Masarykova univerzita Přírodovědecká fakulta

Bakalářská práce

Diagramy Voronoia

Jana Tomšů

Vedoucí bakalářské práce:
doc. RNDr. Martin Čadek, CSc.

Brno Květen 2008

Prohlášení

Prohlašuji, že bakalářskou práci jsem vypracovala samostatně s použitím uvedené literatury a za odborného dohledu doc. RNDr. Martina Čadka, CSc.

V Brně dne ....
Jana Tomšů

Poděkování

Na tomto místě bych ráda poděkovala doc. RNDr. Martinu Čadkovi, CSc. za odborné vedení, laskavý přístup a trpělivost při zpracování této práce.

Obsah

Úvod

Diagram Voronoia je rozdělení roviny na části, které je určeno vzdálenostmi bodů od určitých míst.

Cílem této bakalářské práce je popsat vlastnosti tohoto rozdělení a algoritmus pro jeho konstrukci. V práci je také zmíněna dualita diagramu Voronoia a Delaunayova podrozdělení a možnost využití diagramu Voronoia v praxi.

Součástí práce je program psaný v programovém prostředí Matlab. Tento program pracuje na principu metody zametací přímky. Uživatelské prostředí tohoto programu umožňuje zobrazení diagramu Voronoia, jeho konstrukce a Delaunayovy triangulace pro různé množiny bodů. Tento program je k dispozici na přiloženém CD.

Teoretická část práce je rozdělena do pěti kapitol. V první je popsán diagram Voronoia a jeho vlastnosti. Ve druhé kapitole se mluví obecně o algoritmech a je zde podrobně popsán algoritmus na konstrukci diagramu Voronoia pomocí metody zametací přímky. Třetí kapitola popisuje dualitu diagramu Voronoia a Delaunayova podrozdělení. Čtvrtá kapitola se zabývá programem, popisuje funkce uživatelského prostředí a jsou zde uvedeny zdrojové kódy hlavních částí programu. Poslední kapitola popisuje aplikaci diagramu Voronoia na problém spádových oblastí nemocnic v Brně a uplatnění diagramu Voronoia v robotice. Při tvorbě této práce bylo čerpáno převážně z knihy Computational Geometry: Algorithms and Applications [2], ale také z prací Voronoi diagrams [1] a Geometric data structures and their selected applications [4] a článku [3].

1 Diagram Voronoia

1.1 Základní pojmy a definice

Diagram Voronoia (zjednodušeně V-diagram) budeme definovat v rovině - dvourozměrném euklidovském prostoru. Vzdálenost dvou bodů $x = (x_1, x_2), y = (y_1, y_2)$ v euklidovské rovině je dána vztahem

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}.$$

V této rovině uvažujme množinu aspoň tří bodů $P = (p_1, p_2, ..., p_n)$. Osa o úsečky p_i, p_j daná vztahem

$$o(p_i, p_j) = \{x|\ d(p_i, x) = d(p_j, x)\}$$

je přímka, rozdělující rovinu na dvě poloroviny h_{ij}, h_{ji}. Polorovina

$$h_{ij} = \{x|\ d(p_i, x) < d(p_j, x)\}$$

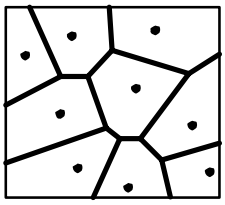
obsahuje všechny body roviny, které jsou blíže k p_i než k p_j. Buňkou $V(p_i)$ bodu p_i budeme nazývat část roviny takovou, že každý bod x buňky je blíže k bodu p_i než k ostatním bodům z množiny P, tedy platí

$$V(p_i) = \{x|\ d(p_i, x) < d(p_j, x) \text{ pro všechna } j \neq i\}.$$

Tuto buňku můžeme dostat jako průnik výše definovaných polorovin

$$V(p_i) = \bigcap_{j=1}^{n} h_{ij}.$$

Pro množinu P bodů v rovině je V-diagram definován jako rozdělení roviny, které každému bodu p_i přiřadí buňku $V(p_i)$. V-diagram je tedy tvořen buňkami, hranicemi buněk, které oddělují dvě buňky a vrcholy, ve kterých se setkávají tři a více buněk.



Obrázek 1: *Příklad diagramu Voronoia.*

1.2 Vlastnosti

Definujme $C_q(P)$ jako kružnici se středem q, která má na obvodu jeden nebo více bodů z P a uvnitř nemá žádný.

Hrana V-diagramu je hranice oddělující dvě buňky $V(p_i)$ a $V(p_j)$. Ta je tvořena body z roviny, které jsou nejblíže právě ke dvěma bodům z P (p_i a p_j), tedy jsou stejně vzdálené od těchto bodů a leží na ose $o(p_i, p_j)$ úsečky p_i, p_j.

Odtud dostáváme, že bod q leží na hraně V-diagramu, právě tehdy, když na kružnici $C_q(P)$ leží právě dva body z P.

Vrchol V-diagramu je místo setkání tří nebo více buněk, je proto stejně vzdálený od tří nebo více bodů z množiny P, ke kterým je nejblíže. Vrchol je tedy středem kružnice na níž leží tyto body, a uvnitř neleží žádný.

Odvodili jsme, že bod q je vrcholem V-diagramu, právě tehdy, když na kružnici $C_q(P)$, leží tři nebo více bodů z množiny P.

Lemma 1.1. *Nechť e je počet hran a v počet vrcholů ve V-diagramu n-prvkové množiny P. Pak $e \leq 3n - 9$ a $v \leq 2n - 5$.*

Důkaz. Podle Eulerovy formule pro rovinné grafy platí

$$v - e + f = 2,$$

kde v je počet vrcholů, e počet hran a f počet oblastí.

V-diagram obsahuje i nekonečné hrany, aby pro něj platila Eulerova formule, přidáme jeden vrchol v nekonečnu. Počet vrcholů bude tedy $v + 1$. Počet oblastí−buněk, je stejný jako počet bodů v množině P, tedy n.

Platí, že každý vrchol náleží nejméně třem hranám, a protože každá hrana má dva vrcholy, dostáváme

$$2e \geq 3(v + 1).$$

Dosazením $e \geq \frac{3}{2}(v + 1)$ do Eulerovy formule dostáváme

$$v + 1 - \tfrac{3}{2}(v + 1) + n \geq 2$$

$$v \leq 2n - 5.$$

Dosazením $v \leq \frac{2}{3}e - 1$ do Eulerovy formule dostáváme

$$\tfrac{2}{3}e - 1 - e + n \geq 2$$

$$e \leq 3n - 9.$$

□

2 Algoritmus

2.1 Vlastnosti algoritmů

Naším cílem je popsat algoritmus pro konstrukci V-diagramu. Nejdříve si řekneme něco obecně o algoritmech.

Algoritmem rozumíme postup, podle kterého se ze vstupních dat vygenerují data výstupní. Vstupní data jsou algoritmu předány před započetím jeho provádění nebo v průběhu jeho činnosti. Mají definovanou množinu hodnot, kterých mohou nabývat. V našem případě jsou to body roviny, zadané svými souřadnicemi.

Algoritmus má alespoň jeden výstup, který je v požadovaném vztahu k zadaným vstupům. V našem případě je výstupem V-diagram.

Základní vlastností algoritmu je, že musí skončit po konečném počtu kroků v konečném čase. Každý krok algoritmu musí být jednoznačně a přesně definován a určen výsledkem kroku předchozího.

Algoritmus nesmí řešit jen jeden konkrétní problém pro konkrétní vstupní data, ale problém obecný pro různé vstupní údaje.

Požadavkem je, aby byl efektivní. Efektivita je popsána časovou a paměťovou asymptotickou složitostí algoritmu.

Asymptotická složitost algoritmu vyjadřuje jak rostou jeho nároky na výpočetní výkon nebo čas v závislosti na množství nebo velikosti vstupních dat. Zapisuje se pomocí Omikron notace jako $O(f(n))$, kde n je veličina popisující velikost vstupních dat.

Nechť $f(n)$ a $g(n)$ jsou funkce definované na N. Potom zápis $g(n) = O(f(n))$ znamená, že existuje $C > 0$ a $n_0 \in N$ tak, že pro všechna $n > n_0$ je $|g(n)| \leq |Cf(n)|$.

2.2 Algoritmus pro konstrukci V-diagramu

K nalezení optimálního algoritmu je třeba určit nejmenší možnou asymptotickou složitost algoritmu.

