Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Využití shlukování pro GIS aplikace

Plzeň 2013 Ondřej Kaas

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 6. května 2013

Ondřej Kaas

Poděkování

Chtěl bych věnovat poděkování Prof. Dr. Ing. Ivaně Kolingerové za její cenné rady, podporu, trpělivost a čas, který mi při konzultacích věnovala. Rovněž děkuji Ing. Jiřímu Skálovi, Ph.D. za jeho odbornou pomoc.

Anotace

Tato bakalářská práce se zabývá použitím metody shlukování v reálních GIS aplikacích.

Cílem práce je vytvořit program využívající shlukovou knihovnu pro shlukování na základě různých atributů dat a nalézt vhodné nastavení shlukování dle GIS atributů.

V úvodu práce je popsána problematika shlukování, za ní následuje popis řešení a diskuze experimentů.

Klíčová slova:

Shlukování, GIS aplikace

Abstract

This Bachelor thesis deals with using clustering methods in real GIS applications.

The goal of this thesis is to create a program that uses the clustering library to cluster on the base of various data attributes and to find the right clustering settings from GIS attributes.

In the beginning the problematics of clustering is described followed by the description of the solution and discussion of experiments.

Keywords:

Clustering, GIS applications

Obsah

1 Úvod

V dnešní době přístrojů, které chrlí enormní množství naměřených dat, je nutné modernizovat dosavadní algoritmy a vymýšlet nové, aby bylo možné zpracovávat takto naměřená data i v případě, že se nevejdou celá do paměti. Jednou z řady metod, které v posledních letech prošly inovací, je metoda shlukování.

Metodu shlukování lze najít v celé řadě technických oborů, jako je např. analýza dat, data mining, vyhledávání informací. Principem shlukování je sloučení podobných elementů dohromady. Druh elementu poté záleží na aplikaci a může být v podstatě jakýkoli. Shlukování lze realizovat celou řadou algoritmů a mnohé z nich řeší i problém velkých dat.

Hlavní náplní práce je vyzkoušení shlukování v oblasti reálných GIS aplikací. Reálné GIS aplikace poskytují vektorová a rastrová data. Vektorová data uchovávají informace o jednotlivých objektech zájmového území formou bodů, linií a polygonů. Objekty jsou sdružovány do vrstev podle určité tématické souvislosti (např. vodstvo, lesy, budovy, památné stromy). U rastrových formátů dat je nositelem informace pixel - může reprezentovat jeden celý objekt, jeho část, nebo je v pixelu ukryto více objektů. Tato data se získávají např. z leteckého snímkování terénu. Data, na kterých bylo v práci experimentováno, představují úsečky z procesu zpřesňujících transformací při digitalizaci katastrálních map.

Potřebné shlukovací algoritmy nebylo nutné implementovat od začátku, ale modifikovat je a doplňit vhodným uživatelským rozhraním. K práci byla poskytnuta shlukovací knihovna od Ing. Jiřího Skály, Ph.D. vytvořená na půdě KIV/ZČU.

Úvodu do problematiky shlukování a představení základních použitých metod shlukování se věnuje kapitola 2. Výběr grafických knihoven a návrh řešení výsledné aplikace a je popsán v kapitole 3. Vlastním experimentům pro ověření funkčnosti a výsledkům se věnuje kapitola 4.

2 Shlukování

Principem shlukování je sloučení většího počtu podobných elementů dohromady a jejich reprezentace menším počtem elementů. Podoba elementu poté záleží na aplikaci a může být v podstatě jakákoli - od bodů v 1D prostoru[1] po 3D objekty, celé digitální obrázky, dokumenty nebo databázeové entity. Společným znakem elementů je jejich možné vyjádření specifickým vektorem. Například, body jsou popsány pomocí jejich prostorových souřadnic. Pro některé složitější abstrakce elementů je nutné najít odpovídající vyjádření pro jejich vlastnosti. Shlukování je NP-těžký problém, a proto výsledky algoritmů jsou pouze aproximací správného řešení.

Výsledkem shlukování budou množiny elementů s největším podobností jejich vektorů v rámci jedné množiny. Každou množinu poté reprezentuje jeden element. Rozhodnutí, jak moc jsou dva elementy podobné, a tedy jestli patří do stejného shluku, se provede pomocí tzv. metriky, popsané v kapitole 2.1.

V některých případech není žádoucí, aby výsledkem byly shluky elementů podobných v rámci všech vlastnostní. Některé souřadnice mohou být důležitější nežli jiné. Například v oblasti digitálního obrazu jsou body (*pixely*) reprezentovány nejen svými polohovými souřadnicemi, ale i souřadnicemi v barevném prostoru RGB. Pokud bude cílem shlukovat pouze na základě podobnosti barevných souřadnic bodů, klasická metrika zde neposkytne kýžené řešení. Je nutné do metriky zanést vztah, který bude zvýhodňovat požadované souřadnice. Toho je docíleno pomocí vah, kterým je věnována kapitola 2.2.

2.1 Metrický prostor

Metrický prostor [jt13] je dvojice $P = (M, p)$, kde M je libovolná neprázdná množina a p je tzv. metrika, což je zobrazení $p : M \times M \Rightarrow R$, které splňuje následující axiomy (pro libovolná $x, y, z \in M$) :

1. Axiom nezápornosti: $p(x, y) \geq 0$
2. Axiom totožnosti: $p(x, y) = 0 \Leftrightarrow x = y$
3. Axiom symetrie: $p(x, y) = p(y, x)$
4. Trojúhelníková nerovnost: $p(x, z) \leq p(x, y) + p(y, z)$

[1] Shlukování hloubkové informace bodu pro renderování.

Vzdálenost mezi dvěma prvky je pojem relativní a může se měřit různě v závislosti na daném prostoru a konkrétní představě. Ona vzdálenost ale bude klíčová ve výpočtu podobnosti dvou elementů.

Každé množině M lze zadat celou řadu různých metrik. Tím se vytvoří různé metrické prostory, které budou mít stejnou základní množinu, tzv. nosič, ale v každém z nich bude jiným způsobem měřena vzdálenost.

Každá z metrik definuje D dimenzionální body jako $x = (x_1, x_2, x_3, ...x_D)$ kde $x_1, x_2, ...x_D$ představují souřadnice daného bodu. Vzdálenost dvou bodů x a y je zapsána jako $d(x, y)$.

Na množině $\mathbb{R}^2$ lze definovat mj. následující metriky:

$x, y \in \mathbb{R}^2, x = (x_1, x_2), x = (y_1, y_2)$:		
$d(x, y) = \sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2}$	...	eukleidovská vzdálenost
$d_m(X, Y) = \lvert y_1 - x_1 \rvert + \lvert y_2 - x_2 \rvert$	...	manhattanská metrika
$d_{max}(X, Y) = max\{\lvert(y_1 - x_1)\rvert, \lvert(y_2 - x_2)\rvert\}$	...	maximová metrika

Tyto metriky lze přirozeně rozšířit na množinu $\mathbb{R}^3$:

$x, y \in \mathbb{R}^3, x = (x_1, x_2, x_3), y = (y_1, y_2, y_3)$:		
$d(x, y) = \sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2 + (y_3 - x_3)^2}$	...	eukleidovská vzdálenost
$d_m(x, y) = \lvert y_1 - x_1 \rvert + \lvert y_2 - x_2 \rvert + \lvert y_3 - x_3 \rvert$	...	oktaedrická metrika
$d_{max}(x, y) = max\{\lvert(y_1 - x_1)\rvert, \lvert(y_2 - x_2)\rvert, \lvert(y_3 - x_3)\rvert\}$	...	maximová metrika

K výpočtu vzdálenosti se nejvíce používá vzorce eukleidovi vzdálenosti, který je jednoduchý a dobře funguje ve většině scénářů.

2.2 Váhy

Pomocí vah lze zanést do vztahu pro výpočet podobnosti elementů důležitost některých souřadnic. Toho je docíleno obohacením vzorce metriky o koeficient váhy příslušné souřadnice. Výpočet vzdálenosti d dvou bodů $x = (x_1, x_2, ...x_D)$ a $y = (y_1, y_2, ...y_D)$ pomocí metriky M, ovlivněný váhami $w = (w_1, w_2, ...w_D)$ je zapsán následujícím vzorcem 2.1.

$$d(x,y) = \sqrt{\sum_{i \in 1...D} (M_i * w_i)^2} \tag{2.1}$$

kde :

D	...	dimenze daného metrického prostoru
M_i	...	rozdíl souřadnic bodů x a y v ose i, tj. $M_i = x_i - y_i$
w_i	...	váha souřadnice i

Souřadnice mezi sebou mohou mít rozdílné intervaly. Pro korektní vliv vah u příslušných souřadnic je nutné, aby všechny souřadnice byly ze stejného intervalu. Toho je docíleno přepočtem hodnoty dané souřadnice na hodnotu souřadnice s největším intervalem ze všech souřadnic. Souřadnice s největším intervalem je vybrána z důvodu zachování přenosti.

Převod souřadnice A do intervalu souřadnice B s největším intervalem se provede následujícím postupem :

1. Vypočti převodní koeficient k dle vzorce:

$$k = \frac{A_{max} - A_{min}}{B_{max} - B_{min}} \tag{2.2}$$

kde:

A_{max}, A_{min}	...	maximální a minimální hodnota souřadnice A
B_{max}, B_{min}	...	maximální a minimální hodnota souřadnice B

2. Vynásob všechny hodnoty z intervalu souřadnic A koeficientem k.

2.3 Shlukovací metody

Existuje mnoho shlukovacích algoritmů, používaných napříč technickými obory. Shlukovací algoritmy mohou být rozděleny podle jejich konkrétních funkcí a principů do dvou protichůdných cest vedoucích k řešení. V následujících odstavcích budou některé z nich představeny [sj13, str. 32,33].

Princip shlukování může být buď hiearchický (*hiearchical*) nebo nehierarchický (*partitional*[2]). Nehierarchický algoritmus rozdělí data mezi přesný počet shluků

[2] v české literatuře se lze setkat s pojmem nehierarchické metody shlukování

(segmentů). Hiearchický algoritmus vytvoří hiearchii malých shluků sloučených do shluků větších tvořících stromovou strukturu zvanou dendrogram. Stupněm shlukování pak lze kontrolovat počet vytvořených úrovní samotné hierarchie.

Jiné možné protichůdné rozdělení algoritmů je aglomerativní (*algomerative*) nebo divizní (*partitional*). Aglomerativní shlukování začíná ve stavu, kdy jsou všechny vstupní elementy považovány za centra shluků a jsou postupně spojována na základě jejich podobnosti do té doby, dokud není splněna zastavovací podmínka. Algoritmus je obvykle zastaven ve chvíli, kdy je vytvořeno požadované množství shluků nebo pokud podobnosti elementů klesnou pod krajní mez, kdy již elementy nemají být přiřazeny k sobě. Divizní algoritmus jde k řešení opačnou cestou. Algoritmus začíná s všemi elementy přiřazenými do jednoho velkého shluku. Ten je poté opakovaně rozdělován na základě nepodobnostní podmínky. Algoritmus opět skončí ve chvíli, kdy je vytvořeno požadované množství shluků nebo shluky jsou tak homogenní, že není potřeba dalšího dělení.

Shlukování může být *hard* nebo *fuzzy*. Hard shlukování přiřazuje každý element právě do jednoho shluku, oproti tomu fuzzy shlukování určuje počet přiřazení jednoho elementu do více shluků.

Shlukovací algoritmy mohou být deterministické (*deterministic*) nebo stochastické (*stochastic*). Mezi stochastické techniky obvykle patří náhodné algoritmy. Ty jsou většinou používány na velká data pro svoji rychlost.

Shlukovací techniky mohou zpracovávat celá data najednou nebo pracovat postupně (inkrementálně). Pokud bude algoritmus zpracovávat celá vstupní data najednou, lze očekávat přesnější výsledky. Inkrementální algoritmus může být rychlejší a díky menší náročnosti na paměť ho lze použít i na velká data. Málá náročnost na paměť plyne z faktu, že si algoritmus neuchovává všechny informace o vstupních datech, pouze nejdůležitější informace o konkrétních shlucích nutných pro další pokračování.

Z možných shlukovacích algoritmů byl vybrán *facility location*, který bude popsán v následující kapitole 2.3.1. Tento algoritmus byl implementován v poskytnuté knihovně [sk09] Ing. Jiřího Skály Ph.D.

2.3.1 Facility location

Algoritmus byl navržen pro velký objem dat, který se zpravidla celý nevejde do paměti počítače a proto se musí načítat po menších blocích (tzv. datastreamové shlukování). Následující odstavec popisuje princip algoritmu (převzato [sj13, str. 35,36]).

