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

Bakalářská práce

Úprava rozvržení bodů v 2D algoritmem přemisťování

Plzeň 2022 Daniel Cába

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Daniel CÁBA**
Osobní číslo:	**A18B0183P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informatika**
Téma práce:	**Úprava rozvržení bodů v 2D algoritmem přemisťování**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s metodami úpravy rozložení bodů v rovině a s problematikou různého rozložení bodů.
2. Navrhněte vhodný algoritmus, který by body přemisťoval do požadovaného rozložení s respektováním uživatelem zadané tolerance jejich vzdáleností.
3. Navržené řešení implementujte, otestujte a zhodnoťte.
4. Algoritmus upravte tak, aby dokázal zaplňovat prázdná místa („díry"), otestujte a zhodnoťte meze jeho možností pro tento úkol.
5. Dosažené výsledky popište v textu práce.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Prof. Dr. Ing. Ivana Kolingerová**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **4. října 2021**
Termín odevzdání bakalářské práce: **5. května 2022**

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 14. října 2021

Prohlášení

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

V Plzni dne 26. června 2022

Daniel Cába

Abstract

This bachelor thesis deals with methods of modifying the distribution of points in the plane and the problem of different point distributions. The aim of the thesis is to learn about the algorithms that address this issue and then to select and test the properties of a suitable algorithm that would move the points to the uniform layout while respecting the user-specified tolerance of their distances.

Abstrakt

Tato bakalářská práce se zabývá metodami úpravy rozložení bodů v rovině a problematikou různého rozložení bodů. Cílem práce je seznámit se s algoritmy, které se na tuto problematiku zaměřují, a následně vybrat a otestovat vlastnosti vhodného algoritmu, který by body přemisťoval do rovnoměrného rozložení s respektováním uživatelem zadané tolerance jejich vzdáleností.

Poděkování

Rád bych tímto poděkoval Prof. Dr. Ing. Ivaně Kolingerové za odborné vedení, podporu a trpělivost, se kterou vedla tuto práci. Zároveň bych rád poděkoval rodině a přátelům za podporu nejen během vypracovávání této práce, ale v celém průběhu studia.

Obsah

1 Úvod

Jak v počítačové grafice tak v mnoha jiných oborech (např. v meteorologii, oboru požární ochrany či biologii) se řeší problém rozmístění objektů po vymezené oblasti. Toto rozmístění může být v praxi různé, ale obvykle je zapotřebí, aby určená oblast byla pokryta rovnoměrně. Tato práce tedy řeší, jakým způsobem je vhodné objekty přesunout, aby rozmístění zvolených objektů (resp. bodů) ve zvolené oblasti (resp. rovině) bylo na konci činnosti rovnoměrné.

První část práce se zabývá základy teorie pravděpodobnosti, která je potřebná pro různé způsoby rozložení bodů. Dále jsou zde vysvětleny pojmy Voroného diagram a CVT, na jejichž základě jsou objasněny algoritmy, kterými lze dosáhnout rovnoměrného rozložení bodů v rovině.

Druhá část práce se zabývá implementací jedné ze zmíněných metod. v páté kapitole je navržena metoda pro implementaci spolu se způsoby jejího otestování. Nad implementovanou metodou jsou prováděny experimenty, které mají za úkol otestovat vhodnost zvoleného algoritmu pro problém rovnoměrného rozmístění bodů v rovině a vyplnění děr v datech. Výsledky provedených experimentů jsou uvedeny v šesté kapitole.

Po závěru jsou v příloze doplněny tabulky s výsledky testů A.1, ze kterých vychází grafy obsažené v textu práce, a uživatelská dokumentace k odevzdávanému programu B.

2 Rozložení bodů v rovině

Tato kapitola zmiňuje několik základních rozložení, která lze použít pro nadefinování pozice bodů. Obecně jsou využívány dva typy rozložení, přičemž první jsou spojitá, tedy rozdělení definovaná pro všechna reálná čísla na daném intervalu. Existují též diskrétní rozdělení, ta jsou však definována pouze pro konečné spočetné množiny, obvykle celých čísel. Jelikož se práce bude zabývat prací s reálnými čísly, tato kapitola bude zaměřena pouze na spojitá rozložení.

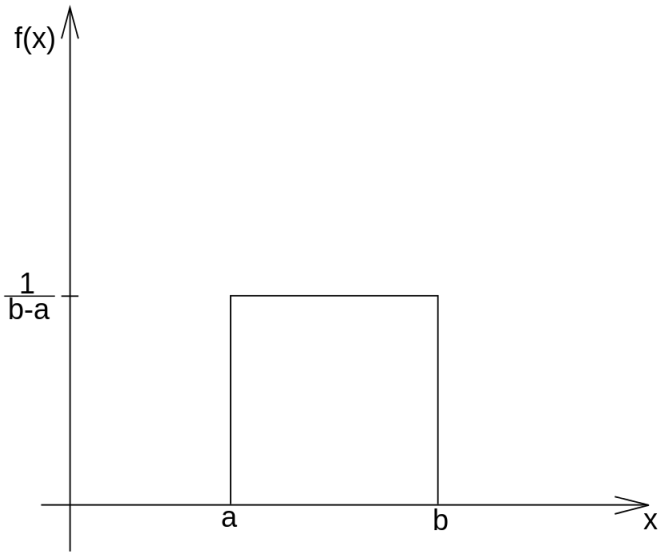
Pro práci s rozloženími je potřeba nejprve definovat hustotu pravděpodobnosti. Jedná se o funkci náhodné veličiny X, kterou značíme $f(x)$, a je definována jako *„limita pravděpodobnosti, že spojitá (kontinuální) náhodná veličina X nabude hodnoty z velmi malého intervalu $(x; x + \Delta x)$ vydělená délkou tohoto intervalu Δx v případě, že se tato délka Δx blíží k nule $(\Delta x \to 0)$ "* [2]. Hustota pravděpodobnosti $f(x)$ je nezáporná reálná funkce, přičemž celkový obsah plochy pod $f(x)$ je vždy roven 1 [2].

2.1 Rovnoměrné rozložení

Náhodná veličina X má rovnoměrné (rektangulární) rozložení s parametry α a β, $-\infty < \alpha < \beta < \infty$, jestliže má hustota pravděpodobnosti této veličiny tvar

$$\begin{aligned} f(x) &= \frac{1}{\beta - \alpha}, \qquad \alpha < x < \beta, \\ &= 0 \qquad\qquad\qquad jinak, \end{aligned} \tag{2.1}$$

jak je znázorněno na obr. 2.1 grafu této funkce [11] [2]. Jinak řečeno, každá hodnota, kterou lze v tomto rozložení dosáhnout, má stejnou pravděpodobnost výskytu. Pokud tedy bude tato funkce použita, náhodné body budou rovnoměrně rozloženy na celé zvolené ploše. V informatice se toto rozdělení často využívá u generátorů pseudonáhodných čísel, který je zabudován v každém statistickém programu a obecně i v programech, které pracují s daty (např. MS Excel) [11].



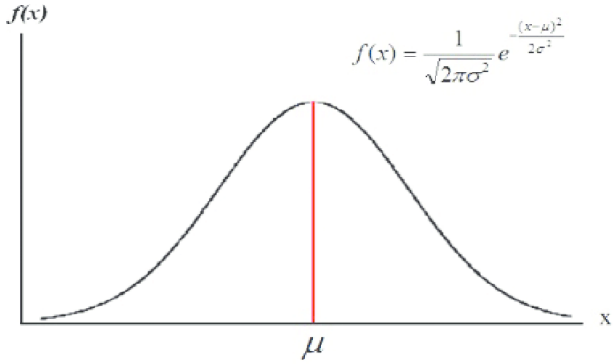
Obrázek 2.1: Graf hustoty $f(x)$ rovnoměrného rozložení [16]

2.2 Normální rozložení

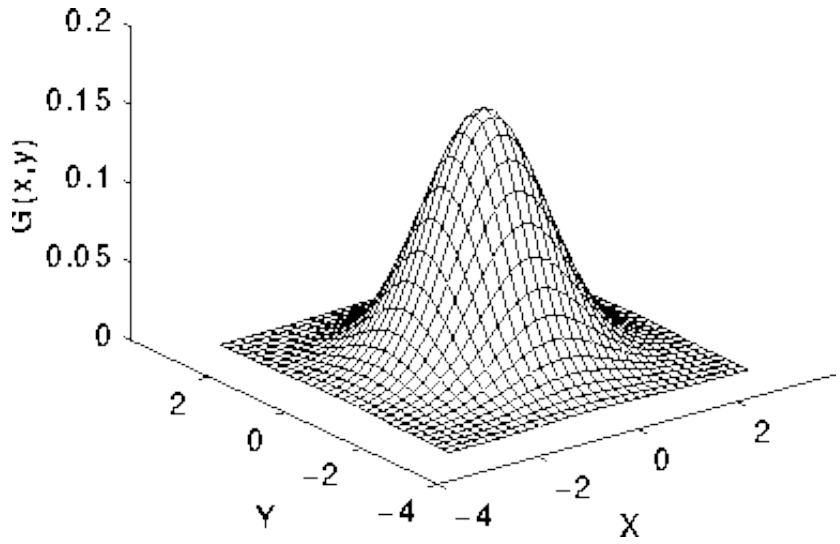
Náhodná veličina X má normální (Laplaceovo-Gaussovo) rozložení s parametry μ a σ^2, $-\infty < \mu < \infty$ a $\sigma^2 > 0$, jestliže její hustota pravděpodobnosti má tvar

$$f(x) = \frac{1}{\mu\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \quad -\infty < x < \infty. \tag{2.2}$$

Řídí-li se náhodná veličina normálním rozložením, pak je hustota pravděpodobnosti symetrická podle bodu μ (viz obr. 2.2). Toto rozložení má zásadní význam v teorii pravděpodobnosti, statistice i mnoha dalších nejen matematických oborech [11] [2]. Jelikož se dále budeme pohybovat ve 2D prostoru, pak je třeba zmínit, že na ploše jsou body vytvářené za pomoci normálního rozložení nejvíce koncentrovány v okolí středu dané oblasti, jak je naznačeno na obr. 2.3. $G(X, Y)$ v obrázku značí spojenou funkci hustoty pro funkce hustoty normálního rozložení na osách X, Y.



Obrázek 2.2: Normální funkce hustoty [12]



Obrázek 2.3: Normální hustota G ve 2D - pro osy X a Y - zobrazena ve 3D grafu [6]

2.3 Beta rozložení

Klasické beta rozložení je dvou parametrické rozložení, kde náhodná veličina X s parametry $a = 0, b = 1, \alpha > 0, \beta > 0$ má tvar hustoty pravděpodobnosti

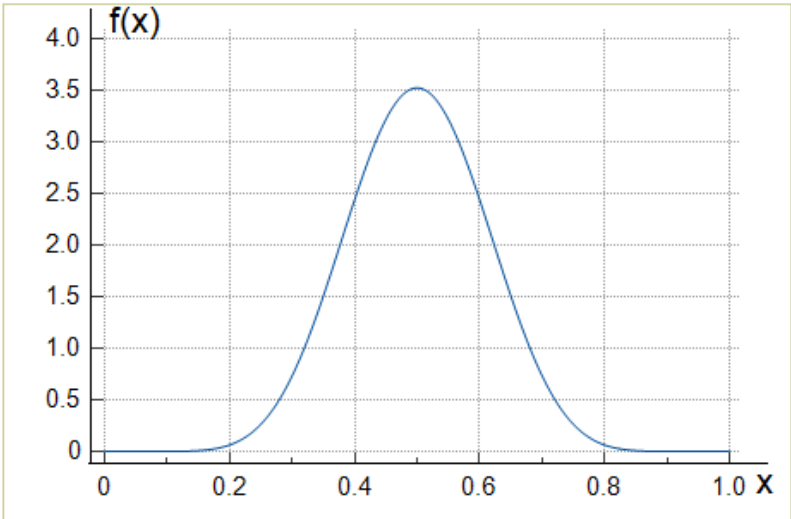
$$\begin{aligned} f(x) &= \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1} \qquad 0 < x < 1, \\ &= 0 \qquad\qquad\qquad\qquad\qquad\qquad jinak, \end{aligned} \tag{2.3}$$

přičemž $B(\alpha, \beta)$ je definováno jako

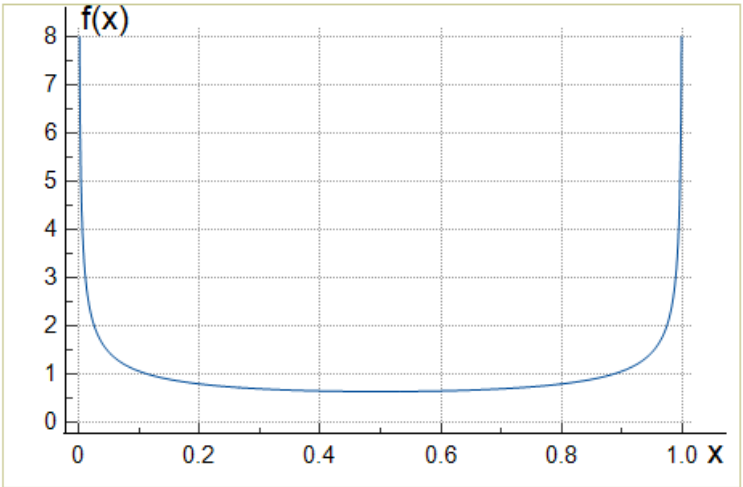
$$B(\alpha,\beta) = \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}\,dx \tag{2.4}$$

Klasické beta rozložení je tedy definované na intervalu $\langle 0, 1\rangle$ a jeho funkce hustoty nabývá rozličných tvarů v závislosti na volbě α a β. v případě, že

budou tyto hodnoty zvoleny jako $\alpha = \beta > 1$, pak vznikne jednovrcholové rozložení kolem středu jednotkové oblasti $[0, 1]^2$ (viz obr. 2.4). V opačném případě, kdy $\alpha = \beta < 1$, vzniká opět symetrická funkce hustoty, tentokrát ve tvaru písmene U (viz obr. 2.5) [11] [2].



Obrázek 2.4: Graf beta funkce hustoty, $\alpha, \beta > 1$ [13]



Obrázek 2.5: Graf beta funkce hustoty, $\alpha, \beta < 1$ [13]

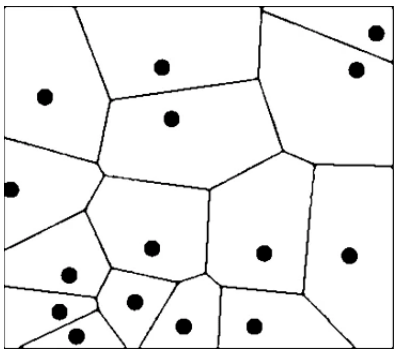
3 Voroného diagramy

Tato kapitola se zabývá Voroného diagramem a Centrálním Voroného diagramem.

3.1 Voroného diagram

Voroného diagram pojmenovaný podle ruského matematika Georgije Feodosjeviče Voroného je jeden ze základních způsobů dekompozice n-rozměrného prostoru na základě vzdálenosti od zvolených bodů, které se v daném prostoru nacházejí. Diagram lze využít v n-rozměrném prostoru, přičemž pro potřeby této práce se dále omezíme na plochu, tedy prostor $X = R^2$ [1] [3].