Uvažujme situaci, kdy leží všechny body množiny P na přímce. V-diagram pro tyto body je tvořen osami úseček bodů ležících na přímce vedle sebe. Vytvořit V-diagram je v tomto případě totéž jako najít pořadí bodů množiny P na přímce, viz [1]. K tomu je potřeba čas $O(n \log n)$. Nejmenší možná asymptotická složitost algoritmu pro konstrukci V-diagramu je tedy rovna $O(n \log n)$.

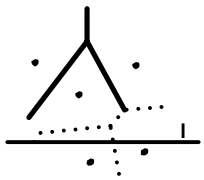
Jeden ze způsobů výpočtu V-diagramu je pomocí průniku polorovin. Hledáme rozdělení roviny na buňky, jednu buňku pro každý bod z množiny P. Jak jsme uvedli v první kapitole, lze buňku bodu p_i vypočítat jako průnik polorovin určených osami úseček p_i, p_j, kde p_j jsou zbylé body z množiny P. Protože však nalezení jedné buňky vyžaduje čas $O(n \log n)$ a počet buněk je n, je celková náročnost takového algoritmu $O(n^2 \log n)$, a tedy není optimální.

Jiný způsob je pomocí metody zametací přímky. Tato metoda má nároky na čas $O(n \log n)$, je tedy optimální, a proto se jí budeme dále zabývat.

2.2.1 Metoda zametací přímky

Zametací přímkou je myšlena vodorovná přímka, která se pohybuje od shora dolů přes celou rovinu a postupně „zametá“ body. Označíme ji l.

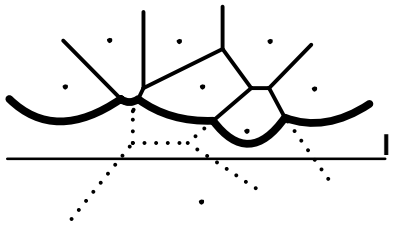
Tato metoda by měla fungovat tak, že pokud se dostaneme zametací přímkou l na místo $y = a$, body pod l neovlivňují výsledek algoritmu nad přímkou l. Toto však neplatí u V-diagramu. Příkladem je situace, kdy prochází přímka l nejvyšším vrcholem nějaké buňky, který ještě nebyl nalezen. Viz obrázek 2.



Obrázek 2: *Body pod přímkou l ovlivňují část plochy nad přímkou.*

K nalezení oblasti nad zametací přímkou l, kterou již nebudou ovlivňovat body pod l využijeme vlastnosti V-diagramu.

Vezněme libovolný bod q z roviny nad přímkou l. Aby tento bod nebyl ovlivněn body z množiny P ležícími pod nebo na přímce l, musí být blíže k nějakému bodu z P ležícímu nad l. Hranice místa nad l, které již nebude ovlivněno body pod přímkou l je tedy určena body, které jsou stejně vzdáleny od bodu z množiny P ležícího nad l a od přímky l. Body stejně vzdálené od bodu a přímky určují parabolu, proto je tato hranice tvořena částmi parabol. Nazývá se bojová linie (v originále beach line). Bojová linie je určena pořadím parabol, které tvoří nějakou její část. Každá parabola je určena bodem z množiny P, který je jejím ohniskem.



Obrázek 3: *Bojová linie.*

Při posunu zametací přímky směrem dolů se směrem dolů posunuje a mění i bojová linie a vytváří přitom hrany a vrcholy V-diagramu.

Bod průniku dvou parabol z bojové linie je místo stejně vzdálené od dvou bodů z množiny P. Leží tedy na hraně V-diagramu.

Pokud dojde k tomu, že se tři paraboly z bojové linie setkají v jednom bodě, bude to bod stejně vzdálený od tří bodů z množiny P a bude tvořit vrchol V-diagramu.

Během průchodu zametací přímky mohou nastat dvě události. K prvé dochází při průchodu přímky l bodem z množiny P. Tato událost se nazývá bodová událost (v originále

site event). Po průchodu přímky l takovým bodem vzniká v bojové linii nová parabola. Vznikem paraboly na bojové linii se objevuje v diagramu nová hrana.

Druhá událost je vznik vrcholu. Nazývá se kruhová událost (circle event), protože nastává v místě průchodu přímky l tzv. kruhovým bodem.

Kruhový bod je nejnižší bod kružnice $C_q(P)$ na níž leží právě tři body z množiny P. Projde-li přímka tímto bodem, setkají se tři paraboly v jednom místě. Tím vznikne vrchol diagramu a zanikne jedna parabola.

Bodovou i kruhovou událost si blíže popíšeme dále.

Lemma 2.1. *Bojová linie se mění pouze při průchodu zametací přímky těmito dvěma událostmi.*

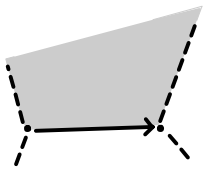
Důkaz viz [2].

2.2.2 Datové struktury

Datové struktury používáme k ukládání dat během průchodu algoritmem.

V-diagram ukládáme do dvojitě souvislého seznamu. Tato datová struktura ukládá záznam pro každou hranu, vrchol a oblast(buňku) V-diagramu.

Jedna hrana diagramu leží na hranici dvou oblastí. K odlišení těchto dvou oblastí, které dělí jedna hrana, se využívá polohran. Polohrany jsou orientované. Každá polohrana leží na hranici pouze jedné oblasti. Tato oblast je určená orientací polohrany, leží od polohrany nalevo. Viz obrázek.



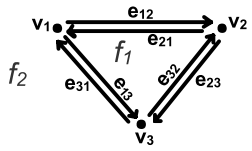
Obrázek 4: *Oblast kterou ohraničuje orientovaná polohrana.*

Z jedné hrany vzniknou dvě polohrany tak, že jedna z nich má počátek v jednom vrcholu hrany, druhá ve druhém a jsou opačně orientované. Těmto dvěma polohranám vzniklým z jedné hrany se říká dvojčata. Díky orientovanosti polohran můžeme určit, jak na sebe navazují. Každá polohrana má svého předchůdce a následovníka. Seznam pro polohrany ukládá dvojče příslušné polohrany, předchůdce, následníka, počáteční vrchol a oblast příslušnou této hraně.

Seznam pro vrcholy ukládá pro každý vrchol jeho souřadnice a polohranu, jejímž je počátečním vrcholem.

Seznam pro oblasti ukládá jednu polohranu ležící na vnější hranici této oblasti. Pokud je oblast neohraničená je tato hodnota prázdná (*nil*). Dále ukládá jednu polohranu ležící na hranici každé oblasti, která leží uvnitř této oblasti. Pokud uvnitř dané oblasti neleží žádné další oblasti, je tato hodnota prázdná.

Zde je příklad dvojitě souvislého seznamu pro jednoduchý graf:



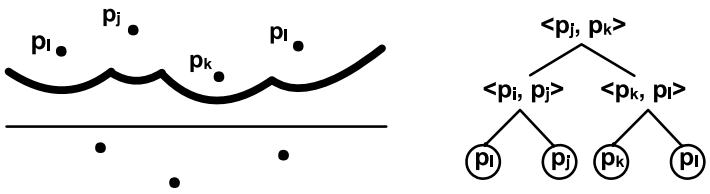
Oblast	Polohrana z vnější hranice	Polohrany vnitřních oblastí
f_1	e_{21}	nil
f_2	nil	e_{12}

Polohrana	Počáteční vrchol	Dvojče	Oblast	Následující	Předchozí
e_{12}	v_1	e_{21}	f_2	e_{23}	e_{31}
e_{21}	v_2	e_{12}	f_1	e_{13}	e_{32}
e_{23}	v_2	e_{32}	f_2	e_{31}	e_{12}
e_{32}	v_3	e_{23}	f_1	e_{21}	e_{13}
e_{31}	v_3	e_{13}	f_2	e_{12}	e_{23}
e_{13}	v_1	e_{31}	f_1	e_{32}	e_{21}

Vrchol	Souřadnice	Polohrana
v_1	$(0, 2)$	e_{12}
v_2	$(2, 2)$	e_{23}
v_3	$(1, 0)$	e_{31}

Další datovou strukturou je vyvážený binární vyhlédávací strom. V něm ukládáme bojovou linii. Binární vyhledávací strom je datová struktura založená na binárním stromu, v němž jsou jednotlivé prvky (uzly) uspořádány tak, aby v tomto stromu bylo možné rychle vyhledávat danou hodnotu. Strom se skládá z uzlů a listů. Každý uzel stromu má právě dva syny - levého a pravého. Každému uzlu je přiřazen určitý klíč, v našem případě $x-$ová souřadnice. Podle hodnot těchto klíčů jsou uzly uspořádány. Levý podstrom uzlu obsahuje pouze klíče menší než je klíč tohoto uzlu a pravý podstrom uzlu obsahuje pouze klíče větší. Koncové uzly, které nemají žádné potomky, jsou listy. V našem algoritmu listy odpovídají jednotlivým parabolám na bojové linii. Jsou seřazeny podle x-ové souřadnice, parabole nejvíce vlevo odpovídá list nejvíce vlevo ve stromu. V každém listu je uložen bod z množiny P, který je ohniskem dané paraboly, a také ukazatel na kruhovou událost, při které daná parabola zmizí (pokud existuje). Kruhové události nejsou známy na začátku provádění algoritmu, během průchodu algoritmem se přidávají nebo odebírají, jak bude ukázáno dále.