V popisu algoritmu jsou používány následující formulace. Nechť písmenem **F** jsou označeny centra shluků (*facilities*) a písmenem **C** označeny přiřazené body ke shluku (*clients*) a všechny body se mohou stát centra shluků. Problém je v rozhodnutí, který z bodů se má stát centrem shluku a které body mají být k němu přiřazeny. Algoritmus rozhoduje na základě ohodnocení, např. za „otevření shluku“ (prohlášení bodu za centrum shluku) je nutné „zaplatit“ cenu **fc** (*facility cost*). Další cenou je spojovací cena (*service cost*), většinou závislá na vzájemné vzdálenosti obou elementů. Analogii problému nalezneme v aplikaci z reálného života. Představme si město, kterému musíme dodávat elektřinu. Máme k dispozici několik míst, kde je možné postavit rozvodnu elektřiny. Postavení rozvoden na všechna možná místa je moc drahé, stejně tak jako připojení všech domácností k jedné centrální rozvodně. Je nutné zjistit, na kterých místech postavit rozvodny a která místa budou pouze připojena. Jinými slovy najít optimální řešení, které povede k minimalizaci nákladů na stavby.

Algoritmus se poté snaží minimalizovat celkovou cenu **Q** definovanou jako

$$Q = \sum_{j \in F} fc + \sum_{i \in C} c_{ij} \tag{2.3}$$

kde:

fc	...	cena za otevření nového centra
F	...	množina center shluků
C	...	množina přiřazených bodů
c_{ij}	...	cena za spojení přiřazeného bodu i k jeho centru shluku j

Vzdálenost je obecně považována za kladnou, symetrickou a splňující trojůhelníkovou nerovnost. Důležité je podotknout, že neexistují žádná omezení mezi množinou center shluků a množinou přiřazených bodů. Množina F může být nezávislá na množině C, podmnožinou C, nebo dokonce shodná s C.

Na začátku shlukování není specifikováno, kolik se má vytvořit center shluků, jako tomu může být u jiných algoritmů. Jediným možným ovlivněním je koeficient ceny za otevření shluku. Vysoká hodnota ovlivní výpočet ceny ve prospěch velkých shluků. Otevření nového centra shluku se stane velmi drahé, a tak se body raději přiřadí k centru, než aby se vytvořil nový shluk. Oproti tomu malá cena vytvoří velké množství shluků. Otevření shluku je levné a proto se mnoho bodů stane centrem shluku.

Metod hledajících minimální cenu ohodnocení existuje velké množství, např. *linear programming rounding*, *primal-dual algorithm* nebo níže popsaný *local search* [sj13, str. 36-38].

Local search

Metoda local search vytváří graf možných řešení. Jednotlivé uzly v grafu představují dané ohodnocené řešení. Uzly jsou poté spojeny hranami, pokud jedno z řešení lze získat z druhého určitým typem modifikace. Algoritmus poté prochází daný graf přes uzly s nižší cenou a hledá lokální minimum. Jinými slovy, takové řešení, které má nejmenší ohodnocení než jeho sousedi. První řešení je náhodně vygenerované, to je dále iterativně vylepšováno lokálními úpravami. Za centrum shluku je zvolen náhodný bod a poté je zjištěno, zdali tímto otevřením bude vylepšeno dosavadní řešení. Pokud jsou v blízkosti nového centra nějaké body, budou k novému centru přiřazeny. Pokud tímto přeřazením vzniknou shluky s malým počtem bodů, budou jeho body také přeřazeny k novému shluku. Algoritmus poté zavádí funkci vylepšení (*gain*), na základě které lze rozhodnout, zda-li bylo nalezeno lepší řešení.

Nyní si popíšeme algoritmus podrobněji s použitím formulací z předešlých odstavců.

Náhodně se vybere bod a prohlásí se za centrum shluku ($j \in F$), nezáleží zdali jím již byl nebo ne, a zjistí se možné vylepšení. Pokud jím nebyl, je nutné zaplatit cenu za jeho otevření. Pokud jsou v blízkosti nějaké body, které mají vzdálenost k jejich dosavadnímu centru shluku větší než k nově vytvořenému j, přeřadí se k j (tím se zmenšuje spojovací cena c). Takto mohou vzniknout centra shluků s malým počtem přiřazených bodů. Tato centra lze uzavřít a ušetřit tak cenu za jejich otevření (fc). Jejich přiřazené body se přiřadí k nově vytvořenému centru j a zaplatí se nová cena za spojení, která může být menší než cena nového spojení. Po těchto upravách je nutné vypočítat funkci *vylepšení*. Pokud bude *vylepšení(j)* > 0, bod j bude prohlášen za centrum shluku (pokud jím již není) a budou provedeny dané úpravy.

Pro definici funkce *vylepšení* je nutné zavést vzdálenostní rozdíl ds_i (*distance spare*), jako rozdíl vzdáleností bodu k jeho dosavadnímu centru shluku a k novému kandidátovi f. Pokud je rozdíl záporný, tedy dosavadní centrum leží blíže než f, nastaví se $ds_i = 0$. Dále je nutné definovat ušetření ceny za uzavření cs_j (*close spare*) jako cenu, kterou lze ušetřit zavřením daného shluku f_j. Tato cena se rovná ceně za vytvoření shluku mínus cena za všechny přeřazené body z f_i do f. Pokud je cena cs_j záporná, tedy nelze nic ušetřit, nastaví se $cs_j = 0$.

Funkci *vylepšení* lze tedy definovat následujícím vzorcem 2.4 :

$$vylepšeni = -fc + \sum_{c_i \in C} ds_i + \sum_{f_j \in F} cs_j \tag{2.4}$$

kde:

fc	...	cena za otevření nového centra, pokud jím již není
ds_i	...	rozdíl spojovacích cen bodu k jeho dosavadnímu a k novému shluku
cs_j	...	ušetřená cena za uzavření shluku a přeřazení všech jeho bodů k novému shluku
c_i	...	všechny přeřazené body
f_j	...	všechny uzavřené shluky

Celý algoritmus se poté provádí právě $N \log N$ krát, kde N je počet potencionálních center shluků, kvůli snížení složitosti a lze ho popsat následujícím pseudokódem:

```
vygeneruj prvotní řešení;
while opakováno < N log N do
    náhodně zvol bod j a prohlaš za centrum shluku;
    if vylepšení(j) > 0 then
        proveď příslušné změny (přeřazení bodů, uzavření center shluků);
    end
end
```

2.3.2 Datastreamové shlukování

Existují tři základní přístupy datastreamového shlukování [sj13, str. 41].

Prvním intuitivním přístupem je rozděl a panuj. Vstupní data jsou rozdělena na několik bloků a každý z nich se poté shlukuje samostatně. Ze zvolených výsledků daných bloků se poté „složí“ výsledné řešení za celá vstupní data. Pokud bude na vstupu velké množství dat, hieararchie může růst do více úrovní.

Dalším možným přístupem je inkrementální shlukování. Shluk je vytvořen s prvním vstupním elementem. Následující elementy jsou buďto přiřazeny k již stávajícím shlukům, nebo jsou prohlášeny za nové centra shluků. Vytvoření nového shluku nebo přiřazení k některému stávajícímu shluku se děje na základě podobnostního ohodnocení. Hlavní výhodou inkrementálního algoritmu jsou jeho malé nároky na pamět, protože algoritmus nemusí ukládat všechna vstupní data do paměti. Další jeho výho-

dou je neiterativní přístup. O vstupním elementu je pouze jednou rozhodnuto a již vícekrát se k němu algoritmus nevrací. Hlavní nevýhodou algoritmu jsou data v náhodném pořadí. Jinými slovy, pokud budou tyto podobné elementy v datastreamu umístěny daleko od sebe, pak je bude nucen algoritmus přiřadit nevýhodně k různým dosavadním shlukům nebo vytvořit nové shluky. Tuto chybu již algoritmus nedokáže opravit.

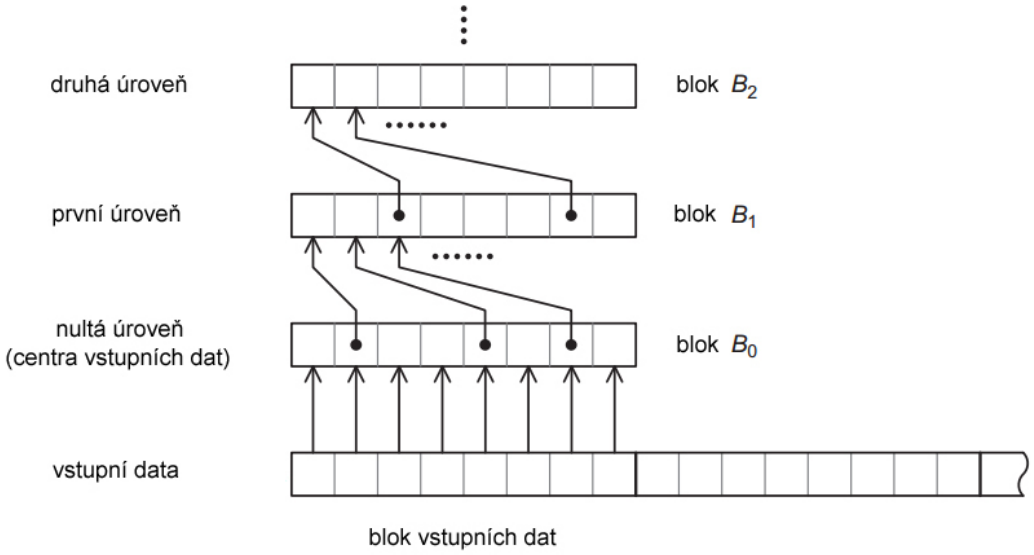
Posledním přístupem, který se v posledních letech stává velmi populární i v jiných odvětvích, je paralelní, distribuované řešení. Algoritmy jsou upraveny tak, aby je bylo možné rozložit na jednotlivé nezávislé etapy. Tyto etapy jsou poté zpracovávány paralelně na více počítačích.

V následující sekci bude popsán hiearchický přístup využívající myšlenku rozděl a panuj [sj13, str. 41-43].

Hiearchické shlukování

Hiearchické datastreamové shlukování rozděluje vstupní data na menší bloky, které poté zpracovává. Po shlukování takového bloku se ohodnocení výsledných center vynásobí počtem přiřazených bodů. Výsledná centra s ohodnocením se poté uloží na externí uložiště (např. na pevný disk). Takto uložená data jsou považována za další datastream a mohou být dále zpracovávána jako původní data. V následujících úrovních bude poté zohledněno jejich dosavadní ohodnocení, kterým bude násobena vzdálenost k jejich novým centrům shlukům. Nové ohodnocení center shluků je poté sumou všech přiřazených bodů (původně také center shluků nižší úrovně).

Algoritmus nejlépe popíše následující Obr. 2.1 [cg13]. Načteme úsek dat do úrovně B_0 a shlukujeme. Výsledné shluky uložíme do vyšší úrovně B_1. Zaplňování vyšší úrovně pokračuje do chvíle, než je danná úroveň plná a je nutné je opět shlukovat. Výsledné shluky opět uložíme do vyšší úrovně.



Obr. 2.1: Hiearchická struktura ukládání shluků

Algoritmus lze popsat následujícím pseudokódem:

while *jsou vstupní data* **do**
  načti úsek vstupních dat do nejnižšího bloku B_i;
  zpracuj (B_i);
end

Funkce zpracuj je poté definována:

shlukuj B_i;
vypočti nové ohodnocení centrům shluků;
přesuň centra shluků do bloku B_{i+1};
if *blok* B_{i+1} *je plný* **then**
  shlukuj B_{i+1};
end

Algoritmus udržuje v každé úrovni maximální počet center shluků. Vyšší úroveň shlukuje až ve chvíli, kdy je potřeba uložit další centra a tím uvolnit místo v paměti.

3 Návrh řešení

Začátek práce spočíval ve vytvoření softwarového zázemí pro práci s knihovnou. Na tuto část práce nebyly kladeny žádné specifické požadavky a bylo čistě na uvážení autora, který směr povede nejsnáze k cíli. Bylo nutné vybrat takový programovací jazyk, který dokáže pracovat se shlukovací knihovnou, bude dostatečně rychlý pro maximální efektivnost a lze v něm použít některou z dostupných grafických knihoven pro vizualizaci.

Po výběru programovacího jazyka a grafické knihovny přišla na řadu samotná implementace, která spočívala ve vytvoření uživatelského rozhraní pro práci s knihovnou. Následně byla ověřena věškerá funkcionalita knihovny a s i tím související implementace aplikace. K tomuto účelu byla vytvořena série uměle vytvořených bodových dat. Ověření funkčnosti na umělých datech se věnuje úvod kapitoly Experimenty a výsledky (4).

