greedy triangulace na grafu (V, C).

```
1: D ← seznam všech diagonál (V, C)
2: Uspořádej prvky D podle délky.
3: for ∀e ∈ D v neklesajícím pořadí do
4:     if e je diagonála (V, E) then
5:         E ← E ∪ {e}
6:     end if
7: end for
```

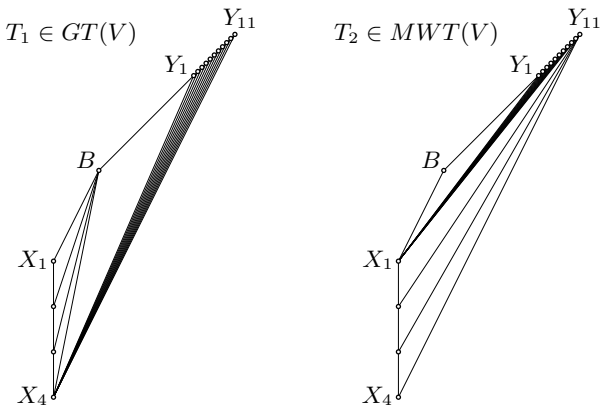
P. D. Gilbert dokázal [12], že greedy triangulace jde najít v čase $O(n^2 \log n)$ s kvadratickou paměťovou složitostí. S využitím tzv. omezených Voronoiových diagramů se pak podařilo vyřešit problém v čase $O(n^2)$ a lineárním prostoru [24]. Zajímavý algoritmus s očekávanou lineární složitostí na rovnoměrně rozmístěných bodech předvedli R. Drysdale a kol. [6]. Převratného výsledku pak dosáhli C. Levcopoulos a D. Krznaric [26], kterým se podařilo dokázat dříve publikovaný algoritmus převádějící DT na GT nejhůře v lineárním čase.

Celková složitost výpočtu GT na obecné množině bodů je tedy $\Theta(n \log n)$ a pro některé speciální množiny vstupních dat je dokonce možné GT najít v očekávaném nebo nejhorším čase $o(n \log n)$ (viz kapitola 3.1.1).

3.2.2 Kvalita GT aproximace

V [23] Levcopoulos dokázal, že dolní odhad kvality GT jako aproximace MWT je $\Omega(\sqrt{n})$. Že se jedná o nejlepší možný odhad pak odvodil společně s Krznaricem v [25].

Konstrukce V množiny n bodů, pro kterou podíl váhy každé GT a MWT je $\Omega(\sqrt{n})$,



Obrázek 3.1: Levcopoulosova množina bodů pro $n = 16$.

vypadá následovně (graficky znázorňuje obrázek 3.1):

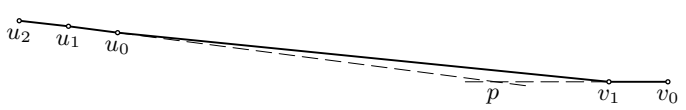
$$B = (1, 1), \tag{3.3}$$
$$X_i = (0, -i), \qquad i = 1, \ldots, \lfloor\sqrt{n}\rfloor, \tag{3.4}$$
$$Y_i = (x, x), x \in [3, 4], \qquad i = 1, \ldots, n - 1 - \lfloor\sqrt{n}\rfloor, \tag{3.5}$$
$$V = \{B\} \cup \{X_i\} \cup \{Y_i\}. \tag{3.6}$$

3.3 Quasi-greedy triangulace

Jak ukázali Levcopoulos a Krznaric v článku [25], mírnou modifikací algoritmu 5 pro greedy triangulaci je možné vylepšit aproximační faktor triangulace vzhledem k MWT na $O(1)$. Stejného faktoru pak dosáhli i pro obecnější problém triangulace s omezujícími hranami.

Definice 3.7 ([25])**.** Řekneme, že $\{v_0, u_0\}$ splňuje v $PSLG$ $G = (V, E)$ tzv. QGT-podmínky pro dvojici (v_1, u_1), jestliže jsou splněny všechny následující body (příklad na obrázku 3.2):

1. diagonála $\{v_1, u_1\}$ tvoří prázdný trojúhelník (v_1, u_0, u_1) s dvěmi hranami v G,
2. diagonála $\{v_0, u_0\}$ protíná $\{v_1, u_1\}$ a tvoří prázdný trojúhelník $\{v_0, v_1, u_0\}$ s dvěmi hranami v G,
3. $|\angle v_1, u_0, u_1| > 135°$ v trojúhelníku (v_1, u_0, u_1),
4. $\|v_0 - u_0\| < 1{,}1 \cdot \|v_1 - u_1\|$,
5. $\|v_0 - p\| < 0{,}5 \cdot \|u_0 - p\|$, kde p je průsečíkem přímek daných dvojicemi bodů (v_0, v_1) a (u_0, u_1),



Obrázek 3.2: Příklad grafu, kde $\{v_0, u_0\}$ splňuje QGT-podmínky pro (v_1, u_1).

6. v G existuje hrana $\{u_1, u_2\}$ taková, že (v_1, u_0, u_1, u_2) tvoří prázdný čtyřúhelník v G a $|\angle u_0, u_1, u_2| > 180°$ v čtyřúhelníku (v_1, u_0, u_1, u_2).

Algoritmus 7 Výpočet omezené quasi-greedy triangulace

Vstup: (V, C) je PSLG.
Výstup: (V, E) je triangulace.

```
1: E ← C
2: while (V, E) není triangulace do
3:     e = {u1, v1} ← nejkratší diagonála v PSLG (V, E)
4:     if ∃{u0, v0} splňující všechny QGT-podmínky pro (u1, v1) nebo (v1, u1) then
5:         E ← E ∪ {u0, v0}
6:     else
7:         E ← E ∪ {u1, v1}
8:     end if
9: end while
```

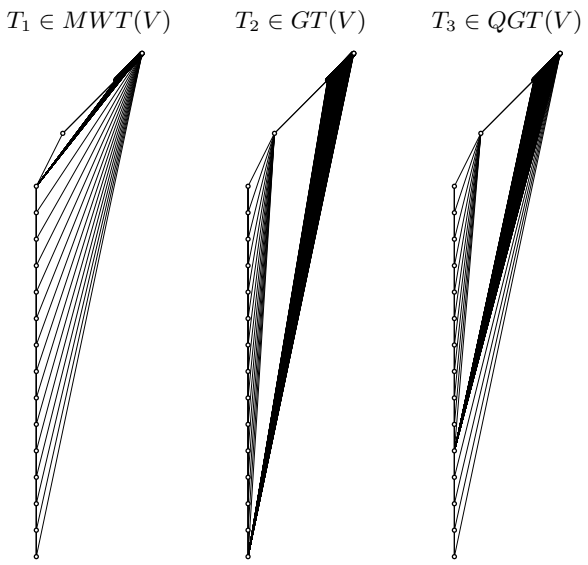
Definice 3.8 (Quasi-greedy triangulace)**.** Nechť (V, C) je PSLG a žádné tři body z V neleží na jedné přímce. Potom omezenou quasi-greedy triangulací ($QCGT$) na grafu (V, C) nazveme každou triangulaci, která může vyjít z algoritmu 7. Prvky množiny C jsou takzvané omezující hrany. Množinu všech omezených quasi-greedy triangulací na (V, C) značíme $QCGT(V, C)$.

Nechť $V \subset \mathbb{R}^2$ je konečná množina bodů, z nichž žádné tři neleží na jedné přímce. Pak prvky množiny $QGT(V) = CQGT(V, \emptyset)$ nazýváme quasi-greedy triangulacemi (QGT) na V.

3.3.1 Složitost výpočtu QGT

QGT-podmínky se dají doplnit do většiny známých algoritmů pro výpočet greedy triangulace. Součástí programu přiloženého k této diplomové práci je i výpočet (omezené) quasi-greedy triangulace pro obecnou polohu bodů v čase $O(n^3)$. Jedná se o jednoduché doplnění podmínek se složitostí $O(n)$ do algoritmu 6.

Levcopoulos a Krznaric v [25] tvrdí, že podobně jako GT, jde i QGT spočítat nejhůře v $O(n \log n)$ modifikací algoritmu z [26]. Tím dostáváme, že složitost problému nalezení quasi-greedy triangulace je $\Theta(n \log n)$.



Obrázek 3.3: Srovnání MWT, GT a QGT na speciální množině 250 bodů.

3.3.2 Kvalita QGT aproximace

Věta 3.2. *Pro libovolný PSLG (V, C) a (omezenou) quasi-greedy triangulaci platí:*

$$\|T_{QGT}\| = O(\|T_{MWT}\|), \qquad \forall T_{QGT} \in QGT(V), \tag{3.7}$$
$$\forall T_{MWT} \in MWT(V), \tag{3.8}$$
$$\|T_{CQGT}\| = O(\|T_{CMWT}\|), \qquad \forall T_{CQGT} \in CQGT(V, C), \tag{3.9}$$
$$\forall T_{CMWT} \in CMWT(V, C). \tag{3.10}$$

Důkaz: Věta je přímým důsledkem tvrzení dokázaných v [25]. □

Chování QGT aproximace na speciální množině bodů znázorňuje obrázek 3.3. Tato množina je mírným posunutím bodů množiny 3.6 (greedy triangulace zde má $\Theta(\sqrt{n})$ aproximační faktor) do obecné polohy a má tvar:

$$B = (1, 1), \tag{3.11}$$
$$X_i = (-\epsilon_n \cdot i^2, -i), \qquad i = 1, \ldots, \lfloor\sqrt{n}\rfloor, \tag{3.12}$$
$$m = n - 1 - \lfloor\sqrt{n}\rfloor, \tag{3.13}$$
$$Y_i = (3 + \frac{i}{m}, 3 + \frac{i}{m} + \epsilon_n \cdot i^2), \quad i = 1, \ldots, m, \tag{3.14}$$
$$V = \{B\} \cup \{X_i\} \cup \{Y_i\}. \tag{3.15}$$

Testy ukázaly, že se rozdíl QGT oproti GT na této množině projeví až pro větší počty bodů. Pro menší n nebyla splněna pátá QGT-podmínka. Se zvyšujícím počtem vrcholů se prodlužuje spodní část triangulace podobná minimální triangulaci a horní část podobná GT zůstává.

Poznámka 3.2. Definice QGT by šla se zachováním $O(1)$ aproximačního faktoru podle [25] rozšířit pro libovolnou polohu vstupních bodů (tj. i s více body na společné přímce).

3.4 Double-greedy triangulace

Jedná se o další modifikaci greedy triangulace, kterou popsal M. Nociar ve své diplomové práci [27] a která se od GT liší pořadím zpracování diagonál. Každou konkrétní hranu e ohodnotíme, narozdíl od klasické Euklidovské délky, celkovou váhou CGT s omezující hranou e. Do PSLG pak postupně přidáváme diagonály v neklesajícím pořadí podle jejich ohodnocení.

Jelikož časová složitost algoritmu dosahuje $O(n^5)$, M. Nociar navrhuje snížit počet uvažovaných hran filtrací hran bez diamantové vlastnosti (viz. definice 4.4). Počet hran se tak v očekávaném případě pro rovnoměrně rozloženou množinu bodů v konvexní oblasti sníží na $O(n)$ a přitom vyřadíme jen hrany, které nepatří do žádné minimální triangulace.

Vyřazením hran z výpočtu CGT dostaneme „zobecněnou greedy triangulaci“, která ne vždy jde dokončit. Jestliže při výpočtu nenajdeme triangulaci, vyřadíme příslušnou omezující hranu z finálního sestavování DGT. Z praktických testů se zdá, že touto úpravou vždy dokončíme triangulaci, ovšem v [27, 9] jsem nenašel potřebný důkaz. V důkazu by bylo nutné uvažovat vlastnosti diamantových hran i zmíněných „zobecněných greedy triangulací“, jinak se v podstatě jedná o NP-úplný problém hledání triangulace v obecné množině kandidátních hran (viz. kapitola 2).

Problém nedokončené triangulace by šel vyřešit tak, že bychom na konci opět spustili celý algoritmus, ovšem bez vyřazení hran. Protože zmíněná situace podle testů téměř nikdy nestává, takto modifikovaná triangulace by si zachovala podobnou očekávanou složitost výpočtu.

Poznámka 3.3. Výše popsané algoritmy, zejména DGT bez filtrace a DGT s filtrací hran na diamantovou vlastnost, definují různé třídy triangulací. Záleží také na přesném tvaru použité zakázané oblasti. Do testů v kapitole 6 jsem zařadil implementaci DGT s vyřazením hran pomocí zakázané oblasti R^{MWT}_{diam} (definice 4.4) a s předpokladem, že vždy dokončí triangulaci. Definici kvůli korektnosti raději uvedu jako DGT na kompletní množině kandidátů.

Podobně je možné zavést omezenou double-greedy triangulaci ($CDGT$). Diagonály v PSLG omezujících hran opět setřídíme podle váhy CGT a v neklesajícím pořadí přidáme do výsledné triangulace. Zde nemá smysl použít diamantovou vlastnost, protože R^{MWT}_{diam} není zakázanou oblastí třídy $CMWT$.

Definice 3.9 (Double-greedy triangulace)**.** Nechť $E_G = (V, E_C)$ je PSLG. Potom omezenou double-greedy triangulací ($CDGT$) na E_G nazveme libovolný možný výsledek algoritmu 8 se vstupní množinou kandidátních hran $E_G = D(G_C)$. Množinu všech omezených double-greedy triangulací značíme $CDGT(V, E_C)$. Dále označme $DGT(V) = CDGT(V, \emptyset)$ množinu všech double-greedy triangulací (DGT) na V.

Algoritmus 8 Výpočet DGT a $CDGT$

Vstup: $G_C(V, E_C)$ PSLG omezujících hran, kandidátní hrany $E_G \subseteq D(G_C)$.

1: **for** $\forall e \in E_G$ **do**
2: $w_e \leftarrow$ váha „zobecněné GT“ na množině kandidátů $E_C \cup \{e\} \cup \{e_0 \in E_G; e_0$ neprotíná $e\}$, případně $infty$, když neexistuje
3: **end for**
4: Uspořádej hrany E_G do neklesající posloupnosti podle w_e.
5: $E \leftarrow E_C$
6: **for** $\forall e \in E_G$ v neklesajícím pořadí **do**
7: **if** $e \in D(V, E)$ **then**
8: $E \leftarrow E \cup \{e\}$
9: **end if**
10: **end for**

Poznámka 3.4. Výsledek algoritmu je pro navzájem různě dlouhé hrany jednoznačný.

3.5 Další aproximace

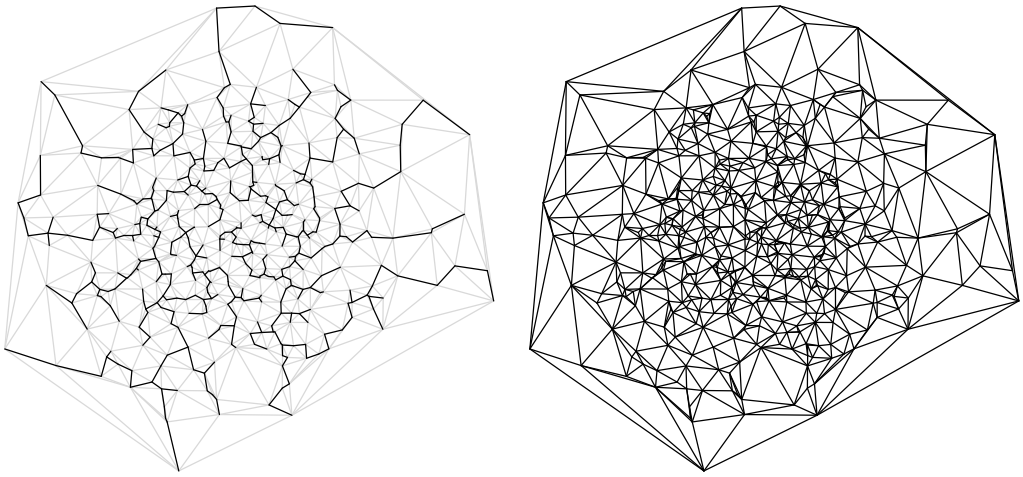
Donedávna byla asymptoticky nejlepší známou aproximací MWT Plaisted-Hong triangulace [28, 29] s aproximačním faktorem $O(\log n)$ a časovou složitostí $O(n^2 \log n)$. Algoritmus nejprve rozdělí konvexní obal vstupní množiny bodů V na konvexní polygony s prázdným vnitřkem, jejichž celková délka je $O(1)$-násobkem váhy MWT. Přidáme-li do grafu hrany $CMWT$, vyjde dohromady $O(\log n)$ aproximační faktor. Pro doplnění triangulace se však používá rychlejší algoritmus, který sice najde o něco horší triangulaci, ovšem se stejným aproximačním faktorem $O(\log n)$.

Další zajímavou aproximací je $CMWT$ triangulace minimální kostry (MST-T). Algoritmus pracuje tak, že pro vstupní množinu bodů spočítá minimální kostru a doplní ji pomocí algoritmu 3 na $CMWT$. Celková složitost výpočtu tak je $O(n^3)$, ale bohužel jde dokázat aproximační faktor $\Omega(n)$ vůči minimální triangulaci [16].

Poznámka 3.5. Kostru pro MST-T jde hledat přes DT, protože každá minimální kostra DT vždy patří mezi minimální kostry V. Složitost minimální kostry v obecném ohodnoceném grafu je $o(m \log n)$ [11], tedy pro triangulaci $o(n \log n)$.

Výpočet minimální kostry s následným doplněním na $CMWT$ jde aplikovat nejen na Delaunayovu triangulaci, ale i na libovolnou jinou triangulaci. Pro greedy triangulaci dostaneme opět časovou složitost $O(n^3)$ a aproximační faktor $\Theta(\sqrt{n})$ [16]. Výsledné triangulace budeme značit zkratkami x-MST-T, kde x je konkrétní třída triangulací, tj. např. GT-MST-T nebo DGT-MST-T.

Za zmínku také stojí perspektivní nápad „look-ahead greedy triangulace“ I. Kolingerové [20], kdy se oproti klasické greedy triangulaci vybírá nejmenší k-tice neprotínajících



MST-T *MWT*

Obrázek 3.4: MST-T množiny 500 bodů.

se diagonál. Pro $k = 1$ tak dostaneme GT a se zvýšením k se pravděpodobně zlepší aproximační faktor výsledné triangulace. Přesnou minimální triangulace pak dostaneme pro $k = 3n - 3 - |E(CH(V))|$. Časová složitost jednoduchého algoritmu je $O(k \cdot n^{2+k})$.

Další typy aproximací lze nalézt například v [9, 21].

Kapitola 4

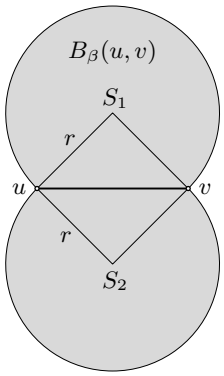
Heuristiky

Smyslem kapitoly bude popsat několik heuristik, které nám umožní v polynomiálním čase najít co největší podgraf nějaké triangulace s minimální váhou (nebo naopak co nejmenší množinu kandidátů na hrany MWT). Kdyby byl výsledný podgraf souvislý, mohli bychom jej pomocí dříve popsaného algoritmu 3 doplnit na minimální triangulaci nejhůře v čase $O(n^3)$.

Polynomiální složitosti bychom dosáhli i pro $k = O(1)$ komponent. Stačí totiž ozkoušet $O(n^{2k-2})$ možností propojení komponent do souvislého grafu a pro každý spočítat $CMWT$ v $O(n^3)$. Jak ukážeme později, celková složitost výpočtu MWT se dá snížit na $O(n^{2+k})$.

4.1 β-skeleton

Definice 4.1 ([17])**.** Nechť $V \subset \mathbb{R}^2$ je konečná množina bodů a $\beta \geqq 1$. Potom β-skeletonem množiny V nazveme graf $G_\beta = (V, E_\beta)$ obsahující právě hrany $\{u, v\} \in E_\beta$ takové, že $u \neq v$ a $\forall w \neq u, v : w \notin B_\beta(u, v)$, kde $B_\beta(u, v)$ je sjednocením uzavřených kruhů o poloměru $\frac{\beta\|u-v\|}{2}$ procházející vrcholy u a v (viz. obrázek 4.1).



Obrázek 4.1: Definice β-skeletonu.

Věta 4.1. *Nechť $V \subset \mathbb{R}^2$ je konečná množina a $\beta_1 \geqq \beta_2 \geqq 1$. Potom β_1-skeleton je podgrafem β_2-skeletonu.*

Důkaz: Tvrzení vyplývá přímo z definice, neboť $B_{\beta_2}(u,v) \subseteq B_{\beta_1}(u,v)$ pro každou dvojici vrcholů z V. □

Věta 4.2. *Pro každou konečnou množinu $V \subset \mathbb{R}^2$ a $\beta \geqq 1$ je β-skeleton PSLG.*

Důkaz: Tvrzení stačí dokázat pro $\beta = 1$, protože podgraf PSLG je zřejmě PSLG. Označme β-skeleton $G_\beta = (V, E_\beta)$. Kdyby nějaká úsečka odpovídající hraně $\{u,v\} \in E_\beta$ obsahovala $w \in V$, pak by $w \in B_\beta(u,v)$, proto G_β je SLG. Nechť hrany G_β $\{u_1,v_1\}$ a $\{u_2,v_2\}$ tvoří dvojici protínajících se úseček mimo své koncové body. Potom je $|\angle u_1,u_2,v_1| < 90°$ a $|\angle u_1,v_2,v_1| < 90°$. Součet úhlů ve čtyřúhelníku je ovšem 360°, tedy alespoň u jednoho z vrcholů u_1, v_1 je ve čtyřúhelníku (u_1,u_2,v_2,v_2) úhel větší než 90° a dostáváme spor s $\{u_2,v_2\} \in E_\beta$. □

Souvislost β-skeletonu s minimálními triangulacemi se podařilo najít Keilovi [17], když dokázal, že pro $\beta = \sqrt{2}$ je G_β podgrafem každé triangulace s minimální váhou.

Tento výsledek se pak rozšiřují Cheng a Xu [14], když dokazují následující dvě tvrzení.

Věta 4.3. *Nechť $V \subset \mathbb{R}^2$ je konečná množina a $T_{MIN} \in MWT(V)$. Potom G_β je podgrafem T_{MIN} pro každé*

$$\beta > \frac{1}{\sin\left(\tan^{-1}\left(\frac{3}{\sqrt{2\sqrt{3}}}\right)\right)} < 1{,}17682. \tag{4.1}$$

Věta 4.4. *Pro každé $\beta \geqq 1$ splňující*

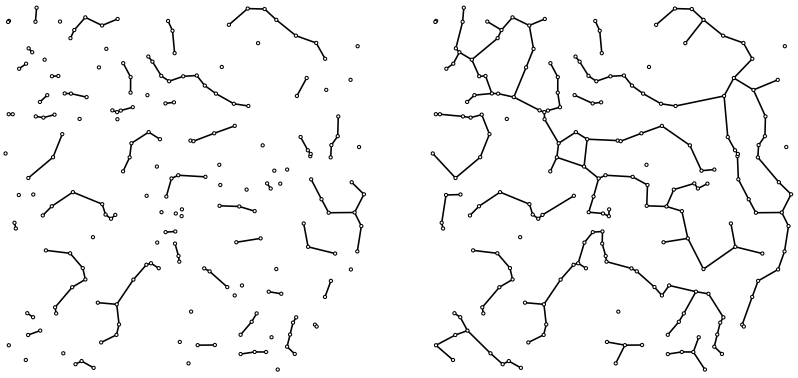
$$\beta < \frac{1}{\sin\left(\frac{\pi}{3}\right)} > 1{,}1547 \tag{4.2}$$

existuje čtyřbodová množina V taková, že β-skeleton V není podgrafem žádné $T_{MIN} \in MWT(V)$.