Obecně tato metoda rozděluje prostor do polygonálních buněk, kdy každá buňka je geometrické místo bodů s nejmenší vzdáleností od jednoho geometrického objektu či více geometrických objektů, obvykle bodů. Dále se v této práci omezíme jen na body. Toto dělení je znázorněno na obr. 3.1, kde je daný prostor rozdělen podle zadané množiny bodů. [1]



Obrázek 3.1: Ukázka jednoduchého Voroného diagramu pro 15 bodů [8]

Nyní definujme Voroného diagram přesněji [9]:

Je dána množina bodů $P = p_1, p_2, ..., p_n$ v R^2. Voroného diagram $V(P)$ znázorňuje rozklad zvoleného prostoru R^2 na oblasti (tj. regiony či buňky) $R_1, R_2, ..., R_i$, pro které platí

$$R_i = \{x : |p_i - x| <= |p_j - x|, \forall j \neq i\} \tag{3.1}$$

Oblasti Voroného diagramu mohou být uzavřené či neuzavřené. Uzavřené jsou ty buňky, které patří bodům na konvexní obálce. Průnik dvou různých

Voroného oblastí R_i a R_j je buďto prázdný nebo sestává z jediné hrany. Vzdálenost $d(p_i, p_j)$ mezi sousedními body Voroného diagramu lze definovat vícero způsoby, pro účely této práce je definována euklidovskou metrikou [1]:

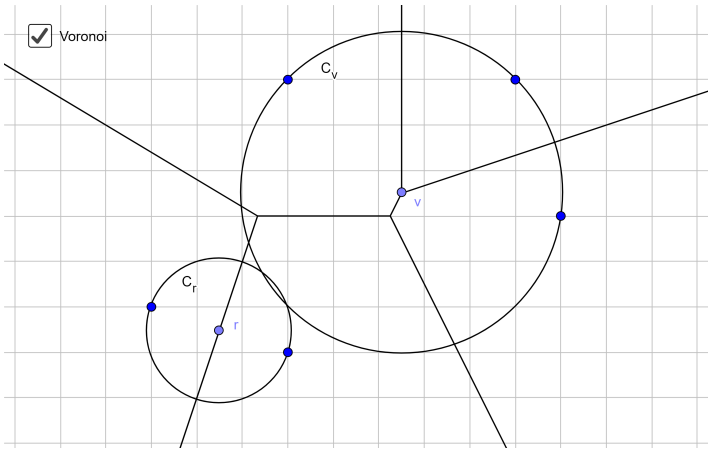
$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{3.2}$$

K vyslovení následujících vlastností je nutné definovat kružnici $C(p)$ se středem v bodě s, která je největší z daného středu a zároveň neobsahuje žádný z bodů množiny P.

→ Bod v je uzlem $V(P)$ právě tehdy, když kružnice $C(v)$ prochází alespoň třemi body z množiny P (viz obr. 3.2).

→ Dvě buňky (Voroného oblasti) s body $p, q \in P$ určují hranu Voroného diagramu $V(P)$ právě tehdy, když existuje bod r takový, že kružnice $C(r)$ prochází z množiny P pouze body p a q a žádnými jinými (viz obr. 3.2).

→ Jestliže žádné čtyři body z množiny P neleží na kružnici, pak každý uzel Voroného diagramu má stupeň 3 [1].



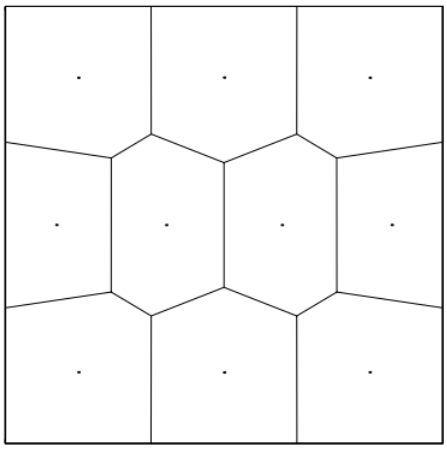
Obrázek 3.2: Ukázka kružnice $C(v)$ z vrcholu diagramu a kružnice $C(r)$ z hrany diagramu u Voroného diagramu pro 5 bodů [10]

3.2 Centrální Voroného diagram

Existuje speciální varianta Voroného diagramu, ve které se každý bod p_i z množiny P nachází v těžišti své buňky R_i. Těžištěm buňky se myslí bod, který se nachází na souřadnicích průměru všech bodů obsažených v dané buňce. Pro výpočet těžiště jsou neuzavřené buňky uzavřeny hranicemi zadaného prostoru. Tato varianta se nazývá centrální Voroného diagram (Central Voronoi Tesselation, CVT) [18].

3.2.1 Definice CVT

Je dána množina bodů $P = p_1, p_2, ..., p_i$, podle které lze dle definice 3.1 rozdělit prostor R^2 na Voroného buňky $R_1, R_2, ..., R_i$. Jelikož diagram poskytuje dostatek informací o každé buňce R_i, je možné určit její těžiště w_i, tedy hmotný střed oblasti. CVT je speciální případ Voroného diagramu, ve kterém $\forall R_i : p_i = w_i$ [3] [7] [17]. Na obr. 3.3 je vyobrazeno CVT vytvořené nad množinou 10-ti bodů. Dle definice se všechny body nacházejí v těžištích svých buněk.



Obrázek 3.3: Ukázka CVT pro 10 bodů [7]

3.2.2 Gershovo tvrzení

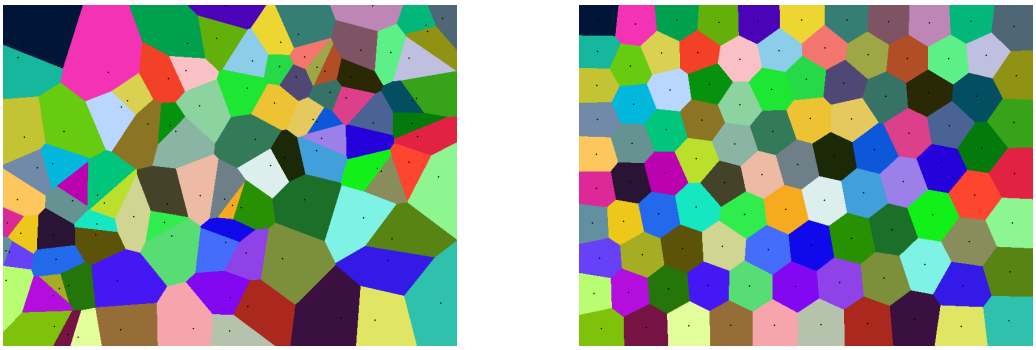
Dle podkapitoly 3.2.1 se v CVT každý bod p_i z množiny P nachází v těžišti své buňky R_i. Du a Wang [4] podle Gershe [5] uvádějí, že s konceptem CVT úzce souvisí Gershovo tvrzení, které bylo dokázáno pro jedno a dvoudimenzionální prostory. Gershovo tvrzení ve zmiňovaném článku [4] říká, že:

„*V asymptotickém smyslu, všechny buňky optimálního CVT tvoří tesselaci a jsou shodné se základní buňkou, jejíž tvar závisí na dimenzi.*"[4].

Na základě tohoto tvrzení víme, že v ideálním CVT ve 2D buňky nabývají tvaru základní buňky. Na ploše R^2 se jedná o tvar pravidelného šestiúhelníku, který je ideální pro rovnoměrné pokrytí plochy R^2 [7] [4]. Lze tedy prohlásit, že čím více šestiúhelníkových buněk se vyskytuje v CVT, tím efektivněji jsou zvolené body v oblasti rozmístěny [7] [4].

4 Iterační algoritmy tvořící CVT

V této kapitole jsou vysvětleny algoritmy, jejichž úkolem je přemístit body ze zadané množiny P tak, aby při vytvoření Voroného diagramu nad touto množinou vznikla CVT. Níže uvedené algoritmy jsou iterační - opakováním běhu každého z těchto algoritmů dochází k postupné přeměně obecného Voroného diagramu do tvaru CVT a spolu s tím k rovnoměrnému rozmístění bodů v prostoru. Na obrázku 4.1 je vyobrazen Voroného diagram a CVT nad množinou 100 bodů .



Obrázek 4.1: Ukázka Voroného diagramu a CVT nad 100 body

4.1 Lloydův algoritmus

Jedním ze základních a nejčastějších způsobů tvorby CVT je Lloydův iterační algoritmus. Od devadesátých let se hojně využívá např. v různých oblastech zpracování signálu, designu, počítačové grafiky, analýzy obrazu [17].

Algoritmus je následující:

→ Je zadána množina K bodů $\{p_i\}_{i=1}^{K}$

→ Nad množinou K je vytvořen Voroného diagram $V(P)$.

→ Pro každou Voroného oblast $\{R_i\}_{i=1}^{K}$ nalezneme její těžiště. Těžiště poté používáme jako nové body $\{w_i\}_{i=1}^{K}$ pro další iteraci.

→ Kroky algoritmu opakujeme, dokud se diagram dostatečně nepřiblíží podobě CVT.

Lloydův algoritmus přináší při dostatečném počtu iterací vysokou rovnoměrnost rozložení bodů. Výhodou tohoto algoritmu je relativně nízký počet iterací potřebný k dosažení dostatečné podobnosti mezi vznikajícím diagramem a ideálním CVT. Hlavní nevýhodou algoritmu je, že při každé iteraci je třeba vytvořit celý aktuální Voroného diagram a vypočítat těžiště každé z oblastí, jelikož výpočet diagramu a určení těžiště uvnitř oblastí je úkol náročný na výpočet. Algoritmus tedy není rychlý co se týče délky běhu jedné iterace. Navíc jeho charakteristické přesouvání všech bodů současně pro dosažení výsledku nemusí být v každém případě vhodné, jelikož existují případy, kdy změna pozice určitých bodů z množiny není žádaná, ba dokonce by mohla poškodit požadovaný výsledek. [17] [7]. Na obrázku 4.1 je ukázán výsledek Lloydova algoritmu pro 100 bodů po 100 iteracích.

4.2 McQueenův algoritmus

McQueenův algoritmus je obecně méně známým algoritmem rozložení bodů. Tento algoritmus svým výsledkem aproximuje CVT, avšak sám přímo s CVT nepracuje. Základem této metody je vzorkování náhodného bodu a následné průměrování posunu měřeného bodu na základě vygenerovaného vzorku [7].

Algoritmus postupuje takto:

→ Do zadané plochy $S \in R^2$ s danou množinou K bodů $\{p_i\}_{i=1}^{K}$ (viz obr. 4.2) je vygenerován náhodný bod ω (viz obr. 4.3, náhodný bod ω je zde reprezentován červenou barvou).

→ V množině K je nalezen bod p_i, který se nachází nejblíže bodu ω.

→ Dochází k výpočtu průměrné souřadnice mezi těmito body (viz obr. 4.4). Do tohoto průměru je následně přesunut bod p_i (viz obr. 4.5). Průměrování vzdáleností ovšem není ideálním řešením, proto je pro posun namísto vzorce pro průměrování

$$p_2 = \frac{\omega + p_1}{2} \tag{4.1}$$

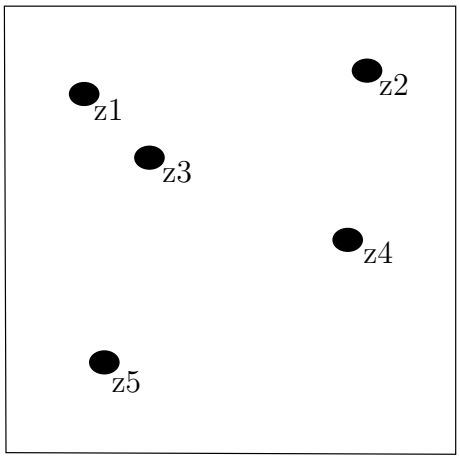
využit optimalizovaný vzorec:

$$p_{i+1} = \frac{\omega + x \cdot p_i}{x + 1} \tag{4.2}$$

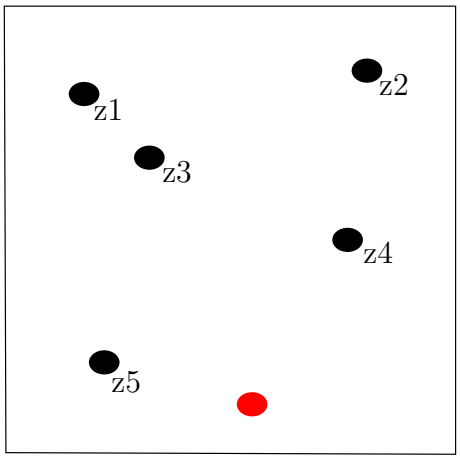
, kde p_i je posouvaný bod, ω náhodný bod, podle kterého je posun proveden a x reprezentuje informaci, kolikrát byl již bod dříve přesunut. K posunu bodu tedy stále dochází, ale použitím tohoto vzorce se starším bodům dodává větší váha (ukázka na obr. 4.6).

→ Celý algoritmus se opakuje, dokud není splněna zastavovací podmínka (obvykle předem zadaný počet iterací).

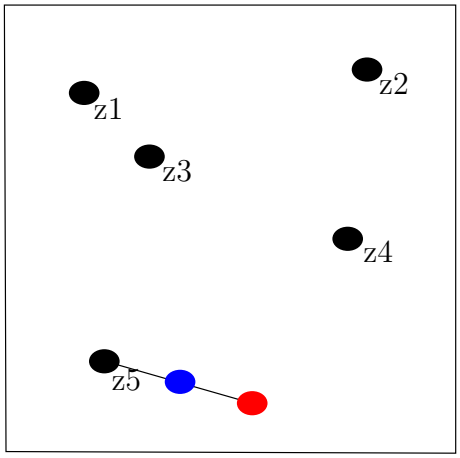
Narozdíl od Lloydova, McQueenův algoritmus nepotřebuje konstruovat Voroného oblasti ani vypočítávat jejich těžiště. A ačkoli nepracuje přímo s Voroného diagramem, McQueenova metoda po dostatečném počtu iterací konverguje k CVT. Tato konvergence však probíhá pomalu, jelikož při každé iteraci tohoto algoritmu dochází k přesunu právě jednoho bodu [7] [18].



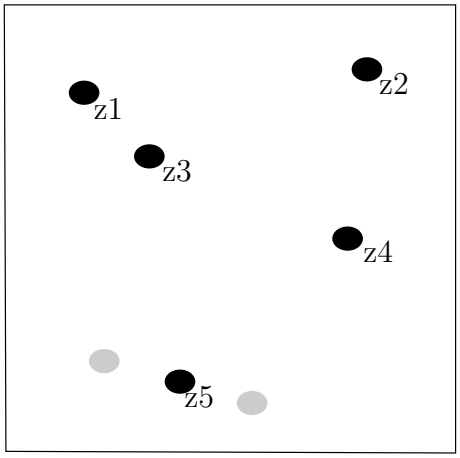
Obrázek 4.2: Zvolená oblast s množinou bodů



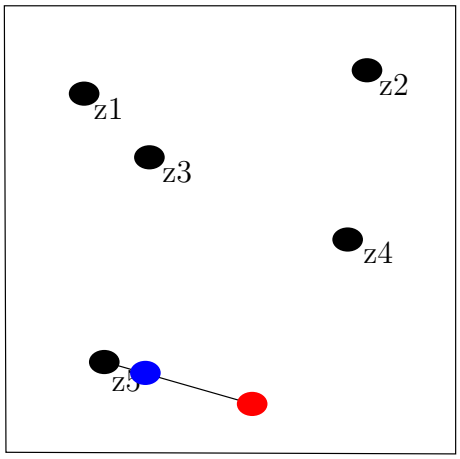
Obrázek 4.3: Generování náhodného bodu



Obrázek 4.4: Výpočet nové pozice bodu



Obrázek 4.5: Přesun bodu na nově určenou pozici



Obrázek 4.6: x-tý přesun bodu

5 Navržené řešení

Následující část práce se zabývá výběrem vhodného algoritmu pro úpravu rozložení bodů v rovině, a to na základě nastudované teorie zabývající se touto tematikou. Dále je zde popsána implementace zvoleného algoritmu a spolu s ní způsob, jakým je měřena kvalita výsledků.

5.1 Výběr algoritmu

Pro řešení zadané problematiky jsem zvolil McQueenův algoritmus, protože jsem o něm nenašel přílišné množství informací ani testů zaměřených na jeho výkon a zajímalo mě jeho chování. Na problémy s rozmístěním bodů v rovině jsou častěji užívané jiné metody, jako například v kapitole 4.1 vysvětlený Lloydův algoritmus.

Oproti ostatním algoritmům nepotřebuje McQueenův pro svůj běh průběžně provádět celkovou analýzu plochy, jeho výpočty nejsou závislé na tvorbě Voroného diagramu. Nevýhodou tohoto algoritmu může být vliv náhody, podle které jsou body přesouvány. Tato náhoda může jak proces urychlit, tak výrazně navýšit potřebný počet iterací, kterými musí tento algoritmus projít, než se dostatečně přiblíží k výslednému CVT. Ačkoli jsou tedy iterace na rozdíl od kupříkladu více využívaného Lloydova algoritmu jednodušší, počet iterací McQueenova algoritmu potřebných pro dosažení požadovaného výsledků obvykle bývá mnohonásobně vyšší.