Uzly stromu reprezentují průniky parabol, ty jsou ukládány ve formě dvojic $< p_i, p_j >$ kde p_i je parabola nalevo od průniku a p_j je parabola napravo. V uzlech je uložen ukazatel na hranu ve dvojitě souvislém seznamu, která obsahuje bod průniku těchto parabol.



Obrázek 5: *Příklad binárního vyhledávacího stromu.*

Poslední datovou strukturou je fronta událostí reprezentující zametací přímku. Má strukturu prioritní fronty. Data v ní uložená jsou uspořádána podle velikosti. Používáme lexikografické uspořádání. Toto uspořádání je definováno vztahem

$$(a, b) \geq (c, d) \Leftrightarrow (b > d \vee (b = d \wedge a \leq c)).$$

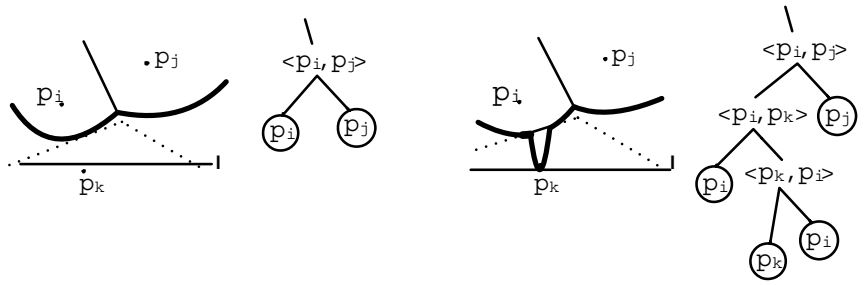
Protože se zametací přímka pohybuje od shora dolů, budou data uspořádána prvně podle souřadnice y a pak podle x. Prvek vkládaný do fronty se zařadí na správné místo a z fronty vybíráme prvek, který je podle uspořádání nejvýše. Fronta událostí obsahuje události, na které bude algoritmus reagovat - body z množiny P a postupně do ní přidáváme nebo z ní odebíráme kruhové body.

2.2.3 Algoritmus

Algoritmus pro konstrukci V-diagramu se skládá ze tří částí.

První je hlavní část, kterou voláme příkazem $VoronoiDiagram(P)$. Vstupním parametrem této části je daná množina P bodů v rovině, výstupním parametrem je dvojitě souvislý seznam, který určuje V-diagram pro vstupní množinu bodů. Tato část simuluje pohyb zametací přímky. Jak prochází zametací přímka jednotlivými body, vybírá je z fronty událostí a reaguje na ně. Aby byl dvojitě souvislý seznam úplný, nesmí obsahovat nekonečné hrany jako přímky a polopřímky. Toho dosáhneme tím, že V-diagram obalíme čtvercem dostatečně velkým, aby se do něj vešly všechny vrcholy nalezené v průběhu algoritmu. Ve výsledku bude tedy dvojitě souvislý seznam navíc obsahovat záznamy pro hrany a vrcholy čtverce a body průniku nekonečných hran s hranami čtverce.

Druhá je část, která reaguje na průchod bodem z množiny P. Volá se příkazem $HandleSiteEvent(p_i)$. Vstupní parametr p_i je bod z množiny P, kterým prochází zametací přímka. Během průchodu bodem vzniká na bojové linii nová parabola. Ta vznikne nad parabolou, která v té době leží přímo nad bodem p_i a vytvoří se dva nové body průniku těchto parabol. Z toho důvodu se nahradí ve stromě list reprezentující parabolu nad bodem p_i podstromem, který má v uzlech nově vzniklé body průniku a v listech tyto dvě paraboly. Viz obrázek.

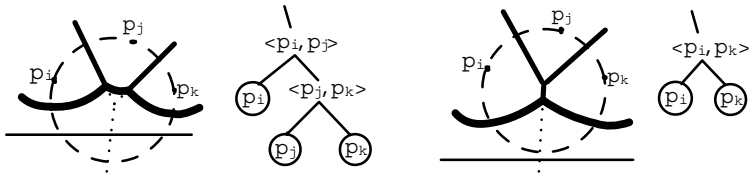


Obrázek 6: *Site event - vznik paraboly a hrany.*

Aby byla práce se stromem efektivní, je třeba, aby byl vyvážený, tj. aby vzdálenost od kořene ke každému listu byla co nejkratší. Proto se po nahrazení listu provede vyvážení stromu. Vznikem průsečíků parabol se objevuje také nová hrana, jdoucí od jednoho

průsečíku ke druhému. Je uložena do dvojitě souvislého seznamu. Dále se kontroluje vznik nových kruhových bodů. Jak bylo popsáno výše, jsou kruhové body nejnižšími body kružnice $C_q(P)$ na níž leží tři body z množiny P. Tyto tři body jsou ohniskem tří sousedních parabol. Zkontrolujeme tedy tři nově vzniklé trojice sousedících parabol. V první trojici je nově vzniklá parabola napravo, ve druhé uprostřed a ve třetí nalevo. Jestliže tři body tvořící tyto trojice parabol dávají kruhový bod, který leží pod zametací přímkou, zapíšeme jej do fronty událostí.

Třetí část je volána při průchodu kruhovým bodem příkazem $HandleCircleEvent(p_l)$. Vstupní parametr p_l je zde kruhový bod, který určuje kružnici $C_q(P)$, na níž leží tři body z množiny P. Tyto tři body jsou ohniskem tří sousedících částí parabol. Při průchodu zametací přímky tímto bodem se setkají tyto tři paraboly v jednom bodě – ve středu dané kružnice. Tím zanikne prostřední parabola. Ze stromu reprezentujícího bojovou linii se proto odstraní list, odpovídající zanikající parabole a aktualizuje se. Místo setkání tří parabol je vrcholem V-diagramu, který se přidá do dvojitě souvislého seznamu. Vznik vrcholu značí konec tří hran, které se v tomto vrcholu setkají. To je třeba zapsat do dvojitě souvislého seznamu. Zánikem paraboly také vznikne nový bod průniku zbylých dvou parabol. Tím vzniká nová hrana. Nakonec se kontrolují kruhové body, protože zánikem paraboly vznikly dvě nové trojice sousedících parabol. Jestliže tři body tvořící tyto dvě trojice parabol dávají kruhový bod, který leží pod zametací přímkou, zapíšeme jej do fronty událostí.



Obrázek 7: *Circle event - vznik vrcholu.*

Lemma 2.2. *Počet částí parabol v bojové linii je vzhledem k počtu bodů v množině P lineární.*

Důkaz. Průchod zametací přímky prvním bodem z množiny P vytvoří jednu parabolu. Každá další bodová událost rozděluje jednu část paraboly na tři části a v kruhových událostech paraboly pouze mizí. Odtud dostaneme pro maximální počet částí parabol v bojové linii vztah: $1 + (n-1)(3-1) = 2n - 1$, kde n je počet bodů v množině P. □

Pseudokód algoritmu má tuto podobu:

VoronoiDiagram(P)

1. Inicializuj prázdný dvojitě souvislý seznam S, prázdný binární vyhledávácí strom T a prioritní frontu Q, která obsahuje všechny body z množiny P s prioritou danou lexikografickým uspořádáním podle souřadnice y.
2. **While** Q je neprázdná,
3. **do** vyber událost s největší prioritou z Q.
4. **If** je vybraná událost p_i bod z množiny P,
5. **then** proveď algoritmus $HandleSiteEvent(p_i)$,
6. **else** proveď algoritmus $HandleCircleEvent(p_i)$.
7. Najdi dostatečně velký čtverec, který ohraničuje všechny vrcholy V-diagramu. Najdi průniky nekonečných hran diagramu s tímto čtvercem a uprav dvojitě souvislý seznam.