Jak je vidět, odhady na obou směrech jsou si velmi blízko a nejde je už o moc vylepšit. Obrázek 4.2 znázorňuje $\sqrt{2}$-skeleton a 1,17682-skeleton množiny 200 bodů.

4.2 Zakázané oblasti

V této části popíšeme lokální oblasti kolem hran, které mají jistou společnou vlastnost pro každou triangulaci z třídy Delaunayových, greedy nebo minimálních triangulací.



Obrázek 4.2: β-skeleton 200 náhodně rozmístěných bodů.

Pokud hrana mezi nějakou dvojicí vrcholů nesplní tuto vlastnost, budeme schopni prohlásit, že hrana jistě nepatří do žádné triangulace z dané třídy, a nebudeme ji muset v dalších výpočtech uvažovat. Při rovnoměrném rozložení vrcholů v konvexní oblasti tak dokonce vyřadíme většinu z $O(n^2)$ hran a zbude jich v průměrném případě $O(n)$.

Definice jsou inspirovány pramenem [7].

Definice 4.2 (Zakázaná oblast)**.** Nechť $e = \{u, v\} \in \binom{\mathbb{R}^2}{2}$, $R \subseteq \mathbb{R}^2$ a $\mathcal{M} \subseteq \mathcal{T}$ je třída triangulací (např. MWT, GT, ...). Označme $R_l = \{x \in R; x \text{ není napravo od } (u, v)\}$, $R_r = \{x \in R; x \text{ není nalevo od } (u, v)\}$.

Řekneme, že R je zakázanou oblastí („exclusion region“) hrany e pro třídu $\mathcal{M}$, jestliže pro každou triangulaci $T \in \mathcal{M}, e \in E(T)$ platí $R_l \cap V(T) = \emptyset \vee R_r \cap V(T) = \emptyset$. Vrcholy V blokují zakázanou oblast R hrany e, pokud $R_l \cap V \neq \emptyset \wedge R_r \cap V \neq \emptyset$.

Věta 4.5 (Vlastnosti zakázaných oblastí)**.** *Pro libovolnou třídu $\mathcal{M} \subseteq \mathcal{T}$, hranu $e \in \binom{\mathbb{R}^2}{2}$ a zakázanou oblast $R \subseteq \mathbb{R}^2$ hrany e pro třídu $\mathcal{M}$ platí:*

(i) Nechť $V \subseteq \mathbb{R}^2$ je konečná množina a $e \in \binom{V}{2}$. Jestliže vrcholy V blokují zakázanou oblast R hrany e, pak e nepatří do žádné triangulace $T \in \mathcal{M}, V = V(T)$.

(ii) Podmnožina R je opět zakázanou oblastí hrany e pro třídu $\mathcal{M}$.

(iii) Jestliže e leží v nějaké triangulaci z $\mathcal{M}$, pak R neobsahuje žádný z koncových bodů hrany e.

(iv) Množina $seg(e) \setminus e$ (úsečka bez krajních bodů) je zakázanou oblastí hrany e pro třídu všech triangulací $\mathcal{T}$. Žádná množina obsahující bod mimo tuto úsečku není zakázanou oblastí hrany e pro třídu $\mathcal{T}$.

(v) $R \cup (seg(e) \setminus e)$ je zakázanou oblastí hrany e pro třídu $\mathcal{M}$.

Důkaz: Tvrzení vyplývají snadno z definice zakázaných oblastí, triangulace a diagonály. □

Poznámka 4.1. Sjednocení zakázaných oblastí nemusí být zakázaná oblast.

Věta 4.6 (O zakázaných oblastech DT)**.** *Pro třídu všech Delaunayových triangulací $\mathcal{M} = \bigcup_V DT(V)$ je zakázanou oblastí každé hrany e otevřený kruh s průměrem $seg(e)$. Žádná jiná otevřená množina obsahující tento kruh už není zakázanou oblastí e pro třídu $\mathcal{M}$.*

Důkaz: (Podle [7]) Nechť $T = (V, E) \in DT(V)$, $e = \{u, v\} \in E$ a R je otevřený kruh s průměrem $seg(e)$. Vezměme kružnici k opsanou některému ze sousedních trojúhelníků k hraně e v T. Z vlastností DT víme, že vnitřek kružnice k (označme K) je prázdný. Jestliže střed k leží na $seg(e)$, pak $K = R$ a R je na obou stranách e prázdná. V opačném případě je prázdná část R na stejné straně e jako střed k.

Vezměme libovolnou otevřenou oblast $R^* \supset R$. Zvolme $p \in R^* \setminus R$ a $q \in R$ tak, aby úsečka pq procházela středem uv. Potom DT množiny $\{u, v, p, q\}$ je tvořena trojúhelníky (p, q, u) a (p, q, v) a R^* nemůže být zakázanou oblastí e pro třídu $\mathcal{M}$. □

Definice 4.3. Pro hranu $e = \{u, v\} \in \binom{\mathbb{R}^2}{2}$ definujeme oblast tvaru „oka“ $R^r_{eye}(e)$ s relativním poloměrem $r \in (0, 1/2)$:

$$K = \left\{ x \in \mathbb{R}^2; \left\| \frac{u+v}{2} - x \right\| \leqq r\|e\| \right\}, \tag{4.3}$$

$$R^r_{eye}(e) = \{tu + (1-t)y, tv + (1-t)y; t \in [0, 1), y \in K\ \} \tag{4.4}$$

Dále pro e zavedeme kosočtvercovou (nebo „diamantovou“) oblast $R^\varphi_{diam}(e)$ s úhlem $\varphi \in (0, \pi/2)$:

$$R^\varphi_{diam}(e) = \left\{ x \in \mathbb{R}^2; u, v \neq x, |\angle uvx| \leqq \varphi, |\angle vux| \leqq \varphi \right\}. \tag{4.5}$$

Věta 4.7 (O zakázaných oblastech GT)**.** *Pro třídu greedy triangulací $\mathcal{M} = \bigcup_V GT(V)$ a hranu $e \in \binom{\mathbb{R}^2}{2}$ jsou zakázané oblasti:*

$$R^{GT}_{eye}(e) = R^r_{eye}(e), \qquad r = \frac{1}{2\sqrt{5}} > 0{,}223, \tag{4.6}$$

$$R^{GT}_{diam}(e) = R^\varphi_{diam}(e), \quad \varphi = \arctan\frac{1}{\sqrt{5}} > 0{,}420. \tag{4.7}$$

Důkaz: Viz [1]. □

Věta 4.8 (O zakázaných oblastech MWT)**.** *Pro třídu všech minimálních triangulací $\mathcal{M} = \bigcup_V MWT(V)$ a hranu $e \in \binom{\mathbb{R}^2}{2}$ je zakázaná dvojice oblastí:*

$$R^{MWT}_{eye}(e) = R^r_{eye}(e), \qquad r = \frac{1}{\sqrt{2}} > 0{,}707, \tag{4.8}$$

$$R^{MWT}_{diam}(e) = R^\varphi_{diam}(e), \quad \varphi = \frac{\pi}{4{,}6} > 0{,}682. \tag{4.9}$$

Následující oblasti kolem hrany e nejsou pro $\mathcal{M}$ zakázané:

$$R^r_{eye}(e), \quad r > 0{,}759, \tag{4.10}$$

$$R^{\varphi}_{diam}(e), \quad \varphi > \frac{\pi}{4{,}23} < 0{,}743. \tag{4.11}$$

Důkaz: Viz [7]. □

Definice 4.4 (Diamantová vlastnost)**.** Výrazy $R^{GT}_{eye}(e)$, $R^{GT}_{diam}(e)$, $R^{MWT}_{eye}(e)$ a $R^{MWT}_{diam}(e)$ budeme označovat množiny popsané v předchozích dvou větách. Řekneme, že hrana e má mezi vrcholy V diamantovou vlastnost („diamond property"), jestliže V neblokuje zakázanou oblast $R^{MWT}_{diam}(e)$.

Poznámka 4.2. Pro zakázané oblasti GT a MWT platí:

$$R^{GT}_{diam}(e) \subset R^{GT}_{eye}(e) \subset R^{MWT}_{eye}(e), R^{MWT}_{diam}(e), \tag{4.12}$$

$$R^{MWT}_{diam}(e) \not\subseteq R^{MWT}_{eye}(e), \tag{4.13}$$

$$R^{MWT}_{eye}(e) \not\subseteq R^{MWT}_{diam}(e). \tag{4.14}$$

V [4] lze najít obecný důkaz, že když je zakázanou oblastí nějaké třídy triangulací kruh ve středu každé hrany e o poloměru $r \cdot \|e\|, r > 0$, pak rovnoměrně rozložená množina vrcholů v konvexní oblasti obsahuje v průměru $O(n)$ nezablokovaných hran. Třídy triangulací s takovým tvarem zakázaných oblastí mají navíc podle [7] konstantní očekávaný faktor aproximace. Přímo pak vyplývá, že váha Delaunayových i greedy triangulací je na náhodné množině V vrcholů v průměrném případě $O(\|MWT(V)\|)$.

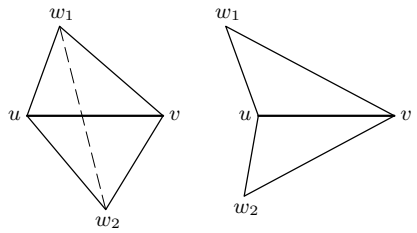
4.3 Lokálně minimální triangulace

Nyní zavedeme třídu tzv. lokálně minimálních triangulací, ve kterých je minimalita definovaná jen přes malé okolí hran. Ukážeme, že triangulace s minimální váhou jsou i lokálně minimální, a že se dá lokální minimalita hran použít jako velmi dobrá heuristika pro výpočet podgrafu MWT. Pro rovnoměrně rozmístěné množiny bodů dokonce vyjde ve většině případů souvislý graf, který umíme doplnit na MWT v polynomiálním čase.

Nechť $T = (V, E)$ je triangulace. Každá hrana $e = \{u, v\} \in E$ buď leží na konvexním obalu V nebo sousedí s nějakou dvojicí trojúhelníků (u, v, w_1) a (u, v, w_2).

Definice 4.5 (Lokálně minimální hrana)**.** Řekneme, že hrana e triangulace T je lokálně minimální, jestliže platí alespoň jedna z následujících podmínek:

1. $e \in E(CH(V))$,
2. $\|e\| \leq \|w_1 - w_2\|$,
3. čtyřúhelník (u, w_1, v, w_2) není konvexní, tj. $T(V, (E \setminus \{e\}) \cup \{\{w_1, w_2\}\})$ není triangulace.



Obrázek 4.3: Okolí lokálně minimální hrany $\{u, v\}$.

Definici lokálně minimální hrany znázorňuje Obrázek 4.3.

Definice 4.6 (Lokálně minimální triangulace)**.** Triangulace T je lokálně minimální, jestliže obsahuje jen lokálně minimální hrany. Množinu všech lokálně minimálních triangulací na množině bodů V značíme $LMT(V)$.

Poznámka 4.3. Obecná lokálně minimální triangulace ve skutečnosti představuje aproximaci MWT, ovšem v této práci použijeme lokální minimalitu spíše jako heuristiku pro výpočet přesné MWT. Proto jsem LMT zařadil do kapitoly o heuristikách.

Věta 4.9 (Lokální minimalita MWT)**.** *Každá triangulace s minimální váhou je lokálně minimální.*

Důkaz: Kdyby nebyla, šlo by zmenšit její váhu prohozením diagonály v některém z čtyřúhelníků. □

Odtud přímo plynou dvě jednoduchá tvrzení:

Důsledek 4.1. *Jestliže je nějaká hrana v každé lokálně minimální triangulaci, pak ji obsahuje i každá triangulace s minimální váhou.*

Důsledek 4.2. *Jestliže nějaká hrana není v žádné lokálně minimální triangulaci, pak nemůže být ani v triangulaci s minimální váhou.*

4.3.1 LMT-skeleton

V textu [5] popisuje Dickerson algoritmus pro výpočet tzv. LMT-skeletonu. Ten je podgrafem každé lokálně minimální triangulace, tedy z důsledku 4.1 vyplývá, že je i podgrafem každé triangulace s minimální váhou.

V algoritmu není definováno, v jakém pořadí zpracovávat prvky $CandEdges$, a výsledek není jednoznačný.

Definice 4.7 (LMT-skeleton [5])**.** LMT-skeletonem konečné množiny $V \subseteq \mathbb{R}^2$ nazveme libovolný PSLG $(V, EdgesIn)$, který může vzniknout jako výsledek Algoritmu 9.

Věta 4.10. *V průběhu (i po skončení) algoritmu 9 obsahuje množina $EdgesIn$ jen hrany, které jsou v každé lokálně minimální triangulaci. Časová složitost algoritmu je $O(n^4)$, paměťová $O(n^3)$.*

Algoritmus 9 Výpočet LMT-skeletonu [5]

Vstup: $V \subseteq \mathbb{R}^2$ je konečná množina bodů v rovině.
Výstup: $EdgesIn$ je LMT-skeleton množiny V.

1: $CandTris \leftarrow$ seznam všech prázdných trojúhelníků na množině V
2: $CandEdges \leftarrow$ seznam všech prázdných hran na množině V
3: $EdgesIn \leftarrow \emptyset$
4: Odeber hrany konvexního obalu z $CandEdges$ a přidej je do $EdgesIn$.
5: **for** $\forall e \in CandEdges$ **do**
6: **if** neexistuje $T_1, T_2 \in CandTris$ tvořící čtyřúhelník, ve kterém je e lokálně minimální, **then**
7: $CandEdges \leftarrow CandEdges \setminus \{e\}$
8: Odeber z $CandTris$ všechny trojúhelníky s krajní hranou e.
9: **else if** e neprotíná žádnou hranu v $CandEdges \cup EdgesIn$ **then**
10: $EdgesIn \leftarrow EdgesIn \cup \{e\}$
11: **end if**
12: **end for**

Důkaz: Nejprve dokážeme, že z $CandEdges$ neodebereme žádnou hranu mimo konvexní obal, která je v nějaké lokálně minimální triangulaci. Nechť algoritmus poprvé zpracovává řádek (7) pro hranu e z nějaké lokálně minimální triangulace T. Protože jsme ale z $CandTris$ dosud neodebrali žádný trojúhelník T, dostáváme se do sporu s podmínkou (6).

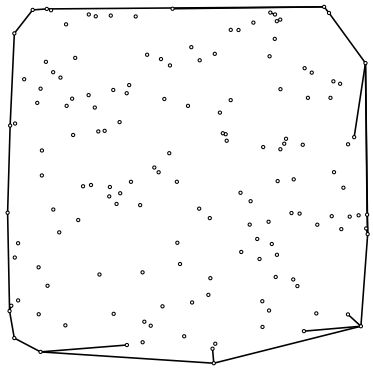
Nechť jsme v průběhu algoritmu přidali hranu e do množiny $EdgesIn$, která není v některé lokálně minimální triangulaci $T = (V, E)$. Na řádku (4) jsme přidali hrany konvexního obalu, a ty zřejmě patří do každé triangulace. V kroku (10) máme zaručeno, že e neprotíná žádná hrana z $CandEdges$. Z první části důkazu ale víme, že $CandEdges \bigcup CH(V)$ obsahuje všechny hrany T, a proto existuje nějaká hrana $f \in CandEdges \bigcap E$ protínající e.

V každém z $O(n^2)$ kroků cyklu přes hrany $CandEdge$ zkoumáme $O(n^2)$ dvojic trojúhelníků. Prázdných trojúhelníků je nejvíce $O(n^3)$ a jdou nalézt v čase $O(n^3)$. □

4.3.2 Rozšířený LMT-skeleton

V algoritmu 9 může odebrání trojúhelníku na řádku (8) změnit výsledek podmínky (9) pro nějakou už zpracovanou hranu e. Kdybychom se k takové hraně vrátili, je možné, že by algoritmus našel větší podgraf triangulací s minimální váhou.

To motivuje k rozšíření algoritmu a cyklus (5) zopakovat pokaždé, když v jeho průběhu odebereme nějakou hranu z $CandEdges$. Složitost algoritmu se zvýší na $O(n^6)$, protože počet kandidátních hran je $O(n^2)$.



Obrázek 4.4: LMT-skeleton 100 rovnoměrně rozmístěných náhodných bodů.

Věta 4.11. *Výsledek rozšířeného algoritmu 9 je jednoznačný i bez definovaného pořadí zpracování hran.*

Důkaz: Nechť $CandTris_1 \neq CandTris_2$ jsou výsledné množiny dvou různých průběhů algoritmu 9 (tj. lišících se pořadím zpracování hran). Podobně označme $CandEdges_i$ a $EdgesIn_i$. Bez újmy na obecnosti $CandTris_1 \not\subseteq CandTris_2$ (jinak prohodíme). Vezměme $T \in CandTris_2 \setminus CandTris_1$, který jsme ze všech takových trojúhelníků odebrali z $CandTris_1$ nejdříve. To mohlo nastat jen při rušení nějaké hrany krajní e trojúhelníku T. Ovšem $e \in CandEdges_2$, tedy existuje dvojice trojúhelníků $T_1, T_2 \in CandTris_2$, ve kterých je e minimální. T_1 i T_2 však musely být při odebírání e i v $CandTris_1$, čímž dostáváme spor a na konci platí $CandTris_1 = CandTris_2$. Odtud pak snadno vyplývá, že $CandEdges_1 = CandEdges_2$, tedy také $EdgesIn_1 = EdgesIn_2$. □

Definice 4.8 (Rozšířený LMT-skeleton [5])**.** Rozšířeným LMT-skeletonem konečné množiny $V \subseteq \mathbb{R}^2$ je PSLG $(V, EdgesIn)$, který vznikne jako výsledek rozšířeného algoritmu 9.

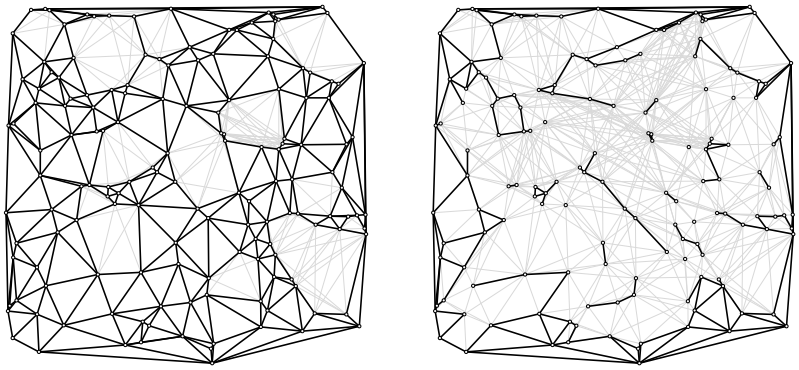
Věta 4.12. *Rozšířený LMT-skeleton konečné množiny $V \subseteq \mathbb{R}^2$ obsahuje jen hrany, které jsou v každé lokálně minimální triangulaci.*

Důkaz: Důkaz je analogický s důkazem věty 4.10. □

Poznámka 4.4. Obrácená implikace neplatí, algoritmus nemusí najít všechny takové hrany.

4.3.3 Modifikovaný LMT-skeleton

Vedle LMT-skeletonu a rozšířeného LMT-skeletonu zavádí Dickerson [5] ještě modifikovaný LMT-skeleton. Pro nalezení MWT stačí, aby počáteční množina kandidátních hran obsahovala libovolnou nadmnožinu hran všech minimálních triangulací. Modifikovaným LMT-skeletonem nazývá Dickerson výsledek rozšířeného algoritmu 9 aplikovaného na množinu hran s diamantovou vlastností, který je jistě podgrafem každé triangulace s minimální váhou. Naše definice bude o něco obecnější:



Obrázek 4.5: Vlevo je rozšířený LMT-skeleton stejné množiny bodů, jako na Obrázku 4.4, včetně zobrazené výsledné množiny $CandEdges \setminus EdgesIn$. Pravý graf znázorňuje průběžný stav algoritmu po pěti z celkových osmi zopakování hlavního cyklu.

Definice 4.9. Nechť $G_G = (V, E_G)$ je SLG obsahující triangulaci. Potom modifikovaným LMT-skeletonem grafu G_G nazveme výsledek rozšířeného algoritmu 9 s počáteční množinou kandidátních hran E_G. Prázdné trojúhelníky se generují jen pro hrany z E_G.

Poznámka 4.5. Důkaz jednoznačnosti ve větě 4.11 funguje na libovolnou počáteční množinu.

Věta 4.13. *Nechť $\mathcal{M} \subseteq LMT(V)$ je neprázdná třída triangulací na množině vrcholů V a $G_G(V, E_G)$ SLG obsahující všechny hrany každé $T \in \mathcal{M}$. Potom modifikovaný LMT-skeleton grafu G_G je podgrafem každé triangulace $T \in \mathcal{M}$ a naopak každá triangulace $T \in \mathcal{M}$ je podgrafem výsledné množiny $CandEdges \cup EdgesIn$.*

Důkaz: Podobný větě 4.10. □

Věta 4.14. *Nechť $G_1 = (V, E_1)$ je podgrafem SLG $G_2 = (V, E_2)$ a G_1 obsahuje triangulaci. Označme $EdgesIn_i$ a $CandEdges_i$ výsledné množiny algoritmu pro výpočet modifikovaného LMT-skeletonu G_i. Potom $EdgesIn_1 \supseteq EdgesIn_2$ a $CandEdges_1 \subseteq CandEdges_2$.*

Důkaz: Stačí mírně upravit důkaz věty 4.11. Kdyby $CandTris_1 \not\subseteq CandTris_2$, dospějeme stejným způsobem ke sporu. Odtud pak plyne, že $CandEdges_1 \subseteq CandEdges_2$ a nakonec $EdgesIn_1 \supseteq EdgesIn_2$. □

Podle předchozích dvou vět můžeme přes modifikované LMT-skeletony korektně počítat podgrafy minimálních triangulací, a to tím lépe, čím méně kandidátů máme na začátku k dispozici. V současnosti jsou takto získané podgrafy aplikované na hrany s diamantovou vlastností základem nejlepších známých heuristik a až na několik známých typů konfigurací většinou vygenerují souvislý podgraf MWT. To byl také důvod, proč jsem modifikované LMT-skeletony použil jako základ vlastní implementace pro výpočet minimálních triangulací, popsaný v kapitole 5.

4.3.4 LMT_k-skeleton

Přestože se tak na náhodných datech děje jen zřídka, nemusí být modifikovaný LMT-skeleton obecné množiny n vrcholů souvislý a může dokonce obsahovat $\Omega(n)$ komponent. Nabízí se otázka, zda by nešlo algoritmus upravit takovým způsobem, aby našel větší počet hran a výsledný podgraf MWT měl méně komponent.

Jedna z možností je popsána v [19]. Lokálně minimální triangulace se zde zobecňují z čtyřúhelníků na větší k-úhelníky a následně se zavede tzv. LMT_k-skeleton. Původní definici z [19] mírně upravíme, ale v podstatě znamená to samé.

Nechť je hrana e součástí triangulace T a $k \geqq 4$. O hraně e pak řekneme, že je lokálně minimální, pokud je na konvexním obalu nebo je součástí minimální triangulace každého l-úhelníku ($4 \leqq l \leqq k$) v T s diagonálou e. l-úhelníky přitom rozumíme jednoduché polygony s obvodem délky l (posloupnost $l+1$ vrcholů), které vzniknou odebráním nějakých hran T.

Je zřejmé, že volbou $k = 4$ dostaneme klasickou lokální minimalitu podle definice 4.5 a zvýšením k vždy zesílíme podmínku minimality. Triangulace s minimální váhou přitom zůstává lokálně minimální pro každé k. Výpočet hran, které se vyskytují v každé lokálně minimální triangulaci, je podobný Dickersonovu algoritmu, jen potřebujeme generovat oproti trojúhelníkům obecné dvojice prázdných disjunktních k_i-úhelníků ($k_1 + k_2 - 2 \leqq k$).