Experimenty na zvoleném algoritmu mají zjistit, jakých výsledků je možné s tímto algoritmem dosáhnout, kolik iterací je zapotřebí pro dosažení dobrých výsledků a zda se může situačně vyplatit jeho využití v praxi.

5.2 Způsoby určení kvality výsledků

K určení kvality výsledků běhů algoritmu byly definovány následující veličiny:

Tou první je vzdálenost D_{dif}. Pro její definici uvažujme Voroného diagram $V(P)$ a v něm jeho buňku R_i vytvořenou kolem bodu $p_i \in P$. Dále definujme množinu bodů $N_i \subset P$, přičemž body náležící do této množiny jsou ty, jejichž buňky sdílejí hranu s buňkou R_i. Tyto body z množiny N_i nazývejme sousedními body pro bod p_i. Mezi p_i a každým z jeho sousedů n_j lze určit vzdálenost $d(p_i, n_j)$. Je-li $d(p_i, n_j)$ nejmenší vzdáleností z p_i k jakému-

koliv z bodů množiny N_i, pak je značena jako $d_{min}(p_i, n_j)$. Je-li $d(p_i, n_j)$ největší vzdáleností p_i k jakémukoliv z bodů množiny N_i, pak je značena jako $d_{max}(p_i, n_j)$. Vzdálenost $d_{min}(p_i, n_j)$, která je nejmenší vzdáleností mezi nejbližšími sousedy v celém grafu, označíme jako $D_{min}(p_i, n_j)$ a vzdálenost $d_{max}(p_i, n_k)$, která je největší vzdáleností mezi nejbližšími sousedy v celém grafu, označíme jako $D_{max}(p_i, n_k)$. Pro statistické záznamy výsledků je využíván maximální rozdíl vzdáleností mezi sousedními body definovaný jako:

$$D_{dif} = D_{max}(p_i, n_k) - D_{min}(p_i, n_j) \tag{5.1}$$

Jelikož iterováním McQueenova algoritmu dochází k aproximaci na rovnoměrné rozložení bodů, vzdálenost mezi všemi sousedními body by měla být srovnatelná. Úspěšnost této metody lze tedy pozorovat na minimalizaci hodnoty D_{dif}.

Jako další faktor pro srovnání kvality výsledků je zavedena míra vyplnění známé díry $K_s(a, b)$, kde a a b reprezentují strany díry. Jelikož je v testech poloha díry známá, je pro názornost míra jejího vyplnění definována následujícím vzorcem:

$$K_s(a, b) = \frac{n(a, b)}{N \cdot ab} \tag{5.2}$$

kde $n(a, b)$ je počet bodů v oblasti díry, N je počet bodů na celé pozorované ploše a a, b reprezentuje obsah díry.

Výsledkem tohoto vzorce je hodnota $K_s(a, b) \in \langle 0; 1 \rangle$, která vyjadřuje, nakolik se běh algoritmu přiblížil k optimálnímu zaplnění zvoleného prostoru. V případě, že by se ve zvolené části jednotkové plochy nacházel ideální počet bodů (například v $\frac{1}{4}$ oblasti by se nacházelo 25 % z celkového počtu bodů), výsledek tohoto vzorce se rovná hodnotě 1. Je-li zvolená část plochy existující dírou a oproti okolí se v ní nenacházejí žádné body, pak $K_s(a, b) = 0$.

5.3 Postup přesouvání bodů

V této sekci je popsán postup vytvořeného programu při práci na zadaných datech. Při spuštění program načte data ze souboru, popř. vytvoří množinu bodů, nad kterou bude pracovat, přičemž programem generovaná množina náhodných bodů se řídí rovnoměrným pravděpodobnostním rozložením.

Každý bod je popsán třemi hlavními informacemi: $[x, y]$ souřadnice, počet provedených posunů bodu a seznam nejbližších sousedů. Pro iterování jsou důležité jeho $[x, y]$ souřadnice a počet posunů, vzhledem k práci nad McQueenovým algoritmem. Informace o nejbližších sousedech jsou využity pro zjištění hodnot d_{min} a d_{max}, které jsou definované v podkapitole 5.2. Tyto hodnoty jsou zaznamenány do souboru pro vytvoření statistické informace.

Při běhu algoritmu je vždy vytvořen náhodný bod p_r na zkoumané jednotkové ploše. Poté program projde pole bodů a nalezne bod p_i, který se nachází nejblíže k náhodnému p_r. Navýší se celkový počet posunů bodu a dle McQueenova vzorce 4.2 je následně proveden přesun. Takto probíhá iterování programu, dokud není potřeba zaznamenat průběžný či konečný výsledek iterování do souboru.

Při zaznamenávání statistik je programem nejprve vytvořen Voroného diagram. Jelikož McQueenův algoritmus Voroného diagram nevyužívá, je tento tvořen pouze jako rastrové zobrazení. Kdyby bylo zapotřebí jej použít pro výpočty, nebylo by to pro jeho nepřesnost možné, avšak vzhledem ke způsobu jeho využití je takováto varianta postačující. Rastrový obrázek diagramu sice není vhodný pro výpočty, avšak díky obarvení je možné průchodem barev různým pixelů dohledat nejbližší sousedy každého z pozorovaných bodů. Obrázek diagramu znázorňující stav aktuální iterace je poté zaznamenán do souboru typu .png. Dále jsou na základě dat a informací o sousedních bodech vypočítány statistiky pro aktuální iteraci. Ty jsou nakonec exportovány do souboru. Tento postup se opakuje, dokud nedojde k ukončení programu, přičemž k ukončení může dojít dosažením předem určeného počtu iterací či dosažením zadané přesnosti.

Možnou schopností programu je též jeho ukončení, je-li splněna zastavovací podmínka. Uvažovanou podmínkou je určená maximální vzdálenost D_{dif} definovaná vzorcem 5.1. Uživatel má možnost sám zadat, jak přesné aproximace CVT, resp. jak malé hodnoty D_{dif} si přeje dosáhnout. Dále v praktické části jsou testovány hranice této hodnoty, jakých je program schopen skrze McQueenův algoritmus dosáhnout.

6 Experimenty a výsledky

Hlavní částí práce jsou experimenty prováděné s McQueenovým algoritmem. Hlavními účely experimentů bylo zjistit možnosti, jaké algoritmus nabízí, analyzovat jeho chování v různých situacích a ověřit hranice jeho využitelnosti v problematice zaplňování děr v datech. Výsledky testů jsou zaznamenány v grafech, které vychází z naměřených hodnot zaznamenaných v tabulkách v příloze A.1.

6.1 Technické prostředky

Pro práci jsem využil vlastní zařízení. Jedná se o notebook HP Pavillion Power 17-ab301nc, bez úprav hardwaru, jehož operačním systémem při zpracovávání této práce byl Windows 10. Program byl vyvíjen ve vývojovém prostředí IntelliJ Idea Community Edition 2021.1.3 v programovacím jazyce Java 11.

6.2 Účel a způsob testování

Následující experimenty probíhají na jednotkové čtvercové oblasti $[0, 1]^2$. Jejich účelem je otestování vlivu polohy a velikosti vzniklé díry v datech na kvalitu výsledku algoritmu. Jelikož McQueenův algoritmus postupem iterací dle svého vzorce 4.2 snižuje vzdálenost posunu bodů v závislosti na počtu posunů, část experimentů se zaměří na modifikaci algoritmu, ve které počet iterací řečeného vzorce 4.2 bude modulován vhodně zvolenou konstantou. Po provedení modulace bude aktuální počet posunů anulován. Následkem toho dochází k jevu dále nazývanému *obnovení* algoritmu - tato modifikace by se též dala popsat jako opětovné spuštění algoritmu pro aktuální stav zpracovávané množiny, tedy dochází k zpětnému navýšení váhy bodů a s tímto spojené zvýšení vzdáleností jejich posunů. Experimenty se zaměří na pozorování vlivu těchto obnovení na výsledek. Ve výsledných grafech jsou uvedené hodnoty D_{dif} (popř. K_s) v průběhu iterování.

Na konci zvoleného počtu iterací program navrací hodnoty a na jejich základě je vyhodnocena kvalita výsledku. Mezi pozorovanými hodnotami jsou hodnoty D_{dif} definované vzorcem 5.1. Jsou-li body rozloženy s iterováním rovnoměrněji, hodnota D_{dif} se postupně zmenšuje. Jako výsledek jsou uváděna srovnání D_{dif} v čase spolu s rozdíly výsledků v díře a mimo ni. Ve vý-

sledcích experimentů není uváděna průměrná vzdálenost mezi sousedícími body, jelikož její změna v čase je minimální. Je tomu tak proto, že posune-li se bod směrem od jiného bodu, přiblíží se k dalšímu a průměr vzdáleností se drží téměř na konstantní úrovni. Jako další faktor pro srovnání kvality výsledku je použita míra vyplnění známé díry K_s definovaná vzorcem 5.2. V experimentech je pozorováno, nakolik se iterováním McQueenovým algoritmem výsledky přibližují ideálnímu stavu - hodnotu D_{dif} se snažíme minimalizovat, K_s se snažíme přiblížit hodnotě 1, pokud není řečeno jinak.

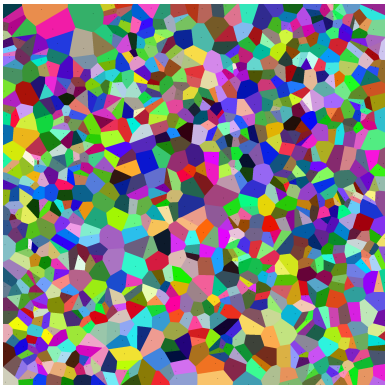
6.2.1 Formát vstupních a výstupních dat

Data o bodech přijímaná programem jsou zapsána v textovém souboru v požadovaném formátu. Na prvním řádku je zaznamenán počet bodů, resp. počet řádků. Zbytek souboru obsahuje $[x; y]$ souřadnice jednoho bodu oddělené mezerou. Pokud jsou na řádku zaznamenány další údaje za souřadnicemi, jsou ignorovány, jelikož pro účely této práce nejsou potřebné. Průběh a výsledky experimentu jsou graficky zaznamenány do souborů typu .png pro náhled. Podstatná data aktuální iterace jsou zaznamenána do textového souboru progress.txt ve složce s výsledky iterací. Výstupní soubor obsahuje na každém řádku záznam o minimální, maximální a průměrné vzdálenosti na celkové ploše a minimální, maximální a průměrné vzdálenosti ve známé díře spolu s hodnotou kvality vyplnění díry v datech. Data jsou uložena pro každou iteraci na jedné řádce, v řečeném pořadí a jsou oddělena mezerou.

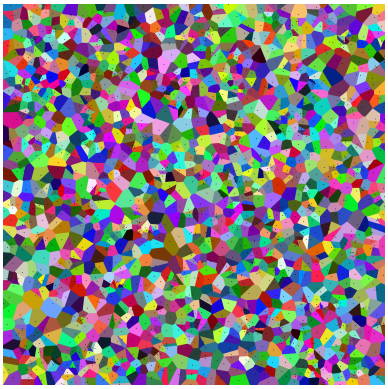
Pro účely této práce bylo vytvořeno 6 datasetů bodů pokrývajících jednotkovou plochu. Body v prvních třech datasetech jsou rozmístěny rovnoměrně pomocí základní javovské knihovny Random [15]. Zbylé tři datasety se řídí beta rozdělením vysvětleném v kapitole 2.3 a byly vytvořeny s pomocí Python knihovny `numpy` [14].

V práci je primárně využit dataset 1 (obr. 6.1) o velikosti 1000 bodů. Tato množina byla zvolena pro testování, jelikož vyšší počet bodů by snižoval názornost výsledků a naopak nižší by je již příliš nízkým počtem mohl negativně ovlivnit. Za účelem sledování možných rozdílů chování při navýšení počtu bodů jsou též v posledním experimentu užity datasety 2 (obr. 6.2) a 3 (obr. 6.3) o velikosti 2000 a 5000 bodů. Datasety obsahující méně bodů nejsou dále testovány, jelikož výskyt nižšího množství bodů je v reálných datech nepravděpodobný. Datasety 4, a 5 vycházejí z beta rozložení, které je popsané v podkapitole 2.3 a oba sestávají z 1000 bodů. Tyto datasety byly zvoleny tak, aby se jejich rozložení od sebe silně odlišovala. Dataset 4 (obr. 6.4)byl navržen tak, aby jeho rozložení bodů na jednotkové ploše bylo možné připodobnit ke 2D Gaussovu rozložení. Dataset 5 je speciálním případem,

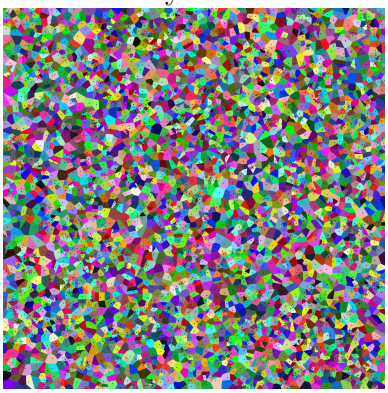
kdy jsou body rozloženy pod křivkou do tvaru písmene U (viz podkapitola 2.3), na jednotkové ploše jsou tedy body rozprostřeny primárně po okraji oblasti (viz obr. 6.5).



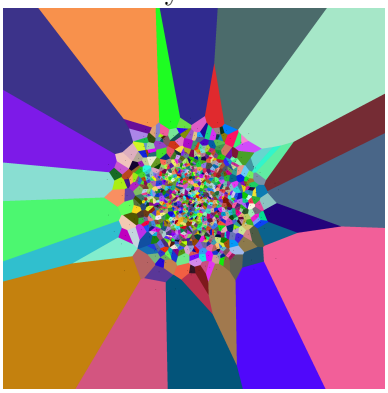
Obrázek 6.1: Ukázka testovací množiny 1000 bodů



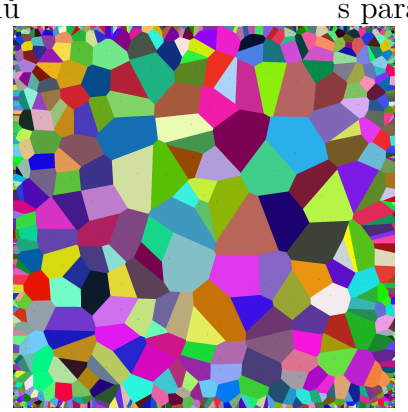
Obrázek 6.2: Ukázka testovací množiny 2000 bodů



Obrázek 6.3: Ukázka testovací množiny 5000 bodů



Obrázek 6.4: Beta rozložení s parametry $\alpha = 20, \beta = 20$

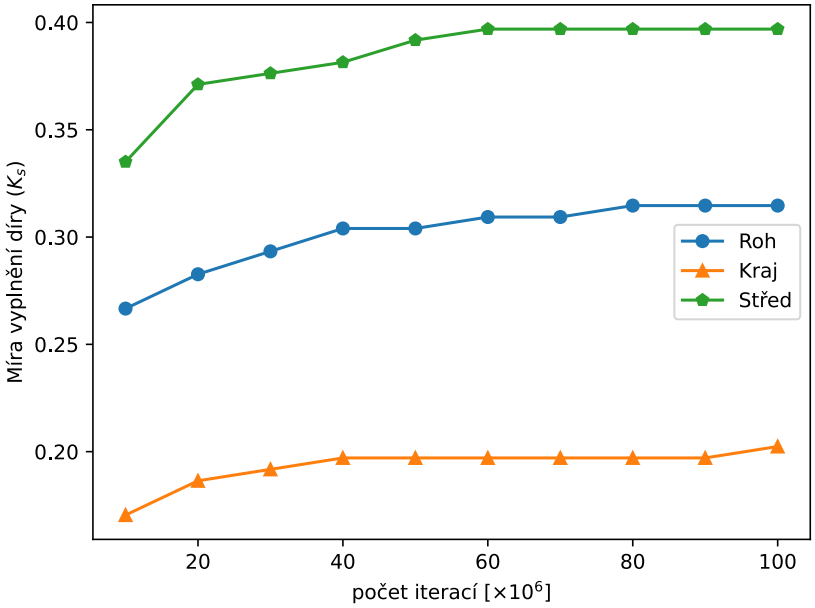


Obrázek 6.5: Beta rozložení s parametry $\alpha = \beta = 0.3$

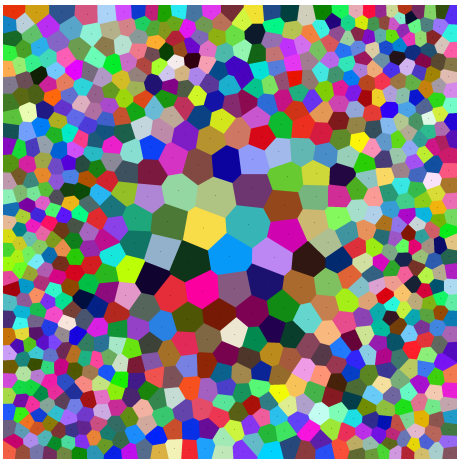
6.3 Vliv pozice díry na její vyplnění

Prvotní experiment ukazuje, nakolik je kvalita vyplnění oblasti závislá na pozici díry v datech. Z vlastností algoritmu vychází, že délka posunu bodu je závislá na množství předchozích přesunů. Jinak řečeno, čím bude díra v datech pro všechny okolní body méně dostupná, tím bude její míra vyplnění K_s nižší, ačkoli vyplnění body bude stále relativně rovnoměrné. Součástí experimentu jsou tři druhy testů, ve kterých je díra vytvořena na 1/4 celkové plochy, a to v jejím rohu, na okraji a ve středu. Následně se algoritmus po 10^8 iterací snaží vzniklou mezeru v datech co nejlépe vyplnit.