Po této etapě vývoje započaly experimenty na reálných geomatických datech. Data představují usečky z procesu zpřesňujících transformací při digitalizaci katastrálních map. Jinými slovy, data, pro která nebyla knihovna původně implementována. Výsledné shluky měly vykazovat podobnost v souřadnicích počátečního bodu, úhlech a délkách. Dalším požadavkem na tyto souřadnice byla možnost omezení maximální hodnoty dané úsečky přiřazené do shluku.

Spíše než rozmístění ovládání a vizualizce aplikace byly pro geomatiky důležité výsledky, které shlukování poskytne. Proto bylo grafické uživatelské rozhraní voleno autorem této práce.

Po prvních experimentech s geomatickými daty se ukázalo nutné zasáhnout do shlukovací knihovny, a to především do používaných metrik. Dosavadní metriky nedokázaly postihnout požadavek omezení maximální hodnoty souřadnice přiřazeného bodu. Pro tento případ byla vymyšlela nová metrika s kterou bylo dále experimentováno.

Volbě programovacího jazyka a grafické knihovny se věnuje kapitola 3.1. Uživatelské rozhraní je popsáno v kapitole 3.2. Popisu úsečkové metriky pro geomatická data se věnuje kapitola 3.3.

3.1 Volba grafické knihovny

Na začátku byly specifikovány požadavky na daný jazyk, resp. grafickou knihovnu. Kromě důležitého importu shlukovací knihovny implementované v `C#` pro .NET Framework 2.0 si aplikace vystačí se základním balíčkem uživatelského rozhraní poskytovaného snad u každého moderního objektového jazyka. Spíše než uživatelské rozhraní bylo nutné vyřešit samotnou vizualizaci dat, kde se předpokládalo vykreslování tisíců bodů.

Nejsilnějšími kandidáty na programovací jazyk aplikace se staly `C#` a `C++`. `C#` díky snadnému importu shlukovací knihovny, `C++` díky grafické knihovně Qt Digitia dostupne na [qt13].

Qt Digitia totiž poskytuje velkou základnu pro vykreslování grafiky s pomocí grafické karty počítače. Při další práci se ale ukázalo nevhodné překládat shlukovací knihovnu v jiném vývojovém prostředí než samotnou aplikaci, to by v budoucnu pouze zdržovalo. Výběr grafických knihoven se tedy omezil pouze na ty podporované v jazyce `C#`.

`C#` nabízí dvě základní knihovny uživatelského rozhraní, které jsou součástí .NET Frameworku. Jsou to Windows Forms [wf13] a WPF [wpf13]. Obě poskytují dostatečně velkou základnu všech ovládacích prvků, které aplikace potřebuje. Proto volba byla pouze na základě rychlejšího proniknutí do funkcionality knihoven, to se nakonec povedlo lépe se starším Windows Forms.

Windows Forms mají pouze jednu nevýhodu, a to, že všechny jejich procesy zatěžují pouze procesor počítače. Z tohoto důvodu byla odezva aplikace při vizualizaci dat velmi pomalá. Proto aplikace pro samotné vykreslování používá `C#` wrapper OpenGL for .NET vyvinutý Ing. Petrem Vaněčkem, Ph.D. [pv12].

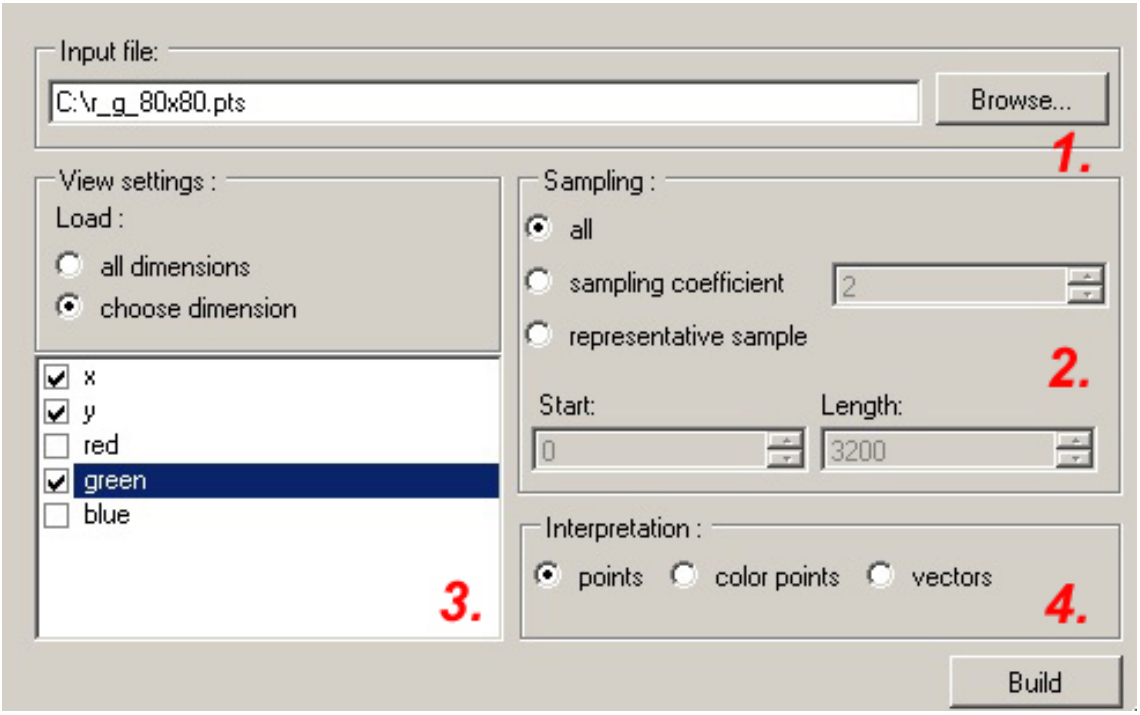
OpenGL for .NET poskytuje veškerou funcionalitu OpenGL 4.3, a tedy i akceleraci za pomoci grafické karty. OpenGL neposkytuje takový komfort jako Qt Digitia, např. přiblížení určité části plátna. Proto bylo nutné tuto základní funkcionalitu implementovat.

3.2 Uživatelské rozhraní

Jelikož důležitost této části práce se ukázala menší, než se původně přepokládalo, budou popsány pouze nejdůležitější prvky ovládání.

3.2.1 Načítání dat

Již při tvorbě formuláře pro výběr vstupních dat (viz Obr. 3.1) bylo bráno za cíl zmenšení objemu dat, která se budou ukládat do paměti počítače. Proto nechybí možnost vzorkování dat nebo výběru souřadnic. Dále je nutné zvolit vizualizaci dat.



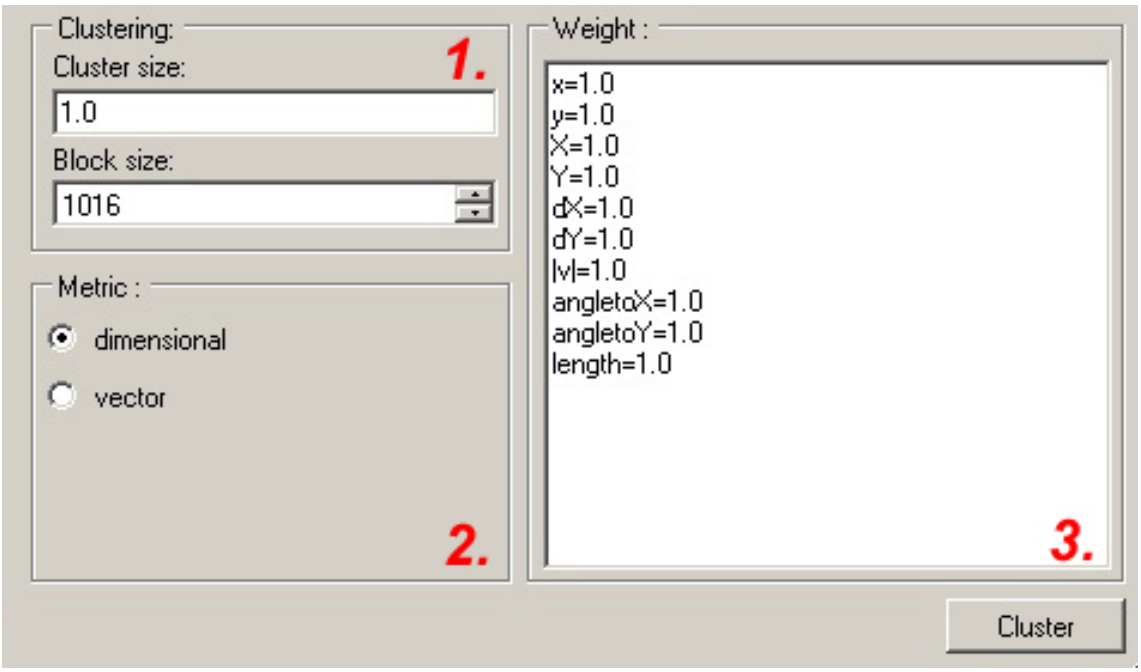
Obr. 3.1: Nabídka pro výběr vstupních dat

Popis vstupního formuláře:

1. zadání vstupního souboru
2. vzorkování
   - all - načtena budou všechna data
   - sampling coefficient - načten bude vždy každý k-tý prvek
   - representative sample - načten bude zvolený interval `<Start, Start+Length>`
3. zvolení interpretace dat

3.2.2 Nastavení shlukování

Pro nastavení vah a zvolení metrik byl vytvořen následující formulář (viz Obr. 3.2). Předpokládala se vysoká frekvence změn v nastavení shlukování, proto je možné tento formulář kdykoli vyvolat, pozměnit jeho nastavení a poté shlukovat již načtená data. Odpadá tak nutnost aplikaci spouštět opakovaně.



Obr. 3.2: Nabídka nastavení vah

Popis nastavení vah:

1. Clustering - volby pro nastavení shluků a velikosti bloku dat
   - Cluster size - nastavení velikosti/počtu shluků. Velká hodnota má za následek několik velkých shluků, malá hodnota poté velké množství malých shluků. Více v podkapitole (4.1.2) v sekci Experimenty a výsledky.
   - Block size - v případě hiearchického shlukování tato hodnota udává velikost bloků dat.
2. Metric - volba metrik pro výpočet podobnosti elementů. Při volbě *dimensional* se použije eukleidovská metrika, při volbě *vector* úsečková (viz kapitola 3.3).
3. Weight - hodnoty příslušných vah souřadnic. Váhy lze zadat v celém intervalu reálných čísel. Nezvýhodněná souřadnice má hodnotu 1.0.

3.3 Úsečková metrika

Geodetický souřadnicový systém je orientován tak, že kladná osa x jde k jihu a kladná osa y směřuje na západ. Geodetická data popisují úsečky. Úsečku l lze popsat 4-dimenzionálním vektorem $t = (x_t, y_t, X_t, Y_t)$, kde x_t, y_t představují počáteční souřadnice úsečky; X_t, Y_t koncové souřadnice úsečky. Požadavek geomatiků na výsledné shluky úseček byla podobnost v následujích souřadnicích:

- počáteční body useček
- směrové vektory useček
- délka úsečky

Během načítání dat jsou proto předpočítány velikosti úhlu a a délky úsečky l dle vzorců (3.1) a (3.2). Úhel je počítán v rozmezí od 0 do 2π od kladné matematické osy x pro jednoznačné určení orientace úsečky.

$$l = \sqrt{(X_t - x_t)^2 + (Y_t - y_t)^2} \tag{3.1}$$

$$a = \begin{cases} \sin\frac{y_t - Y_t}{l}, l \neq 0, & \text{pro } x_t < X_t \wedge y_t > Y_t \\ \frac{\pi}{2} + \sin\frac{y_t - Y_t}{l}, l \neq 0, & \text{pro } x_t > X_t \wedge y_t > Y_t \\ \pi + \sin\frac{y_t - Y_t}{l}, l \neq 0, & \text{pro } x_t > X_t \wedge y_t < Y_t \\ \frac{3\pi}{2} + \sin\frac{y_t - Y_t}{l}, l \neq 0, & \text{pro } x_t < X_t \wedge y_t < Y_t \end{cases} \tag{3.2}$$

Vzniká tak nový popis úsečky:

$$t = (x_t, y_t, X_t, Y_t, a_t, l_t)$$

kde:

x_t, y_t	...	souřadnice počátečního bodu
X_t, Y_t	...	souřadnice koncového bodu
a_t	...	úhel, který svírá směrový vektor úsečky s kladnou poloosou
l_t	...	délka úsečky

Kromě standardní podobnosti úseček bylo požadavkem možnost omezení maximálního povoleného rozdílu souřadnic v daném shluku. Do shluku se nesmí přiřadit

úsečka, která svojí vzdáleností počátečních bodů od centra shluku, úhlem nebo délkou překračuje povolenou mez. Tomuto požadavku ale nelze vyhovět pomocí dosavadní používané eukleidovské metriky, je nutné zavést novou.