S rostoucím k se podstatně zvyšuje časová i paměťová složitost algoritmu, protože počet možných prázdných k-úhelníků je $O(n^k)$. Paměťová složitost by se dala snížit na $O(n^2)$ podobným trikem, jaký se používá pro generování trojúhelníků v [3], ovšem časová je obecně exponenciální v k. Ve většině případů by se dal algoritmus zrychlit postupným zesilováním lokální minimality (nejprve hledat čtyřúhelníky, pak až pětiúhelníky, ...) a počítat jen s nesouvislými oblastmi z předchozího kroku. V nejhorším případě si tak ale nepomůžeme.

Kdybychom uměli pro nějaké konstantní k dokázat, že algoritmus najde souvislý podgraf MWT, měli bychom k dispozici polynomiální algoritmus řešící MWT. Bohužel se zdá, že pro každé k lze sestrojit protipříklad.

Poznámka 4.6. Dokonce si ani nejsem jistý, zda pro $k = \infty$ vyjde minimální triangulace (nebo její souvislý podgraf). Bylo by potřeba dokázat souvislost mezi minimalitou všech jednoduchých polygonů a globální minimalitou triangulace.

4.3.5 Zobecněná lokální minimalita

V předchozích částech kapitoly jsme se snažili najít postupy urychlující výpočet minimální triangulace (MWT). Nyní se pokusíme stejné myšlenky aplikovat na obecnější problémy omezené a zobecněné minimální triangulace ($CMWT$ a $GMWT$).

Definice 4.10 (Zobecněná lokálně minimální hrana)**.** Řekneme, že hrana $e = \{u, v\}$ triangulace $T \in T_G(V, E_G)$ je lokálně minimální mezi kandidáty E_G, jestliže je v T lokálně minimální nebo druhá diagonála příslušného čtyřúhelníku neleží v E_G.

Definice 4.11 (Zobecněná a omezená lokálně minimální triangulace)**.** Nechť $G_G = (V, E_G)$ je SLG. $T \in T_G(V, E_G)$ nazveme zobecněnou lokálně minimální triangulací ($GLMT$) na E_G, jestliže obsahuje jen lokálně minimální hrany mezi E_G. Množinu všech takových triangulací na V značíme $GLMT(V, E_G)$. Pro PSLG $G_C = (V, E_C)$ dále zavedeme množinu

$$CLMT(V, E_C) = GLMT(V, E_C \cup D(G_C)) \tag{4.15}$$

omezených lokálně minimálních triangulací ($CLMT$) na G_C.

Věty i algoritmy pro lokálně minimální triangulace platí podobně i pro zobecněné a omezené lokálně minimální triangulace. Jediným rozdílem je nová podmínka u lokálně minimálních hran. Pro úplnost uvedeme algoritmus 10 řešící ekvivalent rozšířeného LMT-skeletonu pro třídu $GMWT$.

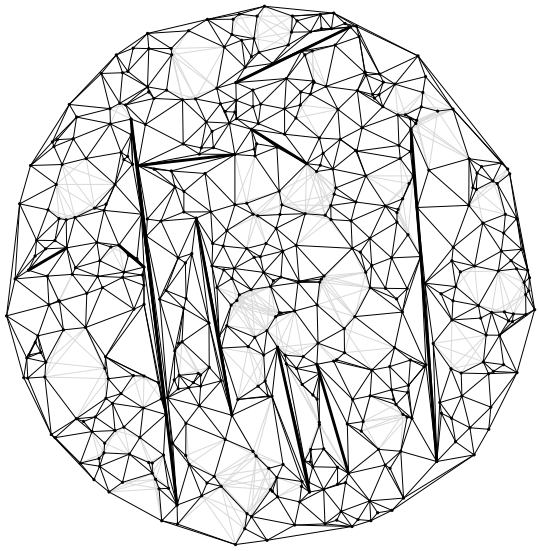
Algoritmus 10 Výpočet rozšířeného $GLMT$-skeletonu

Vstup: (V, E_G) je SLG.
Výstup: $(V, EdgesIn)$ $GLMT$-skeleton grafu (V, E_G).

1: $CandTris \leftarrow$ seznam všech prázdných trojúhelníků na V z hran E_G
2: $CandEdges \leftarrow E_G$
3: $EdgesIn \leftarrow \emptyset$
4: Přesuň hrany konvexního obalu z $CandEdges$ do $EdgesIn$ (když existují).
5: **repeat**
6: $Leave \leftarrow$ true
7: **for** $\forall e \in CandEdges$ **do**
8: **if** neexistuje $T_1, T_2 \in CandTris$ tvořících čtyřúhelník, ve kterém je e lokálně minimální mezi E_G **then**
9: $CandEdges \leftarrow CandEdges \setminus \{e\}$
10: Odeber z $CandTris$ všechny trojúhelníky s krajní hranou e.
11: $Leave \leftarrow$ false
12: **else if** e neprotíná žádnou hranu v $CandEdges \cup EdgesIn$ **then**
13: $EdgesIn \leftarrow EdgesIn \cup \{e\}$
14: **end if**
15: **end for**
16: **until** $Leave$

Poznámka 4.7. Když v E_G není triangulace, pak ji neobsahuje ani výsledná množina $CandEdges$. I v tomto případě je ale výsledek algoritmu jednoznačný a následující definice korektní.

Definice 4.12. Rozšířeným $GLMT$-skeletonem pro SLG (V, E_G) rozumíme výsledný PSLG $(V, EdgesIn)$ algoritmu 10.



Obrázek 4.6: Rozšířený $CLMT$-skeleton 500 bodů.

Složitost algoritmu se oproti rozšířenému LMT-skeletonu zvýší o jeden řád na $O(n^7)$ kvůli testům na existenci diagonály u čtyřúhelníků. To by šlo snadno obejít předpočítáním tabulky s konstantním časem dotazu. V kapitole 5.2 ukážeme, že můžeme, stejně jako u modifikovaných LMT-skeletonů, snížit časovou složitost na $O(n^4)$ a potřebnou paměť na $O(n^2)$.

Podobně se dá definovat rozšířený $CLMT$-skeleton, případně s další množinou kandidátních hran modifikovaný $GLMT$-skeleton a modifikovaný $CLMT$-skeleton.

Poznámka 4.8. Obecně nejde zaměnit modifikovaný LMT-skeleton s $GLMT$-skeletony. U LMT-skeletonu je jiná definice lokální minimality.

Chování $CLMT$-skeletonů jsem prakticky testoval na náhodných množinách bodů s náhodnými omezujícími hranami a zdá se, že podobně jako u klasických LMT-skeletonů zřídkakdy vznikne nesouvislý podgraf $CMWT$. To víceméně splnilo očekávání, protože omezující hrany znamenají pro výpočet podobnou situaci, jako hrany konvexního obalu.

Obecnější případ $GLMT$-skeletonů se těžko vizuálně představuje a testy jsem neprováděl. Ovšem mohlo by být zajímavé zkusit použít tuto heuristiku v problému TRI (hledání triangulace) nebo jiných problémech. Na první pohled TRI s lokální minimalitou příliš nesouvisí, takže by zde asi výsledky mnoho nepřinesly.

Kapitola 5

Rychlý výpočet MWT

Tématem kapitoly je kompletní popis algoritmu, který lze použít pro efektivní výpočet přesné minimální triangulace. Algoritmus je založený na několika známých heuristikách, zejména na výpočtu lineárního očekávaného počtu hran s diamantovou vlastností a na rozšířeném LMT-skeletonu. Přestože pro většinu typů vstupních množin bodů má program polynomiální složitost, lze sestrojit protipříklad, kdy skončí v exponenciálním čase vzhledem k počtu vrcholů.

V textu uvažujeme problém hledání $T \in MWT(V)$, ovšem některé myšlenky jde stejně tak aplikovat na omezené či zobecněné triangulace nebo i zcela odlišné třídy triangulací.

Převážná část popsaného algoritmu je součástí implementace přiložené k diplomové práci. Vynechané části zmíníme na závěr kapitoly společně s následky, které to pro výpočet může znamenat.

5.1 Hledání kandidátních hran

Prvním krokem algoritmu pro výpočet MWT bude vyřadit co nejvíce z celkových $O(n^2)$ hran, nepatřících do žádné minimální triangulace. Z kapitoly 4.2 o zakázaných oblastech víme, že v minimální triangulaci mohou být jen hrany splňující diamantovou vlastnost, kterých je v očekávaném případě pro rovnoměrně rozmístěnou množinu bodů $O(n)$. Vyřazením ostatních hran se tak podstatně zvýší očekávaná efektivita dalších výpočtů.

Nejjednodušším postupem, jak najít hrany s diamantovou vlastností, je pro každou hranu e projít všechny vrcholy a zjistit, zda leží na některé straně zakázané oblasti pro e. Takový algoritmus by ale měl složitost $O(n^3)$, a i v očekávaném případě zřejmě alespoň $\Omega(n^2)$.

V další části textu se pokusíme vylepšit očekávanou složitost pro rovnoměrně rozložené body na $O(n)$ modifikaci algoritmu, jaký popsal Drysdale [6] pro výpočet greedy triangulace. Tvary známých zakázaných oblastí pro MWT jsou podobné jako GT, a proto můžeme stejný postup použít i pro tuto třídu triangulací.

Drysdale provádí výpočet zvlášť pro každý bod $v \in V$ v libovolném pořadí. Nej-

prve rozdělí okolí bodu na konstantní počet stejně velkých úhlů („wedges“) a v každém najde nejbližší vrchol. Poté zvlášť pro každou část prochází body vzestupně podle vzdálenosti od v. Dříve zpracované vrcholy a nejbližší vrcholy ze sousedních částí umožňují v očekávaném případě brzy zajistit, že už dál žádní kandidáti být nemohou, a výpočet lze zastavit. S využitím pravidelné čtvercové sítě $O(n)$ čtverců o straně $\frac{|\mathcal{C}|}{\sqrt{n}}$ Drysdale dokáže, že pro množinu bodů rovnoměrně rozloženou v nějaké konvexní oblasti $\mathcal{C}$ algoritmus pracuje s očekávanou složitostí $O(1)$ pro každý bod, tedy celkově $O(n)$, a najde dohromady $O(n)$ kandidátů na hrany GT.

Vlastní implementované řešení se od Drysdalova liší tím, že okolí bodů zkoumáme celé najednou (tj. ne po jednotlivých intervalech úhlů) a kandidáty na koncové vrcholy hran procházíme ne nutně v setříděném pořadí. Ukážeme, že stačí použít průchod po vrstvách kolem v s postupně se zvyšujícím poloměrem, a přitom v průměru navštívíme jen málo čtverců sítě. Z nalezených kandidátů kolem vrcholu v nakonec vyřadíme všechny, které nemají zablokovanou zakázanou oblast, a plně ji tak využijeme k vygenerování co nejmenšího počtu hran. Celková složitost přitom zůstane lineární v očekávaném případě a umožní nám s pomocí lokálně minimálních triangulací najít v rozumném čase přesnou minimální triangulaci i velkého počtu rovnoměrně rozmístěných bodů.

5.1.1 Kritéria pro vyřazení částí roviny

Předpokládejme, že jsme už prozkoumali kolem vrcholu v nějakou dvojici vrcholů u_1, u_2. V následujících větách dokážeme, že je-li velikost úhlu u_1vu_2 dostatečně malá, můžeme pro nějakou konstantu c vyloučit celou oblast roviny v tomto úhlu, která je od v dál, než $c \cdot \max\{\|v - u_1\|, \|v - u_2|\}$.

Pro úplnost odvodíme nejen konstanty pro diamantovou vlastnost hran MWT, ale budeme uvažovat obecnou třídu triangulací a zakázanou oblast tvaru R^{φ}_{diam} nebo R^{r}_{eye}.

Lemma 5.1. *Nechť $e = (v, u) \in \binom{\mathbb{R}^2}{2}$, $\varphi \in (0, \pi/2)$, $w \in \mathbb{R}^2, w \neq v, u$ a platí:*

$$|\angle uvw| = \alpha \leqq \varphi, \tag{5.1}$$

$$\|v - w\| \leqq \frac{\sin\varphi}{\sin(\varphi + \alpha)}\|e\|. \tag{5.2}$$

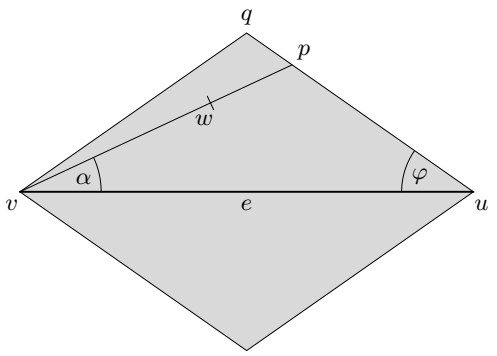
Potom $w \in R^{\varphi}_{diam}(e)$.

Důkaz: Pro $\alpha = 0$ leží w na úsečce vu a tvrzení platí. Nechť $\alpha \in (0, \varphi]$. Označme q vrchol kosočtvercové oblasti $R^{\varphi}_{diam}(e)$ na stejné straně od hrany e, jako je bod w, a p průsečík přímek vw a uq (viz obrázek 5.1). Podle sinové věty dostáváme pro trojúhelník (v, u, p):

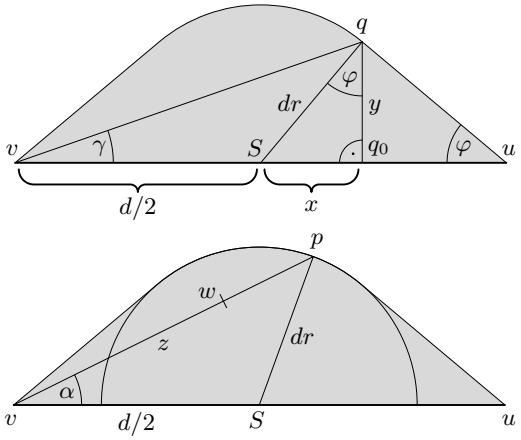
$$\|v - p\| = \frac{\sin\varphi}{\sin(\pi - \varphi - \alpha)}\|e\| = \tag{5.3}$$

$$= \frac{\sin\varphi}{\sin(\varphi + \alpha)}\|e\| \geqq \|v - w\|. \tag{5.4}$$

Odtud plyne, že w leží na úsečce vp a $w \in R^{\varphi}_{diam}(e)$. □



Obrázek 5.1: Důkaz lemma 5.1.



Obrázek 5.2: Důkaz lemma 5.2.

Lemma 5.2. *Nechť* $e = (v, u) \in \binom{\mathbb{R}^2}{2}$, $r \in (0, 1/2)$, $w \in \mathbb{R}^2, w \neq v, u$, $\alpha = |\angle uvw|$ *a*

$$\gamma = \arctan \frac{2r\sqrt{1-4r^2}}{1+4r^2}. \tag{5.5}$$

Platí-li

$$\|v-w\| \leqq \begin{cases} \|e\|, & \alpha = 0, \\ \|e\| \cdot (2r)/\sin(\arcsin 2r + \alpha), & \alpha \in (0, \gamma], \\ \|e\| \cdot \left(\cos\alpha + \sqrt{4r^2 - \sin^2\alpha}\right)/2, & \alpha \in (\gamma, \arcsin 2r], \end{cases} \tag{5.6}$$

pak $w \in R^r_{eye}$.

Důkaz: Pro $\alpha = 0$ leží w na úsečce vu a tvrzení platí. Označme jako na obrázku 5.2 S střed úsečky vu, q bod, kde přechází hranice $R^r_{eye}(e)$ z úsečky na kružnici, q_0 patu kolmice z bodu q na úsečku vu a $d = \|e\|$, vše na stejné straně hrany e jako je vrchol w. Dále označme p průsečík přímky vw s hranicí $R^r_{eye}(e)$ a d délku úsečky vp.

Z trojúhelníku (S, u, q) dostáváme pro $\varphi = |\angle Suq|$:

$$\sin\varphi = \frac{2dr}{d} = 2r, \tag{5.7}$$

$$\cos\varphi = \sqrt{1 - \sin^2\varphi} = \sqrt{1 - 4r^2}. \tag{5.8}$$

Dále v trojúhelníku (S, q_0, q) dopočítáme:

$$x = dr \cdot \sin\varphi = 2dr^2, \tag{5.9}$$

$$y = dr \cdot \cos\varphi = dr\sqrt{1 - 4r^2}. \tag{5.10}$$

Potom pro $\gamma = |\angle q_0 vq|$ platí:

$$\tan\gamma = \frac{2y}{d + 2x} = \frac{2dr\sqrt{1 - 4r^2}}{d + 4dr^2} = \frac{2r\sqrt{1 - 4r^2}}{1 + 4r^2}. \tag{5.11}$$

Pro úhel $\alpha \in (0, \gamma]$ je situace podobná jako v lemma 5.1 a tvrzení lze dokázat sinovou větou pro trojúhelník (v, u, p).

Zbývá možnost, kdy $\alpha \in (\gamma, \arcsin 2r] = (\gamma, \varphi]$. V tomto případě ovšem stačí použít cosinovou větu v trojúhelníku (v, S, p):

$$0 = \left(\frac{d}{2}\right)^2 + z^2 - 2\left(\frac{d}{2}\right) z\cos\alpha - (dr)^2, \tag{5.12}$$

$$z = \frac{d\cos\alpha + \sqrt{d^2\cos^2\alpha - d^2 + 4d^2r^2}}{2} = \tag{5.13}$$

$$= \frac{d\cos\alpha + d\sqrt{4r^2 - \sin^2\alpha}}{2}. \tag{5.14}$$

Podle předpokladů lemma leží bod w na úsečce vp, a tedy i v oblasti $R^r_{eye}(e)$. □

Věta 5.1. *Nechť $\mathcal{M} \subseteq \mathcal{T}$ je třída triangulací, $\alpha > 0$, $d > 0$ a $c > 1$. Dále nechť $V \subseteq \mathbb{R}^2$ je konečná množina vrcholů, $\{v, u_1\}, \{v, u_2\}, \{v, u\} \in \binom{V}{2}$, $u_1 \neq u_2$, $|\angle u_1 v u_2| \leqq \alpha$, $\|v - u_i\| \leqq d \forall i \in \{1, 2\}$, $\|v - u\| \geqq c \cdot d$ a vrchol u leží úhlu $u_1 v u_2$.*

Jestliže je navíc splněn alespoň jeden z následujících bodů, pak hrana $e = \{v, u\}$ neleží v žádné triangulaci z množiny $\mathcal{M} \cap T(V)$.

(i) $\varphi \in (0, \pi/4]$, $\alpha \leqq \varphi$, $R^\varphi_{diam}(e)$ je zakázanou oblastí hrany e pro třídu $\mathcal{M}$ a

$$c \geqq \frac{\sin(\varphi + \alpha)}{\sin\varphi}, \tag{5.15}$$

(ii) $r \in (0, \sqrt{2}/4]$, $\alpha \leqq \arcsin 2r$, $R^r_{eye}(e)$ je zakázanou oblastí hrany e pro třídu $\mathcal{M}$ a

$$c \geqq \begin{cases} \sin(\arcsin 2r + \alpha)/(2r), & \alpha \in (0, \gamma], \\ 2/\left(\cos\alpha + \sqrt{4r^2 - \sin^2\alpha}\right), & \alpha \in (\gamma, \arcsin 2r], \end{cases} \tag{5.16}$$

kde

$$\gamma = \arctan \frac{2r\sqrt{1-4r^2}}{1+4r^2}. \tag{5.17}$$

Důkaz:

(i) Jelikož vrchol u leží v úhlu u_1vu_2 a $|\angle u_1vu_2| \leqq \alpha$, musí být velikost obou úhlů u_ivu menší nebo rovna α. Snadno lze ověřit, že pro $\varphi \in (0, \pi/4]$ je

$$\frac{\sin\varphi}{\sin(\varphi + |\angle u_ivu|)} \geqq \frac{\sin\varphi}{\sin(\varphi+\alpha)}. \tag{5.18}$$

Splnili jsme tak předpoklady lemma 5.1 a pro vrcholy u_i dostáváme $u_i \in R^{\varphi}_{diam}(e)$. Ovšem v_i nemůžou být oba na jedné straně od hrany e. Podle definice 4.2 tak vrcholy V blokují zakázanou oblast $R^{\varphi}_{diam}(e)$ hrany e a z věty 4.5 pak vychází, že e nepatří do žádné triangulace T z množiny $\mathcal{M} \cap T(V)$.

(ii) Analogicky jako (i), jen použijeme lemma 5.2. □

Poznámka 5.1. Věta 5.1 zobecňuje „Corollary 1“ v [6] na libovolnou třídu triangulací se zakázanou oblastí ve tvaru kosočtverce nebo „oka“. Snadno jde pro daný koeficient porovnávající délky hran (c) spočítat potřebný počet klínů kolem vrcholů ($\lceil 4\pi/\alpha \rceil$) a naopak z jejich počtu určit potřebný koeficient. Pro $c = 2$ a oblast R^{GT}_{eye} vychází $\alpha \approx 25{,}84°$, což přesně odpovídá poznámce na konci [6] o nejlepším známém úhlu.

5.1.2 Čtvercová síť

Algoritmus bude potřebovat efektivně vyhledávat koncové body v okolí právě zpracovávaného počátečního vrcholu, pokud možno vzestupně podle vzdálenosti. Navíc pomocí dokázaného kritéria půjde některé větší souvislé části roviny vynechat. K tomuto účelu dobře poslouží průchod „do šířky“ pravidelnou čtvercovou sítí.

Předpokládejme, že vstupní množina n bodů V je rovnoměrně rozložená v nějaké konvexní oblasti $\mathcal{C} \subseteq \mathbb{R}$ o obsahu $|\mathcal{C}| > 0$.

Rovinu rozdělíme na pravidelnou síť čtverců $B_{i,j}$ o straně $\frac{|\mathcal{C}|}{\lceil\sqrt{n}\rceil}$:

$$B_{i,j} = \left[\frac{i\cdot|\mathcal{C}|}{\lceil\sqrt{n}\rceil}, \frac{(i+1)\cdot|\mathcal{C}|}{\lceil\sqrt{n}\rceil}\right) \times \left[\frac{j\cdot|\mathcal{C}|}{\lceil\sqrt{n}\rceil}, \frac{(j+1)\cdot|\mathcal{C}|}{\lceil\sqrt{n}\rceil}\right), \quad i,j \in \mathbb{Z}. \tag{5.19}$$

Potom počet čtverců $B_{i,j}$ takových, že $\mathcal{C} \cap B_{i,j} \neq \emptyset$, je $O(n)$ (uvnitř i na obvodu) a každý z těchto čtverců obsahuje konstantní očekávaný počet vrcholů z V.

Datová struktura bude obsahovat všechny čtverce zasahující do $\mathcal{C}$ (úplně i částečně) a ke každému do paměti uložíme jejich pozici, odkazy na sousední čtverce a spojový seznam vrcholů. Aby šlo v konstantním čase lokalizovat čtverec příslušný danému bodu, vytvoříme navíc pole přes použité indexy i (je jich lineárně mnoho). Pro každé i pak do

tohoto pole uložíme další pole obsahující všechny neprázdné $B_{i,j}$. Celkově má popsaná datová struktura lineární paměťovou složitost pro libovolnou konvexní množinu $\mathcal{C}$.

V přiloženém programu pro výpočet MWT pro jednoduchost počítáme s rovnoměrným rozložením bodů jen v nějaké obdélníkové oblasti. Vyhledávací síť pro n vrcholů tak snadno vytvoříme s lineární časovou složitostí v nejhorším případě.

