Uniwersytet Jagielloński
Wydział Matematyki i Informatyki

Piotr Helm
Nr albumu: 1132708

Analiza algorytmów budowania coresetu dla problemu K-means

Praca licencjacka
na kierunku Informatyka Analityczna

Praca wykonana pod kierunkiem
dr Iwona Cieślik
Instytut Informatyki Analitycznej

Kraków, wrzesień 2020

Streszczenie

W pracy przedstawimy stan wiedzy na temat budowania coresetów w kontekście problemu k-means. W szczególności omówimy techniki konstrukcji coresetów takie jak geometryczna dekompozycja oraz losowe próbkowanie z [15]. Celem pracy jest przedstawienie wyników teoretycznych oraz implementacja technik budowania coresetów [15] [4].

0. Spis treści

1. Wstęp

More is more to jedna z podstawowych doktryn związanych z szeroko rozumiałym Big Data. Więcej danych to więcej informacji, które analizujemy licząc na poznanie ukrytych zależności. W erze Big Data skalowalność rozwiązań jest szczególnie ważna, dlatego celem wielu naukowców jest dostarczenie kompromisu pomiędzy szczegółowością informacji a wymaganiami pamięciowymi. Tutaj warto zwrócić uwagę na dużą wartość takich rozwiązań w praktycznych zastosowaniach.

Sketch-and-solve to popularny paradygmat, który zakłada separacje algorytmu agregującego dane od właściwego algorytmu analizującego. Główną ideą jest redukcja danych tak aby rozmiar zbudowanego zbioru nie był zależny od rozmiaru wejściowych danych lub tylko *trochę* od nich zależał. Następnie aplikowany jest właściwy algorytm, który jest mniej zależy od początkowego rozmiaru danych. W rezultacie wykonuje swoją pracę szybciej, a niekiedy nawet lepiej. Dodatkową zaletą jest fakt, że w większości przypadków nie jest konieczna modyfikacja algorytmu analizującego.

Niestety w kontekście tego paradygmatu największym wyzwaniem jest znalezienie kompromisu pomiędzy stratą jakości danych a ich rozmiarem. To jak określamy charakterystykę ważnych informacji jest ściśle zależne od aplikacji danych. Coresety są strukturą algorytmiczną, która ma na celu indentyfikacje takich cech oraz określenie akceptowalnego kompromisu dla różnych funkcji celu.

Mówiąc ogólniej, mamy na wejściu zbiór danych $A \subset U$, gdzie U oznacza jakieś uniwersum, zbiór potencjalnych rozwiązań C pewnego problemu oraz funkcję celu $f : P(U) \times C \to \mathbb{R}_{\geq 0}$, gdzie $P(U)$ oznacza zbiór potęgowy dla uniwersum. Chcemy znaleźć istotnie mniejszy zbiór danych $S \subset U$, dla którego dla każdego potencjalnego rozwiązania $c \in C$ daje wartość $f(S, c)$ *dobrze* aproksymującą $f(A, c)$. A dokładniej:

$$|f(A, c) - f(S, c)| < \epsilon f(A, c)$$

gdzie $\epsilon > 0$.

Algorytmy budujące coresety są aplikowalne do wielu problemów klasteryzacji. W tej pracy skupimy się na konstrukcjach dla problemu k-means, który należy do klasy problemów *NP-trudnych* [1]. Najpopularniejszym algorytmem heurystycznym dla tego problemu jest algorytm Lloyd'a [13]. Z uwagi na to, że złożoność algorytmu istotnie zależy od rozmiaru wejściowych danych, jest on idealnym kandydatem do optymalizacji poprzez odpowiednią konstrukcję coresetu.

Rozdział 2 niniejszej pracy poświęcony jest wpowadzeniu potrzebnych definicji i pojęć. W rozdziale 3 opiszemy budowę *lightweight coresetu* z pracy [4]. Następnie, w rozdziale 4 przedstawimy konstrukcję *coresetu* bazującą na geomtrycznej dekompozycji problemu korzystając z badań [15].

2. Notacja i niezbędne definicje

W tej pracy przyjmujemy, że działamy w przestrzeni euklidesowej w skończonym wymiarze $d > 0$.

Definicja 2.1. Niech $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$. *Normą typu* l_2 nazywamy odwzorowanie $\|\cdot\| : \mathbb{R}^d \to \mathbb{R}$ określone wzorem:

$$\|x\|_2 = \|x\| = \sqrt{\sum_{i=1}^{d} x_i^2}$$

Załóżmy, że mamy dany konkretny problem optymalizacyjny.

Definicja 2.2. *α-aproksymacją* dla problemu optymalizacyjnego nazywamy wielomianowy algorytm, który dla każdej instancji I problemu minimalizacyjnego oblicza rozwiązanie, którego wartość spełnia:

$$A(I) \leq \alpha \cdot OPT(I)$$

gdzie $A(I)$ jest wartością obliczonego rozwiązania dla instancji I, a $OPT(I)$ jest wartością optymalnego rozwiązania tej instancji. Zmienną α nazywamy *współczynnikiem aproksymacji*. Analogicznie możemy zdefiniować *α-aproksymację* dla problemu maksymalizacyjnego. Jest nim wielomianowy algorytm, który dla każdej instancji I spełnia:

$$A(I) \geq \frac{1}{\alpha} \cdot OPT(I)$$

gdzie $A(I)$ i $OPT(I)$ są zdefiniowane tak jak wyżej.

Definicja 2.3. *Wielomianowym schematem aproksymacji* dla problemu optymalizacyjnego nazywamy rodzinę algorytmów $\{A_\epsilon\}$, która dla danego $\epsilon > 0$ zawiera algorytm aproksymacyjny A_ϵ o współczynniku aproksymacji równym $(1 + \epsilon)$ dla problemu minimalizacyjnego albo $(1 - \epsilon)$ dla problemu maksymalizacyjnego. Złożoność algorytmu A_ϵ jest wielomianowa ze względu na rozmiar instancji problemu podanej na wejściu A_ϵ.

W poprzedniej definicji złożoność algorytmu A_ϵ jest niegraniczona względem $\frac{1}{\epsilon}$. Oznacza to, że złożoność algorytmu A_ϵ może być wykładniczo zależna od $\frac{1}{\epsilon}$. W związku z tym wprowadzimy silniejszą definicję 2.3.

Definicja 2.4. *W pełni wielomianowym schematem aproksymacji* jest wielomianowy schemat aproksymacji, dla którego złożoność algorytmu A_ϵ jest wielomianowa od $\frac{1}{\epsilon}$.

Definicja 2.5. *Centroidem* dla skończonego zbioru punktów $x_1, \ldots, x_k \in \mathbb{R}^d$ nazywamy:

$$C = \frac{x_1 + \cdots + x_k}{k}$$

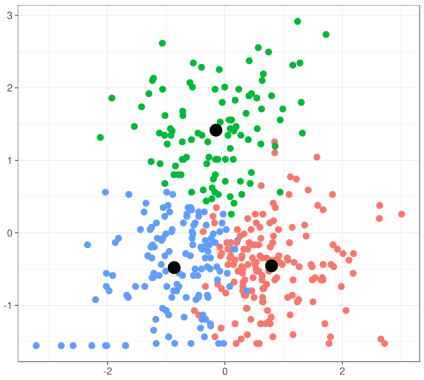
2.1. K-means

Zacznijmy od zdefiniowania problemu, dla którego będziemy analizować konstrukcje coresetów.

Definicja 2.6. *Problem k-means.* Niech X będzie skończonym zbiórem punktów z $\mathbb{R}^d$. Dla danego X chcemy znaleźć zbiór $k \in \mathbb{N}$ punktów $Q \subset \mathbb{R}^d$, który minimalizuje funkcję $\phi_X(Q)$ zdefiniowaną następująco:

$$\phi_X(Q) = \sum_{x \in X} d(x, Q)^2 = \sum_{x \in X} \min_{q \in Q} ||x - q||^2$$

Definicja 2.6 zakłada, że działamy w przestrzeni euklidesowej. Uogólnioną wersję można zdefiniować analogicznie, zamieniając d na odpowiednią funkcję miary w danej przestrzeni.



Rysunek 2.1: Przykład interpretacji problemu k-means. Czarne punkty to centroidy, które stanowią zbiór Q optymalizujący funkcję ϕ.

Problem k-means jest jednym w elementarnych problemów machine learningu. Używamy go m.in w data miningu, segmentacji obrazu czy klasyfikcji danych.

Definicja 2.7. *Problem k-means - wersja ważona.* Niech X_w będzie skończonym zbiórem punktów z $\mathbb{R}^d$ oraz niech w będzie funkcją $w : X_w \to \mathbb{R}_{\geq 0}$. Dla danego X_w oraz funkcji w chcemy znaleźć zbiór $k \in \mathbb{N}$ punktów $Q \subset \mathbb{R}^d$, który minimalizuje funkcję $\phi_{X_w}(Q)$ zdefiniowaną następująco:

$$\phi_{X_w}(Q) = \sum_{x \in X_w} w(x) d(x, Q)^2$$

Wartość funkcji ϕ dla optymalnego rozwiązania oznaczamy $\phi^k_{opt}(X)$ lub OPT. W pracy często będziemy korzystać ze stwierdzenia *optymalne rozwiązanie*,

które oznacza k elementowy zbiór C_{opt}, dla którego wartość funkcji ϕ jest zminimalizowana. Funkcję ϕ w literaturze nazywamy błędem kwantyzacji.

Definicja 2.8. *Klasterem* nazywamy skończony zbioru punktów $x_1, \ldots, x_k \in \mathbb{R}^d$, które łączy pewna wspólna cecha. W naszym kontekście wspólną cechą bedzie przyporządkowanie do tego samego centroidu.

2.2. Coreset

To jak definujemy coreset ściśle zależy od problemu, który optymalizujemy. Zacznijmy od podstawowej definicji coresetu dla problemu k-means.