HandleSiteEvent(p_i)

1. **If** T je prázdný,
2. **then** vlož p_i do T,
3. **else**
4. najdi v T list reprezentující parabolu α, která leží nad bodem p_i.
5. **If** list reprezentující parabolu α obsahuje odkaz na kruhovou událost,
6. **then** vymaž tuto kruhovou událost z fronty událostí Q.
7. Nahraď ve stromu T list reprezentující parabolu α podstromem se třemi listy. Prostřední list bude reprezentovat nově vzniklou parabolu odpovídajici bodu p_i, levý a pravý bude reprezentovat parabolu α odpovídající bodu p_j. V uzlech budou uloženy body průniku těchto dvou parabol $< p_i, p_j >$ a $< p_j, p_i >$. Proveď vyvážení stromu T.
8. Vytvoř záznam pro hrany do dvojitě souvislého seznamu S, které budou vytvářeny průsečíky daných dvou parabol.
9. Zkontroluj, zda přidáním nového bodu p_i nevznikla trojice bodů, vytvářející nový kruhový bod. Jestli ano, přidej ho do fronty událostí Q a k listu reprezentujícímu nově vzniklou parabolu přidej odkaz na tento kruhový bod.

HandleCircleEvent(p_i)

1. Vymaž list, který reprezentuje parabolu zanikající průchodem tímto kruhovým bodem. Aktualizuj strom T a proveď jeho vyvážení.
2. Vymaž všechny kruhové události z fronty Q, které jsou tvořené pomocí zaniklé paraboly.
3. Vytvoř záznam pro vrchol do dvojitě souvislého seznamu S. Nově vzniklým vrcholem je střed kružnice, kterou určuje kruhový bod. Vytvoř záznamy pro hrany v tomto seznamu, které jsou určeny novými průsečíky parabol. Uprav záznamy tří hran, které končí ve vzniklém vrcholu.
4. Zkontroluj, zda zánikem paraboly nevznikla trojice bodů, vytvářející nový kruhový bod. Jestli ano, přidej ho do fronty událostí Q a k příslušnému listu přidej odkaz na tento kruhový bod.

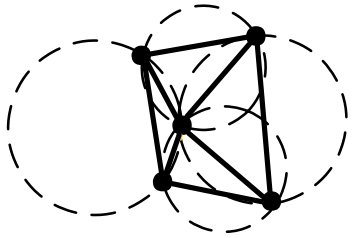
Lemma 2.3. *Algoritmus zametací přímky je optimální. Jeho časová náročnost je* $O(n \log n)$ *a jeho paměťová náročnost je* $O(n)$.

Důkaz. Inicializace fronty a její uspořádání vyžaduje čas $O(n \log n)$. Průchod bodovou událostí (HandleSiteEvent) obsahuje kromě konstantních operací pouze operace probíhající na binárním vyhledávacím stromě. Vyžaduje tedy čas $O(\log n)$. Stejně tak vyžaduje průchod kruhovou událostí (HandleCircleEvent) čas $O(\log n)$. Počet bodových událostí je n. Protože v každé kruhové události vzniká jeden vrchol, je počet kruhových událostí stejný jako počet vrcholů. Lemma 1.1 říká, že počet vrcholů je maximálně $2n-5$. Počet kruhových událostí je tedy také maximálně $2n-5$. Celkem dostaneme $O(n \log n)+n \cdot O(\log n)+(2n-5) \cdot O(\log n) = O(n \log n)$. □

3 Dualita diagramu Voronoia a Delaunayova podrozdělení

Definice 3.1. Delaunayovo podrozdělení definované na množině bodů P, je množina n-úhelníků takových, že vrcholy n-úhelníků jsou body z množiny P a kružnice opsaná každému n-úhelníku neobsahuje uvnitř žádný bod z množiny P.

Triangulací Delaunayova podrozdělení je Delaunayova triangulace. Ta má tu vlastnost, že mezi všemi možnými triangulacemi konvexního obalu dané množiny bodů je to ta triangulace, která maximalizuje minimální úhly trojúhelníků.



Obrázek 8: *Příklad Delaunayovy triangulace.*

Definice 3.2. Jako duální graf F grafu G označujeme takový graf, jehož vrcholy odpovídají stěnám grafu G a hrany vedou mezi každou dvojicí stěn, které sdílejí společnou hranu.

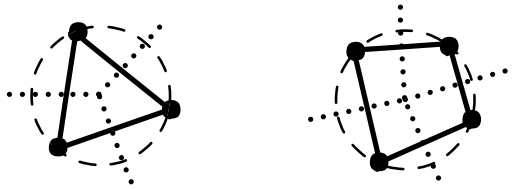
Lemma 3.3. *Diagram Voronoia a Delaunayovo podrozdělení jsou duální grafy.*

Důkaz. V-diagram se skládá z vrcholů, hran a buněk. V každém vrcholu se protínají tři a více hran, které oddělují tři a více buněk V-diagramu. Jak bylo popsáno v kapitole 1 je vrchol V-diagramu tvořen středem kružnice $C_q(P)$, na níž leží alespoň tři body z množiny P. Spojíme-li tyto body úsečkami, dostaneme n-úhelník. Tento n-úhelník náleží Delaunayovu podrozdělení, protože všechny jeho vrcholy jsou body z množiny P a leží na kružnici $C_q(P)$, která splňuje podmínku, že v jejím vnitřku neleží žádný bod z P. Vrchol V-diagramu

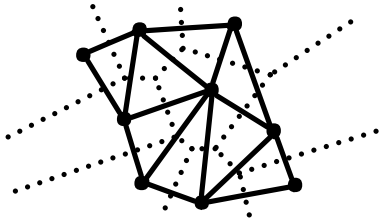
tedy určuje n-úhelník Delaunayova podrozdělení. Naopak pro každý n-úhelník Delaunayova podrozdělení platí, že jeho vrcholy jsou body z množiny P a uvnitř kružnice jemu opsané neleží žádný jiný bod z množiny P. Tato opsaná kružnice je tedy kružnicí $C_q(P)$, jejíž střed je vrcholem V-diagramu. Každý n-úhelník Delaunayova podrozdělení tedy určuje vrchol V-diagramu.

Každá hrana e V-diagramu vychází z nějakého vrcholu v, který je středem nějaké kružnice $C_q(P)$. Tato kružnice určuje n-úhelník Delaunayova podrozdělení. Leží na ní tři a více bodů z P. Hraně e odpovídají právě dva z těchto bodů − e leží na ose spojnice těchto dvou bodů. A protože tyto dva body jsou zároveň vrcholy n-úhelníka, určuje každá hrana V-diagramu hranu Delaunayova podrozdělení, která spojuje tyto dva body a je na ni kolmá. Naopak každá hrana f Delaunayova podrozdělení odpovídá nějakému n-úhelníku tohoto podrozdělení. Tento n-úhelník určuje nějaký vrchol v V-diagramu, který je středem nějaké kružnice $C_q(P)$. Vrcholy hrany f jsou body z množiny P a leží na této kružnici. Protože tyto dva body leží na kružnici $C_q(P)$, určují hranu V-diagramu, která vychází z vrcholu v a leží na ose spojnice těchto dvou bodů.

□



Obrázek 9: *Kružnice $C_q(P)$ (čárkovaně) se třemi a čtyřmi body, Voronoi diagram (tečkovaně) a Delaunayův n-úhelník (plnou čarou).*