Kdybychom ji chtěli rozšířit na libovolnou konvexní oblast $\mathcal{C}$, bylo by nejprve potřeba spočítat konvexní obal množiny V, což by šlo v nejhorším čase $O(n \log n)$ nebo očekávaně v $O(n)$. Složitost výpočtu obsahu $\mathcal{C}$ a sestavení datové struktury by vyšla opět nejhůře lineární.

Při zkoumání okolí vrcholu $v \in V$ vždy prohledáme nějakou souvislou podmnožinu čtverců (souvislou v grafu sousednosti čtverců), počínaje čtvercem obsahujícím v. Podrobněji tento průchod popíšeme dále.

5.1.3 Intervaly úhlů

Vedle pravidelné čtvercové sítě zavedeme pro výpočet kandidátních hran ještě jednu pomocnou strukturu. Dokázané kritérium pro vyloučení částí roviny v okolí počátečního vrcholu v jde použít jen tehdy, je-li mezi některou dvojicí vrcholů dostatečně malý úhel ($\leqq \alpha$).

Označme $(v_0, \ldots, v_{m-1})$ posloupnost dříve zpracovaných vrcholů do vzdálenosti d od počátečního vrcholu v, setříděných podle úhlu kolem v. Úhlem vrcholu v_i rozumíme $\beta_i \in [0, 2\pi)$ takové, že

$$v_i = v + d_i \cdot (\cos \beta_i, \sin \beta_i), \quad d_i > 0. \tag{5.20}$$

Úhel $\beta \in \mathbb{R}$ nazveme uzavřeným, jestliže existuje index i takový, že polopřímka vycházející z vrcholu v pod úhlem β leží v orientovaném úhlu $v_i v v_{(i+1) \bmod m}$ (tj. v úhlu proti směru hodinových ručiček od v_i) a $|\angle v_i v v_{(i+1) \bmod m}| \leqq \alpha$.

Algoritmus bude potřebovat opakovaně zjišťovat, zda jsou dané úhly nebo celé intervaly úhlů uzavřené, navíc občas do struktury přidá nový vrchol. Problém by řešil například obyčejný seznam nebo vyvážený strom, který by zaručoval čas $O(\log m)$ dotazu na konkrétní úhel a stejnou složitost přidání prvku. Protože je ale úhel α konstantní pro celý průběh výpočtu, můžeme snadno vylepšit popsané operace na $O(1)$. I bez této úpravy by nakonec vyšla lineární očekávaná složitost výpočtu kandidátních hran, ovšem snížíme ji alespoň o konstantu a navíc dostaneme lepší složitost v nejhorším případě.

Interval $[0, 2\pi)$ rozdělíme na $k \geqq 2\pi/\alpha$ stejně velkých částí. Pro každou část W_i si zapamatujeme index krajního vrcholu v_{r_i} (resp. v_{l_i}), který patří do intervalu W_i a má nejmenší (největší) úhel β_{r_i} (β_{l_i}). Úhel mezi β_{r_i} a β_{l_i} (pokud existují) je vždy uzavřený, zbylé úhly závisí jen na krajních vrcholech sousedních intervalů. Vrcholy, které nejsou krajní v žádném W_i, tak nic neovlivní, a nemusíme je ve struktuře uchovávat.

Dotaz na uzavřenost daného úhlu β nebo intervalu úhlů lze snadno provést v konstantním čase $O(k)$, stejně jako přidání nového prvku. Většinou dokonce prozkoumáme jen velmi málo intervalů W_i.

5.1.4 Prohledání okolí vrcholu

Nechť c a α jsou konstanty z věty 5.1 a pro množinu V vstupních vrcholů máme vytvořenou čtvercovou síť. Popíšeme, jak nalézt kandidátní hrany vycházející z konkrétního vrcholu v.

Algoritmus nejprve inicializuje strukturu pro intervaly úhlů popsanou v předchozí části. Rovinu budeme procházet po jednotlivých vrstvách

$$L_i = \{x; r_{i-1} > \|x - v\| \leqq r_i\} \tag{5.21}$$

kolem vrcholu v, kde $r_0 = 0$, $r_1 = \Theta(|\mathcal{C}|/\sqrt{n})$ a $r_i = r_1 \cdot c^{i-1}$.

Hlavní myšlenkou je, že pomocí libovolné dvojice vrcholů (v_1, v_2) z vrstev $\bigcup_{j=1}^{i-2} L_j$ svírající s v úhel menší nebo roven α můžeme vyřadit celou část oblasti $\bigcup_{j=i}^{\infty} L_j$ v úhlu v_1vv_2. Ve struktuře pro intervaly úhlů tedy budeme uchovávat vrcholy $\bigcup_{j=1}^{i-2} L_j$ pro aktuálně zpracovávané i, a vynecháme vrcholy i čtverce se zablokovaným úhlem. Postup podrobněji vystihuje zápis algoritmu 11 v pseudokódu.

Věta 5.2. *Nechť $\mathcal{M} \subseteq \mathcal{T}$ je třída triangulací se zakázanou oblastí R^{φ}_{diam} nebo R^{r}_{eye}, $V \subseteq \mathbb{R}^2$ je konečná množina n vrcholů v konvexní oblasti $\mathcal{C}$ a konstanty α, c splňují podmínky věty 5.1. Potom za předpokladu, že dostane na vstupu čtvercovou síť nad oblastí $\mathcal{C}$ popsanou v kapitole 5.1.2 a libovolný vrchol $v \in V$, skončí algoritmus 11 výpočet nejhůře v čase $O(n)$.*

Důkaz: Z volby poloměru $r_1 = \Omega(\frac{|\mathcal{C}|}{\sqrt{n}})$ první vrstvy na řádku (1) snadno plyne, že každý čtverec sítě má neprázdný průnik nejvýše s $O(1)$ vrstvami a že celkový počet neprázdných vrstev je $O(n)$. Proto počet zopakování cyklu (5) je $O(n)$. Krok (7) provedeme pro každý čtverec tolikrát, do kolika zasahuje vrstev, tj. dohromady celkem $O(n) \cdot O(1) = O(n)$. Čtverce v síti procházíme nejvýše jednou a i celková složitost cyklu (11) musí být $O(n)$ (pozn: zpracované čtverce můžeme označit nějakou speciální hodnotou). Poslední cyklus, (21), má podobně jako (7) celkovou složitost $O(n) \cdot O(1) = O(n)$. Množiny vrcholů a čtverců lze reprezentovat spojovými seznamy. Všechny pro ně použité operace tak mají konstantní složitost, což podle kapitoly 5.1.3 platí i pro strukturu W. □

Poznámka 5.2. Z věty 5.2 přímo vyplývá konečnost algoritmu 11.

Věta 5.3. *Za stejných předpokladů, jako ve větě 5.2, najde algoritmus 11 koncové vrcholy $w \in C$ všech hran $\{v, w\}$, které patří do nějaké triangulace $T \in \mathcal{M} \cap T(V)$.*

Důkaz: Předpokládejme pro spor, že algoritmus nenajde nějakou hranu $\{v, w\} \in E(T)$, kde $T \in \mathcal{M} \cap T(V)$, tj. vrchol w není na konci algoritmu prvkem množiny C.

Označme $B_0, \ldots, B_l$ posloupnost čtverců na úsečce vw setříděnou vzestupně podle vzdálenosti od v a B_j první čtverec této posloupnosti, který algoritmus vynechá. Zřejmě $j > 0$, neboť první čtverec posloupnosti obsahuje vrchol v. Protože jsme ovšem zpracovali sousední čtverec B_{j-1} čtverce B_j, musel být celý interval úhlů čtverce B_j na řádku

Algoritmus 11 Výpočet kandidátů v okolí vrcholu

Vstup: Inicializovaná čtvercová síť pro vrcholy V, $v \in V$.
Výstup: $C \subseteq V \setminus \{v\}$ množina kandidátů na koncové vrcholy hran z v.

$d \leftarrow \Theta(\frac{|\mathcal{C}|}{\sqrt{n}})$ ▷ Poloměr r_1 vrstvy L_1
$Cands_1, Cands_2, Cands_3 \leftarrow \emptyset$
$Buckets_3 \leftarrow \{$čtverec obsahující vrchol $v\}$
Inicializace struktury W pro k intervalů úhlů.
repeat ▷ Zpracování vrstvy L_i
  $Cands_4, Buckets_4 \leftarrow \emptyset$ ▷ Vyprázdníme L_{i+1}
  while $\exists B, B \in Buckets_3$ **do** ▷ Zpracujeme čtverce L_i
    $Buckets_3 \leftarrow Buckets_3 \setminus \{B\}$
    if úhel alespoň jednoho bodu B není zablokovaný **then**
      $Cands_3 \leftarrow Cands_3 \cup \{w; w \neq v, w$ je vrchol ve čtverci $B\}$
      while $\exists B_0$ sousední ještě nezpracovaný čtverec k čtverci B **do**
        if celý B_0 je od v dále než d **then** ▷ B_0 zpracujeme až v L_{i+1}
          $Buckets_4 \leftarrow Buckets_4 \cup \{B_0\}$
        else ▷ B_0 zasahuje do L_i
          $Buckets_3 \leftarrow Buckets_3 \cup \{B_0\}$
        end if
      end while
    end if
  end while
  $Cands_5 \leftarrow \emptyset$
  for $\forall w, w \in Cands3$ **do**
    if úhel vrcholu w není zablokovaný **then**
      if $\|v - w\| > d$ **then** ▷ $w \in L_{i+j}, j > 0$
        $Cands_4 \leftarrow Cands_4 \cup \{w\}$
      else ▷ $w \in L_i$
        $Cands_5 \leftarrow Cands_5 \cup \{w\}$
      end if
    end if
  end for
  Přidej vrcholy $Cands_2$ do struktury W.
  $Cands_1 \leftarrow Cands_1 \cup Cands_2$ ▷ $\subseteq \bigcup_{j=1}^{i-1} L_j$
  $Cands_2 \leftarrow Cands_5$ ▷ $\subseteq L_i$
  $Cands_3 \leftarrow Cands_4$ ▷ $\subseteq \bigcup_{j=i+1}^{\infty} L_j$
  $Buckets_3 \leftarrow Buckets_4$
  $d \leftarrow d \cdot c$ ▷ Poloměr další vrstvy r_{i+1}
until $Cands_2 \cup Cands_3 \cup Buckets_3 = \emptyset$
$C \leftarrow Cands_1$

(9) při průchodu vrstvy L_i uzavřený. To ale nemůže nastat. Body B_j na úsečce vw, a tedy i vzdálenější vrchol w, by musely být uzavřené, čímž se dostáváme do sporu s větou 5.1 pro w.

Zpracování koncového vrcholu w úsečky vw na řádku (26) lze dokázat analogicky. Vrchol se tak dostane do množiny $Cands_5$ a v další části výpočtu přes $Cands_2$ a $Cands_1$ do výsledné množiny C. □

Poznámka 5.3. Obráceně implikace neplatí – algoritmus může vrátit i hrany se zablokovanou zakázanou oblastí a dokonce i hrany, které nejsou diagonálami mezi V.

Věta 5.4. *Za stejných předpokladů, jako ve větě 5.2, označme C výslednou množinu vrcholů z algoritmu 11. Nechť $w \in C$ a $\{v, w\}$ je hrana se zablokovanou zakázanou oblastí vrcholy V. Potom i $C \subseteq V$ blokuje tuto oblast.*

Důkaz: Označme R zakázanou oblast hrany $e = \{v, w\}$ pro třídu triangulací $\mathcal{M}$. Vezměme vrchol $u \in R \cap (V \setminus C)$ takový, že má na dané straně e nejmenší možnou velikost úhlu $\angle wvu$. Pro tvar zakázaných oblastí z věty 5.1 zřejmě platí, že body oblasti mají od vrcholu v menší vzdálenost než vrchol w. Proto $\|v-u\| < \|v-w\|$. Vrchol u nemohl být v množině C jen tehdy, když byl jeho úhel v průběhu algoritmu zablokovaný, tedy musí existovat dvojice vrcholů $u_1, u_2 \in C$ blokujících vrchol u. Bez újmy na obecnosti předpokládejme, že u_2 svírá s vw větší úhel než u (definujeme tak pořadí očíslování u_1, u_2). Kdyby u_1 byl na druhé straně od hrany e než vrchol u, dostali bychom se do sporu, neboť dvojice u_1, u_2 by blokovala w a $w \notin C$. V opačném případě u_1 musí ležet v oblasti R ($\|vu_1\| < \|v-u\| < \|v-w\|$) a je na stejné straně jako u. Analogicky lze najít podobný vrchol z C na druhé straně R, který spolu s u_1 blokuje zakázanou oblast R. □

Poznámka 5.4. Kdybychom chtěli vygenerovat právě množinu kandidátních hran (ne jen jejich nadmnožinu), které nejsou zablokované vrcholy V, stačí podle věty 5.4 výsledné kandidáty z algoritmu 11 otestovat jen vůči množině C. Filtrace hran se složitostí $O(|C|^2)$ jde snadno doplnit na konec algoritmu. Další možností je zabudovat filtraci přímo do hlavního cyklu a testovat jen vrcholy z příslušných vrstev a intervalů úhlů, čímž se podstatně sníží konstanta u časové složitosti.

Hrany obsahující vrchol by šlo z výsledných kandidátů vyřadit i rychlejším algoritmem s časovou složitostí $O(|C| \log |C|)$ v nejhorším případě.

5.1.5 Výpočet grafu kandidátů

Spustíme-li algoritmus 11 postupně na všechny vrcholy V, vyjde graf obsahující každou triangulaci z třídy $\mathcal{M}$. Postup stručně popisuje algoritmus 12. Časová složitost v nejhorším případě je $O(n^2)$, případně $O(n^2 \log n)$ nebo $O(n^3)$ s filtrací hran podle poznámky 5.4.

Algoritmus 12 Výpočet kandidátů

Vstup: $V \subseteq \mathbb{R}^2$ konečná množina.
Výstup: (V, E) graf kandidátů na hrany triangulací z dané třídy.

```
1: Vytvoř čtvercovou síť pro vrcholy V.
2: Vytvoř prázdný graf (V, E) a jednoznačně očísluj vrcholy V.
3: for ∀v_i ∈ V do
4:     Spočítej C množinu kandidátů na koncové vrcholy hran z v_i.
5:     for ∀v_j ∈ C do
6:         if i < j then                ▷ Ošetření dvojnásobného přidání hran
7:             E ← E ∪ {{v_i, v_j}}
8:         end if
9:     end for
10: end for
```

5.1.6 Očekávaná složitost

Očekávaná složitost a počet nalezených kandidátů algoritmem 12 pro vstupní body rovnoměrně rozložené v konvexní množině $\mathcal{C}$ by šla odvodit podobným způsobem, jaký předvedl Drysdale [6]. Využil by se fakt, že bereme poloměr první vrstvy $r_1 = O(\frac{|\mathcal{C}|}{\sqrt{n}})$ a další vrstvy zvětšujeme vždy jen o konstantu. Prozkoumaná oblast se tak příliš neliší oproti [6]. Jediný problém s nelinearitou by mohl vzniknout v blízkosti hranice oblasti $\mathcal{C}$, ovšem i v našem případě by šla aplikovat podobná myšlenka a dodefinovat uzavřenost úhlů, jestliže hranice vrstvy opustí $\mathcal{C}$. Pro čtvercový tvar $\mathcal{C} = (0, a)^2$ se tak děje automaticky, protože čtverce jsou vždy celé uvnitř nebo mimo $\mathcal{C}$. Očekávanou složitost ještě ovlivní počet prozkoumaných vrstev, ale ten nemůže být díky $r_1 = \Omega(\frac{|\mathcal{C}|}{\sqrt{n}})$ asymptoticky větší než počet prozkoumaných čtverců.

Dlouhé výpočty pravděpodobnosti nejsou hlavním cílem této diplomové práce a nadále se spokojíme jen s předpokladem, že najdeme pro každý vrchol očekávaný počet $O(1)$ kandidátů v očekávaném čase $O(1)$. Kdyby tomu tak nebylo, vždy by šel aplikovat původní Drysdalův algoritmus, a to nejen pro výpočet GT, ale zřejmě i MWT ($R^{GT}_{eye}(e) \subset R^{MWT}_{diam}(e)$).

Celková očekávaná složitost algoritmu 12 je za zmíněných předpokladů lineární a v průměrném případě najde $O(n)$ kandidátů na hrany triangulací. Bez zvýšení očekávané složitosti také můžeme aplikovat filtraci hran popsanou v poznámce 5.4. Výsledný SLG pak bude vždy obsahovat právě všechny hrany s nezablokovanou zakázanou oblastí.

5.2 Aplikace poznatků o LMT

V předchozí části jsme popsali způsob, jak v očekávaném lineárním čase najít pro rovnoměrně rozmístěné vrcholy v konvexní oblasti průměrně $O(n)$ kandidátních hran s dia-

mantovou vlastností na hrany minimálních triangulací. V nejhorším případě algoritmus funguje v čase $O(n^3)$ a vygeneruje $O(n^2)$ hran.

Nyní se podíváme, jak z těchto kandidátů co nejrychleji získat modifikovaný LMT-skeleton (definice 4.9) společně s podmnožinou zbylých kandidátů na doplnění MWT.

Pro implementaci jsem zvolil mírně upravený algoritmus popsaný Beiroutim v [3]. Hlavní výhodou proti Dickersonovu přístupu z kapitoly 4.3 je to, že dostaneme lepší časovou i paměťovou složitost v nejhorším případě a velmi dobrou $O(d^3 \cdot n)$ složitost pro d maximální stupeň vrcholu vstupního grafu kandidátů. Jak víme, očekávaný počet hran s diamantovou vlastností je pro náhodný bod $O(1)$, tedy celková očekávaná složitost bude mít blízko k lineární.

V této práci doplníme některé důkazy správnosti, které Beirouti neuvádí, a také zmíníme jeden drobný teoretický problém, který by se mohl při výpočtu vyskytnout. Navrhneme, jak by šel tento problém vyřešit se zachováním očekávané složitosti algoritmu. Výsledným grafem pak bude vždy právě modifikovaný LMT-skeleton.

Algoritmus z [3] dělá v podstatě to samé, jako Dickersonův algoritmus [5]. V paměti ovšem nebude potřeba uchovávat seznam prázdných trojúhelníků (menší paměťová složitost) a hlavní cyklus propojíme s hledáním dvojic trojúhelníků v okolí hrany, abychom nemuseli každou hranu zpracovávat $O(n^2)$-krát.

Poznámka 5.5. Práce [3] popisuje další (téměř) lineární variantu hledání kandidátů na hrany MWT.

5.2.1 Prázdné trojúhelníky

Nechť $s = (u, v)$ je orientovaná hrana v SLG $G = (V, E)$ a u orientovaných hran máme uložené informace z definice 2.2. Dále pro s mějme k dispozici proměnné s_l, s_r a s_f, představující odkazy na orientované hrany G. Popíšeme nyní algoritmus, který v lineárním čase a přesně definovaném pořadí projde všechny prázdné trojúhelníky (neobsahující vrchol) nalevo od orientované hrany s. Tím nahradíme potřebné operace na množině $CandTris$ z algoritmu 9, aniž bychom trojúhelníky museli uchovávat v paměti.

Věta 5.5. *Algoritmus 13 najde pro SLG $G = (V, E)$ a orientovanou $s \in S(E)$ všechny prázdné trojúhelníky nalevo od s.*

Důkaz: Budeme postupovat trochu netradičně a místo přímého důkazu sporem spíše odvodíme, jak obecný algoritmus procházející orientované hrany proti směru hodinových ručiček může vypadat.

Nechť je algoritmus v nějakém konkrétním stavu s orientovanými hranami s_l a s_r nalevo od s. Všechny možné vzájemné polohy s_l a s_r znázorňuje obrázek 5.3, kde pro pevnou pozici s a s_r rozdělíme polorovinu vlevo od s na oblasti pro koncový vrchol s_l.

Nyní rozebereme tyto vzájemné polohy a zjistíme, zda lze posunout levou nebo pravou hranu, aniž bychom minuli prázdný trojúhelník.

- (6),(8): Nemůže nastat, jinak by SLG G obsahoval neprázdné hrany.

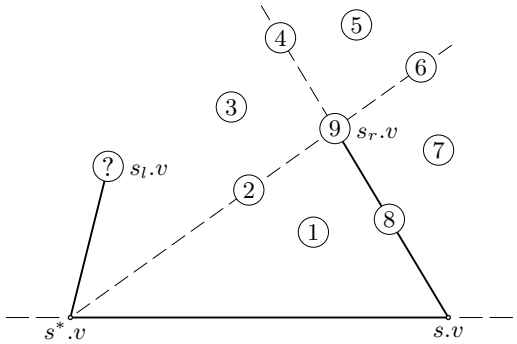
Algoritmus 13 Průchod prázdných trojúhelníků

Vstup: (V, E) je SLG, $s \in S(E)$.

```
1: WalkInit(s)
2: while WalkNext(s) do
3:     ...                    ▷ Provede se pro každý prázdný trojúhelník (s*.v, s.v, s_l.v)
4: end while

5: function WalkNext(s)
6:     s_r ← s_r.ccw
7:     while s_l.v ≠ s_r.v do
8:         if s_l = s_f ∨ s_r = s* then      ▷ Prošli jsme všechny levé nebo všechny pravé hrany
9:             return false
10:        else if s_l.v je vpravo od s_r then
11:            s_l ← s_l.ccw                                  ▷ Posuneme levou hranu
12:        else
13:            s_r ← s_r.ccw                                  ▷ Posuneme pravou hranu
14:        end if
15:    end while
16:    return true                                            ▷ Našli jsme trojúhelník
17: end function

18: procedure WalkInit(s)
19:    s_l ← s.ccw                          ▷ První hrana z s*.v nalevo od s (když existuje)
20:    s_r ← s*              ▷ Potřebujeme první hranu z s.v nalevo od s (když existuje)
21:    repeat
22:        s_r ← s_r.cw
23:    until s_r.v není nalevo od s
24:    s_f ← s                ▷ Najdeme první hranu z s*.v, která už není nalevo od s
25:    repeat
26:        s_f ← s_f.ccw
27:    until s_f.v není nalevo od s
28: end procedure
```



Obrázek 5.3: Průchod prázdných trojúhelníků.

- (4),(5),(6),(7),(8): Lze posunout s_l, neboť každá další orientovaná hrana s_r má $s_l.v$ napravo a s_l s ní nemůže tvořit trojúhelník.
- (2),(3),(4),(5),(6): Lze posunout s_r, neboť každá další orientovaná hrana s_l má $s_r.v$ napravo a s_r s ní nemůže tvořit trojúhelník.
- (1),(2),(8): Trojúhelník $(s^*.v, s.v, s_r.v)$ je neprázdný a s_r zřejmě není součástí jiného. Proto můžeme posunout s_r.
- (9): Našli jsme trojúhelník. Žádný další trojúhelník neobsahuje s_l ani s_r a obojí lze posunout.

Snadno jde ověřit, že funkce WalkInit inicializuje hodnoty s_l a $s_r.ccw$ tak, aby ukazovaly na první (ve smyslu úhlu) orientované hrany nalevo od s, nebo když takové hrany neexistují, aby při prvním spuštěním WalkNext skončil algoritmus s hodnotou false.