Definicja 2.9. (ϵ, k)*-aproksymacja zbioru centroidów.* Niech X będzie skończonym zbiórem punktów z $\mathbb{R}^d$. Skończony zbiór $A \subset \mathbb{R}^d$ nazywamy (ϵ, k)-aproksymacją zbioru centroidów, gdzie $\epsilon \in (0, 1)$ oraz $k \in \mathbb{N}$, jeżeli istnieje k elementowy podzbiór $C \subseteq A$, dla którego zachodzi:

$$|\phi_X(C_{opt}) - \phi_X(C)| \leq \epsilon \phi_X(C_{opt})$$

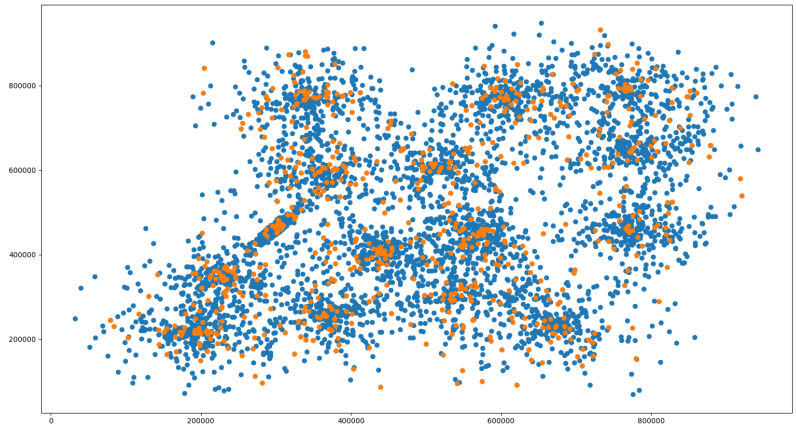
gdzie C_{opt} optymalizuje wartość funkcji ϕ.

Intuicyjnie (ϵ, k)-aproksymację zbioru centroidów możemy rozumieć jako zbiór $A \subset \mathbb{R}^d$, który zawiera k elementowy podzbiór $C \subseteq A$ *dobrze* aproksymujący C_{opt}. Definicja 2.9 pojawia się tutaj z dwóch powodów. Po pierwsze jest ona pewngo rodzaju prekursorem coresetów, a po drugie jest ona konieczna w definicji 2.12.

Definicja 2.10. *Coreset.* Niech X będzie skończonym zbiórem punktów z $\mathbb{R}^d$. Skończony zbiór $C \subset \mathbb{R}^d$ nazywamy (ϵ, k) coresetem, gdzie $\epsilon \in (0, 1)$, jeżeli dla dowolnego $k \in \mathbb{N}$ elementowego zbioru $Q \subset \mathbb{R}^d$ zachodzi:

$$|\phi_X(Q) - \phi_C(Q)| \leq \epsilon \phi_X(Q)$$

Zauważmy, że taka definicja daje nam bardzo moce gwarancje teoretyczne. Wartość funkcji $\phi_C(Q)$ aproksymuje $\phi_X(Q)$ ze współczynnikiem aproksymacji równym $(1 + \epsilon)$ dla dowolnego zbioru k kandydatów Q na rozwiązanie problemu k-means. Jest to na tyle istotne, że w literaturze odróżnimy taką wersję nazywając ją *strong coresetem.* Definicja 2.10 jest wariantem ciągłym, czyli punkty należące do coresetu są dowolnymi punktami przestrzeni, w szczególności nie muszą należeć do zbioru X. W praktyce stosuje się wariant dyskretny, który zakłada, że $C \subseteq X$.



Rysunek 2.2: Pomarańczowe punkty tworzą coreset dla zbioru niebieskich punktów. Parametry: $k = 15$, $\epsilon = 0.1$, $n = 15000$.

Definicja 2.11. *Coreset - wersja ważona.* Niech X_w będzie skończonym zbiórem punktów z $\mathbb{R}^d$ oraz niech w będzie funkcją $w_x : X_w \to \mathbb{R}_{\geq 0}$. Skończony zbiór $C_w \subset \mathbb{R}^d$ wraz z funkcją $w_c : C_w \to \mathbb{R}_{\geq 0}$ nazywamy (ϵ, k) coresetem, gdzie $\epsilon \in (0, 1)$, jeżeli dla dowolnego $k \in \mathbb{N}$ elementowego zbioru $Q \subset \mathbb{R}^d$ zachodzi:

$$|\phi_{X_w}(Q) - \phi_{C_w}(Q)| \leq \epsilon \phi_{X_w}(Q)$$

Tak samo jak w problemie k-means, coreset może być zdefiniowany w wariancie ważonym. Oznacza to, że dla zbioru C z definicji 2.10 musimy zdefiniować funkcję wag $w_c : C \to \mathbb{R}_{\geq 0}$. Nierówność pozostaje bez zmian, ponieważ funkcja ϕ uwzględnia C_w oraz X_w będące zbiorami ważonymi.

Definicja 2.12. *Coreset - weak (w wariancie dyskretnym).* Niech X będzie skończonym zbiorem punktów z $\mathbb{R}^d$. Niech $C \subseteq X$ oraz A jest (ϵ, k)-aproksymacją zbioru centroidów dla zbioru X, gdzie $\epsilon \in (0, 1)$. Jeżeli, dla dowolnego $k \in \mathbb{N}$ elementowego podzbioru $Q \subseteq A$ zachodzi:

$$|\phi_X(Q) - \phi_C(Q)| \leq \epsilon \phi_X(Q)$$

to parę (C, A) nazwiemy słabym coresetem.

Słabość definicji 2.12 wynika z konieczności budowy zbioru A. Intuicyjnie zbiór A pomaga coresetowi w zachowaniu odpowiedniego współczynnika aproksymacji, ponieważ w zbiorze A istnieje *dobry* kandydat na rozwiązanie. W pracy korzystamy wyłącznie z definicji 2.10 oraz 2.11. Z uwagi na to, że nasza praca ma charakter przeglądowy zdefiniowaliśmy wszystkie możliwe warianty coresetów.

3. Lightweight Coreset

Budowa coresetów w kontekście problemu *k-means* ma bardzo długą historię. Za przełomową pracę uznaje się [14], która jako pierwsza przedstawiła w pełni wielomianowy schemat aproksymacji o złożoności $O(n\epsilon^{-2k^2d}\log^k n)$ rozwiązujący problemu k-means, bazując na budowie coresetu.

Wyróżnia się trzy techniki budowania coresetów:

- Geometryczna dekompozycja problemu.
- Losowe próbkowanie zbioru.
- Zawansowane metody algebraiczne.

Pierwsza i trzecia technika cechuje się mocnymi gwarancjami teoretycznymi. Niestety większość rozwiązań jest mało praktyczna i kosztowna czasowo. Losowe próbkowanie w praktyce daje zadowalające wyniki, jednak bardzo często nie daje nam żadnej gwarancji odnośnie optymalności rozwiązania. Autorzy pracy [4] zaproponowali rozwiązanie o nazwie *lightweight coreset*, które w swoich założeniach ma łączyć:

- Prostą implementacje.
- Gwarancje teoretyczne.
- Szybkie działanie oparte na próbkowaniu zbioru danych.

3.1. Lightweight coreset

Zacznijmy od wprowadzenia definicji *lightweight coresetu*.

Definicja 3.1. *Lightweight coreset dla problemu K-means.* Niech $\epsilon > 0$ oraz $k \in \mathbb{N}$. Niech X bedzie n elementowym zbiórem punktów z $\mathbb{R}^d$ wraz ze średnią $\mu(X) = \frac{1}{n}\sum_{i=1}^{n} x_i$. Zbiór $C \subset \mathbb{R}^d$ jest (ϵ, k) lightweight coresetem jeżeli dla dowolnego $k \in \mathbb{N}$ elementowego zbioru $Q \subset \mathbb{R}^d$ zachodzi:

$$|\phi_X(Q) - \phi_C(Q)| \leq \frac{\epsilon}{2}\phi_X(Q) + \frac{\epsilon}{2}\phi_X(\mu(X))$$

Jak możemy zauważyć definicja (3.1) trochę się różni od (2.10). Notacje *lightweight* coresetu możemy interpretować jako relaksację gwarancji teoretycznych zdefiniowanych w (2.10). Wprowadza ona oprócz błędu multiplikatywnego, błąd addytywny.

Głowną motywacją stojącą za konstrukcjami coresetów jest to, żeby rozwiązanie obliczone na tym zbiorze było konkurencyjne z rozwiazaniem optymalnym dla całego zbioru danych. Dlatego w kontekście *lightweight* udowodnimy następujące twierdzenie.

Twierdzenie 3.1. *[4] Niech $\epsilon \in (0, 1]$. Niech X będzie skończonym zbiorem danych z $\mathbb{R}^d$ oraz niech C będzie (ϵ, k) lightweight coresetem dla X. Optymalne rozwiązanie problemu K-means dla X oznaczamy Q_X^*. Optymalne rozwiązanie problemu K-means dla C oznaczamy Q_C^*. Dla takich założeń zachodzi:*

$$\phi_X(Q_C^*) \leq \phi_X(Q_X^*) + 4\epsilon\phi_X(\mu(X))$$

Dowód. Zgodnie z własnością lightweight coresetu otrzymujemy:

$$\phi_C(Q_X^*) \leq (1 + \frac{\epsilon}{2})\phi_X(Q_X^*) + \frac{\epsilon}{2}\phi_X(\mu(X))$$

oraz

$$\phi_C(Q_C^*) \geq (1 - \frac{\epsilon}{2})\phi_X(Q_C^*) - \frac{\epsilon}{2}\phi_X(\mu(X))$$

Wiemy z definicji, że $\phi_C(Q_C^*) \leq \phi_C(Q_X^*)$ oraz $1 - \frac{\epsilon}{2} \geq \frac{1}{2}$. A więc:

$$\phi_X(Q_C^*) \leq \frac{1 + \frac{\epsilon}{2}}{1 - \frac{\epsilon}{2}}\phi_X(Q_X^*) + \frac{\epsilon}{1 - \frac{\epsilon}{2}}\phi_X(\mu(X))$$

$$\leq (1 + 2\epsilon)\phi_X(Q_X^*) + 2\epsilon\phi_X(\mu(X))$$

Zauważając, że:

$$\phi_X(Q_X^*) \leq \phi_X(\mu(X))$$

dowodzimy tezę twierdzenia. □

Twierdzenie 1 dowodzi, że kiedy wartość ϵ maleje koszt optymalnego rozwiązania otrzymanego na zbiorze C zbiega do kosztu rozwiązania otrzymanego na całym zbiorze danych.