Obrázek 10: *Dualita diagramu Voronoia a Delaunayova podrozdělení.*

4 Program

4.1 O programu

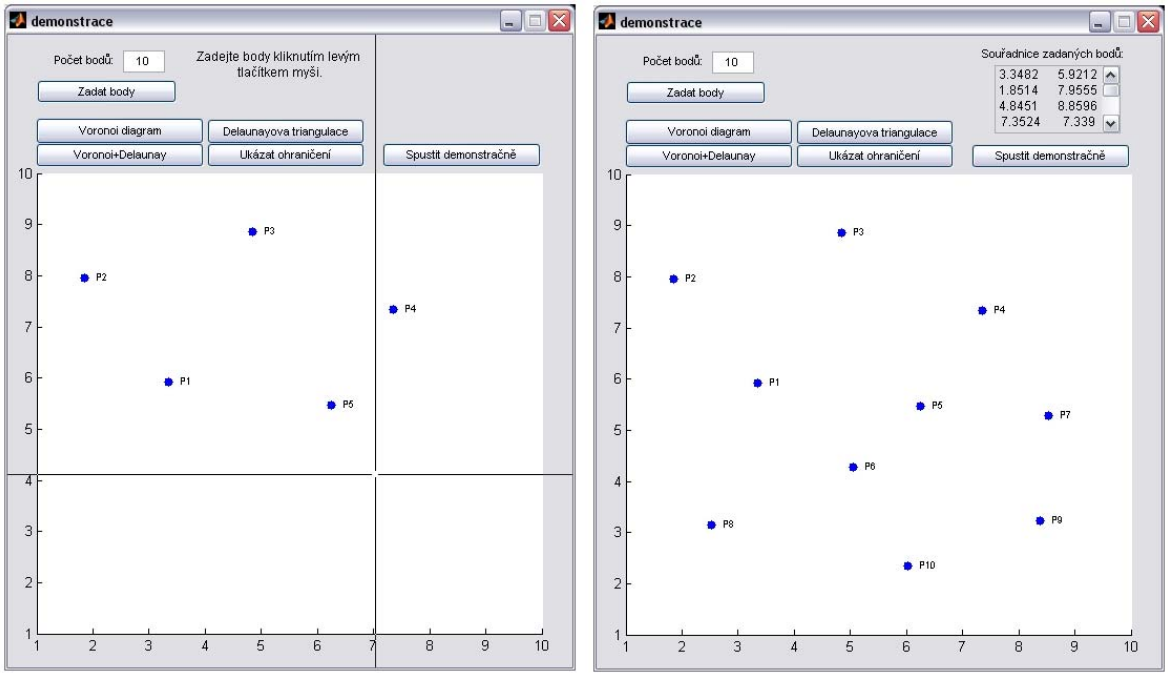
Pro vytvoření programu na prezentaci V-diagramu bylo použito programové prostředí Matlab *(Copyright 1984-2005 The MathWorks, Inc., Version 7.0.4.352 (R14) Service Pack 2, January 29, 2005, License Type: College, License Number: 84911)*, které je dostupné na počítačích Masarykovy univerzity.

Pro demonstraci programu slouží uživatelské prostředí, které bylo také naprogramováno pomocí Matlabu. Celkový program byl překompilován pomocí toolboxu Matlab compilator. Je spustitelný i bez prostředí Matlab, avšak k jeho spuštění je nutné mít nainstalován Matlab Component Runtime verze 7.7.

Pro spuštění z prostředí Matlab (verze 7) stačí nastavit aktuální adresář na adresář na CD nazvaný MSOUBORY a spustit příkaz *demonstrace*.

Prezentace programu pro konstrukci V-diagramu by měla sloužit jako studijní materiál pro studenty předmětů zabývajících se geometrickými algoritmy. Prezentace má několik funkcí.

První funkcí je volba počtu a zadávání bodů. Po spuštění programu je implicitně nastaven počet bodů na deset. Tuto hodnotu je možno měnit, a to v rozmezí od tří do sta. Samotné zadávání probíhá po stisku tlačítka *Zadat body*, a to klikáním myší.



Obrázek 11: *Ukázka programu během a po zadávání bodů.*

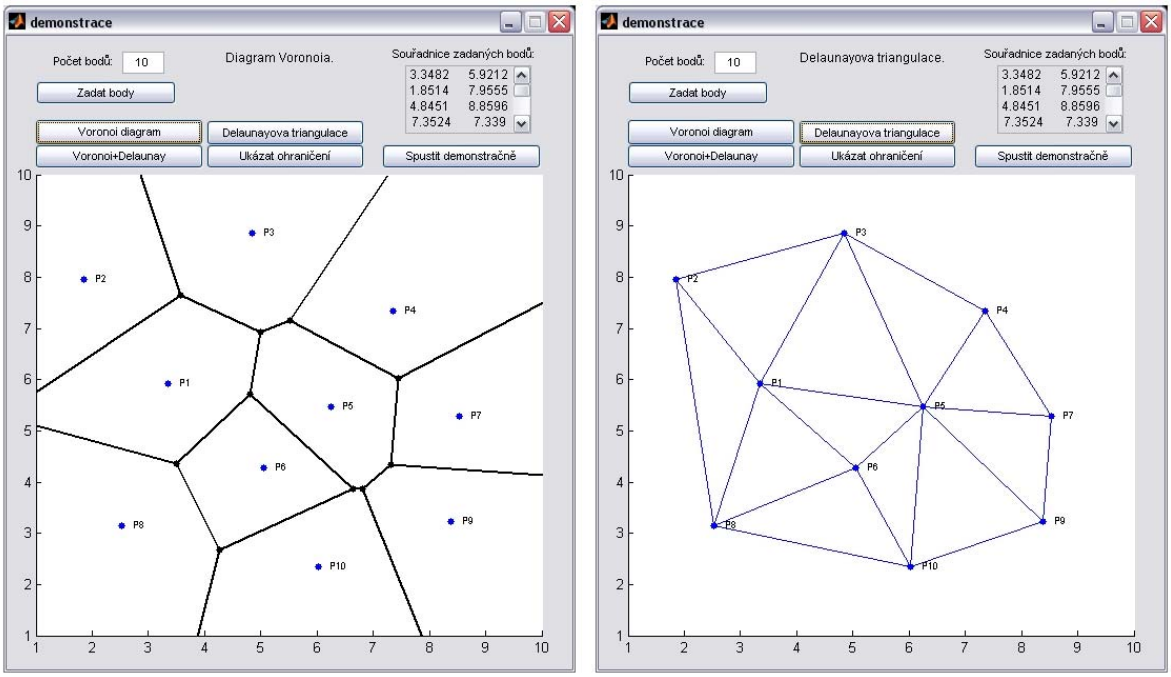
Pokud by chtěl uživatel zadat body ručně, má k dispozici m-soubory programu pro konstrukci V-diagramu. V prostředí Matlab pak k tomu slouží příkaz $VoronoiDiagram(P)$,

kde P je matice souřadnic bodů, zadaná uživatelem.

Popis dalších funkcí:

Voronoi diagram– konstrukce V-diagramu. Po stisknutí tlačítka proběhne algoritmus a výsledek se zobrazí v grafu.

Delaunayova triangulace– vykreslení Delaunayovy triangulace pro danou množinu bodů. Ke konstrukci Delaunayovy triangulace se používá program zabudovaný v Matlabu volaný příkazem $tri = delaunay(P)$ a k jeho vykreslení slouží příkaz $triplot(tri, P)$.