Při vzájemných polohách (5),(6),(7) hran s_l a s_r posouváme vždy s_l o jednu hranu proti směru hodinových ručiček, ve zbylých polohách pak hranu s_r. Podle odvozených tvrzení tak algoritmus nemine žádný prázdný trojúhelník, a protože skončí až prozkoumáním všech s_l nebo s_r nalevo od s, najde všechny prázdné trojúhelníky nalevo od s. □

Věta 5.6. *Algoritmus 13 má pro SLG $G = (V, E)$ a $s \in S(E)$ časovou složitost $O(d_1 + d_2)$, kde d_i jsou počty orientovaných hran nalevo od s v G.*

Důkaz: Při inicializaci projdeme právě všechny takové hrany a konstantní počet dalších hran. Celková složitost volání funkcí WalkNext je opět $O(d_1 + d_2)$, protože v každé iteraci cyklu posuneme jednu z hran a nikdy se nevracíme. □

Poznámka 5.6. Algoritmus v obecném případě může najít i neprázdné trojúhelníky, a to i v případě, kdy G obsahuje kompletní triangulaci. Problém je společný pro náš algoritmus i pro výpočet popsaný v [3]. Snadno jde sestrojit protipříklad grafu s triangulací,

kdy pro nějakou hranu vrátí už první volání WalkNext neprázdný trojúhelník. Uvidíme však, že s trochou opatrnosti takové trojúhelníky nevedou k problémům při řešení MWT a jen málokdy najdeme podgraf lišící se od výsledku Dickersonova algoritmu. Dokážeme také, že se dá problému zcela vyhnout.

Následující větu použijeme pří hledání přesného modifikovaného LMT-skeletonu.

Věta 5.7. *Nechť (u, v, w) je trojúhelník nalevo od $s = (u, v)$ nalezený algoritmem 13 v SLG $G = (E, V)$ a nechť pro každý vrchol x ležící uvnitř trojúhelníku (u, v, w) platí, že $\{\{u, x\}, \{v, x\}\} \cap D(V) \subseteq E$. Potom je trojúhelník (u, v, w) prázdný.*

Důkaz: Označme y jeden z vrcholů uvnitř (u, v, w) nejblíže přímce uv. Potom (u, v, y) je podle předpokladů věty trojúhelník v G a je zřejmě prázdný. Ten jsme ale nemohli najít, protože má levou hranu po směru hodinových ručiček od levé hrany (u, v, w) a druhou hranu proti směru hodinových ručiček od pravé hrany (u, v, w). Dostáváme se tak do sporu s větou 5.5 a tvrzení je dokázáno. □

5.2.2 Doplnění grafu kandidátů

Abychom mohli aplikovat větu 5.7 a hledat tak v lineárním čase prázdné trojúhelníky, musíme do vstupního grafu přidat další pomocné hrany. Tyto hrany by však neměly ovlivnit hledaný modifikovaný LMT-skeleton původní množiny kandidátů. Proto je jen nějak speciálně označíme a mimo funkce WalkInit, WalkNext vždy přeskočíme.

Definice 5.1. Jestliže je pro trojúhelníky SLG $G = (V, E_G)$ a nějakou množinu pomocných hran $E_P \subseteq D(V) \setminus E_G$ splněný předpoklad věty 5.7 v $G = (V, E_G \cup E_P)$, pak množinu E_P nazveme uzávěrem grafu $G = (V, E_G)$.

Ke grafu kandidátů (V, E_G) by jistě stačilo přidat všechny zbylé diagonály $D(V)$, ovšem získali bychom tak úplný graf s $O(n^2)$ hranami a výpočet modifikovaného LMT-skeletonu by se oproti původní očekávané složitosti výrazně zpomalil.

Podívejme se na nějaký konkrétní vrchol $v \in V$ a množinu vrcholů V_v, do kterých vede z v hrana. Pro splnění předpokladu věty 5.7 stačí, abychom přidali všechny prázdné hrany do vrcholů uvnitř konvexního obalu V_v.

Pro rovnoměrně rozložené vstupní body v konvexní oblasti $\mathcal{C}$ víme, že jsme při generování kandidátů v kapitole 5.1 prozkoumali jen konstantní očekávaný počet čtverců a v průměru našli konstantní počet kandidátů z v. Abychom však ukázali konstantní velikost konvexního obalu, potřebujeme silnější tvrzení. Takové se dá najít v [6], kde se dokazuje pro náhodný bod $O(\frac{|\mathcal{C}|}{\sqrt{n}})$ očekávaná prozkoumaná vzdálenost v každém z $O(1)$ intervalů úhlů kolem v (pozn.: poblíž hranice $\mathcal{C}$ se prohledává do větší očekávané vzdálenosti, ale takových bodů je málo). Náš algoritmus dojde asymptoticky stejně daleko, proto i konvexní obal námi nalezených bodů pokrývá v průměru konstantní počet čtverců a vrcholů.

Program doplňující do grafu kandidátů pomocné hrany by mohl pracovat tak, že by pro každý vrchol spočítal konvexní obal, poté s využitím čtvercové sítě přidal hrany

do bodů uvnitř obalu a nakonec vyřadil neprázdné hrany. V nejhorším případě tak docílíme složitost výpočtu uzávěru $O(n^2 \log n)$ a očekávanou $O(n)$. V průměru přidáme jen $O(1)$ pomocných hran ke každému vrcholu.

Poznámka 5.7. Při řešení MWT jsem v praxi nenarazil na zásadní problémy s neprázdnými trojúhelníky a popsané řešení jsem neimplementoval. Jedná se spíše o teoretický návrh.

5.2.3 Algoritmus v $O(n^6)$

Pro přehlednost nejprve uvedeme algoritmus 14, pracující stejně jako původní Dickersonův, a který jen snižuje paměťovou složitost vynecháním množiny $CandTris$.

Definice 5.2. Nechť (V, E) je SLG a $e \in E$ hrana mimo konvexní obal V. Dvojici prázdných trojúhelníků na různých stranách e nazveme certifikátem hrany e, jestliže tvoří čtyřúhelník, ve kterém je e lokálně minimální.

Věta 5.8. *Nechť E_P je uzávěrem (V, E_G). Potom algoritmus 14 skončí se stejnými množinami $EdgesIn$ a $CandEdges$ jako Dickersonův algoritmus výpočtu modifikovaného LMT-skeletonu na V, E_G.*

Důkaz: Alogoritmus se od Dickersonova liší způsobem hledání dvojic trojúhelníků (certifikátů) z $CandTris$ kolem e, kde je e lokálně minimální. Stačí dokázat, že projdeme v obou algoritmech stejnou množinu certifikátů, protože zbytek výpočtu je stejný. To ale přímo vyplývá z věty 5.7, protože algoritmus prochází jen trojúhelníky splňující její předpoklady. (Poznámka: Přesunutím kandidátní hrany mezi pomocné hrany získáme opět uzávěr). □

Poznámka 5.8. Kdybychom umožnili i průchod neprázdných trojúhelníků (např. neúplným uzávěrem nebo vynecháním řádku (20) programu), pak by algoritmus mohl vyřadit méně kandidátních hran. Nic by se ale nezměnilo na faktu, že hrany $EdgesIn$ jsou součástí každé triangulace z příslušné třídy a naopak každá taková triangulace podgrafem $EdgesIn \cup CandEdges$. Pro třídu minimálních triangulací a počáteční množinu hran s diamantovou vlastností testy na náhodných datech ukazují, že se výsledné množiny téměř vždy shodují s modifikovaným LMT-skeletonem, ale existují i protipříklady, kdy tomu tak není.

5.2.4 Certifikáty

Přestože se nám podařilo snížit paměťovou složitost Dickersonova algoritmu, stále zůstává časová náročnost $O(n^6)$. Pro zvýšení efektivity výpočtu použijeme myšlenku z [3].

Zamysleme se nad situací, kdy algoritmus nenajde žádný certifikát nějaké hrany e a vyřadí ji tak z množiny $CandEdges$. Je skutečně nutné znovu zopakovat celý hlavní cyklus (5)?

Algoritmus 14 Výpočet modifikovaného LMT-skeletonu v $O(n^6)$

Vstup: (V, E_G) je SLG s uzávěrem E_P.
Výstup: $(V, EdgesIn)$ modifikovaný LMT-skeleton grafu (V, E_G).

1: $CandEdges \leftarrow E_G$
2: $Impossible \leftarrow E_P$
3: $EdgesIn \leftarrow \emptyset$
4: Přesuň hrany konvexního obalu z $CandEdges$ do $EdgesIn$.
5: **repeat**
6: $Leave \leftarrow$ true
7: **for** $\forall\{u, v\} = e \in CandEdges$ **do**
8: $CertificateFound \leftarrow$ false
9: WalkInit$((u, v))$ ▷ průchod v grafu $(V, EdgesIn \cup CandEdges \cup Impossible)$
10: **while** WalkNext$((u, v))$ **do**
11: WalkInit$((v, u))$
12: **while** WalkNext$((v, u))$ **do**
13: **if** hrany čtyřúhelníku $(u, (u, v)_l.v, v, (v, u)_l.v)$ jsou mezi $EdgesIn \cup CandEdges$ a e je zde lokálně minimální **then**
14: $CertificateFound \leftarrow$ true
15: **end if**
16: **end while**
17: **end while**
18: **if** $CertificateFound =$ false **then**
19: $CandEdges \leftarrow CandEdges \setminus \{e\}$
20: $Impossible \leftarrow Impossible \cup \{e\}$
21: $Leave \leftarrow$ false
22: **else if** e neprotíná žádnou hranu z $CandEdges$ **then**
23: $EdgesIn \leftarrow EdgesIn \cup \{e\}$
24: **end if**
25: **end for**
26: **until** $Leave$

Vyřazením hrany e můžou zaniknout nějaké prázdné trojúhelníky tvořící certifikáty jiných hran, což může ovlivnit jejich vyřazení v další iteraci cyklu (5). Takto ovlivněné hrany ale můžeme snadno pro hranu e najít. Jednou z možností je probrat všechny prázdné trojúhelníky kolem hrany e, protože e může být součástí certifikátu jen z takových trojúhelníků. Další možností je prozkoumat jednoduše všechny hrany vycházející z koncových vrcholů e nebo si certifikáty s hranou e pamatovat ve spojovém seznamu. Každopádně je jasné, že nemusíme při příští iteraci cyklu znovu ověřovat všechny hrany z $CandEdges$, ale jen poměrně malý počet těchto „sousedních“ hran.

Další klíčovou myšlenkou je vyhledávat certifikáty kolem všech hran z množiny $CandEdges$ současně, abychom nemuseli po zneplatnění certifikátu odebráním nějaké hrany vše opakovat. Když se podíváme na řádky (9–12) algoritmu 14, je vidět, že v přesně definovaném pořadí postupně procházíme dvojice prázdných trojúhelníků kolem zpracovávané hrany e, a že by toto pořadí nijak neovlivnilo dočasné přerušení cyklů a přesunutí hran z $CandEdges$ do množiny $Impossible$. Navíc nikde nevznikají nové certifikáty a můžou jen ubývat. Proto když přesouváme nějakou hranu e z $CandEdges$ do $Impossible$, stačí jen najít sousední hrany, pro které je e součástí aktuálního certifikátu, a pokračovat u nich hledáním dalších.

Tyto úvahy jsou základem algoritmu 15, který později využijeme pro výpočet modifikovaného LMT-skeletonu. Funkce NextCertificate postupně hledá certifikáty kolem dané hrany. Zavolá se jednou pro každou hranu mimo konvexní obal a poté pokaždé, když nějaká hrana aktuálního certifikátu zanikne. Druhou možnost zajišťuje procedura InvalidateCertificates, která při odebrání hrany přidá do pomocné množiny $Unknown$ hrany s neplatným certifikátem.

Poznámka 5.9. Pseudokód nedefinuje, jakým způsobem hledat v InvalidateCertificates příslušné certifikáty. Ve vlastní implementaci jsem pro ně zavedl spojové seznamy, kde každý certifikát je prvkem seznamů čtyř krajních hran.

Věta 5.9. *Celková složitost posloupnosti volání funkce* NextCertificate *pro hranu e do první návratové hodnoty false je $O(d^2)$, kde d je maximální stupeň vrcholu grafu s uzávěrem. Procedura* InvalidateCertificates *trvá nejdéle $O(d)$.*

Důkaz: Podle věty 5.6 je složitost každého cyklu přes trojúhelníky $O(d)$ a nijak ji neovlivní ani přesouvání vrcholů z $CandEdges$ do $Impossible$. Test dvojice trojúhelníků na vlastnosti certifikátu, stejně jako přidání i ubrání certifikátu do spojových seznamů podle poznámky 5.9, trvá $O(1)$. Počet ovlivněných certifikátů v InvalidateCertificates pro e je omezený počtem hran z krajních vrcholů e, tedy $O(d)$. □

5.2.5 Průsečíky

V této části vylepšíme časovou složitost hledání průsečíků na řádku (22) algoritmu 14, aby celkově fungovalo v čase $O(n \cdot d^2)$.

Nejprve přesuneme hledání průsečíků z hlavního cyklu až na konec algoritmu, kdy otestujeme každý prvek množiny $CandEdges$ a případné hrany bez průsečíků přidáme

Algoritmus 15 Hledání certifikátů pro modifikovaný LMT-skeleton

Vstup: SLG $(V, EdgesIn \cup CandEdges \cup Impossible)$, množina $Unknown$.

```
1: function NextCertificate({u, v})
2:     if jedná se o první volání funkce pro hranu {u, v} then
3:         WalkInit((u, v))
4:         WalkInit((v, u))
5:         if WalkNext((v, u)) = false then
6:             return false
7:         end if
8:     end if
9:     while true do
10:        if WalkNext((u, v)) = false then
11:            if WalkNext((v, u)) = true then
12:                WalkInit((u, v))
13:            else
14:                return false
15:            end if
16:        else if dvojice prázdných trojúhelníků v (u, v) a (v, u) tvoří certifikát then
17:            return true
18:        end if
19:    end while
20: end function

21: procedure InvalidateCertificates(e)
22:    for ∀e_0, e je součástí aktuálního certifikátu hrany e_0 do
23:        Unknown ← Unknown ∪ {e_0}        ▷ Pro e_0 budeme dál hledat certifikáty
24:    end for
25: end procedure
```

do $EdgesIn$. Zřejmě tak dostaneme množinu $EdgesIn$ z Dickersonova algoritmu, protože v něm $EdgesIn$ obsahuje právě hrany $CandEdges$ bez průsečíků a konvexní obal (důkaz snadný).

K výpočtu průsečíků pak lze použít algoritmus 16.

Algoritmus 16 Hledání průsečíků pro modifikovaný LMT-skeleton

Vstup: SLG $G = (V, EdgesIn \cup CandEdges)$.

```
1: function HasIntersection(s)
2:     s_1 ← s.ccw
3:     while s_1.v je vlevo od s do
4:         s_2 ← s_1.ccw
5:         while s.v je vlevo od s_2 do
6:             if s protíná s_2 then
7:                 return true
8:             end if
9:             s_2 ← s_2.ccw
10:        end while
11:        s_1 ← s_1.ccw
12:    end while
13:    return false
14: end function
```

Věta 5.10. *Nechť za stejných předpokladů, jako ve větě 5.8, $CandEdges$ výsledná množina algoritmu 14 a množina $EdgesIn$ obsahuje konvexní obal vrcholů V. Potom pro $s \in S(CandEdges)$ skončí algoritmus 16 s návratovou hodnotou true, právě když hrana $s.e$ nemá v $G = (V, EdgesIn \cup CandEdges)$ průsečík.*

Důkaz: Funkce vrátí true jen tehdy, když najde konkrétní průsečík. Zbývá opačná implikace. Pro spor vezměme $\{u, v\} \in CandEdges$ hranu protínající $s.e$ nejblíže vrcholu $s^*.v$. Jelikož $\{u, v\} \in CandEdges$, musí mít hrana $\{u, v\}$ nějaký certifikát a existuje prázdný trojúhelník (u, v, w) na stejné straně $\{u, v\}$ jako je bod $s^*.v$. Potom ale $w = s^*.v$, jinak by $\{u, v\}$ neprotínala $s.e$ nejblíže $s^*.v$. Algoritmus zmíněný průsečík najde přes nějakou dvojici hran trojúhelníku (u, v, w) (leží v $EdgesIn \cup CandEdges$). □

Poznámka 5.10. Důkaz předpokládá, že při výpočtu zkoumáme jen prázdné trojúhelníky. Ani v opačném případě se mi však při náhodném generování grafů hran s diamantovou vlastností nepodařilo najít protipříklad, kdy by funkce HasIntersection nenašla ve výsledné množině $CandEdges$ existující průsečík.

Věta 5.11. *Jestliže jsou splněny předpoklady posledního tvrzení a máme k dispozici proměnné $s'.ccw$ u orientovaných hran $s' \in S(EdgesIn \cup CandEdges)$, skončí algoritmus 16 v čase $O(d^2)$, kde d je maximální stupeň vrcholu v G.*

Důkaz: Oba vnořené cykly provedeme nejvýše d-krát a ostatní operace mají konstantní složitost. □

5.2.6 Algoritmus v $O(n \cdot d^3)$

Konečný postup, jak najít v grafu kandidátních hran (V, E_G) s uzávěrem E_P modifikovaný LMT-skeleton, znázorňuje algoritmus 17.

Algoritmus 17 Výpočet modifikovaného LMT-skeletonu v $O(n \cdot d^3)$

Vstup: SLG (V, E_G) s uzávěrem E_P.

1: $EdgesIn \leftarrow \emptyset$
2: $CandEdges \leftarrow E_G$
3: $Impossible \leftarrow E_P$
4: Spočítej cw a ccw u orientovaných hran SLG $(V, EdgesIn \cup CandEdges \cup Impossible)$.
5: Odeber hrany konvexního obalu z $CandEdges$ a přidej do $EdgesIn$.
6: **while** $\exists e \in Unknown$ **do**
7: $\quad Unknown \leftarrow Unknown \setminus e$
8: $\quad$ **if** NextCertificate(e) = false **then**
9: $\quad\quad CandEdges \leftarrow CandEdges \setminus \{e\}$
10: $\quad\quad Impossible \leftarrow Impossible \cup \{e\}$
11: $\quad\quad$ InvalidateCertificates(e)
12: $\quad$ **end if**
13: **end while**
14: Odeber ze SLG hrany $Impossible$.
15: **for** $\forall \{u, v\} \in CandEdges$ **do**
16: $\quad$ **if** HasIntersection$((u, v))$ = false **then**
17: $\quad\quad EdgesIn \leftarrow EdgesIn \cup \{\{u, v\}\}$
18: $\quad$ **end if**
19: **end for**

Věta 5.12. *Nechť $\mathcal{M} \subseteq LMT(V)$ je neprázdná třída triangulací na konečné množině V a SLG (V, E_G) obsahuje hrany každé triangulace z $\mathcal{M}$. Jestliže je množina E_P uzávěrem (V, E_G), pak algoritmus 17 skončí se stejnými množinami $CandEdges$ a $EdgesIn$ jako Dickersonův algoritmus pro hledání modifikovaného LMT-skeletonu.*

Důkaz: Podle kapitoly 5.2.4 vyřadíme přesně stejné hrany z množiny $CandEdges$ jako algoritmus 14, který je podle věty 5.8 identický s Dickersonovým algoritmem. Výsledný množina $CandEdges$ je proto stejná, což díky větě 5.10 o průsečících můžeme říct i o množině $EdgesIn$. □

Věta 5.13. *Složitost algoritmu 17 je $O(n \cdot d^3)$, kde d je maximální stupeň vrcholu v $(V, E_G \cup E_P)$.*

Důkaz: Řádek (4) lze implementovat v čase $O(n \cdot d \log d)$ setříděním orientovaných hran kolem každého vrcholu. Pomocí získaných informací pak snadno v lineárním čase identifikujeme hrany konvexního obalu – řádek (5). Celková složitost kroku (8) je podle věty 5.9 pro každou hranu $e \in E_G$ nejhůře $O(d^2)$ a pro celý graf pak $O(n \cdot d^3)$. Důsledkem je i stejné omezení počtu iterací cyklu (6). Řádek (11) použije díky větě 5.9 nejvýše $O(n \cdot d^2)$ instrukcí. Konečné hledání průsečíků na řádku (16) trvá podle věty 5.11 $O(n \cdot d^3)$, tedy i výslednou složitost algoritmu můžeme asymptoticky odhadnout výrazem $O(n \cdot d^3)$. □

Poznámka 5.11. Shrneme-li poznámky 5.6, 5.7, 5.8 a 5.10 v průběhu textu, program můžeme spustit i s prázdnou množinou pomocných hran E_P. Nemusí tak sice vyjít kompletní modifikovaný LMT-skeleton, ale děje se tak jen zřídka, a i při rozdílu lze množiny $EdgesIn$ a $CandEdges$ použít pro výpočet přesné MWT. Teoreticky také může $CandEdges$ obsahovat hranu protínající $EdgesIn$. V praxi jsem na takový případ nenarazil, ale kdybychom si chtěli být jisti, snadno bychom zmíněné hrany vyřadili v $O(n^2 \cdot d)$ ($|EdgesIn| = O(n)$).

5.3 Jednoduché polygony

Popsaným výpočtem grafu kandidátů a modifikovaného LMT-skeletonu jsme získali PSLG $G = (V, E_{certain})$ obsahující $CH(V)$ a $E_{possible}$ podmnožinu diagonál G. Přitom víme, že G je podgrafem každé triangulace s minimální váhou a naopak libovolná MWT je částí v $(V, E_{certain} \cup E_{possible})$. Obě množiny hran máme společně uložené v datové struktuře SLG s uspořádanými orientovanými hranami podle úhlu kolem počátečních vrcholů.

Některé stěny G tvoří jednoduché polygony, některé mohou být nesouvislé. V této části kapitoly vyřešíme první možnost.

Podívejme se na nějaký konkrétní jednoduchý polygon f s obvodem $(v_0, \ldots, v_l)$. Možným řešením by bylo přímo aplikovat algoritmus hledání $GMWT$ z kapitoly 2.3 se složitostí $O(l^3)$, ale uvidíme, že existuje i rychlejší výpočet na řídkém grafu kandidátů pracující v $O(l \cdot d^2)$, kde d je maximální počet kandidátních hran směřujících do oblasti f z nějakého vrcholu na obvodu.

Postup už nebudeme podrobně popisovat, protože je velmi podobný algoritmu 2, a jen zmíníme rozdíly. Potřebujeme paměťovou složitost $o(n^2)$, takže k uložení informací využijeme dodatečný prostor u existující hran, konkrétně $e.sum$ (analogie $w_{i,j}$) a dvojici proměnných $e.left$ a $e.right$ (náhrada $k_{i,j}$) ukazujících na hrany nejlepšího nalezeného trojúhelníku u hrany e. Vnější cyklus opět probíhá přes všechny vrcholy (resp. orientované hrany) na obvodu, vnitřní cykly pak jen přes kandidátní hrany. Pro hledání trojúhelníků můžeme přímo použít dříve popsanou funkci WalkNext, jen je

potřeba upravit interval prozkoumaných úhlů u krajních vrcholů hrany, abychom našli jen trojúhelníky nezasahující mimo řešený polygon.

Celková složitost algoritmu 18 je pak zřejmě $O(l \cdot d^2)$, protože všechny cykly (i uvnitř WalkNext) iterují jen přes kandidátní hrany uvnitř polygonu.

5.4 Nesouvislé oblasti

Zbývá dořešit poslední problém, totiž doplnit triangulaci stěn $(V, E_{certain})$ s nesouvislou hranicí. Bohužel není znám žádný polynomiální algoritmus řešící tuto úlohu. Vedle „hrubé síly“ s exponenciální složitostí zbývá zkusit další heuristiky, které by (možná) nalezly více jistých hran, nebo nějakou aproximaci. Naštěstí ale v modifikovaném LMT-skeletonu bývají nesouvislé stěny jen zřídka, a když už, tak mívají, až na velmi speciální případy, málo komponent. Nyní popíšeme algoritmus, který doplní přesnou minimální triangulaci stěny f s k komponentami (vnější hranici nepočítáme) a s celkovým počtem l vrcholů na hranici nejhůře v čase $O(l \cdot d^{2+k}) = O(n^{3+k})$, kde d je největší počet hran směřujících z nějakého vrcholu na obvodu do stěny f. Přitom komponentami nemusí být jen izolované body nebo stromy, ale libovolně velké útvary, které příležitostně samy mohou obsahovat další nesouvislé stěny. Složitost vnitřku komponent nijak neovlivní čas výpočtu stěny f.