3.2. Konstrukcja

Opisana w tym podrozdziale konstrukcja lightweight coresetu oparta jest na próbkowaniu z uwzględnieniem ważności danego punktu. Niech q będzie dowolnym rozkładem prawdopodobieństwa na zbiorze X oraz niech $Q \subset R^d$ będzie dowolnym potencjalnym zbiorem rozwiązań mocy k. Wtedy funkcję ϕ możemy zapisać jako:

$$\phi_X(Q) = \sum_{x \in X} q(x)\frac{d(x, Q)^2}{q(x)}$$

Wynika z tego, że funkcja ϕ może być aproksymowana poprzez wylosowanie m punktów z X korzystając z q i przypisując im wagi odwrotnie proporcjonalne do

q. W szczególności musimy zagwarantować, jednostajność wyboru dowolnego zbioru k punktów Q z odpowiednim prawdopodobieństwem $1-\delta$, gdzie $\delta \in (0, \frac{1}{2}]$. Funkcja q może mieć wiele form, autorzy [4] rekomendują postać:

$$q(x) = \frac{1}{2}\frac{1}{|X|} + \frac{1}{2}\frac{d(x,\mu(X))^2}{\sum_{x' \in X} d(x',\mu(X))^2}$$

Algorytm 1

Algorytm na wejściu otrzymuje zbiór $X \subset \mathbb{R}^d$.
procedure LIGHTWEIGHT
$\mu \leftarrow$ średnia dla X
for $x \in X$ **do**
$q(x) = \frac{1}{2}\frac{1}{|X|} + \frac{1}{2}\frac{d(x,\mu(X))^2}{\sum_{x' \in X} d(x',\mu(X))^2}$
$C \leftarrow$ próbka m ważonych punktów z X, gdzie każdy punkt x ma wagę $\frac{1}{mq(x)}$ oraz jest wylosowany z prawdopodobieństwem $q(x)$
return lightweight coreset C

Pierwszy składnik rozkładu q to rozkład jednostajny, który zapewnia, że każdy punkt jest wylosowany z niezerowym prawdopodobieństwem. Drugi składnik uwzględnia kwadrat odległości punktu od średniej $\mu(X)$ dla całego zbioru. Intuicyjnie, punkty, które są daleko od średniej $\mu(X)$ mogą mieć istotny wpływ na wartość funkcji ϕ. Musimy więc zapewnić, odpowiednią częstotliwość wyboru takich punktów. Jak pokazuje pseudokod, implementacja takiej konstrukcji jest całkiem prosta. Algorytm przechodzi przez zbiór danych jedynie dwukrotnie, a jego złożoność to $O(nd)$, gdzie n to rozmiar wejściowych danych, d to wymiar przestrzeni. Warto zwrócić uwagę na to, że nie mamy zależności od k co jest kluczowe w konkeście praktyczności takiego rozwiązania.

3.3. Analiza

W tym podrozdziale pokażemy, że zaproponowany w poprzedniej części algorytm oblicza lightweight coreset dla odpowiedniego m.

Twierdzenie 3.2. *[4] Niech $\epsilon > 0$, $\delta > 0$ oraz $k \in \mathbb{N}$. Niech X będzie skończonym zbiór punktów z $\mathbb{R}^d$ oraz $C \subseteq X$ to zbiór zwracany przez algorytm dla:*

$$m \geq c\frac{dk\log k + \log\frac{1}{\delta}}{\epsilon^2}$$

gdzie c jest stałą. Wtedy z prawdopodobieństwem co najmniej $1-\delta$ zbiór C jest (ϵ, k) lightweight coresetem dla X.

Dowód. Zaczniemy od ograniczenia $q(x)$, dla każdego punktu $x \in X$. W tym celu zdefinujmy funkcję:

$$f(Q) = \frac{1}{2|X|}\phi_X(Q) + \frac{1}{2|X|}\phi_X(\mu(X))$$

gdzie $\mu(X)$ to średnia zbioru X oraz udowodnimy następujący lemat.

Lemat 3.1. *[4] Niech X będzie skończonym zbiórem punktów z $\mathbb{R}^d$ wraz ze średnią $\mu(X)$. Dla każdego $x \in X$ oraz $Q \subset \mathbb{R}^d$ zachodzi:*

$$\frac{d(x,Q)^2}{f(Q)} \leq \frac{16d(x,\mu(X))^2}{\frac{1}{|X|}\sum_{x' \in X} d(x',\mu(X))^2} + 16$$

Dowód. Z nierówności trójkąta oraz z faktu, że $(|a|+|b|)^2 = 2a^2 + 2b^2$, otrzymujemy

$$d(\mu(X),Q)^2 \leq 2d(x,\mu(X))^2 + 2d(x,Q)$$

Uśrednienie dla wszystkich $x \in X$, implikuje:

$$d(\mu(X),Q)^2 \leq \frac{2}{|X|}\sum_{x \in X} d(x,\mu(X))^2 + \frac{2}{|X|}\sum_{x \in X} d(x,Q)$$

$$= \frac{2}{|X|}\phi_X(\mu(X)) + \frac{2}{|X|}\phi_X(Q)$$

To implikuje, że dla każdego $x \in X$ oraz $Q \subset \mathbb{R}$ zachodzi:

$$d(x,Q)^2 \leq 2d(x,\mu(X))^2 + 2d(\mu(X),Q)$$

$$\leq 2d(x,\mu(x))^2 + \frac{4}{|X|}\phi_X(\mu(X)) + \frac{4}{|X|}\phi_X(Q)$$

Dzieląc powyższą nierówność przez wyżej zdefiniowaną funkcję $f(Q)$ dostajemy:

$$\frac{d(x,Q)^2}{f(Q)} \leq \frac{2d(x,\mu(x))^2 + \frac{4}{|X|}\phi_X(\mu(X)) + \frac{4}{|X|}\phi_X(Q)}{\frac{1}{2|X|}\phi_X(Q) + \frac{1}{2|X|}\phi_X(\mu(X))}$$

$$\leq \frac{2d(x,\mu(x))^2 + \frac{4}{|X|}\phi_X(\mu(X))}{\frac{1}{2|X|}\phi_X(\mu(X))} + \frac{\frac{4}{|X|}\phi_X(Q)}{\frac{1}{2|X|}\phi_X(Q)}$$

$$\leq \frac{16d(x,\mu(X))^2}{\frac{1}{|X|}\sum_{x' \in X} d(x',\mu(X))^2} + 16$$

co kończy dowód lematu. □

Powyższy lemat implikuje, że stosunek pomiędzy kosztem kontrybucji $d(x,Q)^2$ jednego punku $x \in X$ a $f(Q)$ jest ograniczony dla każdego $Q \subseteq X$ przez:

$$s(x) = \frac{16d(x,\mu(X))^2}{\frac{1}{|X|}\sum_{x' \in X} d(x',\mu(X))^2} + 16$$

Niech $S = \frac{1}{|X|}\sum_{x \in X} s(x)$. Zauważmy, że:

$$S = \frac{1}{|X|}\sum_{x \in X} s(x) = \frac{1}{|X|}\sum_{x \in X}\Big(\frac{16d(x,\mu(X))^2}{\frac{1}{|X|}\sum_{x' \in X} d(x',\mu(X))^2} + 16\Big)$$

$$= \frac{1}{|X|}\sum_{x \in X}\Big(\frac{16d(x,\mu(X))^2}{\frac{1}{|X|}\sum_{x' \in X} d(x',\mu(X))^2}\Big) + \frac{1}{|X|}\sum_{x \in X} 16$$

$$= \frac{16\sum_{x\in X} d(x,\mu(X))^2}{\sum_{x'\in X} d(x',\mu(X))^2} + 16 = 32$$

dla każdego zbioru X. Dzięki temu możemy zapisać rozkład q jako:

$$q(x) = \frac{1}{2}\frac{1}{|X|} + \frac{1}{2}\frac{d(x,\mu(X))^2}{\sum_{x'\in X} d(x',\mu(X))^2} = \frac{s(x)}{S|X|}$$

dla każdego $x \in X$. Teraz zdefinujmy funkcję:

$$g_Q(x) = \frac{d(x,Q)^2}{f(Q)s(x)}$$

dla każdego $x \in X$ oraz $Q \subset \mathbb{R}^d$. Zauważmy, że dla dowolnego zbioru $Q \subset \mathbb{R}^d$ zachodzi:

$$\phi_X(Q) = \sum_{x\in X} d(x,Q)^2 = S|X|f(Q)\sum_{x\in X}\frac{s(x)}{S|X|}\frac{d(x,Q)^2}{f(Q)s(x)}$$

$$= S|X|f(Q)\sum_{x\in X} q(x)g_Q(x)$$

Następnie podstawiamy z definicji wartości oczekiwanej:

$$\mathbb{E}_q[g_Q(x)] = \sum_{x\in X} q(x)g_Q(x)$$

dzięki temu przekształcamy ostatnie równanie:

$$\phi_X(Q) = S|X|f(Q)\mathbb{E}_q[g_Q(x)]$$

Następnym krokiem jest ograniczenie wartości $\mathbb{E}_q[g_Q(x)]$. Autorzy [4] nie dowodzą wprost tego ograniczenia, powołując się na inne prace [12]. Dowód jest bardzo skompilowany i wykracza tematyką istotnie poza ramy tej pracy, więc go pomijamy. Korzystamy z finalnego ograniczenia:

$$|\mathbb{E}_q[g_Q(x)] - \frac{1}{|C|}\sum_{x\in X} g_X(x)| \leq \frac{\epsilon}{32}$$

Powyższe ograniczenie jest prawdziwe z prawdopodobieństwem $1-\delta$ dla dowolnego $Q \subset \mathbb{R}^d$ o rozmiarze nie większym niż k. Mnożąc obie strony nierówności przez $32|X|f(Q)$ otrzymujemy:

$$|32|X|f(Q)\mathbb{E}_q[g_Q(x)] - \frac{32|X|f(Q)}{|C|}\sum_{x\in X} g_X(x)| \leq \epsilon|X|f(Q)$$

Niech (C,u) będzie ważonym zbiorem, gdzie dla każdego $x \in C$ definujemy funkcję $u(x) = \frac{1}{|C|q(x)}$. Wynika z tego, że:

$$\frac{32|X|f(Q)}{|C|}\sum_{x\in X} g_X(x) = \sum \frac{1}{|C|q(x)} d(x,Q)^2$$

$$= \sum u(x)d(x,Q)^2 = \phi_C(Q)$$

A więc otrzymujemy:

$$|32|X|f(Q)\mathbb{E}_q[g_Q(x)] - \phi_C(Q)| \leq \epsilon|X|f(Q)$$

$$|\phi_Q(Q) - \phi_C(Q)| \leq \frac{\epsilon}{2}\phi_X(Q) + \frac{\epsilon}{2}\phi_X(\mu(X))$$

co kończy dowód twierdzenia 3.2.