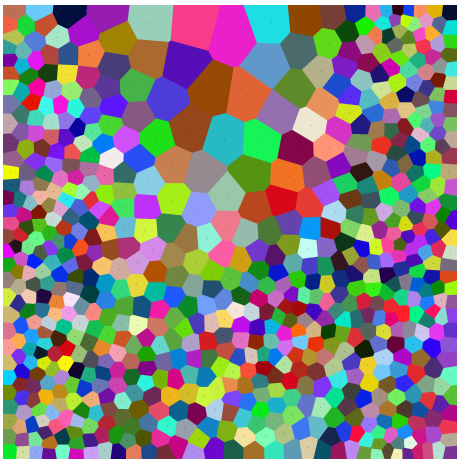
Z výsledků je zřejmé, že části uvnitř díry, které jsou více vzdálené od přesouvaných bodů, se vyznačují nižší hustotou vyplnění než okolí. Z prováděných testů nejlepší vyplnění předvádí díra vzniklá ve středu oblasti. Jelikož se algoritmem přesouvané body nacházely po celém obvodu, byla většina oblasti relativně rovnoměrně vyplněna. Z obrázku Voroného diagramu 6.7 je však zřejmé, že míra vyplnění oblasti původní díry je stále nižší ve srovnání se zbytkem plochy. S vysokým množstvím iterací již dochází pouze k minimálnímu zlepšení.



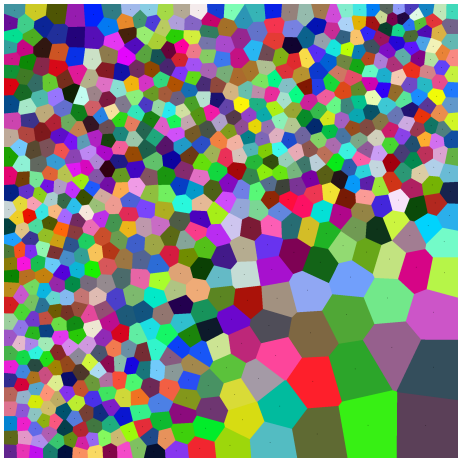
Obrázek 6.6: Experiment 1 - míra vyplnění K_s



Obrázek 6.7: Zaplnění díry - Experiment 1, střed



Obrázek 6.8: Zaplnění díry - Experiment 1, kraj

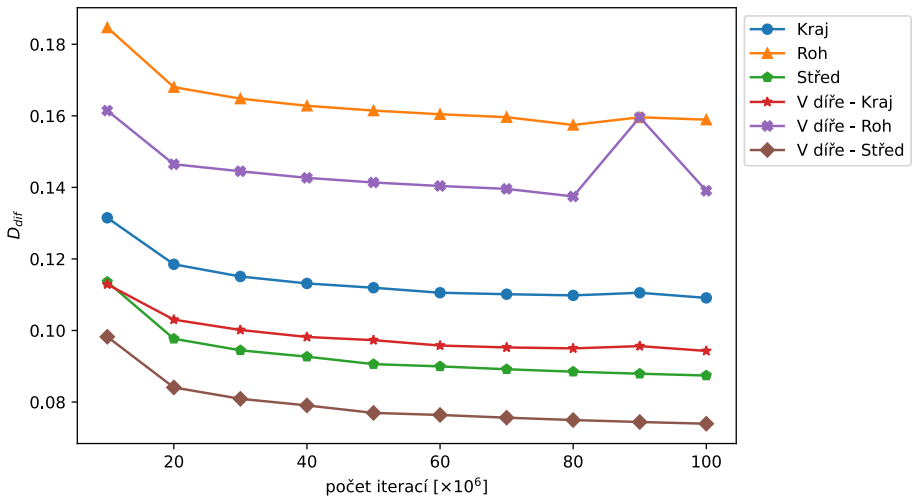


Obrázek 6.9: Zaplnění díry - Experiment 1, roh

U testu s odstraněním bodů na okraji jednotkové plochy $[0, 1]^2$ byl znatelný vliv stěny na míru vyplnění prázdné části, viz obr. 6.8. Jelikož stěna ohraničuje prostor a za jejím okrajem se již žádné body nenacházejí, míra vyplnění se směrem ke středu stěny snižuje, viz obrázek grafu 6.6. Obdobný výsledek ukazuje i graf rozdílů maximálních a minimálních vzdáleností mezi sousedními body ukázaný na obr. 6.10. Z tohoto grafu je však navíc vidět, že rozdíl mezi extrémy je uvnitř díry menší než mimo ni. Tím je ukázáno, že rozložení uvnitř díry je rovnoměrnější než v okolí, avšak dle výsledků viditelných na obrázku grafu 6.6 je zde míra vyplnění oproti okolí nižší. Nejhorší výsledek, co se týče kvality i míry vyplnění, se projevil v testu s dírou v rohu zvolené oblasti, viz obr. 6.9. Jelikož je pozicí díry omezen přístup bodů pouze

na 1/2 hraniční oblasti, rohová část je vyplněna znatelně méně, než oblasti v předchozích testech.

Nakonec experiment ukazuje, že zvolený počet iterací ke konci měl pouze minimální vliv na posuny bodů. Dle vztahu pro výpočet průměrného posunu bodu, po 10^8 iteracích se již průměrný bod přesouvá pouze o zlomek původního posunu, což téměř anuluje vliv na výsledek. Navíc je možné logicky odhadnout, že body, které mají na vyplnění větší vliv, byly přesunuty vícekrát, tedy vzdálenost jejich posunu bude oproti průměru navíc značně snížena.

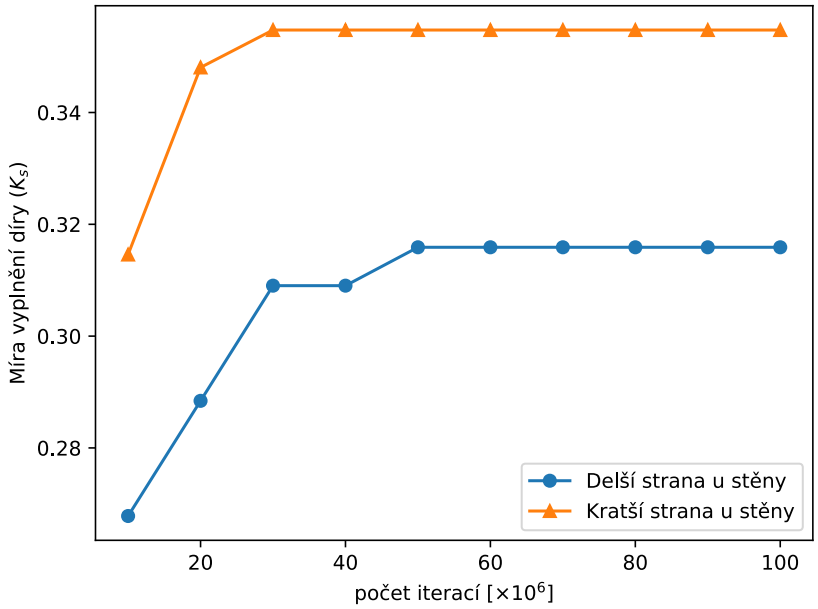


Obrázek 6.10: Experiment 1 - rozdíl max-min vzdáleností mezi sousedními body D_{dif}

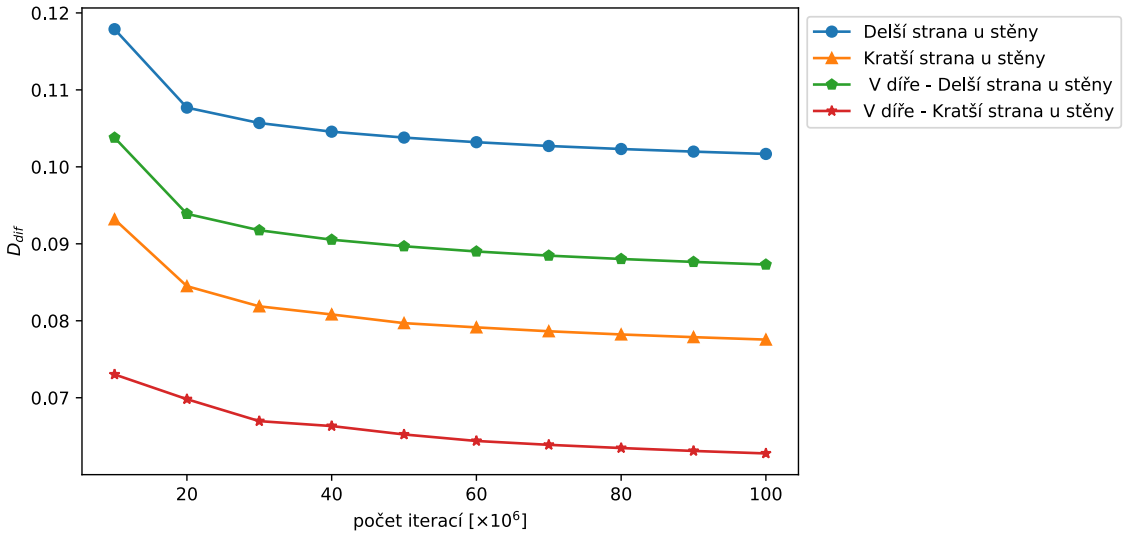
6.3.1 Vliv stěny na vyplnění

Pro doplnění a potvrzení experimentu byla dále vytvořena obdélníková díra, kde $a = 0,6$ a $b = 0,3$. Ta byla přiložena ke stěně jednotkové oblasti ve dvou testech, a to svou kratší a následně delší hranou. Výsledek měl potvrdit, že čím více díra sousedí s hranicí prostoru, tím náročnější je její rovnoměrné vyplnění.

Výsledky testu (viz obr. 6.11 a 6.12) ukazují, že míra vyplnění obdélníkové prázdné oblasti je mnohem vyšší, sousedí-li s okrajem jednotkové plochy svou kratší stěnou. Je-li tedy prázdný prostor dobře přístupný pro okolní body, je ve finále i rovnoměrněji vyplněn.



Obrázek 6.11: Experiment 1 - míra vyplnění K_s u obdélníkové díry



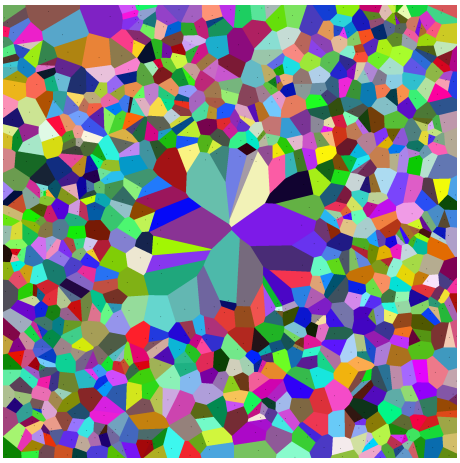
Obrázek 6.12: Experiment 1 - rozdíl max-min vzdáleností mezi sousedními body D_{dif} v obdélníkové díry

6.4 Vliv velikosti díry

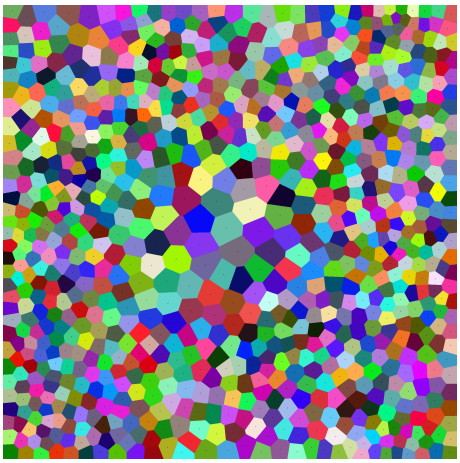
Z předchozího experimentu vycházelo, že nejen pozice díry, ale také její velikost, resp. vzdálenost nejzazších částí od bodů mimo díru, ovlivňuje hustotu vyplnění prostoru. Víme-li, že díra o velikosti 1/4 jednotkové plochy během 10^8 iterací není vyplněna na stejnou průměrnou hustotu zaplnění jako její okolí, je možné předpokládat, že pro vyšší hustotu zaplnění je zapotřebí snížit rozsah prázdné oblasti.

Dále je tedy testována čtvercová díra s délkou hrany $0,3$, voz obr. 6.13. Vytváření menších děr nebylo testováno, jelikož obdobné se již vyskytují v základní množině dat a jsou zde dostatečně dobře vyplňovány. Díry jsou v testech umisťovány do středu jednotkové plochy, jelikož na základě předchozích testů je možné chování za krajních podmínek již dostatečně predikovat.

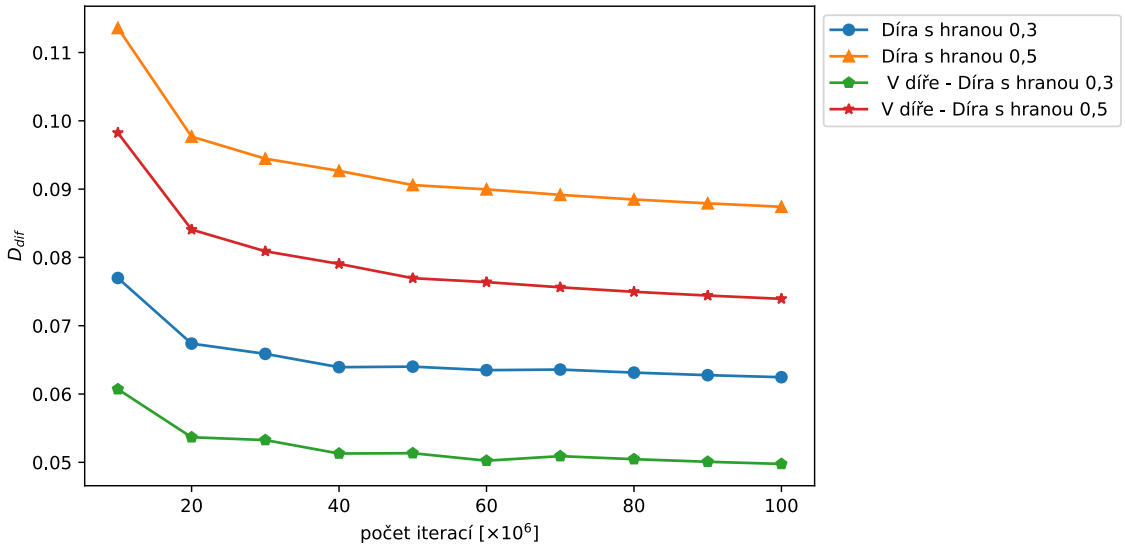
Již z obrázků Voroného diagramů 6.14 a 6.7 je zřetelné, že menší díra je vyplněná lépe než díra o délce hrany $0,5$ zvolená z předchozího experimentu. Porovnáme-li výsledky děr, je z grafu 6.15 patrné, že rozdíl vzdáleností D_{dif} mezi sousedními body je u grafu s menší dírou nižší, dochází zde tedy k rovnoměrnějšímu vyplnění. Graf 6.16 navíc přidává podklad pro tvrzení z předchozího experimentu - jelikož je menší díra okolním bodům přístupnější, její K_s je značně vyšší než u díry větší.