S popisem úsečky (3.3) lze již přistoupit k definici výsledné úsečkové metriky. Úsečková metrika také počítá podobnost dvou elementů, ale navíc je obohacena o kontrolu maximální hodnoty pro danou souřadnici. Pokud rozdíl v některé souřadnici metriky překročí maximální povolenou hodnotu, výsledná vzdálenost úseček se nastaví na nekonečnou. Jsou dány dvě úsečky $t_1 = (x_{t1}, y_{t1}, X_{t1}, Y_{t1}, a_{t1}, l_{t1})$ a $t_2 = (x_{t2}, y_{t2}, X_{t2}, Y_{t2}, a_{t2}, l_{t2})$. Pak jejich vzájemná vzdálenost d s omezením maximální hodnoty vzdálenosti počátečních bodů m_e, úhlu m_a, délky m_l je definována jako:

$$d(t_1, t_2) = \begin{cases} \sqrt{(E * w_e)^2 + (A * w_a)^2 + (L * w_l)^2} & \text{pro } E < m_e \wedge A < m_a \wedge L < m_l \\ \infty & \text{v ostatních případech} \end{cases} \tag{3.3}$$

$E = E(t_1, t_2) = \sqrt{(x_{t2} - x_{t1})^2 + (y_{t2} - y_{t1})^2}$ ... eukleidova vzdálenost počátečních bodů úseček v $\mathbb{R}^2$

$A = A(t_1, t_2) = a_{t2} - a_{t1}$ ... rozdíl úhlů úseček

$L = L(t_1, t_2) = l_{t2} - l_{t1}$ ... rozdíl délek úseček

kde :

$d(t_1, t_2)$	...	vážená vzdálenost úseček t_1 a t_2
$m_e \in \mathbb{R}^+$	...	maximální eukleidovská vzdálenost počátečních bodů úseček
$m_a \in \mathbb{R}^+$	...	maximální rozdíl úhlu úseček
$m_l \in \mathbb{R}^+$	...	maximální rozdíl délek úseček
$w_e \in \mathbb{R}$	...	váha vzdálenosti poč. bodů úseček
$w_a \in \mathbb{R}$	...	váha úhlu úseček
$w_l \in \mathbb{R}$	...	váha délky úseček

Princip výpočtu podobnosti úseček pomocí úsečkové metriky je podobný jako u dosavadních metrik. Úsečková metrika využívá eukleidovu vzdálenost namísto polohových souřadnic x, y. Další rozdíl spočívá v kontrole maximálního rozdílu, kde „nekonečná nepodobnost“ způsobí nenavázání dané úsečky pod centrum shluku. Jinými slovy, navázání pod daný shluk se stane „nesmírně drahé“ a proto se neprovede. Vlivu úsečkové metriky se věnuje kapitola (4.3).

4 Experimenty a výsledky

Experimenty byly prováděny na sestavě s procesorem Intel®Pentium®Dual CPU T3200 (2,00 GHz, 2 jádra) a 3 GB operační pamětí.

Po úspěšné implementaci aplikace bylo nutné ověřit správné chování knihovny. Ověření implementace proběhlo na uměle vytvořených digitálních obrázcích. Následně byla ověřena funkčnost hiearchického shlukování knihovny na uměle vytvořených datech. Poté experimenty směřovaly k hlavnímu cíli, datům z oblasti GIS.

Vlivu nastavení shlukování se věnují podkapitoly (4.1.1) a (4.1.2). Experimenty s hiearchickými daty jsou popsány v podkapitole (4.2). Geodetickým datům a výsledkům s úsečkovou metrikou se věnuje podkapitola (4.3).

4.1 Digitální obrázky a ověření funkčnosti

Body v digitálním obrázku (*pixely*) se od sebe liší svojí pozicí, ale i barvou, která je pro nás přirozenějším kritériem shlukování nežli např. směrový vektor v geodetických datech. Pixel p lze definovat jako $p = (x, y, R, G, B)$

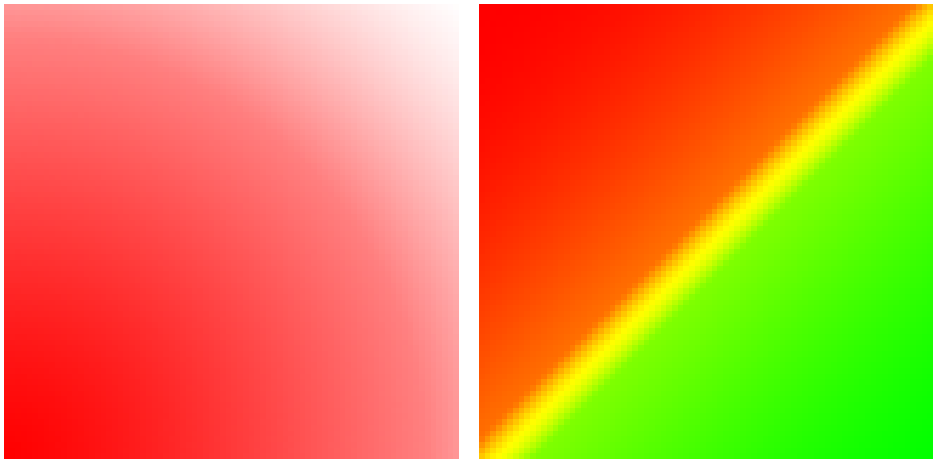
kde:

x, y	...	souřadnice polohy bodu
R, G, B	...	souřadnice barvy bodu v prostoru RGB

Díky barevné rozdílnosti bodů si lze v některých případech všimnout navázání bodu k jinak barevnému shluku. Za tímto účelem byly vytvořeny 2 testovací obrázky (viz Obr. 4.1) na kterých bylo dále experimentováno.

Obr. (4.1a) ukazuje pomalý přechod červené barvy k bílé barvě. Obrázek byl vytvořen za účelem otestování chování shlukování s mírnými změnami v jejich barevných souřadnicích.

Obr. (4.1b) naopak ukazuje rychlý skok mezi barevnými přechody s žlutým diagonálním předělem. Zde bylo cílem otestování chování shlukování při vzájemném „soupeření“ souřadnic barev.



(a) červeno-bílý přechod (b) červeno-zelený přechod

Obr. 4.1: Testované obrázky

4.1.1 Vliv vah na shlukování

Nyní budeme sledovat vliv vah na výsledek shlukování (viz Obr. 4.2). První z experimentů představuje výsledek shlukování, které ignoruje barevnou složku bodů (viz Obr. 4.2b). Jak si můžeme všimnout, metoda vytvořila shluky, pod které patří pouze polohově nejbližší body.

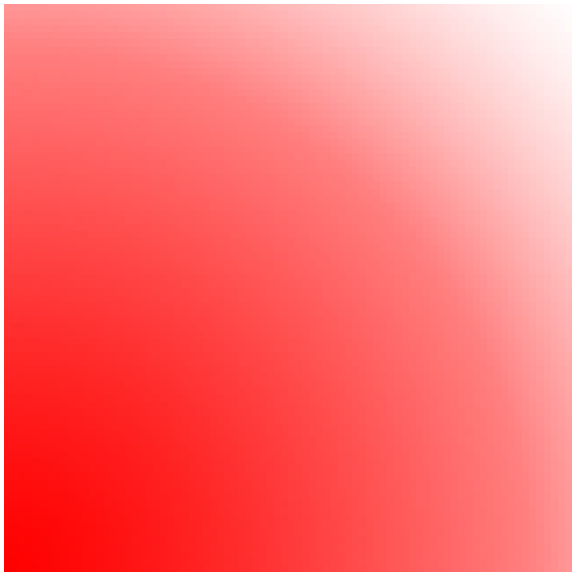
Nyní změníme váhy tak, aby shlukování upředňostnovalo barevnou složku (viz Obr. 4.2c). Ze znalosti vah (viz kapitola 2.2) byly nastaveny váhy takto $x = 0.2$; $y = 0.2; R = 1.0; G = 1.0; B = 1.0$. Zde můžeme jasně vidět vliv barevné složky na utváření shluků.

Do shluku se vždy přiřadí okolní body stejného odstínu červené barvy až do místa, kde „oslabená“ poloha bodu převáží barevnou složku a donutí algoritmus k vytvoření nového shluku. Proto také tvary shluků kopírují tónový rádius obrázku.

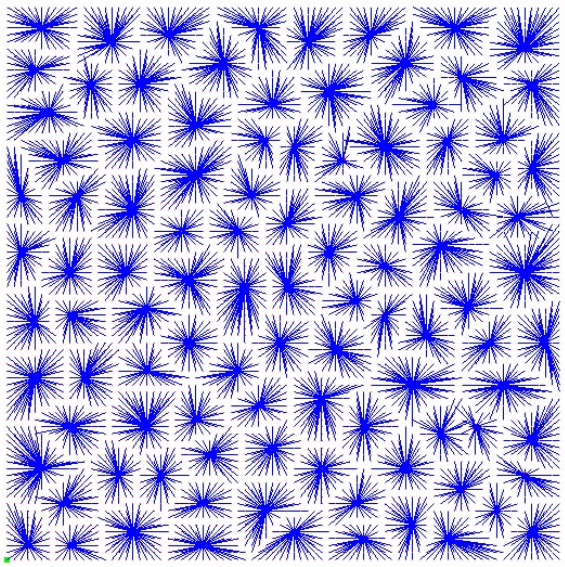
Vliv vah si mužeme ukázat i na soupeření barevných složek mezi sebou (viz Obr. 4.3). Nejdříve si ukážeme výsledek implicitního nastavení pro shlukování, kde jsou všechny souřadnice rovnocenné.

U shlukování implicitním nastavením vah (viz Obr. 4.3b) si můžeme všimnout změny tvarů shluků v místech žlutého přechodu. V tomto případě jsou barevné složky vzájemně rovnocenné a shluky proto tvoří diagonální předěl mezi červenou a zelenou barvou. Tento stav se změní ve chvíli, kdy oslabíme jednu z barevných složek, např. zelenou barvu (viz Obr. 4.3c).

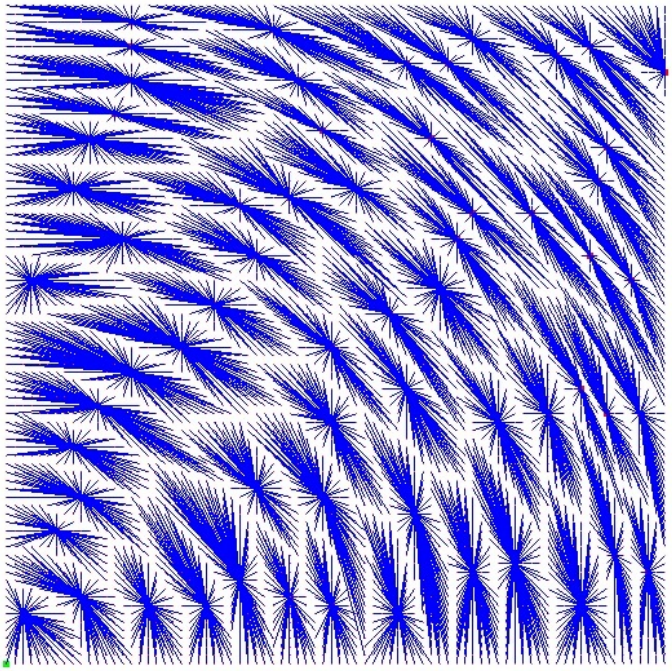
Zde si můžeme všimnout červených shluků na diagonále, pod které nyní patří