□

4. Geometryczna Dekompozycja

W tej części pracy przedstawimy konstrukcję budowy coresetu bazującą na geometrycznej dekompozycji problemu. Punktem wyjścia były badania [15], na których bazuje opisany w podrozdziale 4.4 algorytm. Zgodnie z nimi budowa coresetu w kontekście problemu k-means to wieloetapowy proces, który jest sekwencją algorytmów z prac: [8] [10] [11] [15]. W rozdziale przyjmujemy następujący porządek analizy konstrukcji:

- W sekcjach 4.1, 4.2, 4.3 opiszemy konstrukcje pomocnicze pochodzące z prac: [8], [10], [11].
- W sekcji 4.4 opiszemy właściwy algorytm z pracy [15].

4.1. Algorytm Gonzalez'a

Pierwszy algorytm, który opiszemy to *Farthest point algorithm* z pracy [8]. Jest to pierwszy algorytm aproksymacyjny rozwiązujący problem k-centrów ze współczynnikiem aproksymacji równym 2. Jego złożoność to $O(nk)$, gdzie n jest liczbą punktów danych na wejściu. Algorytm Gonzaleza jest nam potrzebny, ponieważ w rozdziale 4.2, w którym budujemy kratę wykładniczą istotnie korzystamy z rozwiązania problemu k-centrów.

Na potrzeby tego rozdziału wprowadzimy kilka definicji.

Definicja 4.1. *Problem k-centrów.* Niech X będzie skończonym zbiórem punktów z $\mathbb{R}^d$. Dla danego X chcemy znaleźć zbiór $k \in \mathbb{N}$ punktów $Q \subset \mathbb{R}^d$, który minimalizuje funkcję $\rho_X(Q)$ zdefiniowaną następująco:

$$\rho_X(Q) = \max_{x \in X} \min_{q \in Q} \|x - q\|$$

Definicja 4.2. Podział zbioru punktów na $k \in \mathbb{N}$ zbiorów $B_1, \ldots, B_k$, nazywamy *k-podziałem.*

Definicja 4.3. Zbiory B_i k-podziału nazywamy *klastrami.* W każdym klastrze wyróżniamy jeden punkt nazywając go *środkiem.*

Niech $X \subset \mathbb{R}^d$ będzie zbiorem, na którym chcemy rozwiązać problem k-centrów. Zakładamy, że $|X| > k$ ponieważ w przeciwnym przypadku problem k-centrów jest trywialnie rozwiązywalny. Niech $\rho^k_{opt}(X)$ oznacza optymalne rozwiązanie

problemu k-centrów dla zbioru X oraz niech T^* będzie zbiorem k elementowym realizującym $\rho^k_{opt}(X)$.

Algorytm Gonzaleza rozwiązujący problem k-centrów składa się z fazy inicjalizującej oraz $k-1$ faz powiększających. W fazie inicjalizującej wszystkie elementy zbioru X są przypisana do zbioru B_1, który jest pierwszym klastrem. Jeden z elementów tego zbioru oznaczamy jako (t_1) - środek klastra B_1. Wybór tego elementu jest losowy. Podczas j fazy powiększającej, niektóre elementy z istniejącego podziału na klastry $B_1, \ldots, B_j$ trafiają do nowego zbioru B_{j+1}. Dodatkowo jeden z elementów nowego zbioru będzie oznaczony jako (t_{j+1}) - środek klastra B_{j+1}. Budowę zbioru B_{j+1} rozpoczynamy od wyboru punktu v, który należy do jednego ze zbiorów $B_1, \ldots, B_j$ oraz jego odległość do środka klastra, do którego należy jest największa spośród wszystkich punktów z $B_1, \ldots, B_j$. Taki punkt będzie oznaczony jako (t_{j+1}), czyli jest środkiem klastra B_{j+1}. Każdy punkt, dla którego dystans do v jest nie większy niż dystans do środka klastru, w którym się znajduje zostaje przeniesiony do B_{j+1}.

Algorytm 2

Algorytm na wejściu otrzymuje zbiór $X \in \mathbb{R}^d$.
Niech T będzie szukanym zbiorem k-centrów.
Dla każdego punktu $p \notin T$, algorytm trzyma $neighbor(p)$, czyli najbliższy punkt w T dla p oraz $dist(p)$, czyli odległość od punktu p do $neighbor(p)$.
procedure FARTHEST POINT ALGORITHM
    $T \leftarrow \emptyset$
    $dist(p) \leftarrow \infty$ for all $p \in X$
    while $|T| \leq k$ **do**
        $D \leftarrow max\{dist(p) | p \in X - T\}$
        wybierz v z $X - T$ tak, aby $dist(v) = D$
        add v to T
        zaktualizuj $neighbor(p)$ oraz $dist(p)$ dla każdego $p \in X - T$
return T

Algorytm 2 buduje jakiś k-podział oraz zbiór środków klastrów T. Teraz pokażemy, że dla takiego k-podziału wartość funkcji celu $\rho_X(T)$ jest ograniczona przez $2 \cdot \rho_X(T^*)$.

Niech $L = \rho_X(T)$ oraz niech $x_0 \in T$ będzie punktem, dla którego wartość funkcji $\rho_X(T)$ osiągnęła maksimum. Z konstrukcji k-podziału wiemy, że zbiór $T \cup \{x_0\}$ zawiera $k+1$ punktów, gdzie każdy z nich jest odległy od siebie o co najmniej L. Zauważmy, że przynajmniej dla 2 punktów ze zbioru $T \cup \{x_0\}$, ich najkrótsza odległość do zbioru punktów T^* będzie zrealizowana przez ten sam punkt $t \in T^*$. Zatem przynajmniej 2 punkty odległe od siebie o co najmniej L są przyporządkowane do jednego punktu z T^*. Z nierówności trójkąta wynika, że $\rho_X(T^*) \geq \frac{L}{2}$, co z kolei implikuje, że $2\rho_X(T^*) \geq \rho_X(T)$.

4.2. Konstrukcja kraty wykładniczej

W tym podrozdziale omówimy część pracy [10]. Tematem pracy są konstrukcje algorytmów dla problemów k-means oraz k-median. W kontekście problemu k-means autorzy zaproponowali algorytm rozwiązujący problem k-means bazujący na budowie coresetu, który uzyskuje lepszą złożoność od algorytmu przedstawionego w [14]. Matoušek w swojej pracy [14] przedstawił w pełni wielomianowy schemat aproksymacji, którego złożoność to $O(n\epsilon^{-2k^2d}\log^k n)$. Autorzy [10] poprawili tę złożoność uzyskując $O(n+k^{k+2}\epsilon^{-(2d+1)k}\log^{k+1} n\log^k \frac{1}{\epsilon})$. Schemat konstrukcji jest następujący:

- Obliczmy szybką ale niedokładną aproksymację dla problemu k-means z pewną dużą wartością k.
- Obliczoną aproksymację przekształcamy w (ϵ, k) coreset używając kraty wykładniczej.

4.2.1. Szybka aproksymacja dla problemu K-means

Zacznijmy od pierwszej części. Dokładniej, udowodnimy następujące twierdzenie.

Twierdzenie 4.1. *[10] Dla danego zbioru n punktów $P \subset \mathbb{R}^d$ oraz parametru $k \in \mathbb{N}$ możemy obliczyć zbiór X o mocy $O(k\log^3 n)$, dla którego*

$$\phi_P(X) \leq 20\phi^k_{opt}(P)$$

Czas działania algorytmu to $O(n)$ dla $k = O(n^{\frac{1}{4}})$ oraz $O(n\log(k\log n))$ w przeciwnym przypadku.

Twierdzenie 4.1 w pracy [10] ma trochę inną formę. Ograniczenie jest następujące:

$$\phi_P(X) \leq 32\phi^k_{opt}(P)$$

Z analizy, którą zaraz przeprowadzimy wynika, że współczynnik aproksymacji jest równy 20. Autorzy potrzebowali współczynnika równego 32 z uwagi na inne zastosowanie twierdzenia 4.1 niż nasze.

Niech $P \subset \mathbb{R}^d$ będzie danym na wejściu zbiórem n punktów. Chcemy szybko obliczyć aproksymację dla problemu k-means na tym zbiorze, gdzie rozmiar wynikowego zbioru punktów będzie rzędu $O(k\log^3 n)$.

Definicja 4.4. *Złe/dobre punkty.* Dla zbioru punktów X, punkt $p \in P$ nazywamy *złym* jeżeli

$$d(p, X) \geq 2d(p, C_{opt})$$

gdzie C_{opt} jest zbiórem punktów realizującym $\phi^k_{opt}(P)$. Punkt jest *dobry* jeżeli nie jest zły.

Na początku opiszemy procedurę, która dla danego P, wyznacza zbiór punktów X o mocy rzędu $O(k \log^2 n)$ oraz zbiór $P^{'} \subset P$. Zbiór $P^{'}$ będzie zawierać *dobre* punkty dla zbioru X. Intuicyjnie procedura ma na celu przekształcenie obliczonej 2-aproksymacji problemu k-centrów dla zbioru P tak, aby rozwiązanie było aplikowalne do problem k-means z zachowaniem pewnych gwarancji teoretycznych. Punkty *dobre* nie wymagają od nas dodatowej pracy, ponieważ dla każego punktu $p \in P^{'}$ instnieje punkt $x \in X$, który *dobrze* aproksymuje wartość $d(p, C_{opt})$, a dokładniej $d(p, x) \leq 2d(p, C_{opt})$.