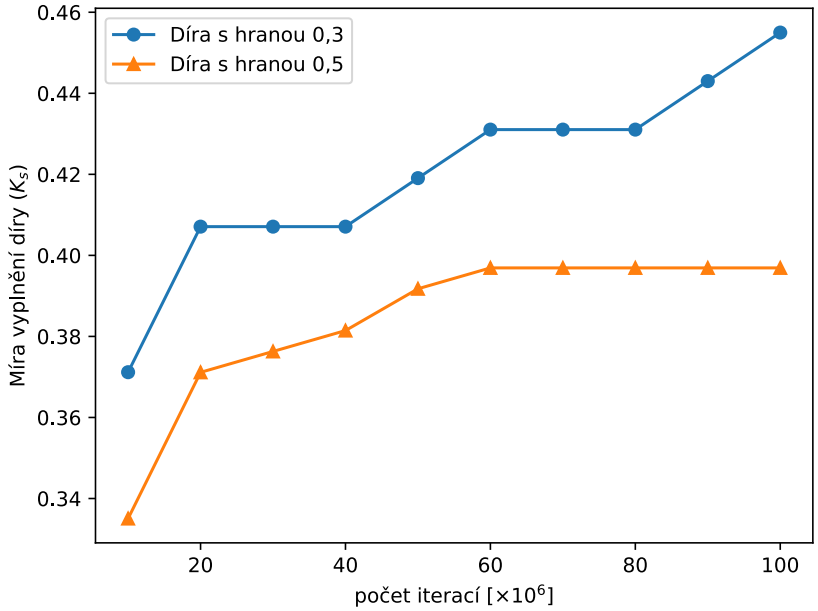
Obrázek 6.13: Zaplnění díry - Experiment 2 - díra o straně $0,3$, začátek



Obrázek 6.14: Zaplnění díry - Experiment 2 - díra o straně $0,3$, po 10^8 iteracích



Obrázek 6.15: Experiment 2 - rozdíl max-min vzdáleností D_{dif} mezi střední a malou dírou

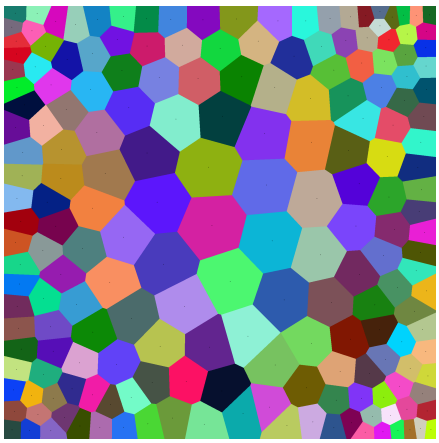


Obrázek 6.16: Experiment 2 - rozdíl míry vyplnění K_s mezi střední a malou dírou

6.4.1 Test s velkou dírou

Jako součást tohoto experimentu byl zařazen i test algoritmu s použitím extrémně velké díry. Z předchozích experimentů vyplývá, že díra bude vyplněna rovnoměrně, avšak hustota jejího zaplnění oproti okolí bude výrazně nižší. Tento test měl za úkol předvést, zda při malém množství bodů na velké ploše budou všechny dostupné body rozmístěny rovnoměrně po celém prostoru. Pokud by tomu tak bylo, v praxi by nemusela být algoritmu poskytnuta celá množina, kde má být vyplněn prázdný prostor, ale pouze oblast okolo díry. Tím by byly přesuny omezeny pouze na nezbytné body a ty, jejichž stabilita v prostoru je z jakéhokoli důvodu klíčová, by nemusely být nijak zvlášť hlídány.

Výsledky tohoto testu dle diagramu 6.17 však tuto myšlenku vyvrátily. Při testování s dírou o délce hrany $0,9$ výsledky ukázaly, že do prázdné oblasti je přemísťovaná pouze část z okolních bodů a přesun ostatních je minimalizovaný. Díra je tedy zaplněna rovnoměrně, ale z pozorování diagramu 6.17 je jasné, že množství bodů vyplňující tuto oblast je oproti okolí stále značně menší, než by bylo zapotřebí. Tento test není nijak graficky přirovnáván k testu s menší dírou, jelikož výsledky by kvůli extrému o ničem nevypovídaly. Míra vyplnění takovéto díry bude vysoká, jelikož počet zbylých bodů je natolik nízký, že jejich část zaplňující díru bude ku celkovému zbylému počtu výrazná. Rozdíly vzdáleností též přinášejí neadekvátně extrémní výsledky, jelikož části bodů v rozích se přesouvají směrem ke středu oblasti minimálně, tedy jejich vzdálenosti jsou extrémně podprůměrné a vzdálenosti blíže ke středu jsou na druhou stranu extrémně velké. Nakonec tedy výsledky tohoto testu potvrzují, že McQueenův algoritmus neodvádí příliš dobrou práci na velkých dírách.

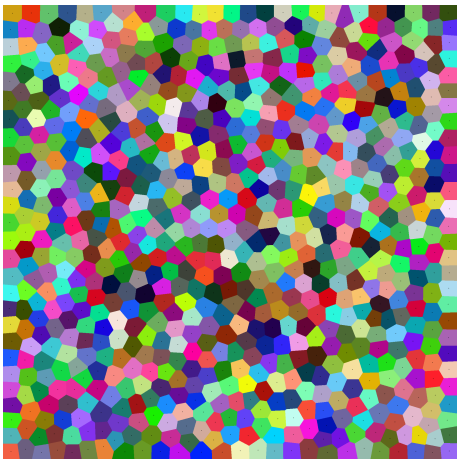


Obrázek 6.17: Zaplnění díry -Experiment 2 - díra o straně $0,9$, 10^8 iterace

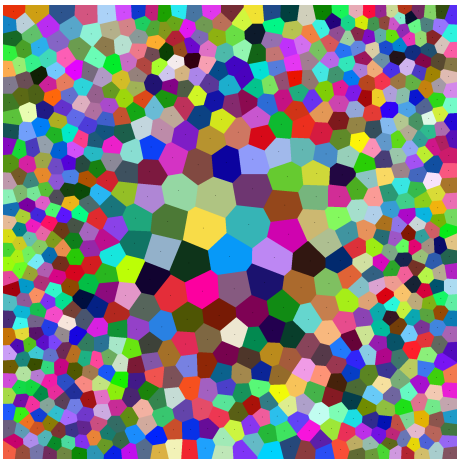
6.5 Experiment s obnovováním algoritmu

Dle dosavadních výsledků je zřejmé, že algoritmus není schopen zaplnit díru obdobnou mírou výskytu bodů, jakou disponuje zbytek oblasti. Je tomu tak pravděpodobně proto, že body z nejbližšího okolí se do díry dostanou, avšak jejich zarovnávání na ploše snižuje dle McQueenova vzorce délku jejich dalších posunů. Po delším běhu algoritmu se body již mohou posouvat pouze o nepodstatné vzdálenosti a k srovnatelnému vyplnění díry s jejím okolím v praktickém čase nedochází. Tento experiment zjišťuje, zda je možné takovýto nedostatek vyřešit obnovením posunu bodů na počáteční vzdálenosti. Postup je tedy takový, že po milionu iterací dojde k obnově běhu algoritmu a počet započítaných posunů každého bodu dle vzorce 5.2 se anuluje. Milion byl zvolen jakožto počet iterací, ve kterém by teoreticky mělo pro tisíc bodů dojít k tisíci posunům každého z nich. Míra posunu je tedy již dosti snížena a možnosti přesunu bodů z větších vzdáleností do díry jsou limitovány. Obnovou má být tedy především řešena míra vyplnění prázdné oblasti.

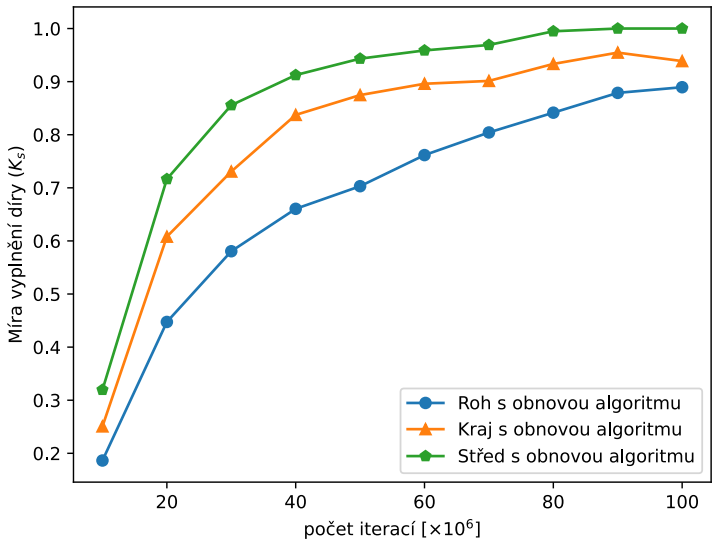
Výsledná kvalita vyplnění je dle grafu 6.20 znatelně vylepšena. Oproti výsledkům bez posunu (srovnáme-li na první pohled diagram 6.18 s diagramem 6.19) se míra vyplnění zlepšila za testovaný počet iterací násobně. Ve středu oblasti, kde původní míra vyplnění K_s nedosahovala ani 40 % v průměru, bez obtíží dosahuje 98 % z předpokládaného množství bodů v původně prázdné oblasti. I díra vzniklá v rohu jednotkové oblasti, jejíž míra vyplnění pouze těsně přesáhla milník 20 % (viz graf na obr. 6.6), se nyní blíží k hranici 90 % kvality vyplnění. Rozdíl průměrných vzdáleností mezi sebou se též znatelně snížil, jak ukazuje graf 6.22. Oproti původním hodnotám v grafu 6.10 dochází opět k rozsáhlému zlepšení. Průměr rozdílů mezi největšími a nejmenšími vzdálenostmi D_{dif}, který v prvním testu dosahoval v nejlepším případě (u konečné hodnoty uvnitř středové díry) vzdálenosti $0,08$, nyní všemi měřenými hodnotami klesl průměrným rozdílem pod hodnotu 0,04. Zajímavostí je, že dochází k zarovnání pod hodnotou $0,03$, avšak průměrné vzdálenosti $0,02$ nikdy nedosáhne.



Obrázek 6.18: Zaplnění díry - Experiment 3 - obnovování algoritmu, 10^8 iterace



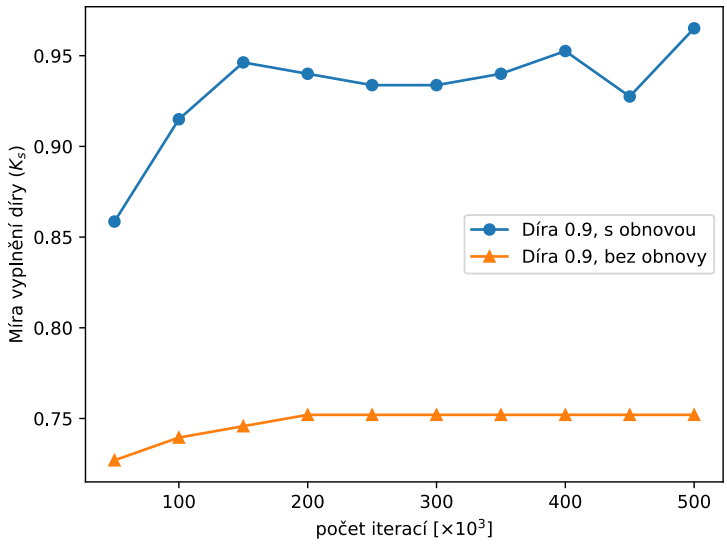
Obrázek 6.19: Zaplnění díry - Experiment 1 - bez obnovování algoritmu, 10^8 iterace



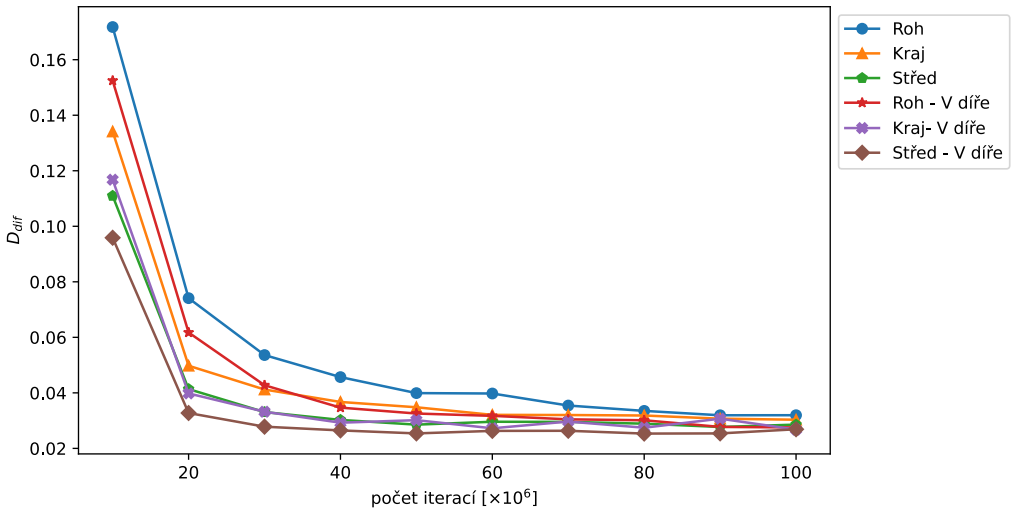
Obrázek 6.20: Experiment 3 - míry vyplnění děr K_s u obnovovaného algoritmu

6.5.1 Test s obnovováním u velké díry v datech

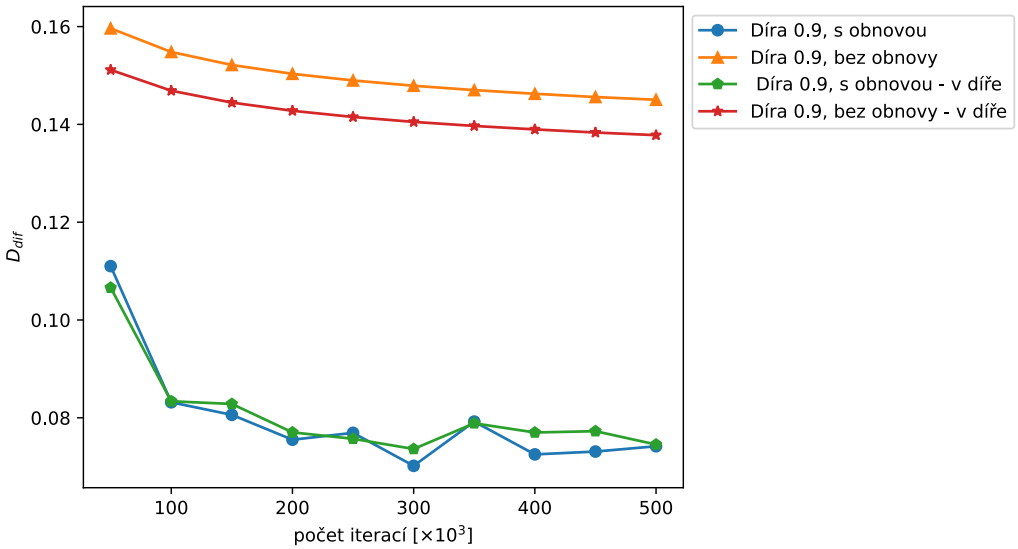
Se znalostí posledního experimentu by bylo vhodné přehodnotit experiment 6.4.1, resp. test s vytvořením velké díry simulující lokální posuny. Díky obnovování umí algoritmus rovnoměrně rozvrhnout body jednotkové oblasti, a tak by bylo vhodné otestovat, zda by tato obnovující vlastnost mohla být využita v užší lokalitě dat na malém množství bodů. Proto následující test



Obrázek 6.21: Experiment 3 - srovnání míry vyplnění K_s pro velkou díru s obnovením a bez

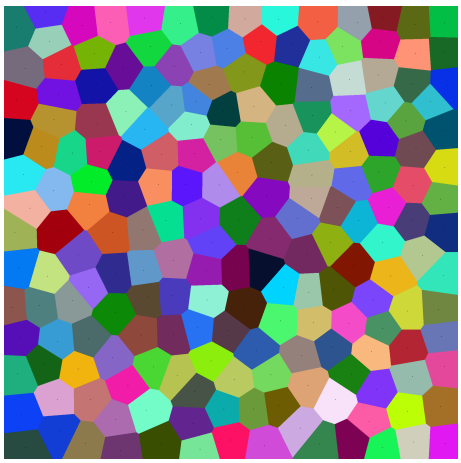


Obrázek 6.22: Experiment 3 - rozdíl max-min vzdáleností D_{dif} mezi sousedními body, při obnovování algoritmu



Obrázek 6.23: Experiment 3 - srovnání rozdílu max-min vzdáleností mezi sousedními body D_{dif}, při obnovování algoritmu u velké díry

opět vytváří velkou díru o délce hrany $0,9$, přičemž jeho úkolem zůstává perfektně rovnoměrné vyplnění. Kvůli nízkému počtu bodů uvnitř byla díra obnovována každých 5000 iterací a výpočet probíhal pouze po $5 \cdot 10^5$ iteracích, jelikož delší běh by s obnovováním u takovéto množiny nedával smysl. Výsledky jsou v tomto případě srovnávány s těmi z experimentu 6.4.1, kdy byl stejný prázdný prostor vyplňován bez obnovování.