(a) Vzorový obrázek



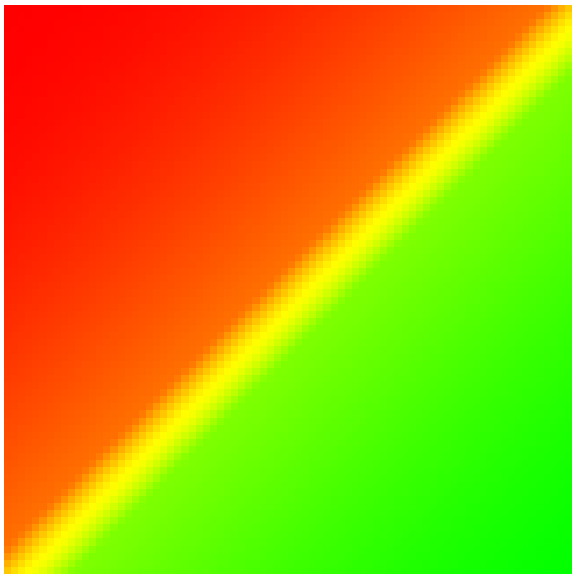
(b) Shlukování dle pozice bodů



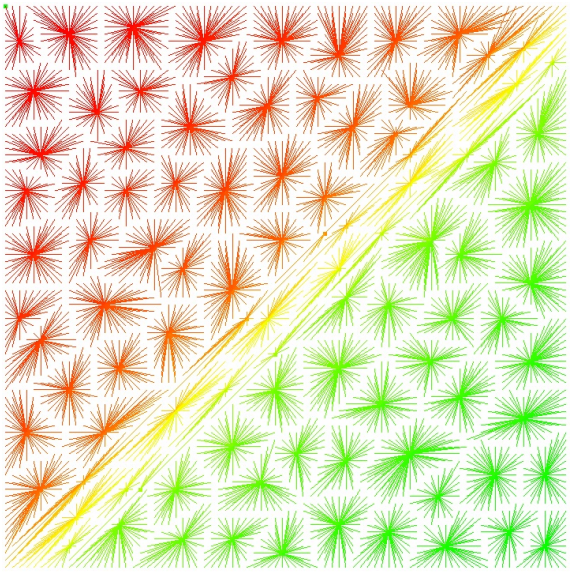
(c) Shlukování upřednostňující barevnou složku bodu

Obr. 4.2: Klasické shlukování

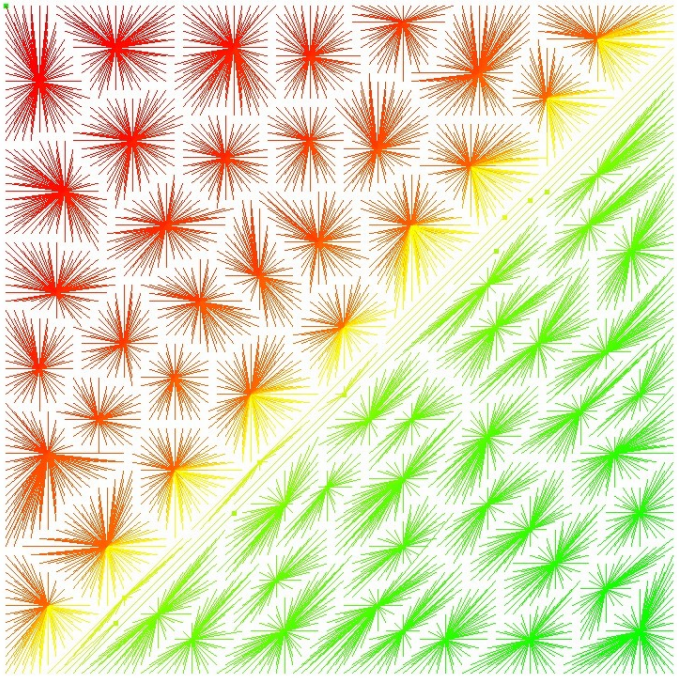
i některé diagonální žluté krajní body. Naproti tomu pod shluky oslabené zelené barvy patří, dle předpokladů, pouze odstíny zelené barvy.



(a) Vzorový obrázek



(b) Shlukování dle všech dimenzí



(c) Shlukování upřednostňující červenou barvu

Obr. 4.3: Klasické shlukování

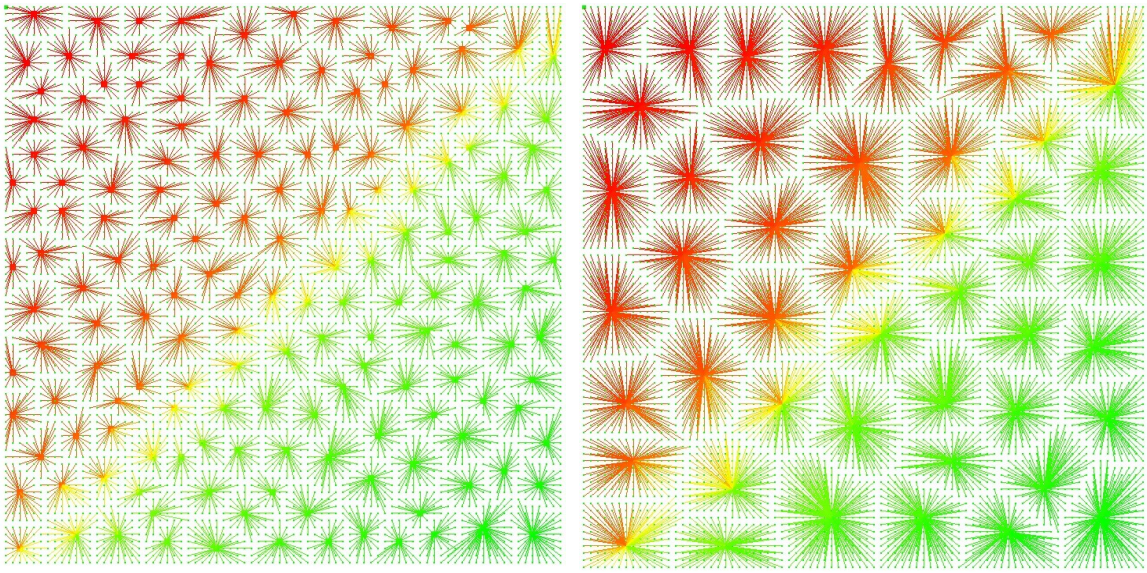
Pod shluky zelené barvy patří pouze body se stejným odstínem zelené, protože světlější nebo tmavší odstín zelené má, díky váhám, menší ohodnocení.

4.1.2 Velikost a počet shluků

Dalším možným koeficientem, kterým lze ovlivnit shlukování, je koeficient ceny za vytvoření nového shluku. Při etapě vylepšování dosavadního lokálního řešení (viz kapitola 2) se koeficientem násobí dosavadní ohodnocení shluku. Tímto koeficientem lze ovlivnit počet a tím i velikost shluků (viz Obr. 4.6).

Obr. 4.4a představuje výsledek při nastavení ceny za vytvoření 0.5, tedy ohodnocení vytvoření nového shluku je o polovinu výhodnější než při implicitním nastavení (koeficient roven 1). Na druhém Obr. 4.4b je výsledek shlukování při koeficientu ceny 3.0, tedy 6-krát větší než na předchozím experimentu. Kromě o poznání větších shluků si povšimněme diagonálního předělu. V těchto místech byly shluky vytvořeny co největší, a to i za cenu přidělení bodů s různými barvami. Za zmínku stojí shluk v levé dolní části Obr. 4.4b, pod který patří body všech tří barev.

Pokud bude koeficient ceny za vytvoření rovný nule, algoritmus prohlásí každý bod za centrum shluku, jelikož vytvoření shluku je ohodnoceno nulou a jakékoli přiřazení by toto ohodnocení pouze zhoršilo.



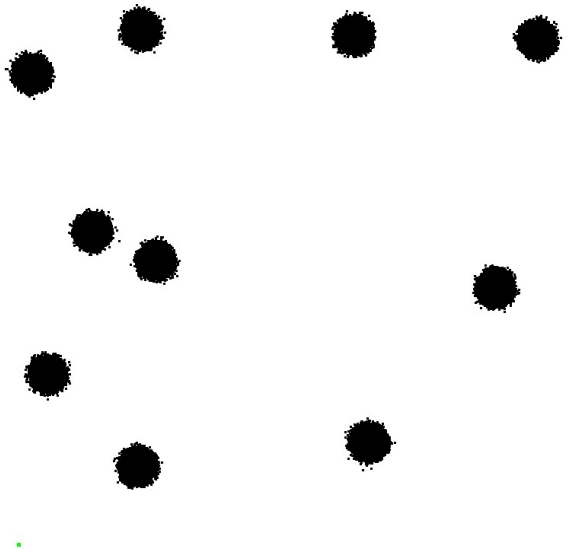
(a) Cena za vytvoření shluku 0.5 (b) Cena za vytvoření shluku 3.0

Obr. 4.4: Klasické shlukování

4.2 Hiearchické úrovně

Hiearchické shlukování vytváří během své činnosti hiearchické úrovně. Vstupní soubor obsahuje na každé řádce souřadnice jednoho bodu. V každé další výstupní úrovni jsou souřadnice centra shluku, za kterými následuje číslo představující index a délku. Index představuje řádek, kde se centrum v předchozí úrovni nachází. Délka poté představuje počet bodů od pozice indexu, které k danému centru patří.

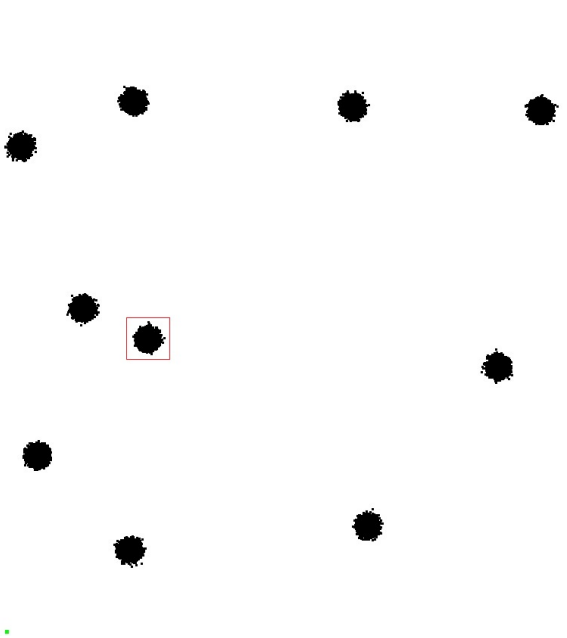
Nyní si předvedeme algoritmus na umělě vygenerovaných datech. Data obsahují 10^6 bodů definované x, y souřadnicí (viz Obr. 4.5). Účelem uspořádání dat byla snadná identifikace správnosti pozice shluků, které se budou vytvářet v místech největší koncentrace bodů.



Obr. 4.5: Vstupní data hiearchického shlukování (10^6 bodů)

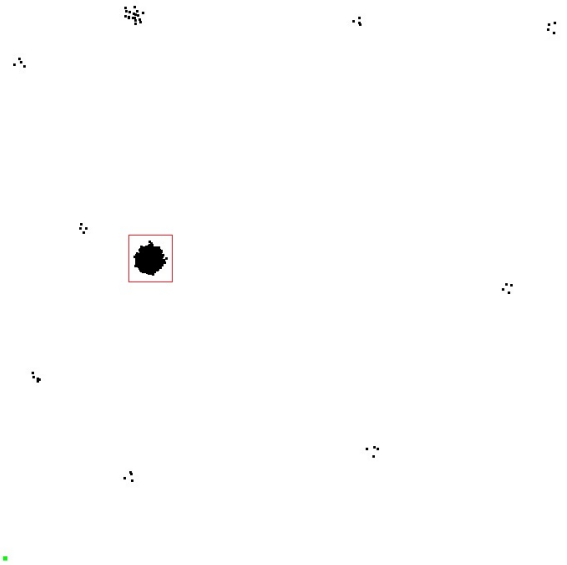
Pokud budeme shlukovat dle polohových souřadnic, výsledkem budou 3 úrovně. První úroveň je stejná jako vzorová data (viz Obr. 4.5). Zbývající 2 úrovně jsou zobrazeny na Obr. 4.6a a 4.6c. Jak si můžeme všimnout, vrcholy se vytvářejí na místech největší koncentrace bodů. Vyšší úrovně hiearchického shlukování poté obsahují řádově nižší množství bodů.

Implementace knihovny, díky struktuře ukládání center shluků, umožnuje po skončení shlukování přístup ke všem výstupním úrovním shlukování a jejich bodům. Tato funkce je demonstrována na Obr. 4.6c, kde jsou zároveň zobrazeny dvě úrovně shlukování (shluk 2. úrovně vyznačený na Obr. 4.6a je kombinován se shluky 3.úrovně). Obr. 4.6b a 4.6d představují přiblížené vybrané části z daných úrovní, kde si můžeme všimnout jejich shody.



(a) Shlukování úroveň 2 (`15 431` bodů)

(b) Přiblížení výběru z 2 úrovně



(c) Shlukování úroveň 3 (`57` bodů) s vizualizací bodů z nižší úrovně

(d) Přiblížení výběru z 3 úrovně

Obr. 4.6: Hiearchické shlukování

4.3 Geodetická data

Data pocházejí z oddělení geomatiky katedry matematiky fakulty aplikovaných věd. Původní data jsou reprezentována spojenými modrými body, které představují koncové a počáteční body úseček. Cílem experimentu je zachytit shluky s malým počtem bodů a izolované body, protože odpovídají tzv. „hrubým chybám“ v datech.