Konstrukcję zbioru X zaczynamy od obliczenia 2-aproksymacji problemu k-centrów dla zbioru P. Niech obliczony zbiór centrów to V. Taką aproksymację dla $k = O(n^{\frac{1}{4}})$ możemy obliczyć w czasie $O(n)$ [9] oraz dla $k = \Omega(n^{\frac{1}{4}})$ w czasie $O(n \log k)$ [7]. Są to wyniki teoretyczne, które zakładamy na potrzebę analizy. W podrozdziale 4.1 przedstawiliśmy algorytm aproksymacyjny z pracy [8], rozwiązujący ten problem w czasie $O(nk)$ dla dowolnego k. Algorytmy o lepszej złożoności są w istotny sposób bardziej skomplikowane, dlatego w naszej implementacji zastosowaliśmy algorytm Gonzaleza. Niech L będzie promieniem takiej aproksymacji, czyli największą odległością pomiędzy punktem $v \in V$ a punktem $p \in P - V$, dla którego punkt v jest centrem. Dla przypomnienia, punkt $v \in V$ będzie centrem dla punktu $p \in P - V$, jeżeli wartość $d(v, p)$ jest najmniejsza spośród wszystkich punktów z V. Ponieważ algorytm z pracy [7] bazuje na algorytmie przedstawionym w podrozdziale 4.1 z pracy [8] to dystans pomiędzy dowolną parą punktów z V wynosi co najmniej L. To implikuje następujące ograniczenia:

$$\left(\frac{L}{2}\right)^2 \leq \phi^k_{opt}(V) \leq \phi^k_{opt}(P) \leq nL^2$$

Dolne ograniczenie argumentujemy tym, że możemy ograniczyć $\phi^k_{opt}(V)$ przez $d(v_1, v_2)^2$, gdzie $v_1, v_2 \in V$. Górne ograniczenie wynika z faktu, że promień aproksymacji jest równy L, zatem dla n elementowego zbioru P wartość funkcji $\phi^k_{opt}(P)$ nie może być większa niż nL^2.

Następnym krokiem konstrukcji jest wylosowanie $\rho = \gamma k \log^2 n$ punktów ze zbioru P, gdzie γ jest stałą, która wynika z analizy, którą zaraz przeprowadzimy. Niech Y będzie zbiorem wylosowanych punktów z P oraz $X = Y \cup V$ będzie zbiorem środków klastrów. W tym kontekście klastrem nazywamy skończony zbioru punktów z $\mathbb{R}^d$, które są przyporządkowane do tego samego środka. Dla $\rho > n$ przyjmujemy $X = P$.

Konstrukcja zbioru X jest stosunkowo prosta. Dużo cięższym zadaniem jest zbudowanie zbioru $P^{'}$, który jest zbiorem *dobrych* punktów dla X.

4.2.2. Konstrukcja zbioru dobrych punktów dla X

Rozpatrzmy zbiór C_{opt}, który jest optymalnym zbiorem centrów problemu k-means dla zbioru P i liczby k. Dla każdego $c_i \in C_{opt}$ tworzymy kulę B_i o środku w c_i. Każda taka kula będzie zawierać co najmniej $\eta = \frac{n}{20k \log n}$ punktów z P. Jeżeli zastosowana do wylosowania zbioru X stała γ jest odpowiednio duża to z wysokim prawdopodobieństwem w każdym B_i jest przynajmniej jeden punkt

z X. Dokładniej:

$$X \cap B_i \neq \emptyset \text{ dla } i = 1 \dots k$$

Niech P_{bad} będzie zbiorem złych punktów z P dla zbioru X. Załóżmy, że dla każdego B_i istnieje punkt $x_i \in X \cap B_i$. Zauważmy, że dla każdego punktu $p \in P \setminus B_i$, dla którego x_i jest najbliższym punktem z X mamy $||p - x_i|| \leq 2||p - c_i||$, ponieważ $V \subseteq X$. W szczególności, jeżeli c_i jest najbliższym punktem z C_{opt} dla punktu $p \in P \setminus B_i$, to taki punkt jest *dobry* dla zbioru X. Zatem z wysokim prawdopodobieństwem jedyne *złe* punkty będą w kulach B_i dla $i = 1, \dots, k$. To implikuje, że z wysokim prawdopodobieństwem liczba złych punktów w P dla zbioru X to co najwyżej $\beta = k\eta = \frac{n}{20 \log n}$.

W takim razie złych punktów nie jest dużo. Mimo tego bezpośrednie wyznaczenie tych punktów jest skomplikowane. Autorzy pracy [10] budują zbiór $P^{'}$ tak aby koszt złych punktów w $P^{'}$ był jak najmniejszy. Koszt punktu p w tym kontekście oznacza jaki wkład ma punkt p w wartość $\phi_{P'}(X)$. Dla każdego punktu w P obliczamy najbliższego sąsiada w X. Niech $r(p) = d(p, X)$ dla każdego punktu $p \in P$. Teraz podzielimy P na zbiory według następującej formuły:

$$P[a, b] = \{p \in P \mid a \leq r(p) \leq b\}$$

A dokładniej:

$$P_0 = P\Big[0, \frac{L}{4n}\Big]$$

$$P_\infty = P\Big[2Ln, \infty\Big]$$

$$P_i = P\Big[\frac{2^{i-1}L}{n}, \frac{2^i L}{n}\Big]$$

dla $i = 1, \dots, M$, gdzie $M = 2\lceil \lg n \rceil + 3$ oraz L to promień aproksymacji. Taki podział możemy wykonać w czasie linowym. Niech P_α będzię ostatnim zbiorem, który zawiera więcej niż $2\beta = \frac{n}{10 \log n}$ punktów. Szukany zbiór zdefiniujemy następująco:

$$P^{'} = V \cup \bigcup_{i \leq \alpha} P_i$$

Chcielibyśmy aby $|P^{'}| \geq \frac{n}{2}$ oraz $\phi_{P'}(X) = O(\phi_{P'}(C_{opt}))$. Teraz udowodnimy, że faktycznie tak zdefiniowane $P^{'}$ spełnia powyższe założenia.

Dowód. Moc zbioru $P^{'}$ jest na pewno równa conajmniej $\Big(n - |P_\infty| - M\frac{n}{10 \log n}\Big)$, gdzie ostatni składnik jest mocą wszystkich zbiorów P_i, które zawierają nie więcej niż 2β punktów i których jest co najwyżej M. Zauważmy, że $P_\infty \subseteq P_{bad}$ oraz $|P_{bad}| \leq \beta$. Zatem:

$$|P^{'}| \geq n - \frac{n}{20 \log n} - M\frac{n}{10 \log n}$$

$$= n - \Big(\frac{n}{10 \log n}\Big)\Big(M + \frac{1}{2}\Big)$$

$$= n - \Big(\frac{n}{10 \log n}\Big)\Big(2\lceil \lg n \rceil + 3 + \frac{1}{2}\Big)$$

$$\geq \frac{n}{2}$$

Jeżeli $\alpha > 0$, to $|P_\alpha| \geq 2\beta = \frac{n}{10 \log n}$. Z uwagi na to jak budujemy $P^{'}$ w najgorszym przypadku wszystkie złe punkty będą w P_α. Wtedy takie punkty będą miały największy wpływ na funkcję ϕ. Niech $Q^{'} = P_\alpha \setminus P_{bad}$. Dla dowolnych punktów $p \in P^{'} \cap P_{bad}$ oraz $q \in Q^{'}$, mamy $d(p, X) \leq 2d(q, X)$, ponieważ $d(p, X) \leq \frac{2^\alpha L}{n}$ oraz $d(q, X) \geq \frac{2^{\alpha-1} L}{n}$. Dodatkowo $|Q^{'}| > |P_{bad}|$, a więc:

$$\phi_{P' \cap P_{bad}}(X) \leq 4\phi_{Q'}(X) \leq 16\phi_{Q'}(C_{opt}) \leq 16\phi_{P'}(C_{opt})$$

Teraz możemy wyprowadzić następujące ograniczenie:

$$\phi_{P'}(X) = \phi_{P' \cap P_{bad}}(X) + \phi_{P' \setminus P_{bad}}(X)$$

$$\leq 16\phi_{P'}(C_{opt}) + 4\phi_{P'}(C_{opt}) = 20\phi_{P'}(C_{opt})$$

Jeżeli $\alpha = 0$, to dla dowolnego punktu $p \in P^{'}$ mamy:

$$(d(p, X))^2 \leq n\Big(\frac{L}{4n}\Big)^2 \leq \frac{L^2}{16n} \leq \frac{L^2}{4n}$$

Zatem:

$$\phi_{P'}(X) \leq \frac{L^2}{4} \leq \phi_V(C_{opt}) \leq \phi_{P'}(C_{opt})$$

bo $V \subseteq P^{'}$. □

Powyższa analiza dowodzi poprawności konstrukcji dla zbiorów X i $P^{'}$. Podsumowując otrzymujemy zbiór X o mocy $O(k \log^2 n)$ oraz zbiór $P^{'}$, dla którego mamy $\phi_{P'}(X) \leq 20\phi_{P'}(C_{opt})$. Czas działania tego algorytmu to $O(n)$ dla $k = O(n^{\frac{1}{4}})$ oraz $O(n \log(k \log n))$ w przeciwnym przypadku [10]. Aby otrzymać taką złożoność kluczowe jest szybkie obliczenie najbliższych sąsiadów punktów. Autorzy [10] proponują zastosowanie algorytmu przedstawionego w pracy [2].

Wróćmy teraz do twierdzenia 4.1, które zostało zdefiniowane na początku podrozdziału 4.2. Chcemy znaleźć zbiór $X^{'}$ o mocy $O(k \log^3 n)$, dla którego jak najwięcej punktów ze zbioru P jest dobrych. Powyżej zdefiniowaną procedurę powtarzamy dla zbioru $P_1 = P \setminus P^{'}$. Analogicznie otrzymamy zbiór $P_1^{'}$ oraz X_1. Kolejny raz aplikujemy procedurę na zbiorze $P_2 = P \setminus (P^{'} \cup P_1^{'})$. Ponieważ za każdym razem zbiór P_i zmieniejsza się co najmniej o połowę to taki proces zakończy się po $O(\log n)$ powtórzeniach. Finalnie otrzymamy zbiór $X^{'} = X \cup X_1 \ldots X_i$ o mocy $O(k \log^3 n)$, dla którego $\phi_P(X^{'}) \leq 20\phi_P(C_{opt})$. Złożoność pozostanie taka sama, czyli $O(n)$ dla $k = O(n^{\frac{1}{4}})$ oraz $O(n \log(k \log n))$ w przeciwnym przypadku.