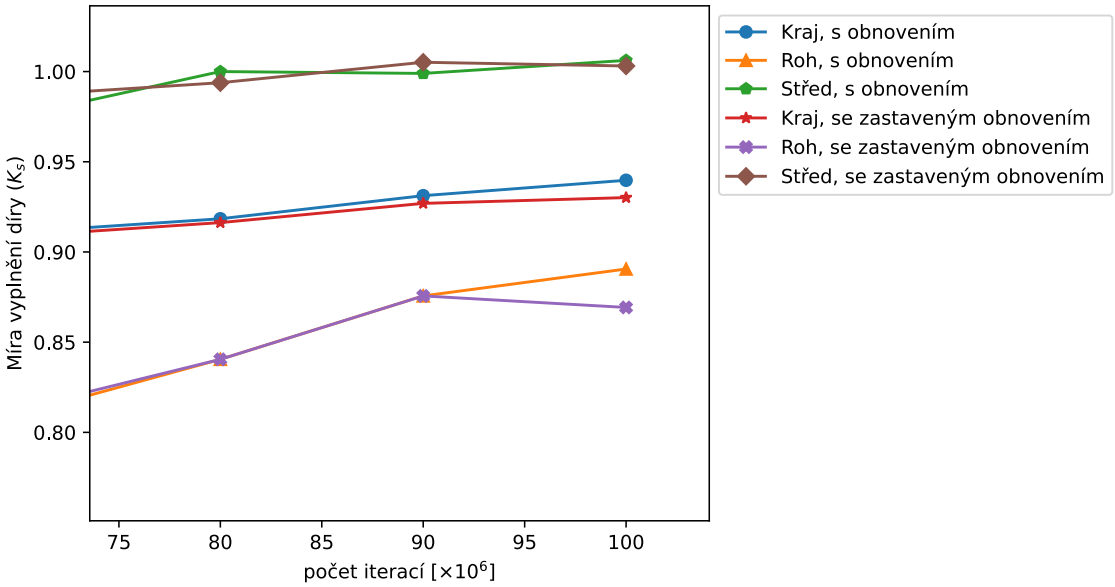


Obrázek 6.24: Zaplnění díry - Experiment 3 - díra o straně $0,9$, $5 \cdot 10^7$ iterace

Z výsledků je jasně zřetelné markantní zlepšení, a to již na diagramu 6.24. Obecná přesnost dle grafu 6.21 i rozdíly mezi maximy sousedních vzdáleností dle grafu 6.23 jsou značně vylepšeny. Pokud by byl tedy použit algoritmus s obnovením na lokalizovanou oblast s nízkým počtem bodů v okolí, dokázal by zde rovnoměrnost rozdělení maximalizovat, byl by tedy již použitelný v praxi.

6.6 Experiment s částečným obnovováním algoritmu

Z předchozího experiment vychází, že obnovováním se McQueenův algoritmus zlepšil ve vyplnění prázdné oblasti a rovnoměrném rozvržení bodů v rovině. Na tomto základu se tedy další experiment zaměří na zvýšení přesnosti vyplnění. Po každé milionté iteraci opět dochází k obnovení algoritmu, přičemž po $8 \cdot 10^7$ iteracích je obnovování zastaveno a algoritmus nyní zarovnává body po dalších $2 \cdot 10^7$ iterací. Očekávaným výsledkem tohoto experimentu byla nejlepší míra rozdělení bodů ze všech experimentů.



Obrázek 6.25: Experiment 4 - srovnání výsledků experimentu 4 s výsledky experimentu 3 - míra vyplněnosti K_s

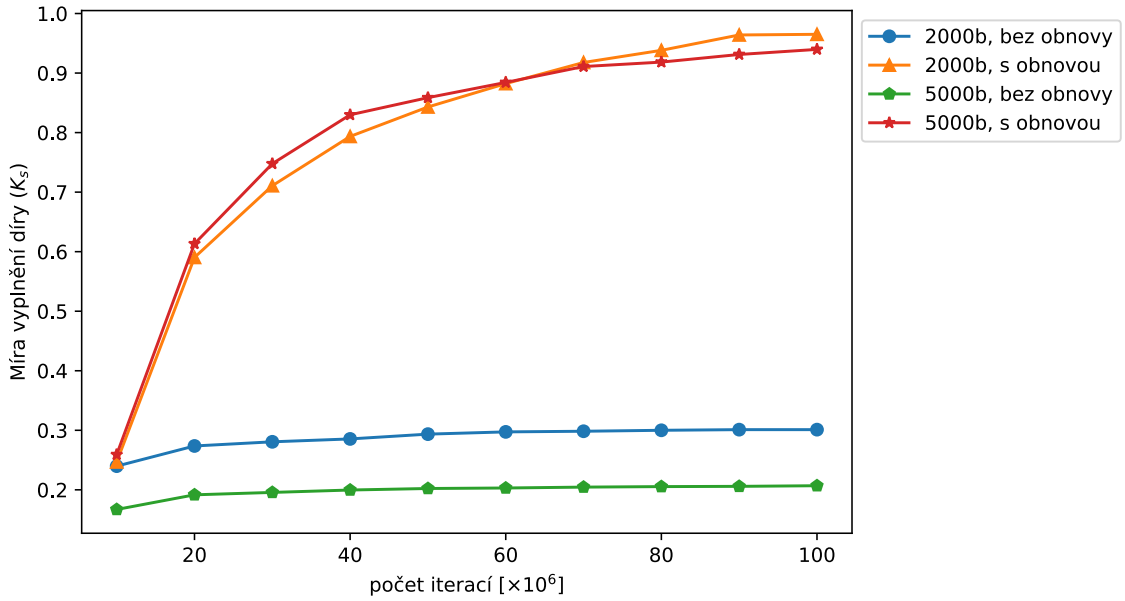
Jelikož průběh do $8 \cdot 10^7$ iterací je extrémně podobný tomu v předchozím experimentu, zaměří se srovnání na zbylý počet iterací. Z grafu 6.25 je

jasné, že na takto malém počtu iterací neměl čistý McQueenův algoritmus bez obnovování dostatečný vliv na větší ovlivnění výsledku. Na základě předchozích experimentů je možné předpokládat, že neobnovovaným iterováním dokáže algoritmus konstantněji dosáhnout lehce zpřesněného výsledku. Dle dat použitých pro graf 6.6 lze odvodit, že užití čistého McQueenova algoritmu ke konci běhu může vést k mírnému zpřesnění výsledku, avšak na základě těchto získaných dat není možné s jistotou potvrdit, že ke zlepšení dojde vždy.

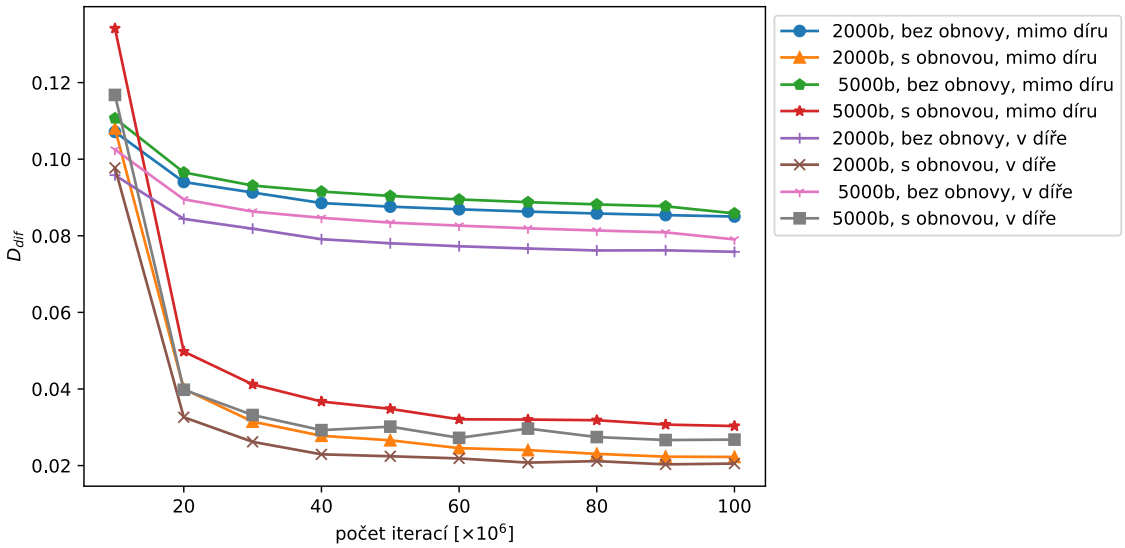
6.7 Experiment s vyšším počtem bodů na ploše

Na množině o počáteční velikosti 1000 bodů došlo k otestování mnoha z vlastností algoritmu a možností jeho využití. Obecně by McQueenův algoritmus neměl vykazovat u množin s více body rozdílné výsledky, ty by měly být srovnatelné s těmi z předchozích testů. Tento experiment se nyní zaměří právě na ověření, že výsledky z předchozích pokusů jsou platné též v množinách s vyšším množstvím bodů a nejedná se o výjimku. Pro toto ověření byly vytvořeny náhodné množiny o velikosti 2000 a 5000 rovnoměrně rozdělených bodů, testován je tedy dvojnásobek a pětinásobek původní množiny.

Dle grafu 6.26 a grafu 6.27 došlo k očekávanému zlepšení při využití obnovovaného algoritmu. Obnovovaný algoritmus konverguje i na větších množinách k násobně lepším výsledkům. Není tedy negativně ovlivněn vysokým množstvím bodů. Zajímavostí je, že i přes navýšené množství bodů se d_{dif} mezi body ku předchozímu obnovovacímu experimentu (viz graf 6.22) příliš neodlišuje. Rozdíly mezi pozicemi díry jsou na větším množství bodů lehce zřetelnější, ale i tak obnovovaný algoritmus dosahuje přesnosti na intervalu $0,02$ až $0,04$. Jak v předchozím experimentu, tak ani zde se nepodařilo s obnovováním algoritmu dosáhnout průměrné blízkosti mezi body rovné $0,02$, vzdálenosti se zarovnávají nad touto hodnotou.



Obrázek 6.26: Experiment 5 - míry vyplněnosti K_s děr u vícebodových množin



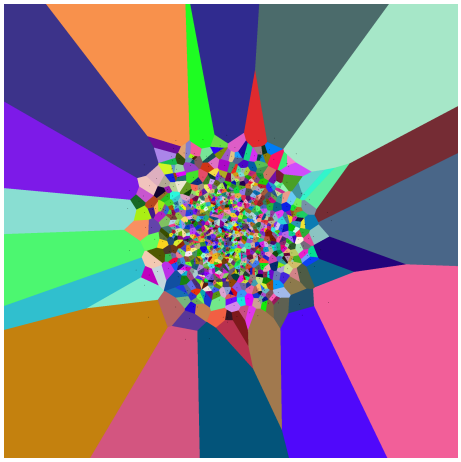
Obrázek 6.27: Experiment 5 - rozdíl max-min vzdáleností mezi sousedními body D_{dif} u vícebodových množin

6.8 Experimenty s beta rozložením

Aby byly testy dostatečně důkladné a nepracovaly pouze s jedním typem rozložení, byly provedeny experimenty pro další dvě množiny vytvořené pomocí beta rozložení. Pomocí parametrů α a β tohoto rozložení byly testované množiny nadefinovány tak, aby připomínaly svým tvarem jiná existující rozložení, přičemž pro každé byl proveden individuální experiment. Následně byly výsledky těchto experimentů srovnány s výsledky experimentů předchozích.

6.8.1 Experiment s rozložením podobným Gaussovu

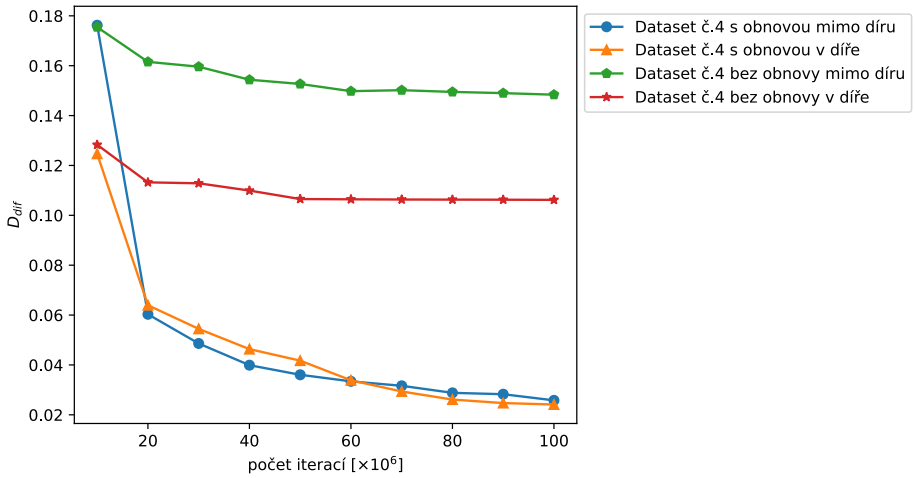
Dataset č. 4 byl navržen tak, aby jeho rozložení bylo podobné Gaussovu. Jak se ukazuje na obr. 6.28, valná většina bodů tohoto rozložení se nachází ve středu oblasti, přičemž celé okolí se dá pro tento experiment považovat za díru v datech. v tomto experimentu byla měřící metoda K_s užita reverzně, tedy sleduje, na kolik se body rozptýlily ze středové oblasti. Pro tento dataset opět dochází ke srovnání obnovované a neobnovované verze McQueenovy metody.



Obrázek 6.28: Beta rozložení s parametry $\alpha = 20, \beta = 20$

Z hodnot měřených v grafu 6.29 je zřejmé, že K_s není zcela vhodné pro měření reverzním způsobem. Předpoklad zněl, že hodnota K_s začne na vysokých hodnotách a bude se k ideálnímu stavu vyplnění (hodnotě 1) přibližovat shora. Jakmile se však body rozptýlily lehce mimo měřenou oblast, došlo k extrémnímu poklesu, který měl vypovídat o rovnoměrnosti rozdělení. To však nebylo pravdivé a dalším iterováním se hodnota K_s přesouváním bodů opět znatelně zvýšila. Až po druhém obdobném skoku dochází k postupněj-

šímu poklesu měřené hodnoty, přičemž po $100M$ iteracích se již průběžně K_s shora přibližuje ideální hodnotě, ačkoli jí stále nedosahuje.