Poznámka 5.12. Algoritmus není inspirován žádném zdrojem, proto zde neuvádím reference.

Věta 5.14. *Algoritmus 19 najde minimální triangulaci libovolné omezené stěny f.*

Důkaz: Snadno lze ukázat, že cyklus (5) prochází právě všechny kombinace neprotínajících se k orientovaných hran s_i z vrcholů v_i pod úhlem $[0, \pi)$. Pořadí zpracování je v lexikografickém pořadí podle úhlů orientovaných hran.

Pro korektnost algoritmu tedy stačí ukázat, že s každou zkoušenou kombinací hran vznikne skutečně simple polygon, a že pro nějakou minimální triangulaci T_f oblasti f zkusíme přidat jen hrany z T_f. Potom minimalita výsledné triangulace vyplývá z korektnosti algoritmu pro hledání zobecněné minimální triangulace jednoduchého polygonu.

Označme $K_i \subseteq V$ vrcholy na vnější hranici i-té komponenty a $\prec$ relaci lexikografického uspořádání vrcholů V primárně podle y-ové a sekundárně podle x-ové souřadnice. V tomto značení řádek (1) najde pro každé i maximální prvek v_i z množiny K_i. Bez újmy na obecnosti můžeme předpokládat, že $v_k \prec v_{k-1} \prec \cdots \prec v_1$, jinak pro potřeby důkazu změníme očíslování komponent.

Nechť $(s_1, \ldots, s_k)$ je nějaká konkrétní kombinace orientovaných hran nalezených na řádku (20). Nyní pomocí indukce dokážeme, že hrany spojují každou komponentu s vnější hranicí. Zřejmě s_1 propojuje komponentu K_1 s vnější hranicí stěny, neboť v_1 je maximální ze všech vrcholů $\bigcup_{i'=1}^{k} K_{i'}$ a úhel volíme v intervalu $[0, \pi)$ kolem v_i. Podobně obecné s_i musí spojovat K_i s nějakou $K_j, j < i$ nebo s vnější hranicí, protože v_i je maximální z $\bigcup_{i'=i}^{k} K_{i'}$. Z indukce existuje cesta z v_j do nějakého vrcholu na vnější

Algoritmus 18 Triangulace jednoduchého polygonu

Vstup: $(V, E_{certain}, E_{possible})$, jednoduchý polygon $s.f$.

```
1: procedure SimplePolygon(s)
2:     for ∀s_1 hrany na obvodu po směru hodinových ručiček do
3:         s_1.e.sum ← 0
4:         s_2 ← s_1^*.ccw
5:         while s_2.e ∈ E_possible do
6:             Cur ← s_2.e
7:             if hranu Cur jsme už dříve navštívili then
8:                 Cur.sum ← ∞        ▷ Hledáme trojúhelníky napravo od diagonály s_2
9:                 WalkInitSimple(s_2^*)
10:                while WalkNext(s_2^*) do
11:                    Left ← (s_2^*)_l.e                    ▷ Levá hrana trojúhelníku
12:                    Right ← (s_2^*)_r.e                   ▷ Pravá hrana trojúhelníku
13:                    Sum ← Left.sum + Right.sum
14:                    if Sum < ∞ ∧ Sum < Cur.sum then
15:                        Cur.sum ← Sum
16:                        Cur.left ← Left
17:                        Cur.right ← Right
18:                    end if
19:                end while
20:                Cur.sum ← Cur.sum + ||Cur||
21:            end if
22:            Označ hranu Cur jakou navštívenou.
23:            s_2 ← s_2.ccw
24:        end while
25:        Podobně prozkoumej trojúhelníky z poslední hrany obvodu.
26:        Sestav nejlepší nalezenou triangulaci z .left a .right.
27:    end for
28: end procedure

29: procedure WalkInitSimple(s)          ▷ Jen trojúhelníky uvnitř polygonu
30:    s_l ← s.ccw, s_r ← s^*, s_f ← s
31:    repeat
32:        s_r ← s_r.cw
33:    until (s_r.v není nalevo od s) ∨ (s_r.ccw ≠ s^* ∧ c_r.ccw ∈ E_certain)
34:    repeat
35:        s_f ← s_f.ccw
36:    until (s_f.v není nalevo od s) ∨ (s_f.cw ≠ s ∧ c_f.cw ∈ E_certain)
37: end procedure
```

Algoritmus 19 Triangulace nesouvislé stěny

Vstup: SLG $(V, E_{certain} \cup E_{possible})$, omezená stěna $s.f$ v PSLG $(V, E_{certain})$.

1: Najdi vnitřní komponenty stěny a u každé maximální vrchol v_i $(i = 1, \ldots, k)$ v lexikografickém uspořádání podle (y, x).
2: U každé kandidátní hrany uvnitř f vynuluj pomocnou proměnnou s počtem průsečíků.
3: Pro každou komponentu najdi $first_i$ první orientovanou hranu z v_i před intervalem úhlu $[0, \pi)$ kolem v_i a $last_i$ první hranu za tímto intervalem.
4: $i \leftarrow 1$, $s_i \leftarrow first_i$, $w_{best} \leftarrow \infty$
5: **while** true **do** ▷ Zkoušení všech kombinací propojení komponent bez průsečíků
6: $s_i \leftarrow s_i.ccw$
7: **if** $s_i = last_i$ **then**
8: **if** $i = 1$ **then**
9: **break** ▷ Vyčerpali jsme všechny kombinace
10: **else**
11: $i \leftarrow i - 1$
12: Sniž počet průsečíků u kandidátních hran protínajících s_i.
13: **end if**
14: **else if** hrana s_i nemá průsečík **then**
15: $w_i = w_{i-1} + \|s_i\|$
16: Zvyš počet průsečíků u kandidátních hran protínajících s_i.
17: **if** $i < k$ **then**
18: $i \leftarrow i + 1$
19: $s_i \leftarrow first_i$
20: **else** ▷ Našli jsme kombinaci bez průsečíků
21: $w = w_k +$ (váha $GMWT$ jednoduchého polygonu)
▷ Stačí upravená fce SimplePolygon,
▷ aby přeskakovala hrany s průsečíky
▷ a jen spočítala váhu triangulace
22: **if** $w < \infty \wedge w < w_{best}$ **then**
23: $w_{best} \leftarrow w$
24: Zapamatuj si posloupnost použitých hran s_i.
25: **end if**
26: Sniž počet průsečíků u kandidátních hran protínajících s_i.
27: **end if**
28: **end if**
29: **end while**
30: Použij nejlepší nalezené propojení komponent.
31: Doplň minimální triangulaci jednoduchého polygonu.

hranici stěny, a tu lze prodloužit přes komponentu K_j a hranu s_i do v_i. Ovšem na řádku (14) jsme vyloučili kombinace protínajících se hran a vybrali jsme jen k hran. Proto nikde nemůže vzniknout cyklus a přidáním hran jsme tak původní stěnu upravili na právě jeden simple polygon (kombinace se dá představit jako strom, kde kořenem je vnější hranice a ostatní vrcholy představují komponenty K_i).

Nechť T_f je nějaká minimální triangulace oblasti f. Vnitřní stěny triangulace tvoří trojúhelníky, a ty mají u každého vrcholu úhel menší než π. Proto z každého vrcholu v_i vede v triangulaci T_f nějaká hrana pod úhlem z intervalu $[0, \pi)$. Takovou kombinaci jsme ale v cyklu (5) prozkoumali a algoritmus nemůže skončit s horší triangulací než T_f. □

Věta 5.15. *Algoritmus 19 skončí nejhůře v čase* $O(l \cdot d^{2+k}) = O(n^{3+k})$.

Důkaz: Inicializaci před hlavním cyklem (5) lze provést v lineárním čase průchodem grafu po vrcholech a hranách ve stěně f, tj. v $O(l \cdot d)$. Současně s tím můžeme také vytvořit pomocný seznam kandidátních hran v f.

Cyklus (5) postupně zkoumá různé posloupnosti orientovaných hran $(s_1, s_2, \ldots, s_{k'})$, $k' \leqq k$, v intervalech $[0, \pi)$ kolem vrcholů v_i. Počet iterací lze omezit množstvím kombinací takových posloupností, který je $O(d^k)$. V každém kroku nejvýše dvakrát projdeme $O(l \cdot d)$ kandidátních hran v f při zvyšování a snižování počtu průsečíků a nejvýše jednou spustíme program pro výpočet zobecněné minimální triangulace simple polygonu s časovou složitostí $O(l \cdot d^2)$. Nalezenou kombinaci si snadno zapamatujme v $O(k) = O(l)$. Výsledné sestavení nejlepší triangulace trvá nejhůře $O(l \cdot d^2)$. Celková složitost algoritmu proto vychází $O(d^k) \cdot O(l \cdot d^2) = O(l \cdot d^{k+2}) = O(n^{3+k})$. □

Poznámka 5.13. Popsaným programem se dá doplnit PSLG s k komponentami na $GMWT$ v čase $O(n^{2+k})$. Stačí přidat konvexní obal, setřídit orientované hrany ve společném SLG jistých a povolených hran kolem svých koncových vrcholů a průchodem grafu doplnit triangulace v jednotlivých stěnách. Snadno ošetříme i neexistenci triangulace.

5.5 Poznámky k implementaci

Popsaný algoritmus je skoro celý součástí programu přiloženého k diplomové práci.

Jak už bylo zmíněno v textu, implementace zjednodušuje generování čtvercové sítě jen na obdélníkovou oblast (místo obecné konvexní oblasti) a může také oproti lineární očekávané složitosti v blízkosti hranice mírně zpomalit. Obojí by šlo podle kapitoly 5.1 poměrně jednoduše vyřešit, čímž bychom dostali lineární očekávanou složitost hledání kandidátů v rovnoměrně rozložené množině vrcholů uvnitř konvexní oblasti.

Další chybějící částí je doplnění grafu kandidátů o uzávěr. Algoritmus by proto teoreticky mohl selhat, kdyby zůstaly v nalezeném „modifikovaném LMT-skeletonu“ průsečíky kandidátních hran s jistými hranami MWT. Na takový případ jsem ale v praxi

ani jednou nenarazil a je možné, že by šlo korektnost i dokázat. Beirouti [3] se o podobné situaci ani nezmiňuje.

Většina výpočtů probíhá exaktně na celých číslech (fixed-point), až na porovnávání součtů odmocnin, které se vyskytuje v algoritmech doplňujících stěny modifikovaného LMT-skeletonu.

Odhad asymptotické složitosti algoritmu vyplývá z předchozího textu. Je roven $O(T_1 + n \cdot d^3 + n \cdot d^{2+k})$, kde T_1 je doba výpočtu kandidátů, n celkový počet vrcholů, d maximální stupeň vrcholu v grafu kandidátů a k maximální počet nesouvislých částí uvnitř stěn modifikovaného LMT-skeletonu. Z testů pro rovnoměrně rozložené množiny bodů ve čtverci (viz kapitola 6) se zdá, že je očekávaná složitost lineární nebo má k lineární velmi blízko. Na druhou stranu existuje protipříklad [3], kdy modifikovaný LMT-skeleton obsahuje stěnu s $\Omega(n)$ komponentami. Odtud vyplývá exponenciální časová složitost algoritmu v nejhorším případě $\Theta(n^n)$.

Cílem implementace nebylo minimalizování konstant u paměťové a časové složitosti a na mnoha místech by šla podstatně zrychlit. Program se snaží pomocí více vláken průběžně v časových intervalech vykreslovat stav výpočtu, z čehož plyne nutnost netriviální synchronizace přístupů ke grafovým strukturám. Na efektivitě se také podepisuje poměrně velká obecnost algoritmů a struktur, které společně využívá více různých algoritmů, včetně výpočtu DT, GT, DGT a jiných triangulací.

Kapitola 6

Srovnání triangulací

V této kapitole srovnáme různé třídy triangulací, zejména jejich aproximační faktor vůči minimálním triangulacím. Porovnávat budeme jednak teoreticky pomocí dokázaných asymptotických odhadů tak prakticky na náhodně generovaných i reálných množinách vrcholů.

6.1 Testované triangulace

Tabulka 6.1 obsahuje přehled testovaných triangulací, včetně časové složitosti jejich výpočtu a aproximačního faktoru v nejhorším i průměrném případě pro rovnoměrně rozloženou množinu bodů v konvexní oblasti.

Minimální triangulace (MWT) se při testování počítala pomocí algoritmu popsaného v kapitole 5, odkud také vyplývá její očekávaná složitost v průměrném případě. Písmeno d značí maximální očekávaný stupeň vrcholu v uzávěru grafu kandidátů (hrany s diamantovou vlastností doplněný o hrany podle kapitoly 5.2.2) a k značí maximální očekávaný počet komponent uvnitř stěn modifikovaného LMT-skeletonu. Přestože zjednodušená implementace algoritmu nepočítá uzávěr a spoléhá na absenci průsečíků v modifikovaném LMT-skeletonu, ani jednou během testování neselhala.

Další testovanou triangulací byla greedy triangulace (GT). Podrobný popis, včetně odkazů na důkazy uvedených odhadů v tabulce, lze nalézt v kapitole 3.2. Konstantní očekávaný faktor GT a DT vyplývá z poznámky na konci kapitoly 4.2 o zakázaných oblastech.

Implementace GT nejprve v očekávaném lineárním čase vygeneruje v průměru lineární počet kandidátů pomocí algoritmu z výpočtu MWT, setřídí je a do výsledné triangulace pak postupně přidává hrany bez průsečíků. Průsečíky konkrétní hrany lze nalézt v očekávaném konstantním čase, jak je popsáno v [6]. Složitost použitého algoritmu je v průměrném případě $O(n \log n)$ a snadno by šla vylepšit na $O(n)$ úpravou třídění hran.

Delaunayova triangulace (DT) se, kvůli jednoduchosti implementace, počítala z GT prohozením diagonál v čtyřúhelníkách, kde hrana není lokálně Delaunayova. Počet ta-

	nejhorší případ		průměrný případ	
	složitost	faktor	složitost	faktor
MWT	$O(n^n)$	1	$O(n \cdot d^3 + n \cdot d^{2+k})$	1
DT	$\Theta(n \log n)$	$\Omega(n)$	$\Theta(n)$	$\Theta(1)$
GT	$\Theta(n \log n)$	$\Theta(\sqrt{n})$	$\Theta(n)$	$\Theta(1)$
QGT	$\Theta(n \log n)$	$\Theta(1)$	$\Theta(n)$	$\Theta(1)$
DGT	$O(n^5)$	—	$O(n^3)$	—
$MST\text{-}T$	$O(n^3)$	$\Omega(n)$	$O(n^3)$	$\Theta(1)$
$GT\text{-}MST\text{-}T$	$O(n^3)$	$\Theta(\sqrt{n})$	$O(n^3)$	$\Theta(1)$
$QGT\text{-}MST\text{-}T$	$O(n^3)$	$\Theta(1)$	$O(n^3)$	$\Theta(1)$
$DGT\text{-}MST\text{-}T$	$O(n^5)$	—	$O(n^3)$	—
$LMT\text{-}CQGT$	$O(n^4)$	$\Theta(1)$	$O(n \cdot d^3 + n \cdot l^2)$	$\Theta(1)$
$LMT\text{-}CQGT\text{-}CMST\text{-}T$	$O(n^4)$	$\Theta(1)$	$O(n \cdot d^3 + n \cdot l^2)$	$\Theta(1)$

Tabulka 6.1: Asymptotické srovnání triangulací.

kových prohození na testovaných datech obvykle nebyl velký, což umožnilo porovnat DT s MWT i na poměrně velkých množinách dat. Třídu Delaunayových triangulací podrobněji popisuje kapitola 3.1.

Odhady pro quasi-greedy (QGT) a double-greedy triangulace (DGT) lze nalézt v kapitolách 3.3 a 3.4. Implementované algoritmy jsou poměrně pomalé, zejména u DGT, proto testování těchto triangulací probíhalo jen na menším množství bodů. Jelikož jsem v prostudovaných materiálech nenašel potřebné důkazy, v tabulce chybí asymptotické odhady aproximačních faktorů pro DGT, .

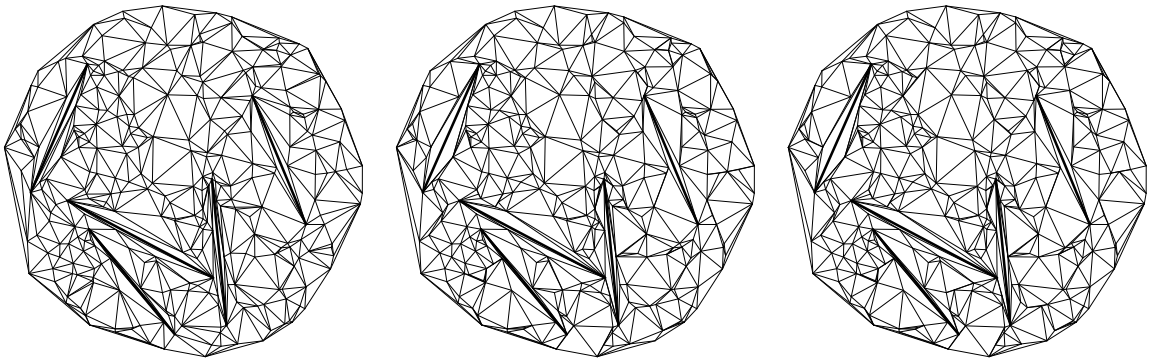
Další čtveřice triangulací v tabulce, $MST\text{-}T$, $GT\text{-}MST\text{-}T$, $QGT\text{-}MST\text{-}T$ a $DGT\text{-}MST\text{-}T$, představuje minimální omezené triangulace minimálních koster DT, GT, QGT a DGT. Kostru jde snadno najít nejhůře v $O(n \log n)$. Vzniklý PSLG pak doplníme na $CMWT$ v čase $O(n^3)$. Kvalita takto upravené aproximace zřejmě není horší než původní aproximace. O dolních odhadech $MST\text{-}T$ a $GT\text{-}MST\text{-}T$ se zmiňuje kapitola 3.5.

Zbývající dvojice triangulací kombinuje více heuristik a aproximací. $LMT\text{-}CQGT$ nejprve spočítá modifikovaný LMT-skeleton vstupní množiny bodů. Stěny vzniklého PSLG pak každou zvlášť doplníme na triangulaci pomocí $CQGT$, což lze snadno provést v čase $O(l^3)$, kde l je počet vrcholů na hranici stěny. Odtud vyplývá celková složitost algoritmu $O(n \cdot d^3 + n \cdot l^2) = O(n^4)$. Protože v každé stěně $CQGT$ aproximuje $CMWT$ s faktorem $O(1)$ a protože modifikovaný LMT-skeleton je podgrafem MWT, musí mít celá triangulace $O(1)$ aproximační faktor vůči MWT.

Poslední testovaná triangulace, $LMT\text{-}CQGT\text{-}CMST\text{-}T$, navíc spočítá minimální omezenou kostru $CQGT$ v každé stěně modifikovaného LMT-skeletonu a doplní ji dynamickým programováním na $CMWT$. Omezenou kostrou rozumíme podmožinu hran s co nejmenší váhou, které spojují nesouvislé části PSLG. Zřejmě tak vylepšíme triangulaci každé stěny a od minimální se může lišit jen lokálně ve stěnách modifikovaného LMT-skeletonu s nesouvislou hranicí. Protože $CMWT$ i omezenou minimální kostru lze počítat lokálně v každé stěně, vyjde stejná časová složitost jako u $LMT\text{-}CQGT$.

	nejhorší případ		průměrný případ	
	složitost	faktor	složitost	faktor
$CMWT$	$O(n^n)$	1	$O(n^4 + n^{3+k})$	1
CDT	$\Theta(n \log n)$	$\Omega(n)$	$O(n \log n)$	—
CGT	$O(n^3)$	$\Omega(\sqrt{n})$	$O(n^3)$	—
$CQGT$	$O(n^3)$	$\Theta(1)$	$O(n^3)$	$\Theta(1)$
$CDGT$	$O(n^5)$	—	$O(n^5)$	—
$CMST$-T	$O(n^3)$	$\Omega(n)$	$O(n^3)$	—
CGT-$CMST$-T	$O(n^3)$	$\Omega(\sqrt{n})$	$O(n^3)$	—
$CQGT$-$CMST$-T	$O(n^3)$	$\Theta(1)$	$O(n^3)$	$\Theta(1)$
$CDGT$-$CMST$-T	$O(n^5)$	—	$O(n^5)$	—
$CLMT$-$CQGT$	$O(n^4)$	$\Theta(1)$	$O(n^4)$	$\Theta(1)$
$CLMT$-$CQGT$-$CMST$-T	$O(n^4)$	$\Theta(1)$	$O(n^4)$	$\Theta(1)$

Tabulka 6.2: Asymptotické srovnání omezených triangulací.



Obrázek 6.1: Příklad omezených triangulací na množině 300 bodů.

Podobně by šlo vymyslet mnoho dalších kombinací heuristik a aproximací MWT. Pro testování jsem zvolil právě zmíněnou dvojici triangulací kvůli tomu, že se dá dokázat jejich aproximační faktor $O(1)$ a že většinou LMT-skeleton obsahuje jen malé stěny se souvislou hranicí.

Program podporuje také výpočet omezených verzí zmíněných triangulací. Pro úplnost uvádím i pro ně tabulku 6.2 odhadů složitostí a aproximačních faktorů, ovšem nebudu je podrobněji rozebírat ani testovat na náhodných datech. Samotné generování omezujících hran v rozumném pravděpodobnostním rozdělení by bylo tématem na samostatnou práci. Implementované algoritmy nejsou příliš rychlé, protože pro omezené triangulace nejde přímo použít implementovaný algoritmus generující lineární očekávané počty kandidátních hran. Žádná kosočtvercová oblast totiž obecně není zakázanou oblastí omezených tříd triangulací.

Počet bodů	*MWT*	*GT*	*LMT-CQGT--CMST-T*
1 000	0,4 s	0,7 s	0,5 s
5 000	1,9 s	4,1 s	2,3 s
10 000	3,9 s	9,0 s	4,7 s
25 000	10,3 s	24,7 s	11,8 s
50 000	21,0 s	52,1 s	23,9 s
75 000	31,6 s	80,5 s	36,3 s
100 000	43,8 s	—	48,7 s

Tabulka 6.3: Rychlost výpočtu vybraných triangulací.

6.2 Srovnání na náhodných datech

Nejvíce testů jsem prováděl na rovnoměrném pravděpodobnostním rozdělení bodů ve čtverci, na které jsou implementované algoritmy nejlépe stavěny a umožňují tak spočítat i velké množiny bodů. Největší *MWT* se podařilo spočítat na stroji Optiplex GX620 Dell, 2GB RAM, 3,2GHz s dvěma procesory z množiny 900 000 náhodných bodů ve čtverci, kdy výpočet trval necelých 6 minut. Zbylé testy pak probíhaly na stroji Pentium 4 Northwood, 512MB RAM, 2,8GHz. Velikost vstupních dat nejvíce limitovala operační paměť, po jejímž zaplnění značně poklesla efektivita výpočtu. Program příliš nešetří pamětí a přístupy k datům jsou velmi nelokalizované.

Orientační rychlost výpočtu některých triangulací uvádí tabulka 6.3. Výpočet *GT* měl oproti *MWT* větší časové a paměťové nároky kvůli menší zakázané oblasti a většímu počtu kandidátních hran.