4.2.3. Krata wykładnicza

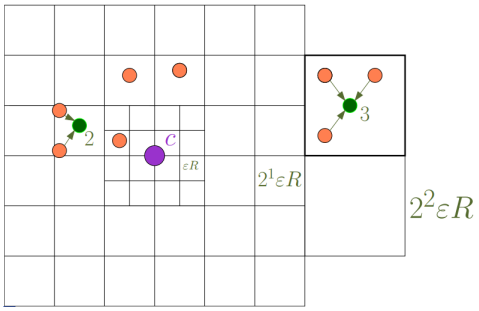
Przejdzmy teraz do kluczowej części tego podrozdziału, czyli budowy kraty wykładniczej, która jest autorską konstrukcją [10]. Krata wykładnicza jest obiektem geometrycznym zbudowanym z komórek, które są hiperześcianami. Każda krata posiada punkt początkowy, od którego zaczynamy konstrukcję

przylegających do siebie komórek o zwiększających się długościach boków. Komórki zawierają punkty z danego na wejścu zbioru danych oraz punkt początkowy, który jest ustalonym punktem z tego zbioru. Konstrukcję dzielimy na $\log n$ poziomów, gdzie n jest mocą wejściowego zbioru. Komórki o tych samych długościach boków należą do tego samego poziomu. Na rysunku 4.1 widzimy 3 poziomową kartę wykładniczą w $\mathbb{R}^2$. Komórki są kwadratami a punktem początkowym jest c.

Niech $P \subset \mathbb{R}^d$, $|P| = n$ oraz niech $A = \{x_1, \ldots, x_m\}$ będzie zbiorem punktów, dla którego zachodzi $\phi_P(A) \leq c\phi^k_{opt}(P)$, gdzie c jest stałą. Nasze A otrzymamy z konstrukcji opisanej w 4.2.1 dla parametru $k = O(k\text{poly}\log n)$ oraz zbioru P, gdzie $\text{poly}\log n = \bigcup_{c\geq 1} O(\log^c n)$. Przedstawiony w tym podrozdziale algorytmy oblicza (k, ϵ)-coreset, gdzie $\epsilon \in (0, 1)$ oraz $c = 20$.

Niech $R = \sqrt{\frac{\phi_P(A)}{cn}}$ będzie dolnym ograniczeniem dla $R^{\phi}_{opt}(P, k) = \sqrt{\frac{\phi_{opt}(P,k)}{n}}$ Niech P_i będzie podzbiorem punktów z P, dla których punkt $x_i \in A$ jest najbliższym sąsiadem. Dla dowolnego $p \in P_i$, mamy $||px_i|| \leq \sqrt{xn}R$, ponieważ $||px_i||^2 \leq \phi_P(A)$ dla $i = 1, \ldots, m$.

Kolejnym krokiem będzie budowa kraty wykładniczej wokół każdego punktu x_i. Niech $Q_{i,j}$ będzie kwatratem o boku długości $R2^j$ o środku w punkcie x_i dla $j = 0, \ldots, M$, gdzie $M = \lceil 2\lg(cn) \rceil$, który jest równoległy do osi układu współrzędnych dla danej przestrzeni. Następnie, niech $V_{i,0} = Q_{i,0}$ oraz niech $V_{i,j} = Q_{i,j} \setminus Q_{i,j-1}$ dla $j = 0, \ldots, M$. Kolejnym krokiem będzie przekształcenie $V_{i,j}$ w kratę, której komórki będą długości $r_j = \frac{\epsilon R 2^j}{10cd}$ oraz niech G_i oznacza wynikową kratę wykładniczą dla $V_{i,0}, \ldots, V_{i,M}$. Mając G_i obliczamy dla każdego punktu z P_i komórkę, do której należy. Dla każdej niepustej komórki z kraty wybieramy losowy punkt do niej należący, który będzie jej reprezentantem. Do takiego punktu przypisujemy wagę, która będzie równa liczbie punktów z komórki, dla której jest reprezentantem. Po przejściu całej kraty otrzymamy zbiór S_i takich punktów. Definujemy $S = \bigcup_i S_i$, który jest (ϵ, k) coresetem. Dowód tego, że zbiór S jest (ϵ, k) coresetem pomijamy z uwagi na jego mechniczność i żmudność. Można do znaleźć w [10]. Zauważmy, że $|S| = O\left(\frac{|A| \log n}{(c\epsilon)^d}\right)$, ponieważ każda krata ma $\log n$ poziomów, a każdy poziom stałą liczbę komórek.



Rysunek 4.1: Krata wykładnicza, gdzie $c = x_i$. W najbliższym otoczeniu punktu c widzimy kratę $V_{i,0}$ z długością komórki równej $r_0 = \epsilon R$. Dla uproszczenia pomijam zmienne w mianowniku wcześniej podanej definicji r_j. Kolejny poziom to krata $V_{i,1}$ o długości komórek równej $r_1 = 2\epsilon R$. Zielone punkty to reprezentanci danej komórki, którym przypisujemy odpowienie wagi równe liczbie punktów w komórce.

4.3. Heurystyka single swap

W tym podrozdziale opiszemy heurystykę single swap [11], która jest przykładem techniki *local search*. Metoda local search jest heurystyką, która zaczyna od dowolnego rozwiązania S problemu optymalizującego. Następnie sprawdza czy niewielka zmiana rozwiązania S poprawia wartość funkcji celu problemu. Jeżeli tak, to wykonuje tę zmianę. Heurystyka kończy swoje działanie w momencie kiedy zmiana w rozwiązaniu nie poprawia funkcji celu. W efekcie otrzymujemy lokalnie optymalne rozwiązanie dla danego na wejściu problemu optymalizującego. Algorytm opisany w tym podrozdziale jest 25-aproksymacją problemu k-means, która zakłada, że na wejściu dostajemy zbiór kandydatów na centra C oraz zbiór n punktów $P \subset \mathbb{R}^d$. Autorzy [11] w celu wyznaczenia C wykorzystują algorytm z pracy [14], którą zastąpimy pracą [10] z uwagi na lepsze gwarancje teoretyczne. Korzystając z algorytmu opisanego w podrozdziale 4.2.3 wyznaczamy zbiór C, który będzie obliczonym (ϵ, k)-coresetem.

Heurystyka *single swap* działa poprzez wybranie początkowego zestawu k centrów S ze zbioru kandydatów na centra C, a następnie wielokrotnej próbie ulepszenia rozwiązania poprzez usunięcie jednego centrum $s \in S$ i zastąpienie go innym centerm $s^{'} \in C - S$. Początkowy stan S może zostać zainicjalizowany losowo albo może być obliczony za pomocą omówionego w podrozdziale 4.1 algorytmu Gonzaleza. Niech $S^{'} = S - \{s\} \cup \{s^{'}\}$ będzie nowym zbiorem centrów, gdzie taką operację na zbiorze S nazwyamy *wymianą* a punkty s, $s^{'}$ *swap parą*. Jeżeli $\phi_P(S^{'}) \leq \phi_P(S)$ to zastępujemy zbiór S zbiorem $S^{'}$, w przeciwnym przypadku S pozostaje bez zmian. W praktyce taki proces powtarzamy do momentu, kiedy $|\phi_P(S^{'}) - \phi_P(S)| < \epsilon$. Formalnie można udowodnić, że dla każdego $\epsilon > 0$, po wielomianiwej liczbie wymian punktów s, $s^{'}$ algorytm zakończy swoje działanie. Autorzy nie dowodzą tego wprost ale powołują się na pracę [3].

Dla uproszczenia zakładamy, że algorytm kończy się kiedy pojedyńcza wymiana

elementów s, $s^{'}$ nie poprawia wyniku. Taki zbiór centrów nazwiemy *1-stable*.

Wprowadźmy notację:

$$\Delta(u,v) = d(u,v)^2 = \sum_{i=0}^{d}(u_i - v_i)^2 = (u-v)\cdot(u-v)$$

gdzie punkty interpretujemy jako wektory oraz operacja $(\cdot)$ jest iloczynem skalarnym.

$$\Delta(S,v) = \sum_{u\in S} d(u,v)^2$$

$$\phi_P(S) = \Delta(S) = \sum_{q\in P}\Delta(q,s_q) = \sum_{q\in P} d(q,s_q)^2$$

gdzie s_q to najbliższy punkt dla q w S oraz $P \subset \mathbb{R}^d$.

Definicja 4.5. Zbiór S nazywamy *1-stable*, jeżeli:

$$\Delta(S - \{s\} \cup \{c\}) \leq \Delta(S)$$

dla dowolnych $s \in S$ oraz $c \in C_{opt}$.

Definicja 4.6. Niech $N_S(s)$ oznacza sąsiedzctwo punktu s, czyli zbiór punktów z P, dla których punkt s jest najbliżej spośród punktów z S.

Autorzy pracy [11] w analizie powołują się na kryterium dla lokalnej minimalności rozwiązania problemu k-means [6]. Mówi ono, że dla dowolnego lokalnie minimalnego rozwiązania C_{min} problemu k-means, każde centrum $c_i \in C_{min}$ jest centroidem jego sąsiedztwa. Rozwiązanie C_{min} nazywamy *centroidalnym*. Zatem, ponieważ punkty interpretujemy jak wektory oraz z powyższego kryterium:

$$s = \frac{1}{|N_S(s)|}\sum_{u\in N_S(s)} u$$

W ramach tego podrozdziału udowodnimy następujące twierdzenie.