První vstupní data obsahují vesnice Plzeňského kraje České republiky, a to konkrétně Komušín (viz Obr. 4.7a), Loučim (viz příloha A.1) a Pavlovsko (viz příloha A.1). Délky úseček v těchto datech jsou zanedbatelné oproti rozsahu souřadnic, proto vizualizace vypadá jako by se jednalo o bodová data.

Druhý balíček dat představuje nameřená data za celé Čechy (viz Obr. 4.7b). Úsečky těchto dat jsou řádově větší než u předešlých vesnic, z důvodu většího měřítka. Náhled je pořízen tak, aby bylo vidět koncové a počáteční body. Ve výsledcích shlukování je přiblíženo pouze na počáteční body, aby bylo možné sledovat utváření shluků.

Experimenty probíhaly na všech poskytnutých datech se stejnými výsledky, proto budou předvedeny pouze výsledky na datech Komušin a Čechy, ostatní lze najít v příloze A.2.

Nejdříve byla otestována klasická eukleidovská metrika (podkapitola 4.3.1) a poté upravená úsečková metrika (podkapitola 4.3.2). Ve výsledcích shlukování je použita červená barva pro centrum shluku, ke kterému jsou černou úsečkou připojeny přiřazené úsečky.

(a) Komušín

(b) Čechy

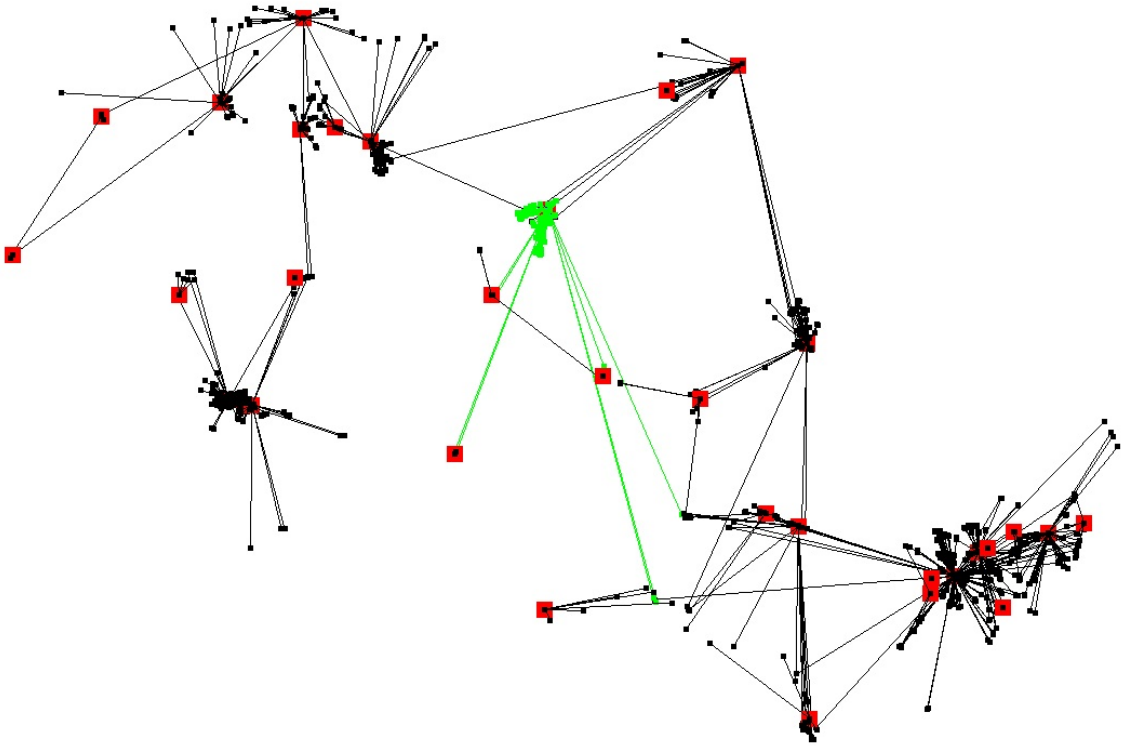
Obr. 4.7: Geodetická data

4.3.1 Eukleidovská metrika

Geomatická data popisují úsečky zadané počátečními a koncovými vrcholy, směrovou složkou dX, dY a délkou.

Nastavení vah probíhalo v souladu s požadavky zadavatelů: Shluky mají být vytvořeny tak, aby úsečky v jednom shluku byly „dostatečně blízko u sebe“, tj. s podobným počátečním vrcholem, úhlem s kladnou poloosou x a podobnou délkou a identifikovat tzv. „hrubé chyby“, tedy úsečky, které mají velmi rozdílné souřadnice oproti ostatním. Tyto „hrubé chyby“ budou představovat shluky s velmi malým počtem přiřazených úseček nebo budou naprosto osamocené.

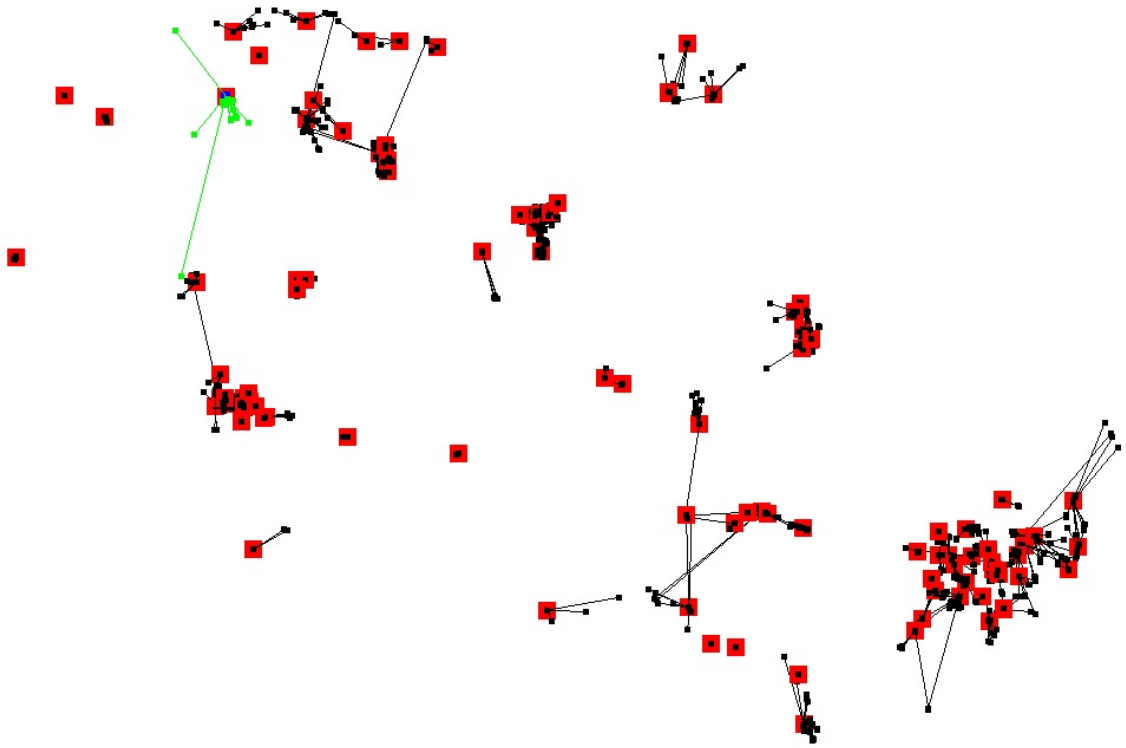
Těmto požadavkům odpovídá nastavení vah eukleidovské metriky $x_t = 1.0; y_t = 1.0; X_t = 0.0, Y_t = 0.0, a_t = 1.0, l_t = 1.0$ (viz popis úsečky v podkapitole 3.3). Výsledkem jsou, shluky pod které patří i vzdálené úsečky, jako je například zeleně zvýrazněný shluk na Obr. 4.8. Přiřazené vzdálené úsečky se natolik podobají svým úhlem a délkou, že tyto veličiny „převážily“ polohové souřadnice. Přiřazení tak vzdálených úseček se zdá být na první pohled chybou, je ale nutné si uvědomit měřítko naměřených dat.



Obr. 4.8: Komušín, shlukování eukleidovskou metrikou

Přiřazování vzdálených úseček k shlukům můžeme nepřímo ovlivnit koeficientem ceny za vytvoření shluku (viz podkapitola 4.1.2). Pokud nastavíme tento koeficient dostatečně malý, bude výhodnější vytvořit nový shluk nežli přiřadit zmíněné vzdálené úsečky. Vznikne tak mnoho malých shluků, pod které budou patřit pouze nejblížší podobné úsečky. Vzdálené úsečky, které by jinak patřily k těmto shlukům, utvoří vlastní shluk (viz Obr. 4.9).

Tímto způsobem lze splnit i požadavek na maximální hodnoty souřadnic úseček v shluku, nelze ho však přímo vynutit, jak můžeme vidět na zvýrazněném shluku. Zde opět podobnost směrového úhlu a délky usečky převážila polohové souřadnice. Na Obr. 4.9 lze již rozpoznat „hrubé chyby“, shluky s malým počtem bodů nebo přímo osamocená centra shluků.



Obr. 4.9: Komušín, shlukování eukleidovskou metrikou s malou cenou za vytvoření shluku

4.3.2 Úsečková metrika

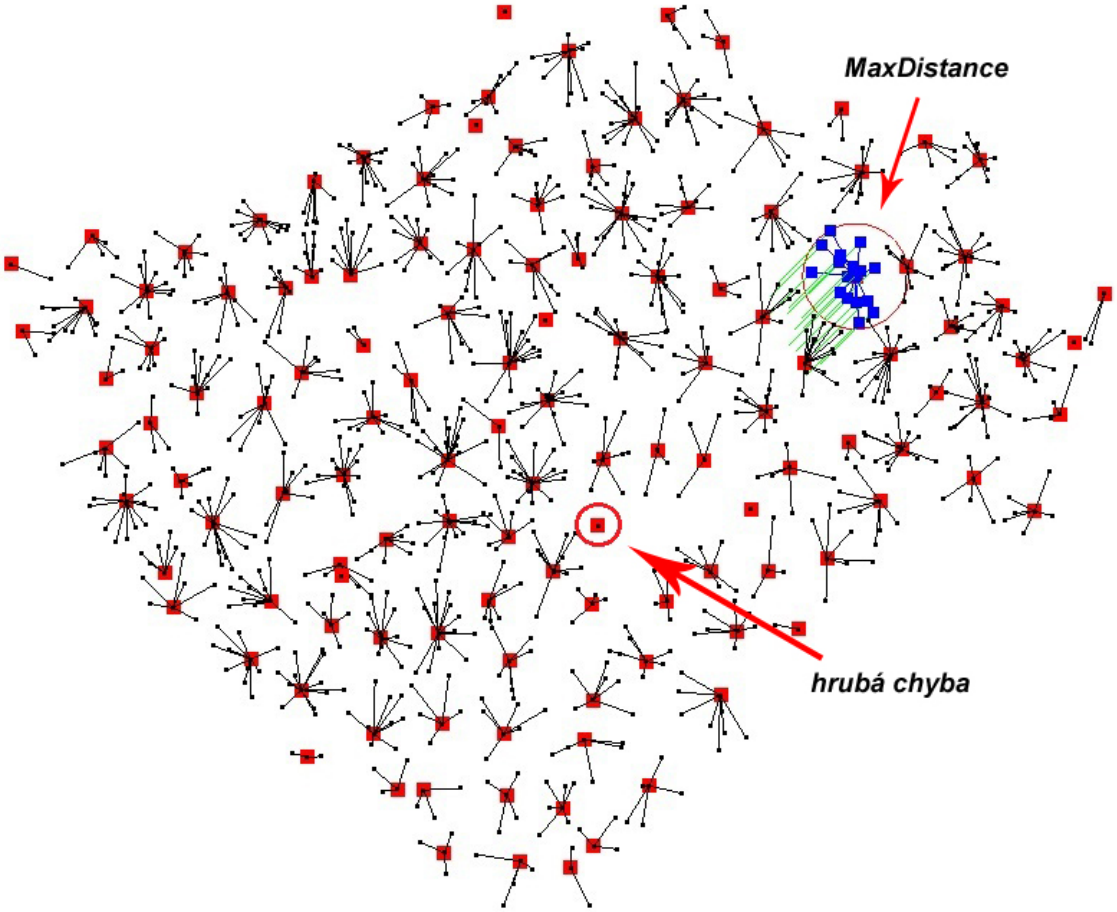
Zvolení maximálních hodnot souřadnic je na znalosti dat, experimenty měly pouze prokázat dodržení těchto hodnot danou metrikou. Váhy úsečkové metriky

byly nastaveny např. $DistanceWeight = 1.0; AngleWeight = 1.0; LengthWeight = 1.0; MaxDistance = 7000.0; MaxAngle = 90.0; MaxLength = 3.0$, (viz vzorec 3.3 v podkapitole 3.3)

kde:

$DistanceWeight$	$\in \mathbb{R}$	...	váha eukleidovské vzdálenosti počátečních bodů
$AngleWeight$	$\in \mathbb{R}$	...	váha úhlu úsečky
$LengthWeight$	$\in \mathbb{R}$	...	váha délky úsečky
$MaxDistance$	$\in \mathbb{R}^+$	...	maximální vzdálenost počátečních bodů úseček
$MaxAngle$	$\in \mathbb{R}^+$	...	maximální rozdíl úhlů úseček
$MaxLength$	$\in \mathbb{R}^+$	...	maximální rozdíl délek úseček

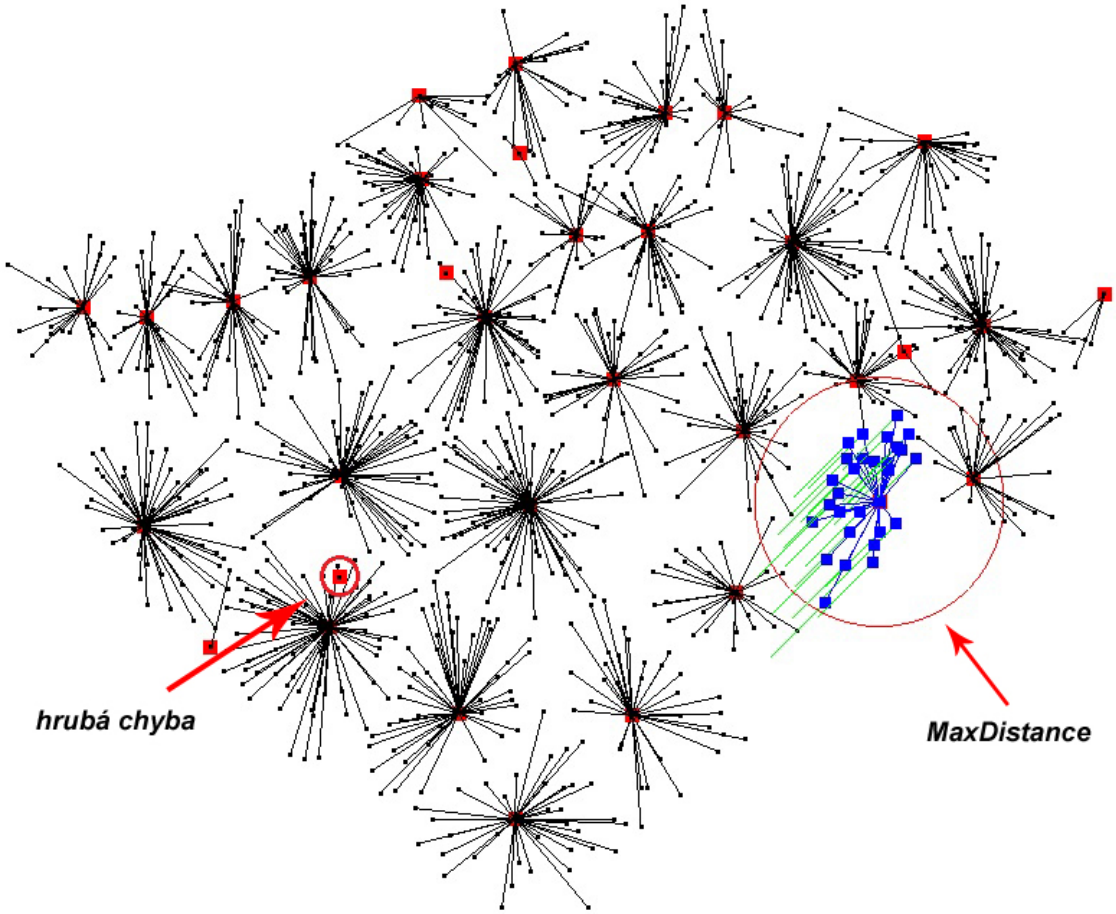
Výsledek shlukování vidíme na Obr. 4.10.



Obr. 4.10: Čechy, shlukování úsečkovou metrikou

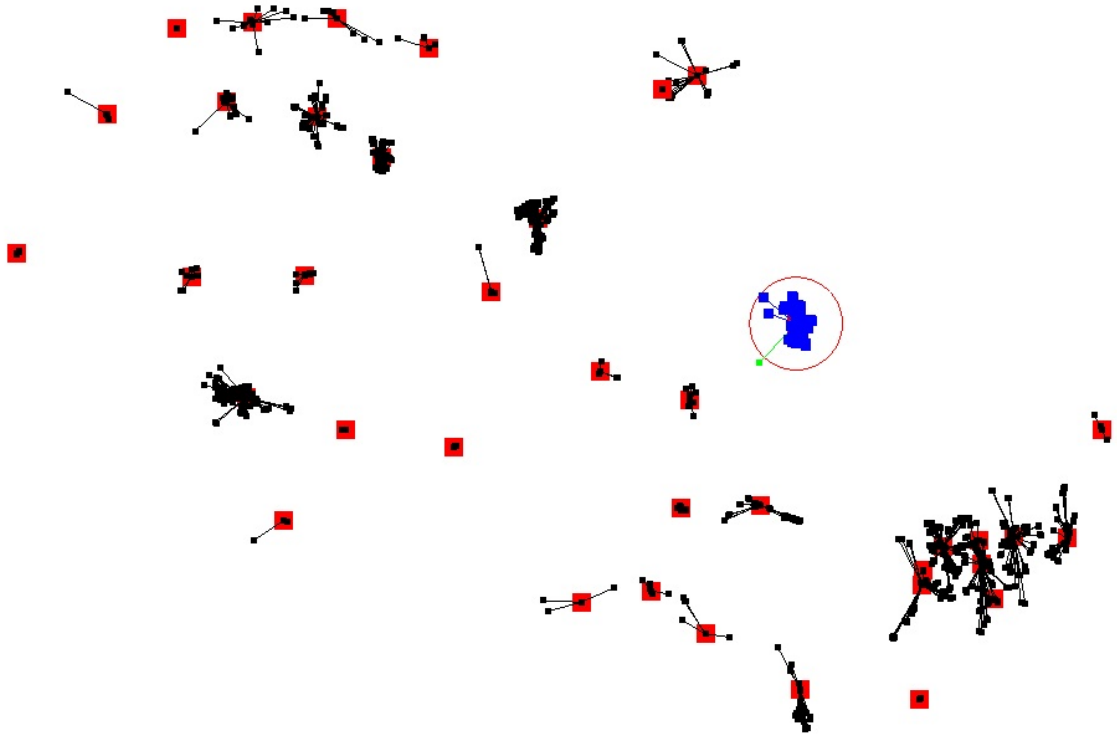
Zelené úsečky vycházející z počátečních bodů představují směrový úhel přiřazených bodů. Můžeme si všimnout jejich rovnoběžnosti ovlivněné váhou úhlové souřadnice. Úsečky přiřazené k centru shluku dodržely požadovanou vzdálenost zobrazenou červenou kružnicí se středem v centru shluku. V těchto datech byly také identifikovány „hrubé chyby“, např. červeně označené centrum shluku ve středu Obr. 4.10, které nemá přiřazené žádné body. Abychom zjistili vliv maximálních hodnot, nastavíme např. $MaxDistance = 10^3$ (viz Obr. 4.11).

Můžeme si všimnout menšího počtu shluků, protože se pod shluk mohly přiřadit i vzdálenější body. I zde byla identifikována „hrubá chyba“ v levé části obrázku. Doposud jsme neměnili žádné jiné parametry (např. cenu za vytvoření shluku) kromě vah souřadnic.



Obr. 4.11: Čechy, úsečková metrika, $MaxDistance = 10^3$

Je nutné si uvědomit, že algoritmus neprochází všechna navázání úseček, pouze se náhodně snaží vylepšit dosavadní řešení. Proto se může stát, že některá přiřazení, která jinak svými vlastnostmi nevyhovují maximálním přípustným hodnotám, nebudou opravena, jak můžeme vidět na následujícím případu shlukování dat Komušín, kde byla nastavena maximální vzdálenost od centra shluku $MaxDistance = 500$ (viz Obr. A.3). Zde zeleně zvýrazněný bod přesahuje danou maximální vzdálenost a je ohodnocen úsečkovou metrikou podobností nekonečno. Tento nedostatek odstraníme zvětšením počtu iterací nebo použitím aglomerativního přístupu shlukování (viz kapitola 2.3).



Obr. 4.12: Komušín, úsečková metrika, $MaxDistance = 500$

V experimentech byla ověřena funkčnost shlukovací knihovny na uměle vytvořených i reálných datech. Byl prokázán vliv nastavení na utváření jednotlivých shluků. Dosavadní implementace programu dokáže postihnout geomatické požadavky, ať použitím eukleidovské nebo úsečkové metriky. V poskytnutých datech byly také identifikovány „hrubé chyby“, shluky o malém počtu přiřazených úseček.

5 Závěr

V práci byly vysvětleny základy problematiky shlukování a popsány vybrané algoritmy z této oblasti. Metoda shlukování byla použita na uměle vytvořených datech ověřující správnou implementaci programu. Dále byly popsány výsledky shlukování na reálných datech z GIS aplikací. Dosavadní experimenty prokázaly použitelnost metody shlukování v geomatických datech. Shlukování, díky své pružnosti nastavení parametrů dokázalo vyhovět všem geomatickým požadavům.

Předpokládá se pokračování práce v rámci společného výzkumu.

A Přílohy

A.1 Vizulizace

Obr. A.1 ukazuje data Loučim.

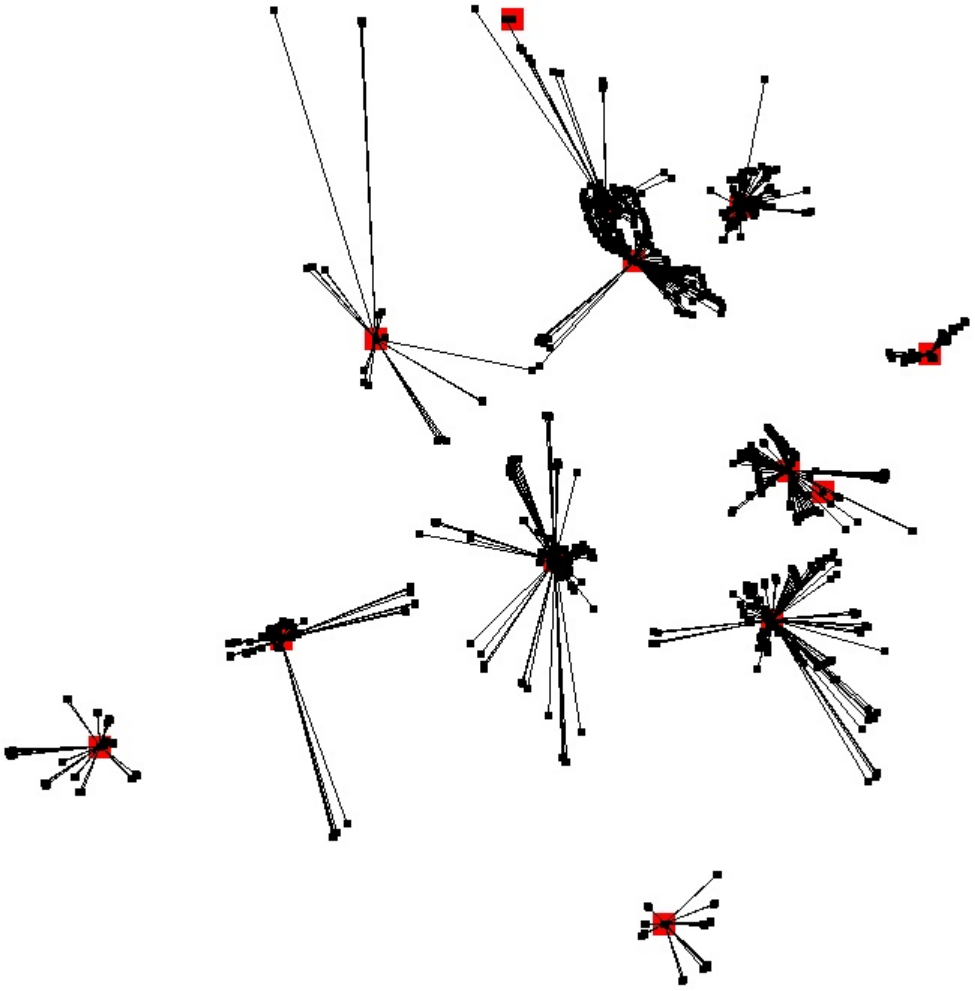
Obr. A.1: Loučim

Obr. A.2 ukazuje data Pavlovsko.