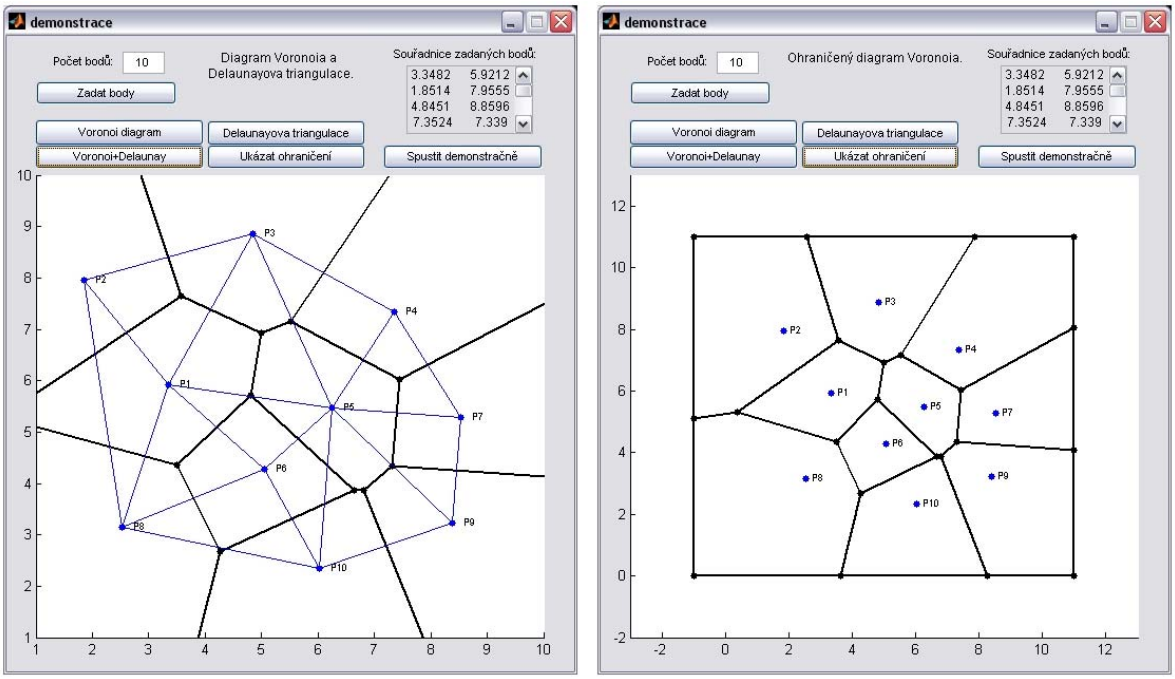


Obrázek 12: *Ukázka vykreslení V-diagramu a Delaunayovy triangulace.*

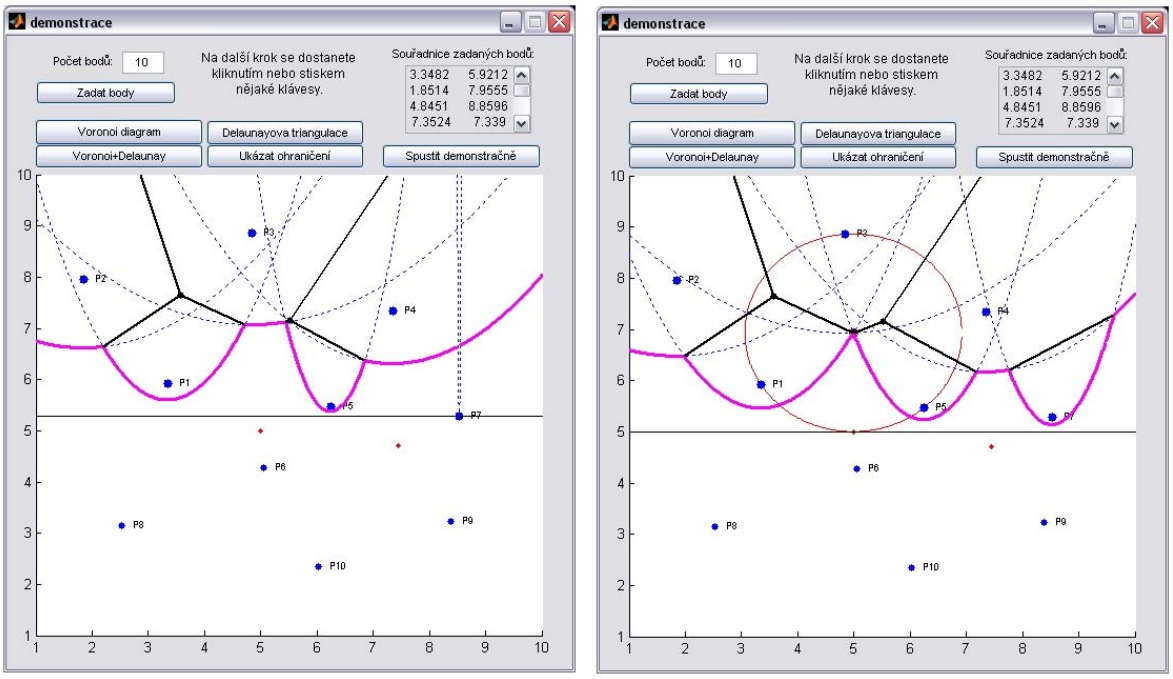
Voronoi+Delaunay– zobrazení V-diagramu a Delaunayovy triangulace v jednom grafu. Zobrazuje dualitu Delaunayova podrozdělení a V-diagramu. Delaunayova triangulace je triangulací Delaunayova podrozdělení. Jestliže žádné čtyři body množiny P neleží na kružnici, je Delaunayova triangulace přímo duální k V-diagramu.

Ukázat ohraničení– zobrazení V-diagramu společně s jeho hranicemi, které byly nalezeny při konstrukci V-diagramu.

Spustit demonstračně– spustí program, který demonstruje krok po kroku konstrukci V-diagramu. Jednotlivé kroky zobrazují bodové a kruhové události, bojovou linii, zametací přímku a část V-diagramu, která již není ovlivnitelná body pod zametací přímkou. Na další krok se postupuje kliknutím nebo stiskem nějaké klávesy.



Obrázek 13: *Ukázka zobrazení duality a hranic V-diagramu.*



Obrázek 14: *Ukázka vykreslení bodové a kruhové události.*

4.2 Zdrojové kódy

function [vrcholy,hrany] = VoronoiDiagram(P)
Funkce VoronoiDiagram vykreslí pro dané body v rovině jejich diagram Voronoia.
Použití:
VoronoiDiagram(P) nebo VoronoiDiagram(n)
VoronoiDiagram(P) ... vstupem je matice P bodů x,y (v prvním sloupci souřadnice x ve druhém sloupci souřadnice y).
VoronoiDiagram(n) ... vstupem je přirozené číslo n, které udává počet bodů. Body se zadávají graficky.
Výstupem je seznam vrcholů a hran V-diagramu a grafický výstup - vykreslený diagram.

$[cax, args, nargs] = axescheck(varargin:)$; ... kontrola počtu vstupních parametrů
$error(nargchk(1, 1, nargs))$; ... error když je nesprávný počet vstupních parametrů
$P = args1$;
$if\ length(P) == 1$ ... když je zadáno pouze jedno číslo
$n = P$;
$P = zadejBody(n)$;
end

inicializace fronty Q:
Q1 ve sloupcích - souřadnice x, souřadnice y,
Q2 - odkaz kruhového bodu na list,
Q3 - název bodu
$P = init(P)$;
clear $[Q1\ Q2\ Q3\ P2]$;
$[P2, Q1, Q2, Q3] = initQ(P)$; ... v Q1 jsou nyní všechny body z množiny P, v Q2 jsou odkazy inicializovány na záporné číslo a v Q3 jsou uloženy názvy bodů. Pro bod z množiny P např. 'P01', pro kruhovou událost např. 'K01' - slouží pro větší přehlednost programu. P2 - pomocná proměnná.
clear T;
$T = Tree()$; ... inicializace stromu T
inicializace seznamu:
$vrcholy = []$; ... vrcholy: (1.souřadnice x, 2.souřadnice y, 3.hrana)
$hrany = []$; ... hrany: (1.první bod, 2.druhý bod, 3.dvojče, 4.vrchol)

$while\ s > 0$ ... dokud je fronta Q neprázdná
$[souradnice\ pozice] = maximum(Q1, Q2)$; ... maximum(Q1) vrací souřadnice události s největší prioritou a její pozici ve frontě
$odkaz = Q2(pozice, :)$; ... odkaz kruhové události na parabolu, není-li to kruhová událost obsahuje záporné číslo
$jmeno = Q3(pozice, :)$; ... jméno události
vymazání události z fronty:
$Q1(pozice, :) = []$;
$Q2(pozice, :) = []$;
$Q3(pozice, :) = []$;
$if\ odkaz < 0$ ... jestliže je vyjmutá událost bodová
$[T, Q1, Q2, Q3, vrcholy, hrany] = HandleSiteEvent(souradnice, jmeno, T,$
$Q1, Q2, Q3, vrcholy, hrany, souradnice(2) - 0.00001, P2, P)$;

$else$ ... je to kruhová událost
$[T, Q1, Q2, Q3, vrcholy, hrany] = HandleCircleEvent(odkaz, T,$
$Q1, Q2, Q3, vrcholy, hrany, souradnice(2) - 0.001, P2, P);$
end
$[s, t] = size(Q1);$
end

$if\ length(vrcholy > 0)$
$[hrany, vrcholy, a, b, c, d] = ohraniceni(hrany, vrcholy, P);$ ... ohraničení diagramu
end
cla
$vykresliBody(P, vrcholy);$ ... vykreslení všech vrcholů a bodů z P
$vykreslihrany(hrany, vrcholy);$ ... vykreslení všech hran V-diagramu

function [T, Q1, Q2, Q3, vrcholy, hrany] = HandleSiteEvent(souradnice, jmeno, T, Q1, Q2, Q3, vrcholy, hrany, l, P2, P)
Funkce volaná při průchodu bodem p_i z množiny P.