Twierdzenie 4.2. *[11] Niech S będzie zbiorem k punktów spełniającym definicję 1-stable oraz niech C_{opt} będzie optymalnym zbiorem dla problemu k-means. Wtedy zachodzi następującą nierówność:*

$$\Delta(S) \leq 25\Delta(C_{opt})$$

Lemat 4.1. *[11] Dla danego skończonego podzbioru S punktów z $\mathbb{R}^d$, niech c będzie centroidem dla S. Wtedy dla dowolnego $c^{'} \in \mathbb{R}^d$, $\Delta(S,c^{'}) = \Delta(S,c) + |S|\Delta(c,c^{'})$.*

Dowód. Z definicji $\Delta(S,c^{'})$ otrzymujemy:

$$\Delta(S,c^{'}) = \sum_{u\in S}\Delta(u,c^{'}) = \sum_{u\in S}(u-c^{'})(u-c^{'})$$

$$= \sum_{u\in S}((u-c)+(c-c^{'}))((u-c)+(c-c^{'}))$$

$$= \sum_{u \in S}((u-c)(u-c)) + 2((u-c)(c-c^{'})) + ((c-c^{'})(c-c^{'}))$$

$$= \Delta(S,c) + 2\Big((c-c^{'})\sum_{u \in S}(u-c)\Big) + |S|((c-c^{'})(c-c^{'}))$$

$$= \Delta(S,c) + |S|\Delta(c,c^{'})$$

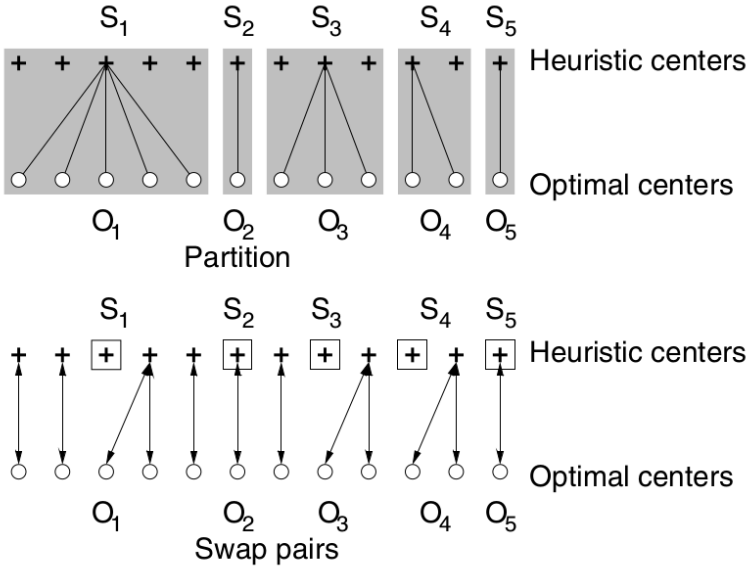
Ostatnie przejście korzysta z faktu, że jeżeli c jest centroidem S to z definicji zachodzi $\sum_{u \in S}(u-c) = 0$. □

Na potrzebę dowodu załóżmy, że znamy zbiór C_{opt}. Dla każdego optymalnego $c \in C_{opt}$ wyznaczamy s_c, który jest najbliższym punktem w S dla punktu c. Punkt s_c nazywamy heurystycznym centrum dla punktu c. W takim kontekście powiemy, że c jest *schwytane* przez s_c. Tutaj warto zaznaczyć, że każde optymalne centrum jest schwytane przez jedno heurystyczne centrum, ale każde heurystyczne centrum może chwytać kilka optymalnych centrów. Heurystyczne centrum nazwiemy *samotnym* jeżeli nie chwyta żadnego optymalnego centrum.

Dowód. Dowód twierdzenia 4.2 zaczniemy od zdefniowania podziału S oraz C_{opt} na zbiory $S_1, \ldots, S_r$ oraz $O_1, \ldots, O_r$ dla pewnego r, gdzie $|S_i| = |O_i|$ dla $i = 1, \ldots, r$.

Dla każdego heurystycznego centrum s_i, które schywało jakąś liczbę $m \geq 1$ optymalnych centrów, tworzymy zbior S_i, który będzie zawierał s_i oraz dowolne $m-1$ osamotnionych heurystycznych centrów. Analogicznie, zbior O_i będzie zawierał wszystkie optymalne centra schwytane przez s_i. Rysunek 4.2 obrazuje tak zdefiniowany podział.

Wygenerujemy teraz *swap par* dla utworzonego podziału. Dla każdego podzbioru podziału takiego, że $|S_i| = |O_i| = 1$, tworzymy z ich elementów parę. Dla każdego podzbioru podziału, który zawiera więcej schwytanych centrów, czyli $|S_i|, |O_i| \geq 1$, tworzymy pary między osamotnionymi heurystycznymi centrami z S_i a optymalnymi centrami z O_i. Każde optymalne centrum jest związane z jednym heurystycznym oraz każde osamotnione centrum jest przyporządkowane co najwyżej dwóm optymalnym centrom. Centra łaczymy dowolnie. Rysunek 4.2 przedstawia przykładowe swap pary.



Rysunek 4.2

Wyznaczymy teraz ograniczenie górne na zmianę funkcji Δ po wymianie punktów ze swap pary (s, o). Zaczniemy od obliczenia najbliższych centrów z $S - \{s\} \cup \{o\}$ dla zadanego na wejściu zbioru P. Niech $N_X(x)$ będzie *sąsiedztwem* punktu x, czyli podzbiorem punktów z P, dla których x jest najbliższym punktem spośród punktów z X. Dla punktów, które należą do $N_{C_{opt}}(o)$ zmiana Δ będzie następująca:

$$\sum_{q \in N_{C_{opt}}(o)} (\Delta(q, o) - \Delta(q, s_q))$$

Każde $q \in N_S(s) \setminus N_{C_{opt}}(o)$ straciło przypisane mu centrum s zatem punkt q musi otrzymać nowe centrum. Niech o_q będzie oznaczało najbliższe centrum dla punktu q. Skoro $q \notin N_{C_{opt}}(o)$ to $o_q \neq o$, zatem s nie schwytało o_q. Zatem po skorzystaniu ze swap pary (s, o), s_{o_q}, najbliższe heurystyczne centrum dla o_q nadal istnieje. Zmiana Δ po wyborze nowych centrów jest co najwyżej równa:

$$\sum_{q \in N_S(s_i) \setminus N_{C_{opt}}(o_i)} (\Delta(q, s_{o_q}) - \Delta(q, s))$$

Na tym etapie dowodu konieczne będzie wprowadzenie dwóch lematów.

Lemat 4.2. *[11] Niech S będzie zbiorem k punktów spełniającym definicję 1-stable oraz niech C_{opt} będzie optymalnym zbiorem dla problemu k-means. Wtedy zachodzi następującą nierówność:*

$$0 \leq \Delta(C_{opt}) - 3\Delta(S) + 2R$$

gdzie $R = \sum_{q \in P} \Delta(q, s_{o_q})$.

Dowód. Na potrzeby dowodu ustalmy swap parę (s, o). Ponieważ S jest 1-stable to:

$$0 \leq \Delta(S) - \Delta(S - \{s\} \cup \{o\})$$

$$= \sum_{q \in N_{C_{opt}}(o)} (\Delta(q, o) - \Delta(q, s_q)) + \sum_{q \in N_S(s) \setminus N_{C_{opt}}(o)} (\Delta(q, s_{o_q}) - \Delta(q, s))$$

Aby rozszerzyć sumę na wszystkie możliwe swap pary zauważmy, że dla każdego optymalnego centrum, jest ono wymienione tylko raz. Zatem każdy punkt q kontrybuuje w pierwszej sumie tylko raz. Po drugie zaważmy, że różnca w drugiej sumie jest zawsze niezerowa, dlatego rozszerzając zakres sumowania możemy tylko zwiększyć sumaryczny wynik.

$$0 \leq \sum_{q \in P} (\Delta(q, o_q) - \Delta(q, s_q)) + \sum_{q \in P} (\Delta(q, s_{o_q}) - \Delta(q, s_q))$$

$$0 \leq \sum_{q \in P} \Delta(q, o_q) - 3 \sum_{q \in P} \Delta(q, s_q) + 2 \sum_{q \in P} \Delta(q, s_{o_q})$$

$$0 \leq \Delta(C_{opt}) - 3\Delta(S) + 2R$$

□

Wcześniej zdefiniowane R nazywamy sumarycznym kosztem przepisania centrów. Korzystając z lematu 4.1 przekształcamy:

$$R = \sum_{o \in C_{opt}} \sum_{q \in N_{C_{opt}}(o)} \Delta(q, s_o) = \sum_{o \in O} \Delta(N_{C_{opt}}(o), s_o)$$

$$= \sum_{o \in C_{opt}} (\Delta(N_{C_{opt}}(o), o) + |N_o(C_{opt})| \Delta(o, s_o)$$

$$= \sum_{o \in C_{opt}} \sum_{q \in N_{C_{opt}}(o)} (\Delta(q, o) + \Delta(o, s_o))$$

$$\leq \sum_{o \in C_{opt}} \sum_{q \in N_{C_{opt}}(o)} (\Delta(q, o) + \Delta(o, s_q))$$

$$= \sum_{q \in P} (\Delta(q, o_q) + \Delta(o_q, s_q))$$

gdzie ostatnia nierówność bazuje na fakcie, że dla każdego $q \in N_{C_{opt}}(o)$ mamy $\Delta(o, s_o) \leq \Delta(o, s_q)$. Następnie korzystamy z nierówności trójkąta.

$$R \leq \sum_{q \in P} \Delta(q, o_q) + \sum_{q \in P} (d(o_q, q) + d(q, s_q))^2$$

$$= \sum_{q \in P} \Delta(q, o_q) + \sum_{q \in P} (d(o_q, q)^2 + 2d(o_q, q)d(q, s_q) + d(q, s_q)^2)$$

$$= 2 \sum_{q \in P} \Delta(q, o_q) + \sum_{q \in P} \Delta(q, s_q) + 2 \sum_{q \in P} d(o_q, q)d(q, s_q)$$

$$= 2\Delta(C_{opt}) + \Delta(S) + 2 \sum_{q \in P} d(o_q, q)d(q, s_q)$$

Lemat 4.3. *[11] Niech $\langle o_i \rangle$ oraz $\langle s_i \rangle$ będą ciągami liczb rzeczywistych, dla których zachodzi:*

$$\delta^2 = \frac{\sum_i s_i^2}{\sum_i o_i^2}$$

dla pewnego $\delta > 0$. Wtedy:

$$\sum_{i=1}^{n} o_i s_i \leq \frac{1}{\delta} \sum_{i=1}^{n} s_i^2$$

Dowód. Z nierówności Schwarza:

$$\sum_{i=1}^{n} o_i s_i \leq \Big(\sum_{i=1}^{n} o_i^2 \Big)^{\frac{1}{2}} \Big(\sum_{i=1}^{n} s_i^2 \Big)^{\frac{1}{2}}$$