Průměrný relativní rozdíl mezi nalezenými triangulacemi a triangulací s minimální váhou shrnuje tabulka 6.4. Náhodně vygenerované množiny bodů se vždy použily jako společný vstup všech tříd. Stejné hodnoty v řádcích nějakého sloupce tak typicky znamenají rovnost všech testovaných triangulací z daných tříd.

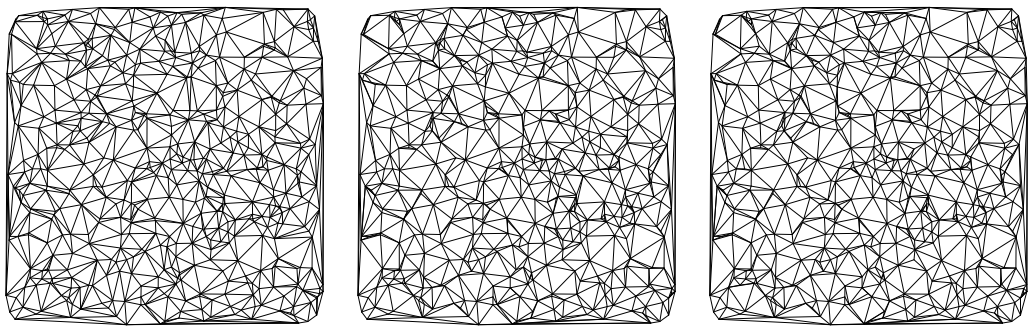
Od minimální triangulace se podle očekávání nejvíce lišila Delaunayova triangulace, kde průměrný aproximační faktor dosahoval hodnot kolem 1,02. Rozdíl je také vidět na obrázku 6.2. Trojúhelníky v minimální triangulaci mají často tvar blízký rovnostrannému trojúhelníku nebo bývají velmi úzké. Středně velké úhly se v *MWT* vyskytují méně, narozdíl od *DT*, která maximalizuje minimální úhel v triangulaci. Poblíž konvexního obalu jsou „nehezké“ podlouhlé trojúhelníky u všech testovaných triangulací.

Mnohem lepší výsledky vykazovala greedy triangulace, přímo související s délkami hran, a nebo její modifikace. Velmi se také osvědčilo vylepšení triangulace pomocí výpočtu *CMWT* minimální kostry, zejména u *DGT-MST-T* a *LMT-CQGT-CMST-T*, kdy se pro všechny testované množiny bodů tyto triangulace shodovaly s *MWT*. Dobré vlastnosti *LMT-CQGT-CMST-T* se dají vysvětlit spojitostí modifikovaného *LMT*-skeletonu v naprosté většině případů. Bohužel velmi pomalá implementace *DGT* neumožnila najít *DGT-MST-T* větších množin bodů.

Za zmínku také stojí to, že *GT* a *QGT* vyšly vždy stejně. Aby byly splněny *QGT-*

	10 bodů 10 000 testů	20 bodů 1 000 testů	30 bodů 1 000 testů	50 bodů 1 000 testů
DT	1,01407371	1,01843754	1,02068394	1,02242867
GT	1,00031399	1,00048952	1,00056544	1,00071098
QGT	1,00031399	1,00048952	1,00056544	1,00071098
DGT	1,00006644	1,00012329	1,00026122	—
MST-T	1,00017078	1,00017325	1,00011584	1,00011149
GT-MST-T	1,00000010	1,00000311	1,00000465	1,00000013
QGT-MST-T	1,00000010	1,00000311	1,00000465	1,00000013
DGT-MST-T	1,00000000	1,00000000	1,00000000	—
LMT-CQGT	1,00031399	1,00048952	1,00056544	1,00071098
LMT-CQGT-CMST-T	1,00000000	1,00000000	1,00000000	1,00000000
	100 bodů 1 000 testů	200 bodů 100 testů	500 bodů 100 testů	1 000 bodů 100 testů
DT	1,02385212	1,02390520	1,02521774	1,02522663
GT	1,00076668	1,00101305	1,00086869	1,00095673
QGT	1,00076668	1,00101305	—	—
MST-T	1,00009944	1,00005634	1,00008652	—
GT-MST-T	1,00000024	1,00000246	1,00000050	—
QGT-MST-T	1,00000024	1,00000246	—	—
LMT-CQGT	1,00076668	1,00101305	1,00086869	1,00095673
LMT-CQGT-CMST-T	1,00000000	1,00000000	1,00000000	1,00000000
	5 000 bodů 50 testů	20 000 bodů 20 testů	50 000 bodů 10 testů	100 000 bodů 5 testů
DT	1,02480832	1,02372177	1,02370990	1,02365764
GT	1,00092896	1,00095712	1,00097164	1,00094095
LMT-CQGT	1,00092899	1,00095716	1,00097166	1,00094096
LMT-CQGT-CMST-T	1,00000000	1,00000000	1,00000000	1,00000000

Tabulka 6.4: Aproximační faktory rovnoměrného rozložení bodů ve čtverci.

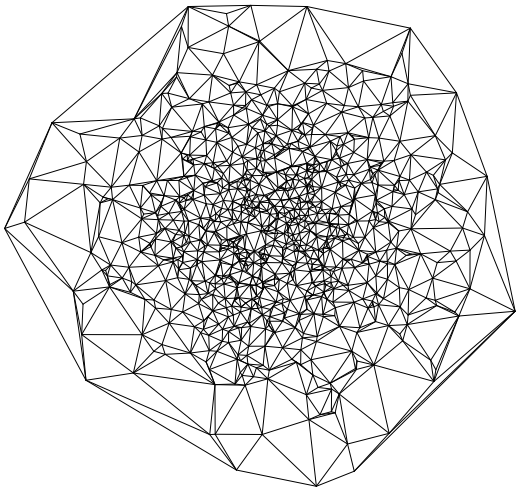


DT *MWT* *GT*

Obrázek 6.2: Příklad 500 rovnoměrně rozložených bodů ve čtverci.

	30 bodů 1 000 testů	100 bodů 1 000 testů	1 000 bodů 100 testů	10 000 bodů 30 testů
DT	1,02169405	1,02533532	1,02515373	1,02410598
GT	1,00059993	1,00087774	1,00099387	1,00094796
QGT	1,00059993	1,00087774	—	—
DGT	1,00026479	—	—	—
MST-T	1,00021565	1,00015886	—	—
GT-MST-T	1,00000338	1,00000118	—	—
QGT-MST-T	1,00000338	1,00000118	—	—
DGT-MST-T	1,00000000	—	—	—
LMT-CQGT	1,00059993	1,00087774	1,00099387	1,00094796
LMT-CQGT-CMST-T	1,00000000	1,00000000	1,00000000	1,00000000

Tabulka 6.5: Aproximační faktory normálního rozložení bodů.



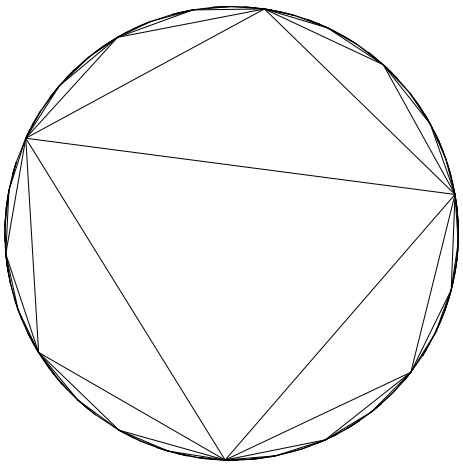
Obrázek 6.3: Příklad *MWT* 1000 bodů v normálním pravděpodobnostním rozdělení.

podmínky, musí totiž několik bodů ležet ve velmi speciální pozici v blízkosti přímky. Rozdíl se ukáže později u více specifických rozdělení.

Výsledky pro normální pravděpodobnostní rozdělení shrnuje tabulka 6.5. Většina triangulací vykazuje o trochu horší aproximační faktor než u předchozích testů, ale jinak je situace velmi podobná. Implementovaný algoritmus si dokázal poradit i s poměrně velkými množinami dat. Příčinou bude nejspíš podobnost s rovnoměrným rozdělením, podíváme-li se do nějaké menší oblasti. Diamantovou vlastnost pak má opět malá podmnožina diagonál a ani stěny v rozšířeném skeletonu nejsou velké.

	10 bodů 10 000 testů	30 bodů 1 000 testů	100 bodů 100 testů	300 bodů 30 testů
GT	1,00197235	1,00540085	1,00770787	1,00984495
QGT	1,00197235	1,00540085	1,00770787	—
DGT	1,00342582	1,09304801	—	—
$GT\text{-}MST\text{-}T$	1,00000000	1,00000000	1,00000000	1,00000000
$QGT\text{-}MST\text{-}T$	1,00000000	1,00000000	1,00000000	—
$DGT\text{-}MST\text{-}T$	1,00000000	1,00000000	—	—
$LMT\text{-}CQGT$	1,00197235	1,00540085	1,00770787	1,00984495
$LMT\text{-}CQGT\text{-}CMST\text{-}T$	1,00000000	1,00000000	1,00000000	1,00000000

Tabulka 6.6: Aproximační faktory rovnoměrného rozložení bodů na obvodu kružnice.

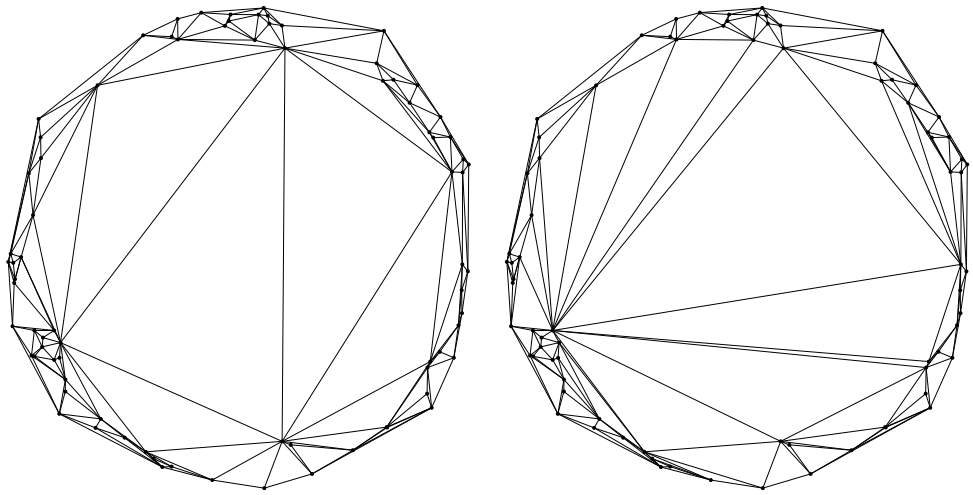


Obrázek 6.4: Příklad MWT pro 300 náhodných bodů na obvodu kružnice.

Problém s rychlostí nastal až u testů množin bodů rovnoměrně rozložených po obvodu kružnice (tabulka 6.6). V této situaci má všech $\Theta(n^2)$ hran diamantovou vlastnost a očekávaná složitost výpočtu se podstatně zvýší. Pokud jde o kvalitu aproximace, problémy se projevily zejména u DGT. Zdá se, že i výběrem „nevýhodné“ tětivy mohou mít omezené greedy triangulace malou váhu, a algoritmus pak jejich výběrem zablokuje mnohem lepší možnost. Vylepšením triangulace pomocí minimální kostry vždy vyšla minimální triangulace. Kostra totiž bývá na obvodu kružnice a $CMWT$ se počítá pro celou stěnu uvnitř konvexního obalu. V tabulce není uvedena DT, jelikož pro body na společné kružnici je libovolná triangulace Delaunayova a zjištěné výsledky by závisely na konkrétní implementaci.

	10 bodů 10 000 testů	30 bodů 1 000 testů	100 bodů 100 testů	300 bodů 30 testů
DT	1,04629415	1,04876297	1,04814540	1,04213418
GT	1,00211682	1,00409578	1,00401028	1,00304678
QGT	1,00211703	1,00409789	1,00401028	1,00304678
DGT	1,00260522	1,02199120	—	—
MST-T	1,00001016	1,00003428	1,00006521	1,00009013
GT-MST-T	1,00000000	1,00000000	1,00000000	1,00000060
QGT-MST-T	1,00000000	1,00000000	1,00000000	1,00000060
DGT-MST-T	1,00000000	1,00000000	—	—
LMT-CQGT	1,00211682	1,00409607	1,00401028	1,00304678
LMT-CQGT-CMST-T	1,00000000	1,00000000	1,00000000	1,00000000

Tabulka 6.7: Faktory náhodného rozložení bodů v blízkosti obvodu kružnice.



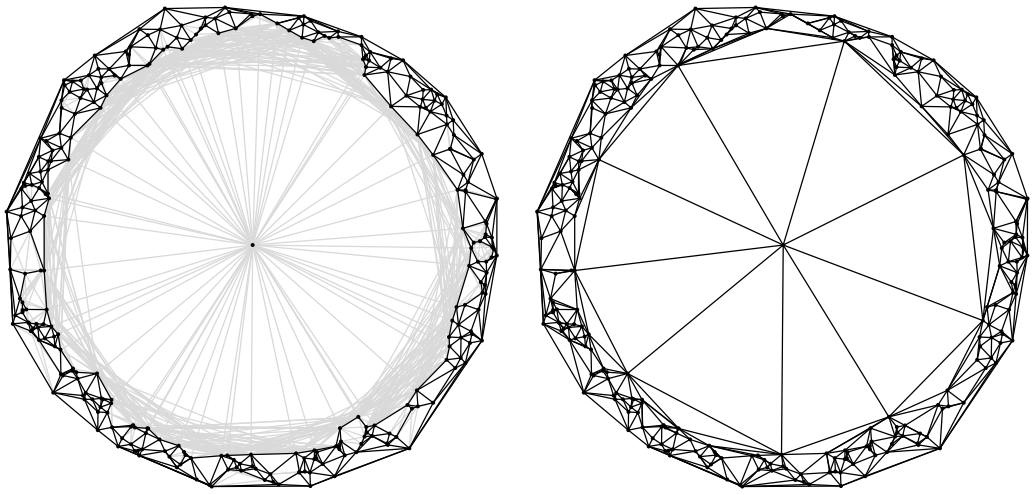
MWT *DGT*

Obrázek 6.5: Příklad 80 bodů v blízkosti kružnice.

Tabulka 6.7 znázorňuje další pravděpodobnostní rozdělení, kdy se vždy nejprve najde náhodný bod na obvodu kružnice a poté posune podle normálního rozdělení. Výsledné body jsou tak v blízkosti obvodu kružnice, což vede k zajímavým výsledkům u některých tříd triangulací. Jednou z nich je *QGT*, kdy se konečně na náhodných datech projevil rozdíl proti *GT*. *DGT* podobně jako u předchozího rozdělení často blokuje nevýhodné „tětivy“. Přes minimální kostru už ne vždy vylepšíme triangulaci na minimální.

	10 bodů 10 000 testů	30 bodů 1 000 testů	100 bodů 100 testů	300 bodů 30 testů
DT	1,02212679	1,06858355	1,08883722	1,07691621
GT	1,00012925	1,00141517	1,00221241	1,00234168
QGT	1,00012925	1,00141589	1,00221241	1,00234168
DGT	1,00002028	1,00183969	—	—
MST-T	1,00131486	1,00028557	1,00026048	1,00014155
GT-MST-T	1,00001376	1,00004019	1,00005687	1,00000199
QGT-MST-T	1,00001376	1,00004019	1,00005687	1,00000199
DGT-MST-T	1,00000143	1,00001141	—	—
LMT-CQGT	1,00012925	1,00141517	1,00221241	1,00234168
LMT-CQGT-CMST-T	1,00000000	1,00000078	1,00005687	1,00000199

Tabulka 6.8: Faktory náhodného rozložení bodů v blízkosti kružnice s bodem uprostřed.

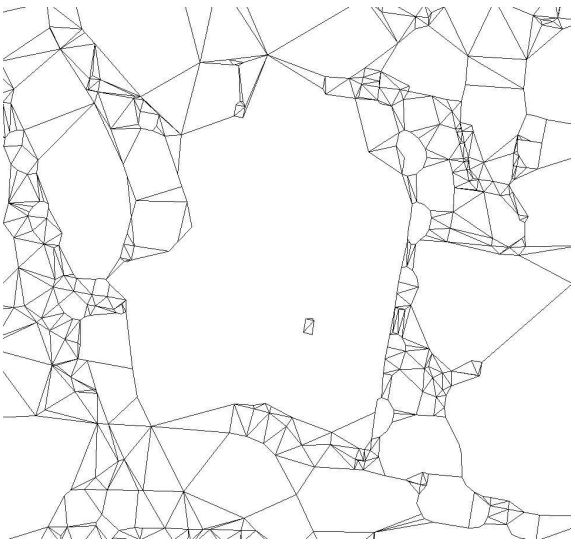


modifikovaný *LMT*-skeleton *MWT*

Obrázek 6.6: Příklad 300 bodů v blízkosti kružnice kolem středového bodu.

Přidáním dalšího bodu do středu kružnice (tabulka 6.8), se projevila rozdílná váha všech testovaných tříd triangulací vůči minimální triangulaci, včetně *DGT-MST-T* a *LMT-CQGT-CMST-T*. V modifikovaném *LMT*-skeletonu často vznikne stěna s nesouvislou hranicí (bod uprostřed) a minimální hrana spojující střed s „obvodem“ (vybraná do minimální kostry) pak může zablokovat minimální triangulaci.

Porovnáme-li efektivitu výpočtů a aproximační faktory, pak nejlepší výsledky z testovaných aproximací vykazovala triangulace *LMT-CQGT-CMST-T*. Faktor aproximace je v nejhorším případě $\Theta(1)$ a jen málokdy se liší od 1. Hodnoty se na testovacích datech pohybovaly v intervalu $[1, 1{,}00323964)$.



Obrázek 6.7: Příklad nespojité stěny v modifikovaném LMT-skeletonu.

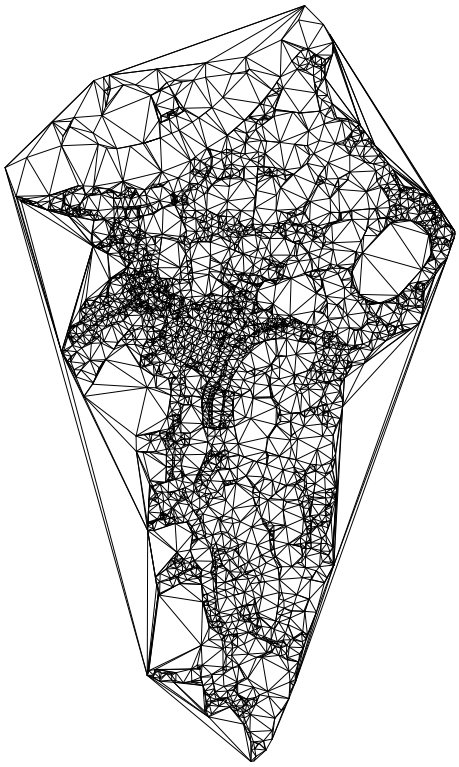
6.3 Srovnání na reálných datech

Některé třídy triangulací, konkrétně DT, GT, MWT a LMT-$CQGT$-$CMST$-T, jsem testoval i na reálných datech o velikostech 4 897 až 70 433 bodů ze zdroje: B.Žalik, University of Maribor, Database of Terrain Data. Dvojici menších množin s 4 897 a 13 829 body lze najít na obrázcích 6.8 a 6.9.

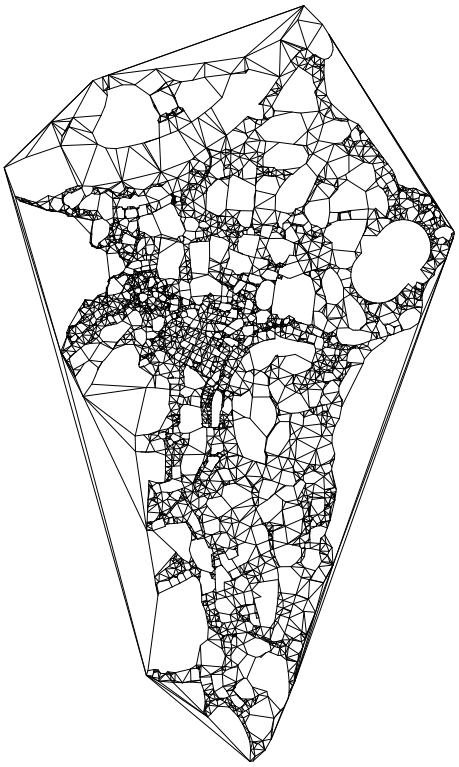
Podobně jako u náhodně generovaných dat je i u „pravidelných“ dat znát poměrně velký rozdíl minimální a Delaunayovy triangulace ve tvaru typických trojúhelníků. Greedy triangulace má k MWT mnohem blíže, ale ve všech případech se liší. Modifikovaný LMT-skeleton vyšel u téměř všech množin souvislý, z čehož vyplývají velmi dobré výsledky pro kombinovanou triangulaci LMT-$CQGT$-$CMST$-T. V průběhu výpočtu se vyskytla jediná nesouvislá stěna, a to hned nezvykle s nebodovou komponentou. Tuto stěnu znázorňuje obrázek 6.7.

Pro MWT jsem pozoroval značný rozdíl v efektivitě výpočtu na zkoumaných reálných množinách oproti rovnoměrně rozloženým množinám bodů. V datech se vyskytovaly poměrně velké oblasti bez bodů, zpomalující průchod čtvercovou sítí při výpočtu hran s diamantových oblastí. V modifikovaných LMT-skeletonech navíc vznikaly stěny s mnoha body na obvodu, které mají velký stupeň. Proto i výpočet modifikovaného LMT-skeletonu a následné doplnění stěn na $CMWT$ probíhá pomaleji než u rovnoměrně rozložených dat. Přesto se v rozumné době několika minut podařilo spočítat všechny testované množiny.

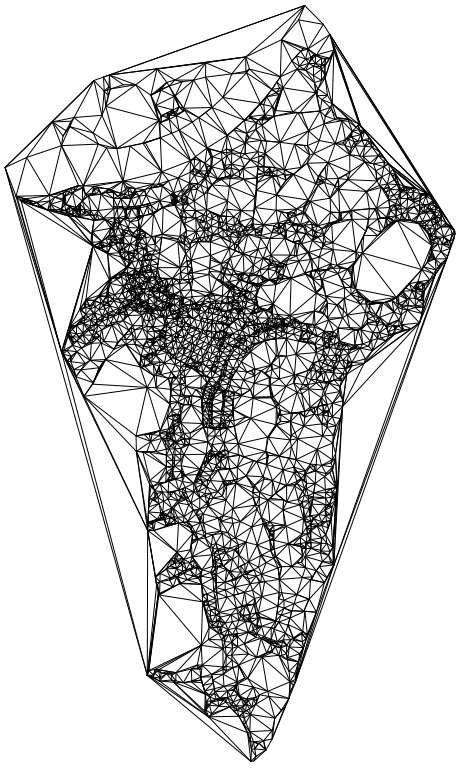
Největší čas zabralo vždy generování grafu kandidátů. Průchod velkými prázdnými oblasti by se dal částečně urychlit například vybudováním nějaké hierarchické struktury (quad-tree, *kd*-tree, R-tree, ...) místo pravidelné čtvercové sítě, čímž by pravděpodobně vznikla lepší heuristika pro testovaný typ reálných dat.