Obr. A.2: Pavlovsko

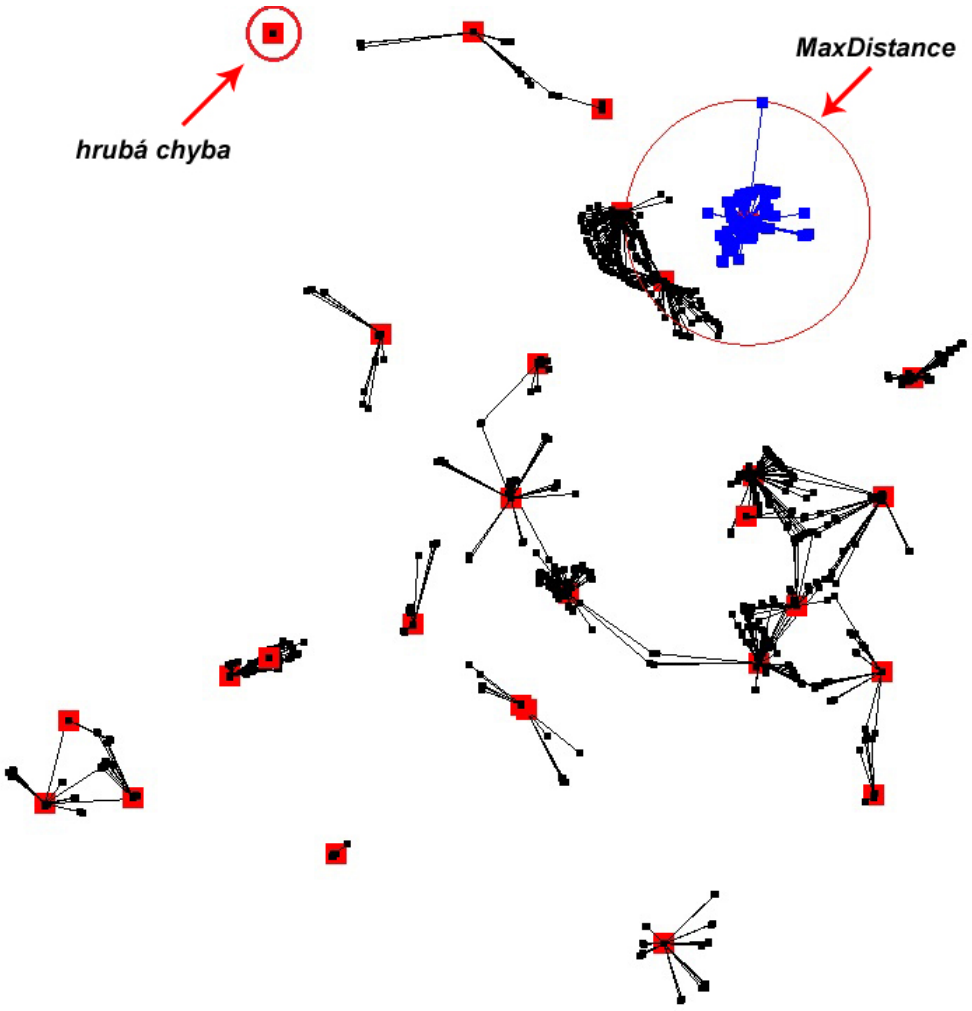
A.2 Experimenty s metrikami

Shlukování dat Loučim eukleidovskou metrikou při nastavení implicitních vah je ukázáno na Obr. A.3.



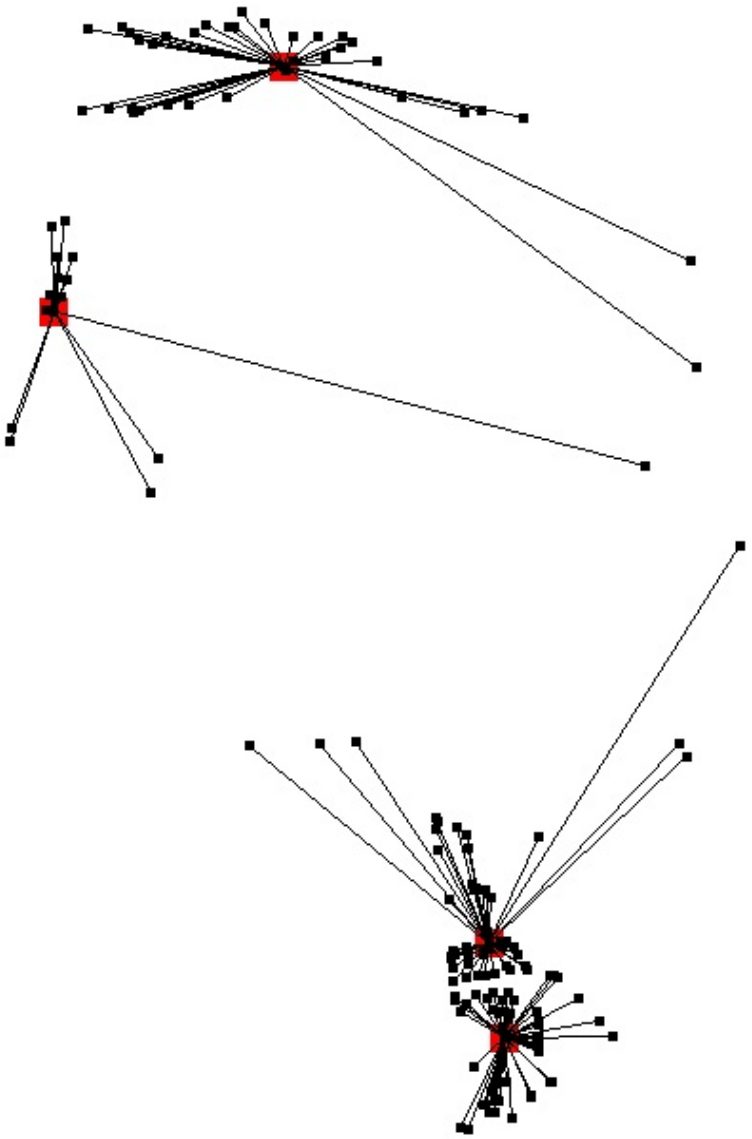
Obr. A.3: Loučim, shlukování eukleidovskou metrikou

Shlukování dat Loučim úsečkovou metrikou při nastavení vah:
$DistanceWeight = 1.0; AngleWeight = 1.0; LengthWeight = 1.0;$
$MaxDistance = 1000.0; MaxAngle = 90.0; MaxLength = 3.0$ poskytnulo výsledek ukázaný na Obr. A.4. MaxDistance je v pravé části Obr. A.4 znázorněna jako kružnice s poloměrem 1000. Nalezená „hrubá chyba“ je na Obr. A.4 zdůrazněna kružnicí.

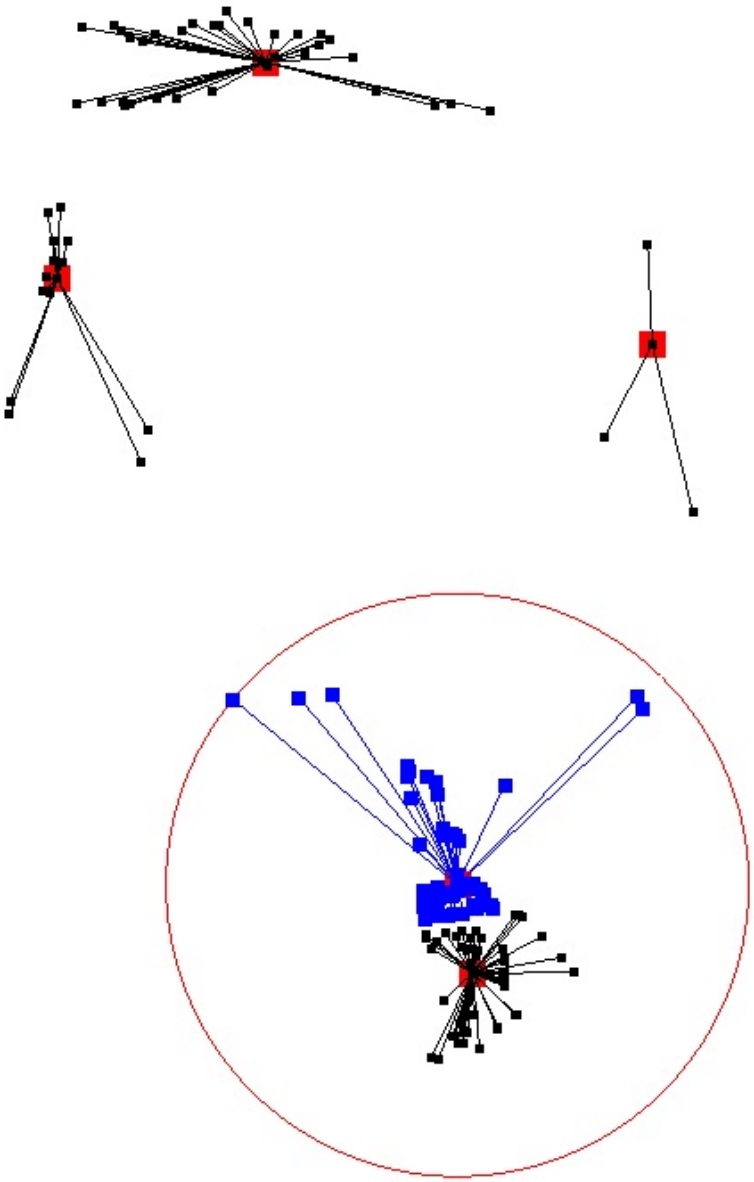


Obr. A.4: Loučim, shlukování úsečkovou metrikou

Shlukování dat Pavlovsko eukleidovskou metrikou při nastavení implicitních vah je na Obr. A.5. Protože v tomto experimentu není omezena maximální povolená vzdálenost, ke shlukům jsou přiřazeny i „poměrně vzdálené“ body. Obr. A.6 ukazuje, že při použití omezení maximální možné vzdálenosti je tento nežádoucí efekt odstraněn. Váhy pro Obr. A.6 byly nastaveny takto: $DistanceWeight = 1.0$; $AngleWeight = 1.0$; $LengthWeight = 1.0$; $MaxDistance = 1000.0$; $MaxAngle = 90.0$; $MaxLength = 3.0$.



Obr. A.5: Pavlovsko, shlukování eukleidovskou metrikou



Obr. A.6: Pavlovsko, shlukování úsečkovou metrikou

Literatura

[sj13] Skála, Jiří. *Algoritmy pro manipulaci s velkými geometrickými daty*, disertační práce. Západočeská univerzita v Plzni, fakulta aplikovaných věd, Plzeň, 2012. 111 l., str. 28-43, vedoucí práce Ivana Kolingerová.

[sk09] Skála, Jiří - Kolingerová, Ivana. *Data Stream Hierarchical Clustering Library* [software]. 2009. http://www.kiv.zcu.cz/vyzkum/software/2009/hier-clust-datastream.html

[cg13] Skála, Jiří - Kolingerová, Ivana. *Dynamic hiearchichal triangulation of a clustered data stream.* Computers & Geosciences, Elsevier, 37(8): 1092-1101, 2011.

[jt13] Tomeček, Jan. *Texty k prednáškám z MMAN3* [online]. [cit. 2013-04-30]. Dostupné z: http://aix-slx.upol.cz/∼tomecek/vyuka/ma1i/metr_prostory.pdf

[pv12] Vaněček, Petr. *OpenGL for .NET* [online]. [cit. 2013-03.09]. http://herakles.zcu.cz/∼pvanecek/OpenGL4net/

[wf13] Microsoft .NET Framework. *Windows Forms* [online]. [cit. 2013-01.02] http://msdn.microsoft.com/en-us/library/dd30h2yb.aspx

[wpf13] Microsoft .NET Framework. *Windows Presentation Foundation* [online]. [cit. 2013-01.02] http://msdn.microsoft.com/en-us/library/ms754130.aspx

[qt13] Qt Digitia. *Qt* [online]. [cit. 2013-05-02]. Dostupné z http://qt.digia.com