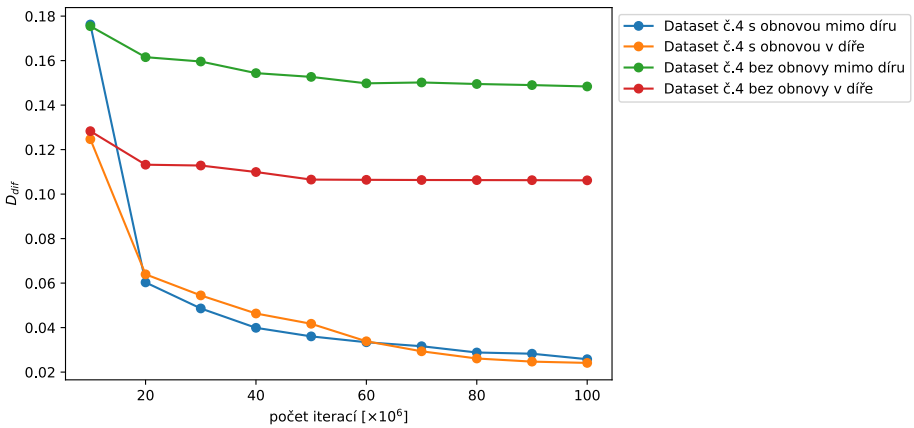


Obrázek 6.29: Experiment 6 - míry vyplnění K_s pro beta rozdělení s obnovou algoritmu a bez

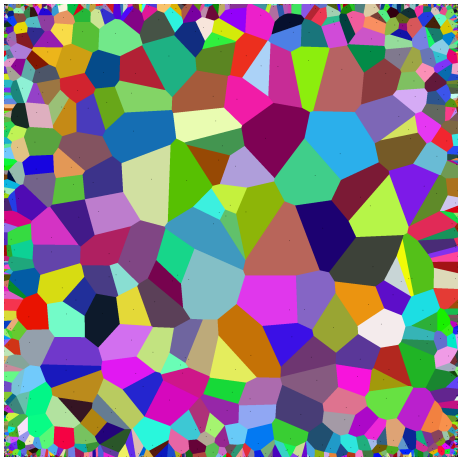
Pro hodnotu D_{dif} je v tomto případě velice důležité, že s jejím měřením začíná program až po vyšším množství iterací - jelikož jsou body ze začátku všechny u sebe, jejich vzdálenosti d_{min} a d_{max} lze považovat za velice podobné, tedy hodnota D_{dif} je zpočátku opět nevypovídající. Avšak po několika iteracích je malé množství bodů odděleno ze skupiny směrem do prostoru a hodnota opět začíná nabývat vypovídající hodnoty. Dle grafu v obr. 6.30 je zřejmé, že výsledky z předchozích experimentů jsou platné i zde - zatímco obyčejný McQueenův algoritmus konverguje k výsledku extrémně pomalu, jeho obnovovaná verze se přibližuje rovnoměrnému rozložení vyšší rychlostí.

6.8.2 Experiment s rozložením do tvaru U

Pro poslední experiment byly pro beta rozložení zvoleny parametry α a β tak, aby tvořily symetrickou křivku ve tvaru připomínajícím písmeno 'U' (viz obr. 2.3 v podkapitole 2.3). Vytvořením náhodných bodů na tomto základě pro $2D$ prostor vznikla množina, ve které se většina bodů nachází na okrajích oblasti, viz obr. 6.31. Pro tento experiment byl tedy opět spuštěn obnovovaný algoritmus. Pro jeho podobnost bylo využito srovnání výsledků s experimentem 3, konkrétně s experimentem vzniku velké díry ve středu oblasti 6.4.1.

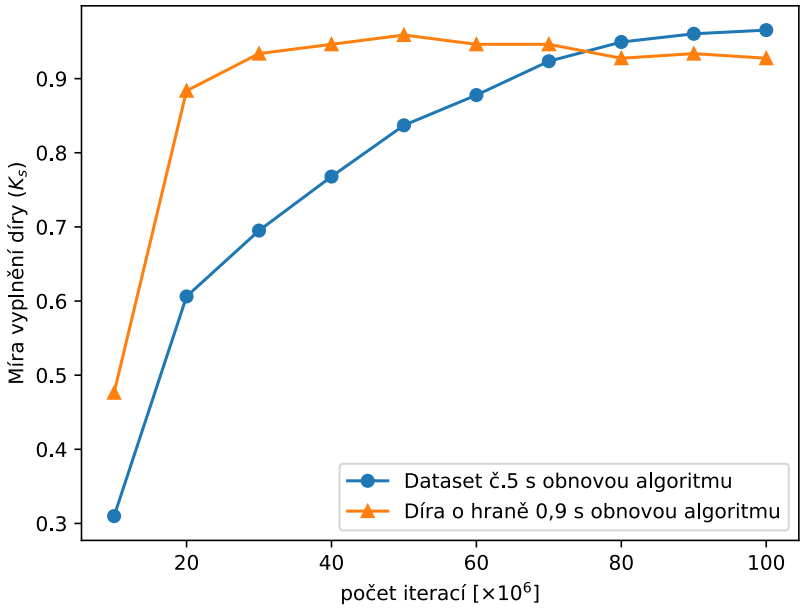


Obrázek 6.30: Experiment 6 - rozdíl max-min vzdáleností mezi sousedními body D_{dif} pro beta rozdělení s obnovou algoritmu a bez



Obrázek 6.31: Beta rozložení s parametry $\alpha = \beta = 0,3$

Dle grafu 6.32 byla volba tohoto srovnání správná. Postup obou množin probíhá velice obdobným způsobem, avšak zlepšení rozložení bodů z datasetu č. 6 probíhá pozvolněji, ne skokově jako u velké díry v rovnoměrném rozložení. Je tomu tak proto, že tento dataset již obsahoval malé množství bodů ve sledované oblasti. Jelikož se ne všechny body nacházely na samotném okraji jednotkové oblasti, prvotní posuny nejsou natolik extrémní, díky čemuž mají body více času na pravidelné rozložení po celém prostoru. Díky tomuto malému množství bodů, které se ve sledovaném prostoru nachází předem je tedy tento způsob měření rázem spolehlivější.



Obrázek 6.32: Experiment 6 - míra vyplněnosti K_s mezi beta rozložením a dírou v normálním rozložením o straně $0,9$

6.9 Časové nároky na výpočet

Časová náročnost iterování tímto algoritmem závisí primárně na datové struktuře pro uložení bodů, jelikož algoritmus opakovaně hledá nejbližší bod. Program byl vytvořen za účelem testování možností využití CVT pro zaplňování děr, přičemž pro tyto účely nebylo zapotřebí optimalizovat jeho rychlost. Implementaci lze tedy považovat za pomalou, navíc s rychlostí běhu závislou na počtu použitých bodů. Při iterování nad množinou o 1000 bodů jedna iterace v průměru trvá kolem 4 μs, vykreslení diagramu a záznam statistik zase 2,5 s. Při práci s množinou o velikosti 5000 bodů jedna iterace v průměru trvala 10 μs a doba potřebná k vykreslení diagramu a záznamu statistik kolem celých 12 s.

Jelikož program není vytvořen za účelem optimálního výkonu, rychlost jeho běhu nebyla měřena v čase, ale v počtu provedených iterací, jak je ukázáno v každém z grafů. Rychlost dosažení výsledku závisí na zvolené přesnosti. Chceme-li dosáhnout nejvyšším možným tempem nejlepšího dosažitelného výsledku za pomoci McQueenova algoritmu, dle závěrů této práce je vhodné použít obnovení posunu bodů algoritmu a při každém obnovení kontrolovat, zda již zvolené přesnosti nebylo dosaženo.

6.10 Shrnutí výsledků experimentů

Experimenty prokázaly, že McQueenův algoritmus je schopen postupným iterováním rovnoměrně rozložit body v rovině s dírou. Míra vyplnění díry je obvykle nižší než u jejího okolí kvůli vlastnostem posunu v tomto algoritmu. Navíc je míra vyplnění díry v datech závislá na velikosti a pozici díry. Pokud mají okolní body k díře nižší přístup, kvalita jejího vyplnění je nižší. Tento algoritmus je tedy vhodnější na vyplnění menších děr, jelikož jejich zaplnění je kvalitativně lepší, než je tomu u větších děr. Obnovováním průběhu McQueenova algoritmu po určitém počtu iterací je možné zlepšit úroveň zaplnění i u větších děr. Toto vylepšení algoritmu také dosahuje vyšší přesnosti umístění bodů. Experimenty neprokázaly, že by nová aplikace klasického algoritmu bez obnovy vylepšila a zpřesnila výsledky. Aniž by tedy bylo zapotřebí algoritmus přímo upravovat, pouze správnou prací s ním je možné dosáhnout kvalitních výsledků v oblasti vyplnění děr v datasetu bodů jejich přemisťováním.

7 Závěr

Cílem této práce bylo implementovat a otestovat algoritmus, který umožní upravovat rozložení bodů v rovině přesouváním. K naplnění tohoto cíle bylo nutné se seznámit s problematikou rozložení bodů a s metodami úpravy rozložení bodů. První část práce se tedy zabývá základními rozloženími bodů, Voroného diagramem a CVT, na kterých jsou postaveny iterační metody pro úpravu rozložení bodů algoritmem přemisťování. Na základě zpracované teorie je navržen McQueenův algoritmus jakožto možné řešení daného problému.

V druhé části je vytvořen program implementující McQueenův algoritmus, konkrétně jeho schopnosti rovnoměrné úpravy rozložení bodů a rovnoměrného vyplnění děr v datech. Na základě testů je potvrzeno, že tento algoritmus je vhodný pro úpravu rozložení bodů a při správné implementaci je též schopný vyplnit díry v datech správným přemístěním zadaných bodů na ploše. Výsledky ukazují, že algoritmus je v základu schopný vyplnit rovnoměrně malé díry v datech, při úpravě nemá problém ani se zaplněním velkých děr. Algoritmus je nejméně vhodný na užití, pokud se díra v datech nachází v rohu oblasti a nejvhodnější, nachází-li se díra v jejím středu.

Podle výsledků testů je tedy McQueenův algoritmus vhodným řešením pro úpravu rozložení bodů v rovině a zaplnění děr, pokud je cílem získat rovnoměrnější rozmístění bodů v rovině. Pro dosažení výsledku je však zapotřebí vysoké množství iterací.

V budoucnu je možné výsledky práce použít pro srovnání s jinými metodami rozložení bodů. Též je možné uvažovat o otestování, zda je tento algoritmus vhodný pro navýšení rovnoměrnosti rozložení bodů ve vícedimenzionálních prostorech.

Literatura

[1] Agoston, M. K. *Computer Graphics and Geometric Modelling: Implementation and Algorithms*. Springer, c2005. ISBN 1-85233-818-0.

[2] Bílková, D. – Budinský, P. – Vohánka, V. *Pravděpodobnost a statistika*. Vydavatelství a nakladatelství Aleš Čeněk, 2009. ISBN 978-80-7380-224-0.

[3] Chaloupka, L. *Aplikace Voroného diagramů v plánování dráhy robotů* [online]. Bakalářská práce. Vysoké učení technické v Brně, 2009. [cit. 2022/05/01]. Dostupné z: `https://dspace.vutbr.cz/bitstream/handle/11012/14290/final-thesis.pdf?sequence=8&isAllowed=y`.

[4] Du, Q. – Wang, D. The optimal centroidal Voronoi tessellations and the gersho's conjecture in the three-dimensional space. *Computers & Mathematics with Applications*. 2005, 49, 9-10, s. 1355–1373. ISSN 0898-1221. doi: 10.1016/j.camwa.2004.12.008. Dostupné z: `https://www.sciencedirect.com/science/article/pii/S0898122105001550`.

[5] Gersho, A. Asymptotically optimal block quantization. *IEEE Transactions on Information Theory*. 1979, 25, 4, s. 373–380. doi: 10.1109/TIT.1979.1056067.

[6] GetRidBug. *How can I create a 3D Gaussian Distribution graph in Illustrator* [online]. GetRidBug. [cit. 2022/06/22]. Dostupné z: `https://getridbug.com/graphic-design/how-can-i-create-a-3d-gaussian-distribution-graph-in-illustrator/`.

[7] Gunzburgerz Max, D. Q. F. V. L. H.-C. J. L. R. V. P. J. W. X. B. J. *Centroidal Voronoi tessellations* [online]. May 2003. [cit. 2022/05/01]. Dostupné z: `https://kmh-lanl.hansonhub.com/uncertainty/meetings/gunz03vgr.pdf`.

[8] Kang, J. M. *Voronoi Diagram*, s. 1232–1235. Springer US, Boston, MA, 2008. doi: 10.1007/978-0-387-35973-1_1461. Dostupné z: `https://doi.org/10.1007/978-0-387-35973-1_1461`. ISBN 978-0-387-35973-1.

[9] Kolingerová, I. *Prezentace k přednáškám Vybrané algoritmické metody* [online]. [cit. 2022/04/15]. Dostupné z: `http://afrodita.zcu.cz/~kolinger/VAM/VAM5.zip`.

[10] Lee, A. *Voronoi Diagram of 5 points* [online]. The Eurographics Association and Blackwell Publishing. [cit. 1.6.2022]. Dostupné z: https://www.geogebra.org/m/Bkn8sw7v.

[11] Marek, L. *Pravděpodobnost*. Praha: Professional Publishing, 2012. ISBN 978-80-7431-087-4.

[12] Martínez-Blanco, M. D. R. et al. *Generalized Regression Neural Networks with Application in Neutron Spectrometry*. 10 2016. doi: 10.5772/64047. ISBN 978-953-51-2704-8.

[13] MedCalc. *Beta distribution functions* [online]. MedCalc Software Ltd. [cit. 2022/06/18]. Dostupné z: https://www.medcalc.org/manual/beta-distribution-functions.php.

[14] NumPyDevelopers. *numpy.random.beta* [online]. [cit. 2022/06/15]. Dostupné z: https://numpy.org/doc/stable/reference/random/generated/numpy.random.beta.html.

[15] Oracle. *Class Random* [online]. Oracle. [cit. 2022/06/15]. Dostupné z: https://docs.oracle.com/javase/8/docs/api/java/util/Random.html.

[16] Pajs. *Hustota rovnoměrného rozdělení* [online]. Wikimedia Commons. [cit. 2022/06/18]. Dostupné z: https://commons.wikimedia.org/wiki/File:Rovnomerne_rozdeleni_hustota.svg.

[17] Pelikán, J. *Náhodné rozmisťování bodů v rovině* [online]. Computer Graphics Group, KSVI MFF, Charles University. [cit. 2022/05/26]. Dostupné z: https://cgg.mff.cuni.cz/~pepca/papers/placementPelikan2014.pdf.

[18] Šedivý, T. *Pomocné programové vybavení a experimenty pro vizualizaci modelů terénu* [online]. Bakalářská práce. Západočeská univerzita v Plzni, 2017. [cit. 2022/06/10]. Dostupné z: http://hdl.handle.net/11025/27682.