$if\ isempty(T)$ ... jestliže je strom prazdný, vlož p_i jako list
$T = [jmeno,' nil'];$
$T(end + 1, :) =' nilnil';$
$T(end + 1, :) =' nilnil';$
$else$ ...když není prázdný
$[kde, o] = najdi(T, souradnice(1), P, P2, l);$ ... najdi ve stromě list reprezentující parabolu, která leží nad bodem p_i, (v proměnné kde bude uložen odkaz na tento list).
Jestliže tento list odkazuje na kruhovou událost, vymaž ji z fronty událostí.
$pom = T(kde + 1, :);$
$odkaz = pom(4 : 6);$
$if\ not(strcmp(odkaz,' nil'))$
$a = find((Q3(:, 1) == odkaz(1))\&(Q3(:, 2) == odkaz(2))\&(Q3(:, 3) == odkaz(3)));$
$Q1(a, :) = [];$
$Q2(a, :) = [];$
$Q3(a, :) = [];$
end
Zaměň list reprezentující parabolu podstromem se třemi listy.
$pi = pom(1 : 3);$
$pk = jmeno;$
$T = zamen(T, pi, pk, kde);$
Vytvoř záznam pro hrany do dvojitě souvislého seznamu, které budou vytvářeny průsečíky daných parabol.
$prvni = str2num(pi(2 : 3));$
$druha = str2num(pk(2 : 3));$
$if\ isempty(hrany)$
$hrany = [prvni, druha, 2, NaN];$
$hrany(end + 1, :) = [druha, prvni, 1, NaN];$
$else$
$[sd] = size(hrany);$
$hrany(end + 1, :) = [prvni, druha, s + 2, NaN];$

$hrany(end+1,:) = [druha, prvni, s+1, NaN];$
end
Zkontroluj, zda přidáním nového bodu p_i nevznikla trojice bodů, vytvářející nový kruhový bod.
$[s,d] = size(T);$
$kruh1 = []; kruh2 = [];$
$listy = toListy(T);$ ... najde odkazy na všechny listy ve stromě
$aa = [];$
$for\ i = 1 : length(listy)$
$if\ strcmp(T(listy(i)+1, 1:3), jmeno)$
$aa = listy(i);$
$poz = i;$
end
end
$s = length(listy);$
$if\ (poz - 2 > 0)\&(not\ strcmp(T(listy(poz-1)+1,:),'nilnil'))\&(not\ strcmp(T(listy(poz-2)+1,:),'nilnil'))$
$kruh1 = najdiKruhovouUdalost(listy(poz-2), listy(poz-1), aa, P, P2, l, T, vrcholy);$
$ss1 = fsouradnice(T(listy(poz-2)+1, 1:3), P, P2);$
$ss2 = fsouradnice(T(listy(poz-1)+1, 1:3), P, P2);$
$ss3 = fsouradnice(T(aa+1, 1:3), P, P2);$
end
$if\ (poz + 2 <= s)\&(not\ strcmp(T(listy(poz+1)+1,:),'nilnil'))\&(not\ strcmp(T(listy(poz+2)+1,:),'nilnil'))$
$kruh2 = najdiKruhovouUdalost(aa, listy(poz+1), listy(poz+2), P, P2, l, T, vrcholy);$
$st1 = fsouradnice(T(aa+1, 1:3), P, P2);$
$st2 = fsouradnice(T(listy(poz+1)+1, 1:3), P, P2);$
$st3 = fsouradnice(T(listy(poz+2)+1, 1:3), P, P2);$
end
$if(not\ isempty(kruh2))\&(not\ isempty(kruh1))\&((kruh1 == kruh2))$
$if\ (breakx(ss2(1), ss2(2), ss3(1), ss3(2), l) - breakx(ss1(1), ss1(2), ss2(1), ss2(2), l)) > (breakx(st2(1), st2(2), st3(1), st3(2), l) - breakx(st1(1), st1(2), st2(1), st2(2), l))$
$kruh1 = [];$
else
$kruh2 = [];$
end
end
Jestliže vznikl nový kruhový bod, přidej ho do fronty událostí Q a k listu reprezentujícímu nově vzniklou parabolu přidej odkaz na tento kruhový bod.
$if\ not\ isempty(kruh1)$
$jm = najdiJmeno(Q3);$
$Q1(end+1,:) = kruh1;$
$Q2(end+1,:) = listy(poz-1);$
$Q3(end+1,:) = jm;$
$T(listy(poz-1)+1, 4:6) = jm;$
end
$if\ not\ isempty(kruh2)$
$jm = najdiJmeno(Q3);$

$Q1(end+1,:) = kruh2;$
$Q2(end+1,:) = listy(poz+1);$
$Q3(end+1,:) = jm;$
$T(listy(poz+1)+1, 4:6) = jm;$
end
end

function [T, Q1, Q2, Q3, vrcholy, hrany] = HandleCircleEvent(odkaz,T, Q1, Q2, Q3, vrcholy, hrany, l, P2, P)
Funkce volaná při průchodu kruhovým bodem.

Vymaž list, který reprezentuje zanikající parabolu a aktualizuj strom T.
if $(odkaz/2 == round(odkaz/2))$
$bratr = odkaz - 1;$
else
$bratr = odkaz + 1;$
end
$n = round(odkaz/2) - 1;$
$i = 1;$
$pom = bratr;$
$otec = T(n+1,:);$
$[S, a1, a2, a3] = najdiVrchol(T, odkaz, P, P2, l);$ … vrací souřadnice vrcholu a odkazy na trojice parabol, které se učastní kruhové události.
$listy = toListy(T);$
$i = find(listy == odkaz);$
$levy = listy(i-1);$
$pravy = listy(i+1);$
$i = find(Q2 == levy);$
$levy = Q3(i,:);$ … v proměnné *levy* je uložen levý soused zanikající paraboly
$i = find(Q2 == pravy);$
$pravy = Q3(i,:);$ … v proměnné *pravy* je uložen pravý soused zanikající paraboly
if not $strcmp(T(bratr+1, 4), 'P')$
$T(odkaz+1,:) = 'nilnil';$
end
$[TQ2] = posun2(T, Q2, n, bratr);$
if $strcmp(otec, [a1(1:3)a2(1:3)])$
$T = prejmenuj(T, [a2(1:3), a3(1:3)], [a1(1:3), a3(1:3)]);$
else
$T = prejmenuj(T, [a1(1:3), a2(1:3)], [a1(1:3), a3(1:3)]);$
end
Vymaž všechny kruhové události z Q, které jsou tvořené pomocí zaniklé paraboly.
if not $isempty(levy)$
$a = find((Q3(:,1) == levy(1))\&(Q3(:,2) == levy(2))\&(Q3(:,3) == levy(3)));$
if not $isempty(a)$
$T(Q2(a)+1, 4:6) = 'nil';$
$Q1(a,:) = [];$
$Q2(a,:) = [];$

```
        Q3(a,:) = [];
    end
end
if not isempty(pravy)
    a = find((Q3(:,1) == pravy(1))&(Q3(:,2) == pravy(2))&(Q3(:,3) == pravy(3)));
    if not isempty(a)
        T(Q2(a)+1,4:6) =' nil';
        Q1(a,:) = [];
        Q2(a,:) = [];
        Q3(a,:) = [];
    end
end
```

Vytvoř záznam pro vrchol a nově vzniklou hranu do dvojitě souvislého seznamu a uprav záznamy pro hrany, které v tomto vrcholu končí.

```
[s,d] = size(hrany);
[sv,dv] = size(vrcholy);
hrany(end+1,:) = [str2num(a1(2:3)), str2num(a3(2:3)), s+2, sv+1];
hrany(end+1,:) = [str2num(a3(2:3)), str2num(a1(2:3)), s+1, NaN];
if isempty(vrcholy)
    vrcholy = [S(1), S(2), s+1];
else
    vrcholy(end+1,:) = [S(1), S(2), s+1];
end
pom = find((hrany(:,1) == str2num(a2(2:3)))&(hrany(:,2) == str2num(a1(2:3))));
hrany(pom,4) = sv+1;
pom = find((hrany(:,1) == str2num(a3(2:3)))&(hrany(:,2) == str2num(a2(2:3))));
hrany(pom,4) = sv+1;
```

Zkontroluj zda zánikem paraboly nevznikla trojice bodů vytvářející nový kruhový bod.

```
[s,d] = size(T);
listy = toListy(T);
kruh1 = []; kruh2 = [];
aa = [];
for i = 1:length(listy)-1
    if (strcmp(T(listy(i)+1,1:3),a1(1:3)))&(strcmp(T(listy(i+1)+1,1:3),a3(1:3)))
        aa = listy(i);
        poz = i;
    end
    if (strcmp(T(listy(i)+1,1:3),a3(1:3)))&(strcmp(T(listy(i+1)+1,1:3),a1(1:3)))
        aa = listy(i);
        poz = i;
    end
end
s = length(listy);
if (poz-1 > 0)&(not strcmp(T(listy(poz-1)+1,:),'nilnil'))&(not strcmp(T(listy(poz+1)+1,:),'nilnil'))
    kruh1 = najdiKruhovouUdalost(listy(poz-1), aa, listy(poz+1), P, P2, l, T, vrcholy);
    ss1 = fsouradnice(T(listy(poz-1)+1,1:3), P, P2);
```