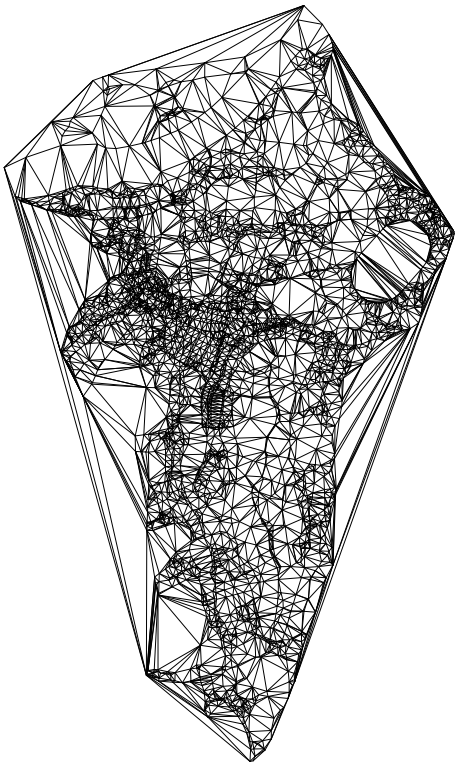
MWT, *LMT-CQGT-CMST-T*



modifikovaný *LMT*-skeleton

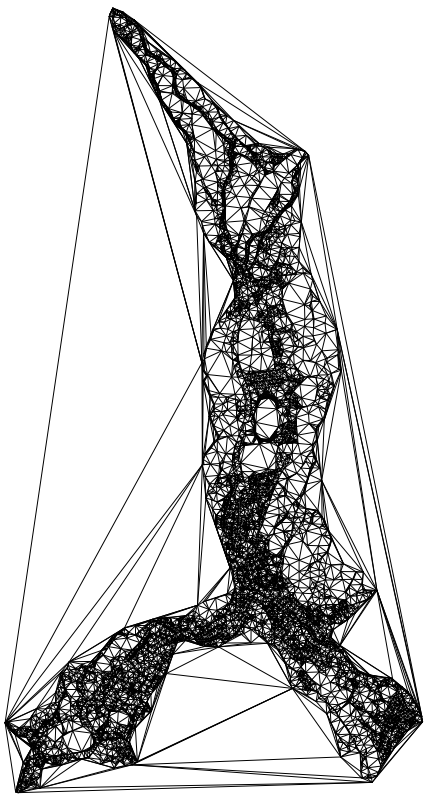


GT

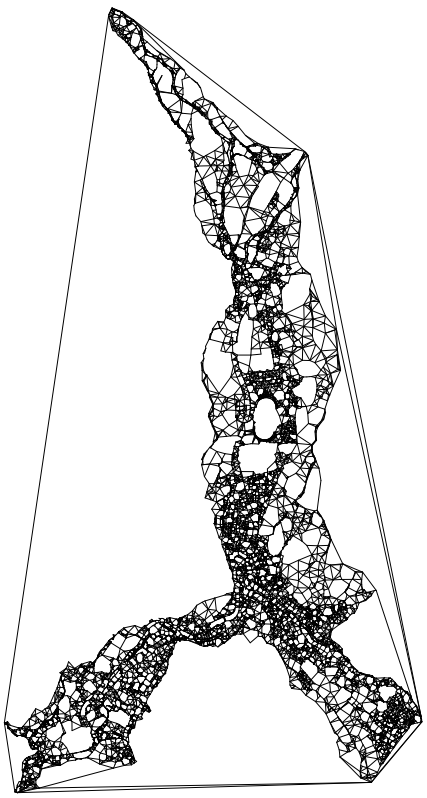


DT

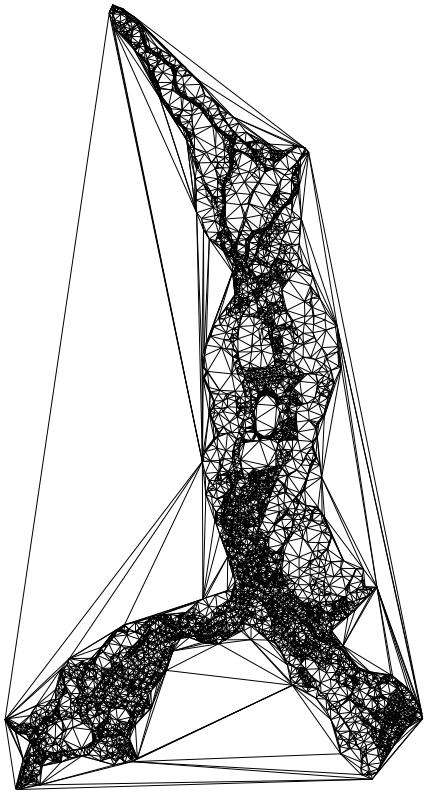
Obrázek 6.8: Příklad množiny 4 897 bodů.



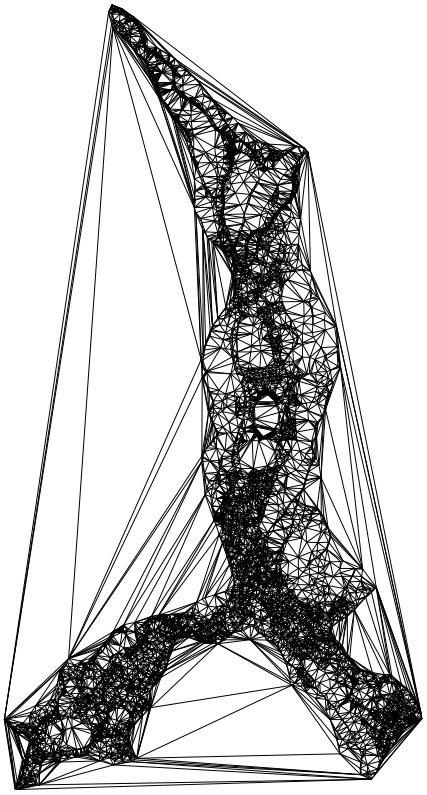
MWT, *LMT-CQGT-CMST-T*



modifikovaný *LMT*-skeleton



GT



DT

Obrázek 6.9: Příklad množiny 13 829 bodů.

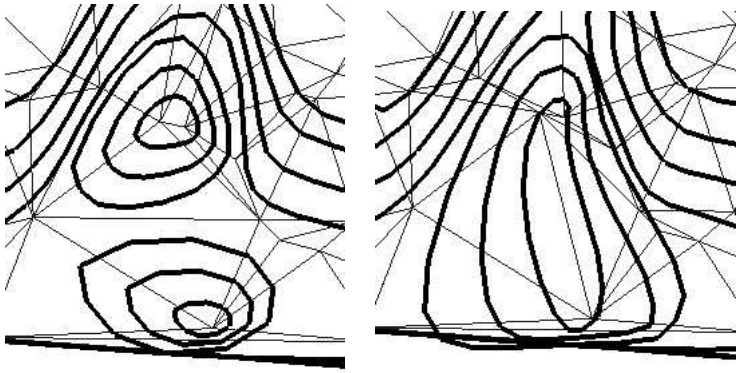
Kapitola 7

Aplikace při výpočtu vrstevnic

Triangulace s minimální váhou jsem zkusil aplikovat v konkrétní úloze výpočtu vrstevnic, kdy na vstupu máme k dispozici množinu bodů s nadmořskými výškami, a společně s greedy triangulací porovnat s nejčastěji používanou Delaunayovou triangulací. V této úloze se většinou triangulace používá jako první krok výpočtu. Ze sítě bodů sestrojíme „prostorovou“ síť trojúhelníků s třetími souřadnicemi rovnajícími se nadmořským výškám vrcholů, která představuje interpolaci terénu pro následné hledání vrstevnic.

Pro výpočet vrstevnic z dané triangulace jsem použil hotový program V. Strycha z jeho diplomové práce [30]. Příklad jedné z testovaných množin vrcholů znázorňuje obrázek 7.3, kde jsou vidět vstupní triangulace i s výslednými vrstevnicemi.

Při zkoumání výsledků na různých množinách jsem oproti [30, 22] nedospěl k ničemu novému. Tvar typických trojúhelníků minimální triangulace se velmi podobá greedy triangulaci a většinou mají tvar blízko rovnostrannému trojúhelníku nebo jsou velmi úzké. Oproti tomu Delaunayova triangulace obsahuje méně extrémně úzkých trojúhelníků, jelikož maximalizuje minimální úhel v triangulaci. Na druhou stranu v ní ale v mnohem větší míře najdeme středně úzké podlouhlé trojúhelníky.



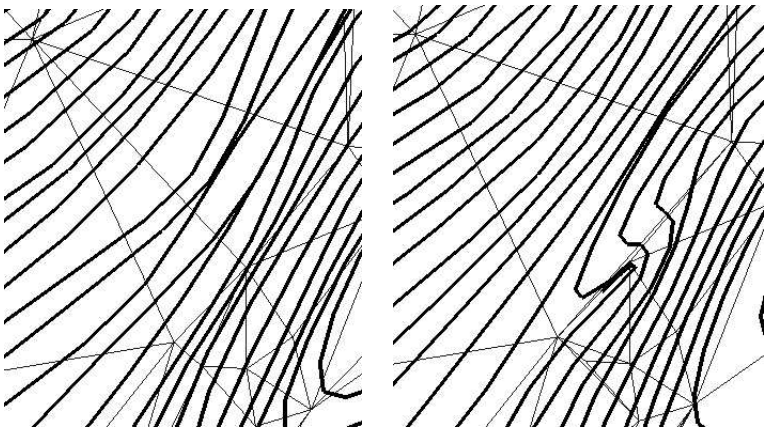
DT *MWT*

Obrázek 7.1: Velký rozdíl vrstevnic při prohození diagonály.

Nelze jednoznačně říct, která triangulace se pro vrstevnice hodí nejvíce, protože

existují příklady, kde „chybují“ všechny zmíněné třídy triangulací. Pro kvalitu výsledné sítě vrstevnic je spíše důležitá hustota vstupních vrcholů, zejména v oblastech s velmi členitým terénem. Jak ukazuje obrázek 7.1, může v opačném případě i malá změna v triangulaci znamenat podstatný rozdíl ve výsledku.

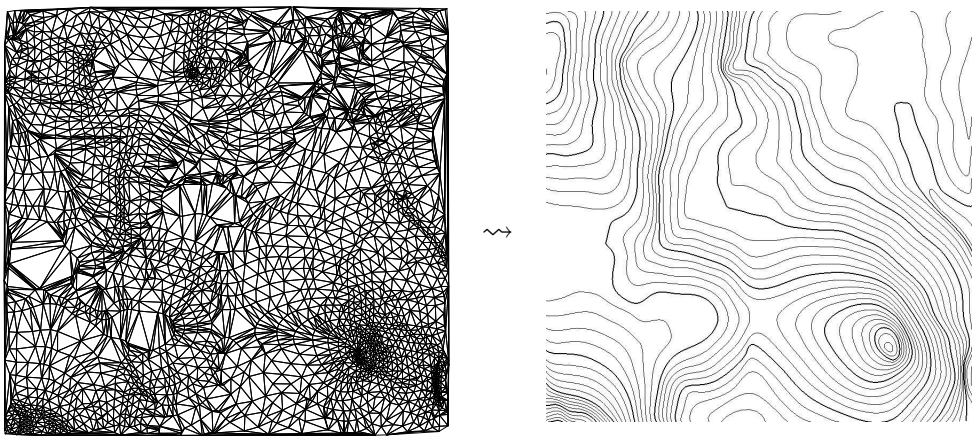
Jeden z problémových případů pro greedy triangulaci nebo minimální triangulaci znázorňuje obrázek , kde úzký trojúhelník způsobí extrémně velké zakřivení vrstevnice. Podobné situace ovšem existují i pro Delaunayovy triangulace [30, 22].



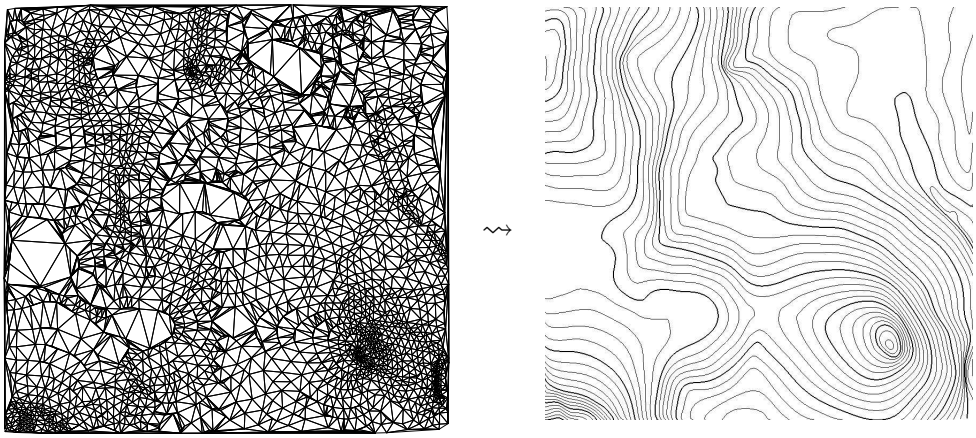
Obrázek 7.2: Problém úzkých trojúhelníků při výpočtu vrstevnic.

Z testů vyplývá, že je pro výslednou kvalitu vrstevnic v podstatě jedno, kterou ze zmíněných tříd triangulací při výpočtu použijeme. Rozdíly nebývají velké a všechny problémové situace jen těžko obecně vyřešíme automatizovanou rovinnou triangulací. Lepší výsledky by spíše přineslo zahrnutí třetí souřadnice jako kritérium pro triangulace nebo ruční nastavení povinných hran uživatelem ve složitých situacích.

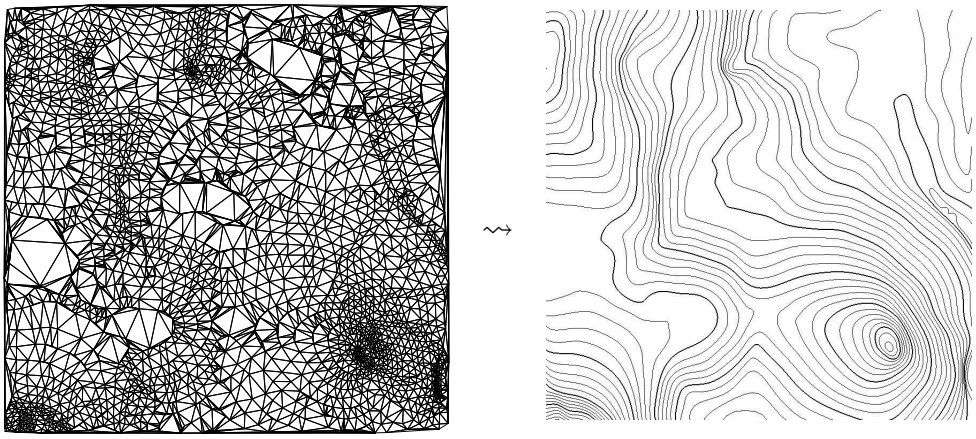
Kdybychom se rozhodovali na základě efektivity algoritmu, vyjde nejlépe Delaunayova triangulace, pro níž jsou známy jednoduché a rychlé algoritmy s $O(n \log n)$ časovou složitostí v nejhorším případě, případně greedy triangulace se stejnou asymptotickou složitostí. Minimální triangulace jde sice pro rovnoměrně rozmístěné body počítat také poměrně rychle, ale mohly by nastat situace, kdy by algoritmus velmi zpomalil. To samé platí pro *LMT-CQGT-CMST-T* s časovou složitostí $O(n^4)$.



DT



GT



MWT

Obrázek 7.3: Použití různých triangulací ve výpočtu vrstevnic.

Kapitola 8

Shrnutí výsledků

Na závěr diplomové práce mohu s čistým svědomím říct, že se podařilo splnit všechny zadané úkoly, kterými bylo prostudování současného stavu problematiky minimálních triangulací, navržení vhodného postupu pro přibližné nebo přesné řešení a porovnání na vhodných datech a v některé konkrétní úloze s jinými známými přístupy.

Zaměřil jsem se zejména na návrh vlastního řešení optimalizovaného na rovnoměrně rozložená data. Algoritmus byl inspirován několika známými přístupy, ovšem podstatnou část jsem odvodil a dokázal samostatně. Narazil jsem také na drobný problém v původních zdrojích a navrhnul jeho řešení se zachováním časové složitosti. Výsledný algoritmus funguje nejlépe na rovnoměrně rozmístěných množinách bodů, kde vykazuje (téměř) lineární složitost a umožňuje tak spočítat přesnou minimální triangulaci až statisíců bodů řádově během několika minut. Lze ovšem použít i pro libovolná jiná vstupní data a jen ve velmi speciálních případech dosáhne exponenciální složitosti.

Popsaný algoritmus, společně s výpočty některých dalších druhů triangulací, jsem bez využití speciálních knihoven nebo převzatého kódu implementoval v prostředí Windows pod Delphi 7.0. Program je, včetně zdrojového kódu a testovacích dat, přiložený k diplomové práci.

Minimální triangulace jsem porovnal s jinými známými třídami triangulací a experimentálně zjistil na náhodně generovaných množinách vrcholů jejich aproximační faktory. Na základě experimentů a poznatků o různých heuristikách a aproximacích jsem navrhnul vlastní třídu triangulací, která kombinuje heuristiku modifikovaného LMT-skeletonu s omezenou quasi-greedy triangulací a s vylepšením triangulace přes minimální kostru (LMT-$CQGT$-$CMST$-T). Tato aproximace má časovou složitost nejhůře $O(n^4)$ a pro náhodně rozloženou množinu bodů lze spočítat v (téměř?) lineárním očekávaném čase. Na náhodně generovaných datech se jen zřídka lišila od optimální triangulace a i v obecném případě se dá dokázat její aproximační faktor $\Theta(1)$.

Nakonec jsem zkusil použít minimální triangulace jako vstup pro úlohu výpočtu vrstevnic a stručně porovnal s Delaunayovou triangulací. Minimální triangulace většinou obsahují trojúhelníky podobného tvaru jako greedy triangulace a výsledky experimentů se tak shodovaly s citovanými prameny srovnávajícími DT s GT.

Tématem pro další navazující práci by mohlo být vylepšení výpočtu MWT pro

méně rovnoměrné množiny vstupních dat, především když obsahují rozsáhlé oblasti bez bodů. Pomoct by mohla například náhrada pravidelné čtvercové sítě při průchodu rovinou nějakou hierarchickou strukturou.

Dále by mohlo být zajímavé porovnat různé třídy omezených triangulací a zjistit, zda pro ně nešla dokázat podobná zakázaná oblast, jako u neomezené verze, například zkombinováním s viditelností v omezujícím PSLG. Kdyby šly odvozené výsledky použít pro efektivní výpočet $o(n^2)$ kandidátních hran omezené triangulace (nejlépe $O(n)$), získali bychom v průměrném případě poměrně rychlý výpočet tříd omezených triangulací CGT, $CMWT$ a $CLMT$-$CQGT$-$CMST$-T.

Seznam obrázků

Seznam tabulek

Seznam algoritmů

Literatura

[1] Aichholzer O. (1995): Local properties of triangulations. *In Proc. 11th European Workshop on Computational Geometry CG '95*, 27–30.

[2] Aurenhammer F. (1991): Voronoi Diagrams – A Survey of a Fundamental Geometric Data Structure. *ACM Computing Surveys*, **23**(**3**), 345–405.

[3] Beirouti R. (1997): A fast heuristic for finding the minimum weight triangulation. *Technical Report TR-97-11*, 10.

[4] Dickerson M., Drysdale R., McElfresh S. A., Welzl E. (1994): Fast greedy triangulation algorithms. *In Symposium on Computational Geometry*, 211–220.

[5] Dickerson M. T., Montague M. H. (1996): A (usually?) connected subgraph of the minimum weight triangulation. *In SCG '96: Proceedings of the Twelfth Annual Symposium on Computational Geometry*, 204–213.

[6] Drysdale R., Rote G., Aichholzer O. (1995): A simple linear time greedy triangulation algorithm for uniformly distributed points. *IIG-Report-Series 408*, TU Graz, Austria. Presented at the Workshop on Computational Geometry.

[7] Drysdale R., McElfresh S. A., Snoeyink J. (2001): *On exclusion regions for optimal triangulations.* Discrete Applied Mathematics, **109**(**1**–**2**), 49–65.

[8] Dwyer R. A. (1989): Higher-dimensional Voronoi diagrams in linear expected time. *SCG '89: Proceedings of the Fifth Annual Symposium on Computational Geometry*, 326–333.

[9] Ferko A. (2004): Approaching the Minimum Weight Triangulation Problem. *habilitační práce*, Univerzita Komenského v Bratislavě.

[10] Fortune S. (1992): Voronoi Diagrams and Delaunay Triangulations. *In Du, D.A. and Hwang, F.K. (eds.): Euclidean Geometry and Computers.* World Scientific Publishing, Singapore, 193–233.

[11] Gabow H. N., Galil Z., Spencer T., Tarjan R. E. (1986): Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *J. ACM*, **42**, 321–328.

[12] Gilbert P. D. (1979): New results in planar triangulation. *Master's thesis.*

[13] Heath L. S., Pemmaraju S. V. (1992): New results for the minimum weight triangulation problem. *Technical report, Blacksburg*, VA, USA.

[14] Cheng S-W., Xu Y-F. (1996): Approaching the largest beta-skeleton within a minimum weight triangulation. *In Symposium on Computational Geometry*, 196–203.

[15] Chin F., Wang C. A. (1999): Finding the Constrained Delaunay Triangulation and Constrained Voronoi Diagram of a Simple Polygon in Linear Time *SIAM J. Comput.*, **28**(**2**), 471–486

[16] Jansson J. (1995): Planar Minimum-Weight Triangulations. *MS Thesis*, Rep. LU-CS-EX:95-16, Lund University, Sweden.

[17] Keil J. M. (1994): Computing a subgraph of the minimum weight triangulation. *Comput. Geom. Theory Appl.*, **4**(**1**), 13–26.

[18] Klincsek G. T. (1980): Minimal triangulations of polygonal domains. *Annals of Discrete Mathematics*, 121–123.

[19] Ferko A., Niepel L., Kolingerová I., Magová. I. (1997): Better subgraph of minimum weight triangulation. *In Spring Conference on Computer Graphics sccg 97*, conference proceedings, Bratislava.

[20] Kolingerová I. (1999): Greedy Triangulation Improvement over Lookahead Search. *Computer Engineering and Informatics CE&I ´99*, conference Proceedings, 34–39, STU Košice, Herlany.

[21] Kolingerová I. (1999): Rovinné triangulace. *Habilitační práce*, University of West Bohemia, Pilsen, Czech Repusblic.

[22] Kolingerová I., Strych V., Čada V. (2004): Using Constraints in Delaunay and Greedy Triangulation for Contour Line Improvement. *Keynote spech for the Computer Graphics and Geometric Modelling Workshop*, Computational Science - ICCS 2004, Krakow, Poland, 123–130.

[23] Levcopoulos C. (1987): An $\Omega(\sqrt{n})$ lower bound for the nonoptimality of the greedy triangulation. *Inf. Process. Lett.*, **25**(**4**), 247–251.

[24] Levcopoulos C., Lingas A. (1992): Fast algorithms for greedy triangulation. *BIT*, **32**(**2**), 280–296.

[25] Levcopoulos C., Krznaric D. (1996): Quasi-greedy triangulations approximating the minimum weight triangulation. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms).*

[26] Levcopoulos C., Krznaric D. (1999): The greedy triangulation can be computed from the delaunay triangulation in linear time. *Comput. Geom. Theory Appl.*, **14**(**4**), 197–220.

[27] Nociar M. (2002): Triangulácie v rovine a teréne. *MSc thesis*. Bratislava, Comenius University 2004.

[28] Plaisted D. A., Hong J. (1987): A heuristic triangulation algorithm. *J. Algorithms*, **8**(**5**), 405–437.

[29] Smith W. D. (1989): Implementing the Plaisted-Hong min-length plane triangulation heuristic. Nepublikováno.

[30] Strych V. (2003): Triangulace a editování vrstevnic. *Diplomová práce*, Západočeská univerzita v Plzni.