Seznam obrázků

A Přílohy

A.1 Tabulky

Měření/poč. iterací	20M iterací	40M iterací	60M iterací	80M iterací	100M iterací
Roh, K_s	0,201	0,208	0,208	0,210	0,213
Kraj, K_s	0.298	0.305	0.309	0.310	0,310
Střed, K_s	0,376	0.386	0,392	0,394	0,396
Roh, D_{dif} mimo díru	0.165	0.161	0.159	0.159	0.151
Kraj, D_{dif} mimo díru	0.115	0.112	0.110	0.109	0.110
Střed, D_{dif} mimo díru	0.094	0.090	0.089	0.087	0.087
Roh, D_{dif} v díře	0.144	0.141	0.139	0.139	0.131
Kraj, D_{dif} v díře	0.100400	0.097	0.095	0.094	0.095
Střed, D_{dif} v díře	0.081123	0.077	0.075	0.074	0.073

Tabulka A.1: Výsledné hodnoty z Experimentu 1

Měření/poč. iterací	20M iterací	40M iterací	60M iterací	80M iterací	100M iterací
b u stěny, K_s	0.296	0.303	0.307	0.307	0.309
a u stěny, K_s	0.345	0.353	0.354	0.356	0.357
b u stěny, D_{dif} mimo díru	0.105	0.103	0.102	0.102	0.101
a u stěny, D_{dif} mimo díru	0.082	0.079	0.078	0.077	0.077
b u stěny, D_{dif} v díře	0.091	0.089	0.088	0.087	0.087
a u stěny, D_{dif} v díře	0.067	0.065	0.063	0.063	0.062

Tabulka A.2: Výsledné hodnoty z experimentu 1.1 - testování obdélníku a = 0,3 b = ,6

Měření/poč. iterací	20M iterací	40M iterací	60M iterací	80M iterací	100M iterací
Díra o straně 0.3, K_s	0.414	0.419	0.423	0.428	0.433
Díra o straně 0.9, K_s	0.736	0.743	0.744	0.747	0.748
Díra o straně 0.3, D_{dif} mimo díru	0.066	0.064	0.063	0.062	0.062
Díra o straně 0.9, D_{dif} mimo díru	0.154	0.150	0.147	0.146	0.145
Díra o straně 0.3, D_{dif} v díře	0.053	0.051	0.050	0.050	0.049
Díra o straně 0.9, D_{dif} v díře	0.146	0.142	0.140	0.138	0.137

Tabulka A.3: Experiment 2, Výsledky pro díry o stranách 0.3 a 0.9

Měření/poč. iterací	20M iterací	40M iterací	60M iterací	80M iterací	100M iterací
Rohová díra, K_s	0.558	0.714	0.804	0.867	0.907
Krajní díra, K_s	0.731	0.856	0.903	0.929	0.949
Středová díra, K_s	0.849	0.964	0.980	0.989	1.008
Roh, D_{dif} mimo díru	0.054	0.039	0.035	0.032	0.030
Kraj, D_{dif} mimo díru	0.042	0.033	0.031	0.030	0.030
Střed, D_{dif} mimo díru	0.034	0.029	0.028	0.027	0.027
Roh, D_{dif} v díře	0.044	0.030	0.029	0.026	0.029
Kraj, D_{dif} v díře	0.033	0.029	0.028	0.027	0.027
Střed, D_{dif} v díře	0.027	0.026	0.026	0.025	0.025

Tabulka A.4: Experiment 3, Výsledky pro obnovovaný algoritmus

Měření/poč. iterací	20M iterací	40M iterací	60M iterací	80M iterací	100M iterací
Bez obnovy, K_s	0.637	0.658	0.674	0.675	0.683
S obnovou, K_s	0.913	0.947	0.937	0.950	0.947
Bez obnovy, D_{dif} mimo díru	0.211	0.199	0.194	0.191	0.188
S obnovou, D_{dif} mimo díru	0.083	0.075	0.070	0.072	0.074
Bez obnovy, D_{dif} v díře	0.200	0.189	0.184	0.181	0.178
S obnovou, D_{dif} v díře	0.083	0.077	0.073	0.076	0.074

Tabulka A.5: Experiment 3, Výsledky při obnovení díry o straně 0,9

Měření/poč. iterací	20M iterací	40M iterací	60M iterací	80M iterací	100M iterací
K_s - Rohová díra	0.560	0.716	0.812	0.867	0.869
K_s -Krajní díra	0.737	0.851	0.903	0.929	0.930
K_s - Středová díra	0.848	0.955	0.983	1.005	1.003
Roh, D_{dif} mimo díru	0.053	0.041	0.037	0.033	0.031
Kraj, D_{dif} mimo díru	0.040	0.034	0.032	0.030	0.028
Střed, D_{dif} mimo díru	0.034	0.029	0.027	0.027	0.024
Roh, D_{dif} v díře	0.042	0.033	0.031	0.029	0.026
Kraj, D_{dif} v díře	0.032	0.030	0.028	0.026	0.024
Střed, D_{dif} v díře	0.030	0.025	0.026	0.025	0.023

Tabulka A.6: Experiment 4, Výsledky zarovnání po obnovení, množina 1000 bodů

Měření/poč. iterací	20M iterací	40M iterací	60M iterací	80M iterací	100M iterací
Bez obnovy, K_s	0.279	0.292	0.297	0.301	0.301
S obnovou, K_s	0.702	0.839	0.910	0.961	0.968
Bez obnovy, D_{dif} mimo díru	0.091	0.087	0.086	0.085437	0.084
S obnovou, D_{dif} mimo díru	0.031	0.027	0.025	0.022	0.022
Bez obnovy, D_{dif} v díře	0.081	0.078	0.076	0.076	0.075
S obnovou, D_{dif} v díře	0.026	0.022	0.022	0.020	0.020

Tabulka A.7: Experiment 5, Výsledky pro 2000 bodovou množinu

Měření/poč. iterací	20M iterací	40M iterací	60M iterací	80M iterací	100M iterací
Bez obnovy, K_s	0.195	0.202	0.204	0.205	0.207
S obnovou, K_s	0.507	0.634	0.716	0.773	0.817
Bez obnovy, D_{dif} mimo díru	0.093	0.090	0.088	0.087	0.085
S obnovou, D_{dif} mimo díru	0.031	0.024	0.021	0.020	0.018
Bez obnovy, D_{dif} v díře	0.086	0.083	0.082	0.080	0.078
S obnovou, D_{dif} v díře	0.027	0.020	0.018	0.017	0.016

Tabulka A.8: Experiment 5, Výsledky pro 5000 bodovou množinu

Měření/poč. iterací	20M iterací	40M iterací	60M iterací	80M iterací	100M iterací
Beta - K_s u rozlož.podobného Gaussovu - K_s	1.614	1.932	1.452	1.208	1.083
Beta - K_s u U rozložení	0.689	0.823	0.918	1.208	1.083

Tabulka A.9: Experiment 6, Výsledky testovaných beta rozložení, míra vyplnění K_s

Měření/poč. iterací	20M iterací	40M iterací	60M iterací	80M iterací	100M iterací
Beta - Gauss, mimo díru	0.047	0.036	0.031	0.027	0.025
Beta - U - mimo díru	0.067	0.058	0.043	0.031	0.0266
Beta - Gauss, v díře	0.055	0.041	0.030	0.024	0.023187
Beta - U - v díře	0.053	0.042	0.037	0.029	0.027

Tabulka A.10: Experiment 6, Výsledky testovaných beta rozložení, vzdálenosti D_{dif}

B Uživatelská dokumentace

V odevzdaných materiálech se nachází program mcqueen.jar. Jeho primárním úkolem je iterování bodů McQueenovým algoritmem. Podle uživatelem připraveného scénáře přijímá více variací vstupních parametrů. Vstupní parametry ovlivňují průběžné i konečné výstupy.

Aby byl program na počítači spustitelný, je potřeba mít nainstalovanou Javu, v nejlepší variantě Javu v.11, ve které byl vyvíjen, a která je tedy doporučena pro zaručení funkčnosti jeho běhu.

B.1 Vstupní parametry programu

Pro svůj běh program přijímá parametry pro příkazovou řádku. Následuje popis možností, jaké program přijme jako parametry pro své spuštění. Varianty jsou dále v textu děleny podle počtu zadaných parametrů.

Zadávané parametry jsou zpracovávány po jednom. Narazí-li program na chybný parametr, vypíše výjimku. Výjimky se liší podle předpokládané chyby. Za výjimkou je též vypsána nápověda s možnostmi, které uživatel může při spuštění využít.

B.1.1 Vstupní parametry potřebné pro spuštění

Následuje seznam vstupních parametrů, které program akceptuje. Sekce jsou vypisované v pořadí možného zadání parametrů, přičemž každá obsahuje všechny varianty, které může uživatel využít.

Parametr 1 - vstupní body pro iterování

První parametr obsahuje cestu k textovému souboru, ve kterém se nachází body, nad kterými má program proběhnout. Body musí náležet do jednotkového čtverce a jejich vstupní formát v souboru musí být řízen charakteristikou zmíněnou v B.2. Pokud by soubor s body nebyl nalezen, dochází k vypsání chybové hlášky [ERROR - File with points not found]. Pokud by formát souboru byl chybný, popř. pokud by byl soubor prázdný, dojde k vypsání hlášky ERROR - Format File.

Druhou variantou prvního parametru je celé kladné číslo (dále zmiňované jako x) omezené velikostí datového typu Integer. Toto číslo reprezentuje počet náhodných bodů, které jsou následně vytvořeny a uloženy do souboru

x_rand_points.txt. Dále program pracuje právě s nimi. Tato množina náhodně generovaných bodů se řídí rovnoměrným spojitým rozdělením.

Parametr 2 - počet iterací / zastavovací podmínka

Pro druhý parametr je očekávána celočíselná hodnota udávající celkový počet prováděných iterací nad množinou bodů. Po provedení celkového počtu iterací dochází k uložení dat na konci běhu. Dle B.2 je vytvořena grafická reprezentace pod názvem

McQueen -x points, iteration no.y.png.

Uživatel zde má též množnost zadat zastavovací podmínku, po které již nemá program pokračovat. Jejím zápisem je s-d, kde d je číselná hodnota vzdálenosti $D_{dif} \in$ (0,1). Když program dosáhne, má za úkol zastavit svůj průběh. Také je zde možnost zadat příkaz ve variantě s-d-e, kde hodnota e za druhou pomlčkou reprezentuje počet iterací, které mají proběhnout, než začne kontrola podmínky zastavení probíhat (např.: s-0.3 − 20000, tedy po 20000 iteracích, začni kontrolovat podmínku a pokud $D_{dif} <= 0.3$, zastav program).

Parametr 3 - požadovaná akce

Tento parametr počítá s tím, že za ním bude následovat množina souřadnic díry, která má být pozorována. Bude-li na třetí pozici zadán příkaz SEEK, program v bodech pouze sleduje vývoj v řečené oblasti. Pokud je zadán příkaz CREATE, na dále zaznamenaných souřadnicích je vytvořena díra v datech a je zaznamenána její existence pomocí výstupních souborů. Program toleruje zadání parametru malými písmeny i variantu zadání pouze prvního písmena. Pokud by byl parametr zadán chybně, program vypíše chybovou hlášku ERROR - wrong action.

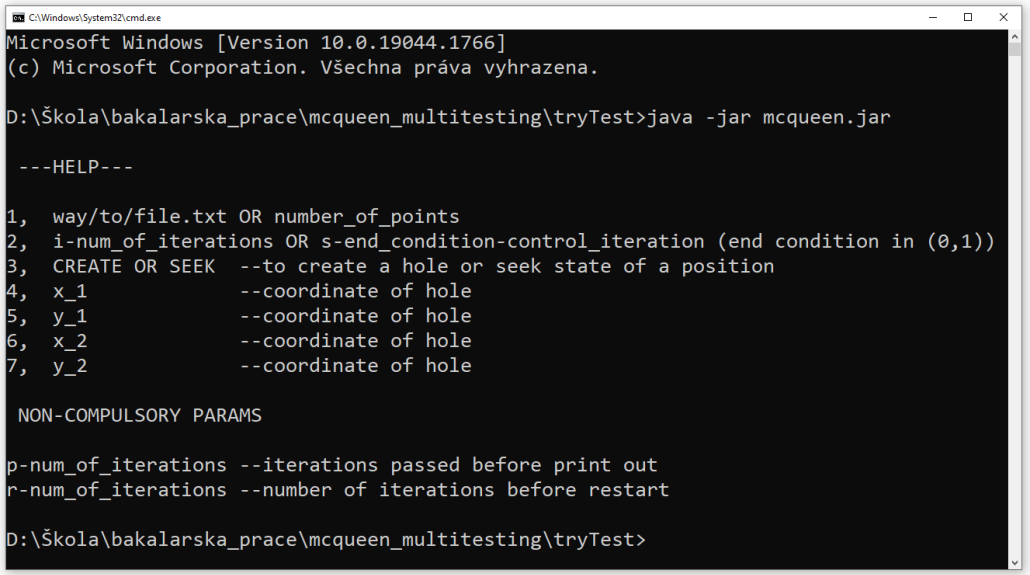
Parametry 4 - 7 - souřadnice sledované oblasti/díry

Následující čtyři parametry obsahují souřadnice sledované oblasti. Výběr počítá se vzorovou obdélníkovou oblastí, přičemž parametry jsou souřadnicemi levého horního a pravého spodního rohu obdélníku. Zadávané souřadnice jsou očekávány v pořadí x_1 y_1 x_2 y_2, přičemž musí náležet do jednotkové oblasti. Pokud by byly souřadnice zadány chybně, program vypíše následující hlášku: ERROR - Wrong parameters.

Parametr 8 a 9 - průběžné informování o výsledcích a obnovení algoritmu

Osmý parametr není povinný k zadání. Pokud zadán nebyl, jednoduše program proběhne celý a na konci zapíše do souborů výsledky. Při zadání osmého parametru, tedy p-a, je po a-té iteraci propočítán a zaznamenán průběžný výsledek do výstupních souborů. Devátý parametr r-y obsahuje informaci, že po z-té iteraci má dojít k obnově posunu bodů v algoritmu.

Pokud jsou zadány oba dva parametry, dochází k záznamu průběžných výsledků vždy při x-té iteraci. Je zde však možnost, že je při zadávání osmý parametr vynechán a je na jeho pozici pouze devátý. V tomto případě při každém obnovení běhu algoritmu na z-té iteraci dojde zároveň k zaznamenání průběžného výsledku do výstupních souborů, dle B.2. V případě chyby na vstupu u osmého parametru program vypíše chybovou hlášku ERROR - print param. v případě, že se chyba objeví na obnovovacím parametru, ať je použit za devátý či osmý, program vypíše chybovou hlášku ERROR - reset param.
Pokud uživatel zadá vstupní parametry správně, pak program provádí iterace do daného počtu iterací, popř. dokud není splněna zastavovací podmínka. Na konci běhu vypíše do konzole uživateli vzkaz $Programm finished$ a ukončí se. Pokud uživatel nezadal žádné parametry, popř. parametry zadal nesprávně, pak mu program vypíše příslušnou chybovou hlášku a spolu s ní nápovědu obsahující jednoduché popisy vstupních parametrů. Ukázka výpisu nápovědy do konzole je vyobrazena na obr. B.1



Obrázek B.1: Výpis nápovědy parametrů do konzole

B.2 Vstupní a výstupní soubory

Jako vstupní soubory program přijímá textové dokumenty obsahující informace o bodech, se kterými bude následně pracovat. Na první řádce souboru je zaznamenán počet bodů dané množiny. Každý další řádek obsahuje x a y souřadnici daného bodu v jednotkovém čtverci. Počet řádků je roven počtu bodů, tedy hodnotě na prvním řádku. Souřadnice x a y jsou seřazeny za sebou v tomto pořadí, přičemž jsou oddělené mezerou. Je-li na řádku dále mezerou oddělená jakákoli informace navíc, program ji ignoruje.

Výstupem programu může být soubor obsahující souřadnice bodů, formát takového souboru odpovídá formátu souboru vstupního a dá se zpětně též jako vstup použít. Průběžným výstupem programu je rastrový obrázek Voroného diagramu. Jedná se pouze o grafickou reprezentaci, ukázku stavu množiny. Z důvodu nepřesnosti ze zaokrouhlení hodnot není vhodné použít takovýto obrázek jakožto vstup pro další výpočty. Finálním výstupním souborem při iterování je textový soubor statistics.txt, kde na každé jeho řádce je jeden průběžný výsledek běhu zaznamenaný po určeném počtu iterací. Údaje na každé řádce jsou oddělené mezerou a zaznamenávané v pořadí: minimální vzdálenost mezi body v okolí dané díry, maximální vzdálenost mezi body v okolí dané díry, minimální vzdálenost mezi body v díře, maximální vzdálenost mezi body v díře, aktuální míra vyplnění díry body určená dle vzorce 5.2.