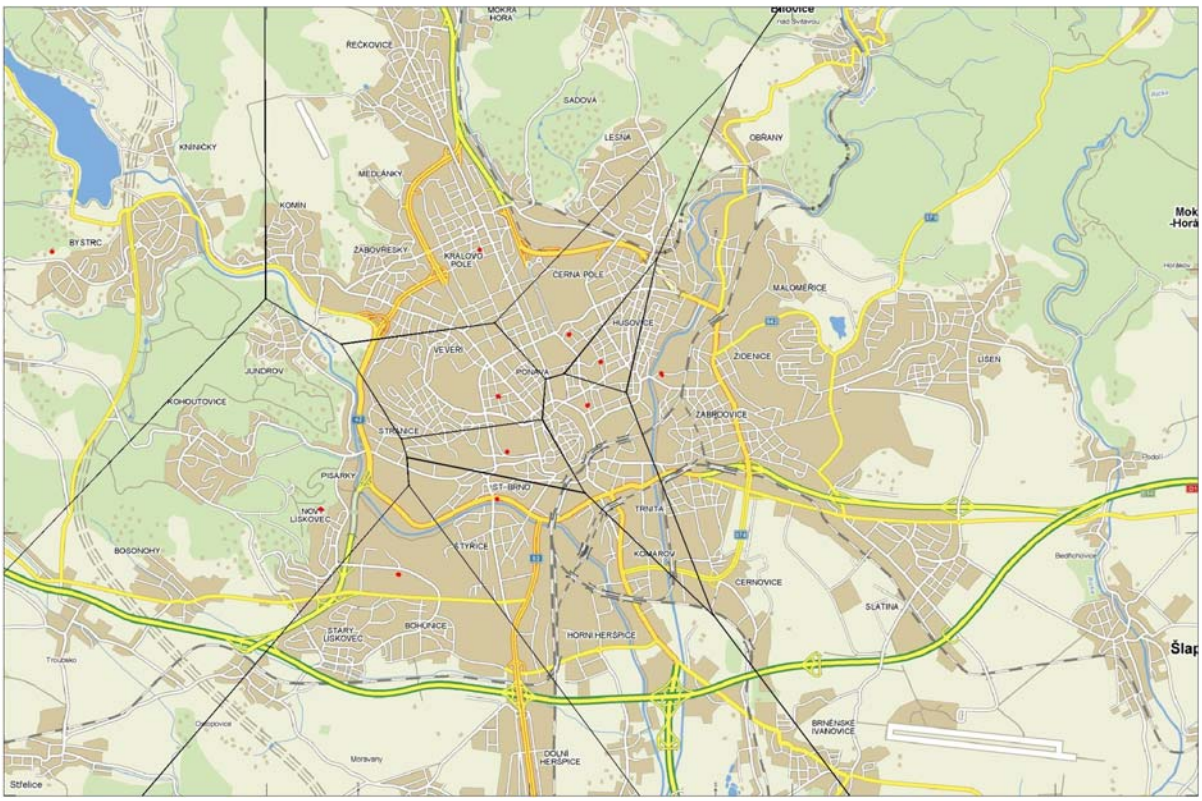
$ss2 = fsouradnice(T(aa+1, 1:3), P, P2);$
$ss3 = fsouradnice(T(listy(poz+1)+1, 1:3), P, P2);$
end
$if\ (poz+2 <= s)\&(not\ strcmp(T(listy(poz+1)+1,:),'nilnil'))\&(not\ strcmp(T(listy(poz+2)+1,:),'nilnil'))$
$kruh2 = najdiKruhovouUdalost(aa, listy(poz+1), listy(poz+2), P, P2, l, T, vrcholy);$
$st1 = fsouradnice(T(aa+1, 1:3), P, P2);$
$st2 = fsouradnice(T(listy(poz+1)+1, 1:3), P, P2);$
$st3 = fsouradnice(T(listy(poz+2)+1, 1:3), P, P2);$
end
$if\ (not\ isempty(kruh2))\&(not\ isempty(kruh1))\&((kruh1 == kruh2))$
$if\ (breakx(ss2(1), ss2(2), ss3(1), ss3(2), l) - breakx(ss1(1), ss1(2), ss2(1), ss2(2), l)) > (breakx(st2(1), st2(2), st3(1), st3(2), l) - breakx(st1(1), st1(2), st2(1), st2(2), l))$
$kruh1 = [];$
else
$kruh2 = [];$
end
end
Jestliže vznikl nový kruhový body, přidej ho do fronty událostí Q a k příslušnému listu přidej odkaz na tento kruhový bod.
$if\ not\ isempty(kruh1)$
$jm = najdiJmeno(Q3);$
$Q1(end+1,:) = kruh1;$
$Q2(end+1,:) = aa;$
$Q3(end+1,:) = jm;$
$T(aa+1, 4:6) = jm;$
end
$if\ not\ isempty(kruh2)$
$jm = najdiJmeno(Q3);$
$Q1(end+1,:) = kruh2;$
$Q2(end+1,:) = listy(poz+1);$
$Q3(end+1,:) = jm;$
$T(listy(poz+1)+1, 4:6) = jm;$
end end

5 Aplikace V-diagramu

5.1 Aplikace na problém spádových oblastí nemocnic

Problém nalezení spádových oblastí nemocnic vede k V-diagramu. Rovinou na které V-diagram hledáme, je mapa oblasti a body z množiny P jsou jednotlivé nemocnice.

Na obrázku 15 je uveden příklad pro nalezení spádových oblastí brněnských nemocnic. Rovinou je zde mapa Brna. Souřadnice bodů reprezentujících jednotlivé nemocnice byly nalezeny pomocí jejich GPS souřadnic (mezi nemocnice byly započítány i polikliniky).



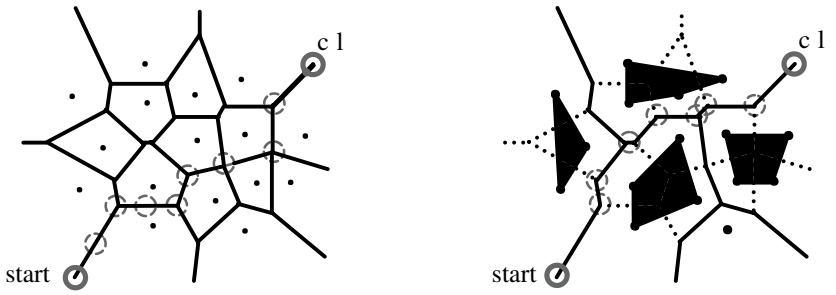
Obrázek 15: *Spádové oblasti nemocnic (červené body) v Brně.*

5.2 Plánování pohybu robota

V-diagram se dá využít při tvorbě umělé inteligence robotů. Příkladem je pohyb robota, kdy chceme dostat robota z místa startu do místa cíle tak, aby se co nejbezpečněji vyhnul překážkám. Viz [4].

Jestliže jsou překážky malé, mohou se nahradit body. Tyto body tvoří množinu P, pro kterou se najde příslušný V-diagram. Robot se pak ze startu do cíle bude pohybovat po hranách tohoto V-diagramu.

V reálných situacích bývají ale překážky složité. V takových případech se za body množiny P vezmou vrcholy překážek. Z nalezeného V-diagramu pro tyto body se poté odstraní hrany, které protínají překážky. Robot se pak bude pohybovat po zbylých hranách tohoto V-diagramu.



Obrázek 16: *Vlevo trasa pohybu robota kolem bodových překážek, vpravo trasa kolem složitějších překážek.*

Literatura

[1] F. Aurenhammer, R. Klein, *Voronoi diagrams*, Elsevier Science B.V., 1994, elektronicky dostupné na <http://www.pi6.fernuni-hagen.de/publ/tr198.pdf>.

[2] M. de Berg, O. Cheong, M. van Kreveld, M. Overmars, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, Third Edition, March 2008.

[3] A. Zivner, *Asymptotická složitost*, článek v encyklopedii wikipedie, 2006, elektronicky dostupné na <http://cs.wikipedia.org/wiki/Asymptotick%C3%A1_slo%C5%BEitost>.

[4] M. Šeda, *Geometric data structures and their selected applications*, World academy of science, 2006, elektronicky dostupné na <www.waset.org/pwaset/v11/v11-12.pdf>.