$$= \Big(\frac{1}{\delta^2} \sum_{i=1}^{n} s_i^2 \Big)^{\frac{1}{2}} \Big(\sum_{i=1}^{n} s_i^2 \Big)^{\frac{1}{2}}$$

$$= \frac{1}{\delta} \sum_{i=1}^{n} s_i^2$$

□

Niech $\langle o_i \rangle$ będzie ciągiem $d(q, o_q)$ oraz niech $\langle s_i \rangle$ będzie ciągiem $d(q, s_q)$ dla wszystkich $q \in P$. Z tego wynika, że współczynnik aproksymacji możemy przedstawić jako:

$$\alpha = \delta^2 = \frac{\Delta(S)}{\Delta(C_{opt})} = \frac{\sum_{q \in P} d(q, s_q)^2}{\sum_{q \in P} d(q, o_q)^2} = \frac{\sum_{i=1}^{n} s_i^2}{\sum_{i=1}^{n} o_i^2}$$

gdzie S jest zbiorem 1-stable uzyskanym przez wcześniej przedstawiony algorytm. Korzystając z lematu 4.3:

$$R \leq 2\Delta(C_{opt}) + \Delta(S) + 2 \sum_{q \in P} d(o_q, q) d(q, s_q)$$

$$\leq 2\Delta(C_{opt}) + \Delta(S) + \frac{2}{\delta} \sum_{q \in P} d(q, s_q)^2$$

$$= 2\Delta(C_{opt}) + \Delta(S) + \frac{2}{\delta} \Delta(S)$$

$$= 2\Delta(C_{opt}) + \Big(1 + \frac{2}{\delta} \Big) \Delta(S)$$

Z lematu 4.2 wiemy, że:

$$0 \leq \Delta(C_{opt}) - 3\Delta(S) + 2R$$

$$= \Delta(C_{opt}) - 3\Delta(S) + 2\Big(2\Delta(C_{opt}) + \Big(1 + \frac{2}{\delta} \Big) \Delta(S) \Big)$$

$$\leq 5\Delta(C_{opt}) - \Big(1 - \frac{4}{\delta} \Big) \Delta(S)$$

Powyższą nierówności możemy przekształcić do postaci:

$$\frac{5}{1-\frac{4}{\delta}} \geq \frac{\Delta(S)}{\Delta(C_{opt})} = \delta^2$$

$$5 \geq \delta^2\Big(1-\frac{4}{\delta}\Big)$$

$$0 \geq (\delta-5)(\delta+1)$$

To implikuje, że $\delta \leq 5$, zatem współczynnik omawianego algorytmu możemy ograniczyć przez $\alpha = \delta^2 \leq 25$, co kończy dowód twierdzenia 4.2. □

4.4. Podsumowanie

W tej części przedstawimy algorytm budowania coresetu z [15]. Algorytm ten rozpoczynamy od wykonania 10-aproksymacji dla problemu k-means korzystając z pracy [11]. W części 4.3 opisaliśmy konstrukcję, której użyjemy w implementacji. Ma ona trochę większy współczynnik aproksymacji ale jak sami autorzy [15] stwierdzają w swojej pracy, nie ma to dużego znaczenia w kontekście całości. Dowolna aproksymacja o stałym błędzie może zostać użyta w tej metodzie. Na potrzeby analizy zakładamy, że wykonujemy algorytm 10-aproksymacyjny dla problemu k-means i uzyskaliśmy zbiór $C^{'}$ oraz, że mamy dany na wejściu zbiór punktów $A \subset \mathbb{R}^d$.

Geometryczna dekompozycja bazuje na dyskretyzacji punktów z A, czyli na zgrupowaniu ze sobą najbliższych punktów, a następnie zbudowaniu ważonego zbioru punktów S o zredukowanym rozmaiarze. Taką technikę mogliśmy już zobaczyć w części 4.2, gdzie odpowiednio grupowaliśmy punkty w komórki kraty wykładniczej.

Praca [15] przedstawia inną technikę, która bazuje na budowaniu kul o wykładniczo rosnącym promieniu wokół każdego punktu z $C^{'}$. Dla przybliżenia idei konstrukcji załóżmy, że znamy $OPT = \phi^k_{opt}(A)$. Algorytm zaczyna od budowy kul o promieniach równych $\frac{1}{n}OPT$ a kończy na promieniach równych $10OPT$, gdzie n to moc zbioru A. Dla takich kul budujemy ϵ-pokrycie kuli. W pracy pod pojęciem *kula* rozumiemy sferę w przestrzeni o wymiarze d.

Lemat 4.4. *[16] Niech U będzie sferą jednostkową w $\mathbb{R}^d$. Wtedy dla dowolnego $\epsilon \in (0,1)$, istnieje ϵ-pokrycie B o rozmiarze $\Big(1+\frac{2}{\epsilon}\Big)^d$, czyli dla każdego punktu $p \in U$ zachodzi:*

$$\min_{b \in B} ||p-b|| \leq \epsilon$$

Niestety nie istnieją efektywne metody budowania takich ϵ-pokryć. My w analizie przyjmujemy, że $|B| = \epsilon^{-O(d)}$. Jako referencję jak zbudować taką konstrukcję autorzy [15] sugerują analizę pracy [5]. Jest to problematyczne w kontekście implementacji jednak tę kwestię poruszymy w następnym rozdziale.

Lemat 4.5. *[15] Niech a, b, c będą punktami z* $\mathbb{R}^d$. *Wtedy dla dowolnego* $\epsilon \in (0,1)$ *zachodzi:*

$$\Big|||a-c||^2 - ||b-c||^2\Big| \leq \frac{12}{\epsilon}||a-b||^2 + 2\epsilon||a-c||^2$$

Lemat 4.6. *Niech A będzie zbiorem n punktów z* $\mathbb{R}^d$, B^i *będzie kulą o promieniu* $r_i = \frac{2^i}{n}\sum_{x\in A}||x||^2$ *o środku w punkcie o współrzędnych zerowych* $(0,\ldots,0)$ *oraz niech* S^i *będzie* $\frac{\epsilon}{3}$*-pokryciem kuli* B^i *dla* $i = 1,\ldots,\log 10n$. *Zdefinujemy* $S = \bigcup_{i=0}^{\log 10n} S^i$. *Wtedy:*

$$\sum_{x\in A}\min_{s\in S}||x-s||^2 \leq \epsilon^2\sum_{x\in A}||x||^2$$

Dowód. Niech A_{close} będzie podzbiorem punktów z A, dla których kwadrat normy euklidesowej jest równy co najwyżej $\frac{1}{n}\sum_{x\in A}||x||^2$ oraz niech A_{far} będzie zbiorem pozostałych punktów zbioru A. Ponieważ $|A_{close}| \leq n$ oraz z definicji ϵ-pokrycia wynika, że:

$$\sum_{x\in A_{close}}\min_{s\in S^0}||x-s||^2 \leq |A_{close}|\frac{1}{n}\frac{\epsilon^2}{9}\sum_{x\in A_{close}}||x||^2 \leq \frac{\epsilon^2}{9}\sum_{x\in A_{close}}||x||^2$$

Dla każdego punktu x ze zbioru A_{far} istnieje takie i, że $x \in B^i \setminus B^{i-1}$ dla $i \in \{1,...,\log 10n\}$. Zatem:

$$\min_{s\in S^i}||x-s||^2 \leq \frac{\epsilon^2}{9}r_i^2 \leq \frac{4\epsilon^2}{9}r_{i-1}^2 \leq \frac{4\epsilon^2}{9}||x||^2$$

Sumując po wszystkich punktach otrzymujemy:

$$\sum_{x\in A}\min_{s\in S}||x-s||^2 \leq \frac{\epsilon^2}{9}\sum_{x\in A_{close}}||x||^2 + \frac{4\epsilon^2}{9}\sum_{x\in A_{far}}||x||^2 < \epsilon^2\sum_{x\in A}||x||^2$$

□

Powyżej zdefiniowana procedura zaczyna konstruckję w punkcie o współrzędnych zerowych $(0,\ldots,0)$. W dowodzie twierdzenia 4.3 wykorzystamy lemat 4.6, aplikując go do każdego punktu z C'.

Twierdzenie 4.3. *Dla dowolnego zbioru n punktów* $A \subset \mathbb{R}^d$ *istnieje* (ϵ,k)*-coreset dla problemu k-means zawierający* $O(k\epsilon^{-d}\log n)$ *punktów, gdzie d to wymiar (skończony) przestrzeni.*

Dowód. Dla każdego z k centrów ze zbioru C', który obliczyliśmy korzystając z algorytmu 10-aproksymacyjnego [11], tworzymy $\log 10n$ kul o rożnych promieniach. Dla każdej takiej kuli o promieniu r obliczamy $\frac{\epsilon}{16}r$-pokrycie. Niech S będzie sumą wszystkich pokryć kul oraz niech $B(x)$ będzie najbliższym punktem w S dla każdego punktu $x \in A$. Z lematu 4.6 wynika, że:

$$\sum_{x\in A}||x-B(x)||^2 \leq \Big(\frac{\epsilon}{16}\Big)^2\sum_{x\in A}||x||^2$$

Zauważmy, że koszt punktów $A_c \subseteq A$, dla których $c \in C'$ jest centrem wynosi $\sum_{x \in A_c} ||x - c||^2$. Jeżeli założymy, że punkt c jest początkiem przestrzeni to $\sum_{x \in A_c} ||x - c||^2 = \sum_{x \in A_c} ||x||^2$. Zatem, ponieważ C' został uzyskany algorytmem 10-aproksymacyjnym uzyskujemy:

$$\sum_{x \in A} ||x - B(x)||^2 \leq \Big(\frac{\epsilon}{16}\Big)^2 \cdot 10 \cdot OPT$$

Aby pokazać, że S jest szukanym zbiorem rozpatrzmy dowolny zbiór k centrów C:

$$\begin{aligned}
&\Big| \sum_{x \in A} \min_{c \in C} ||x - c||^2 - \sum_{s \in S} \min_{c \in C} ||s - c||^2 \Big| \\
&\leq \Big| \sum_{x \in A} \min_{c \in C} ||x - c||^2 - \sum_{x \in A} \min_{c \in C} ||B(x) - c||^2 \Big| \\
&\leq_{\text{lemat 4.5}} \frac{12}{\epsilon} \sum_{x \in A} ||x - B(x)||^2 + 2\epsilon \sum_{x \in A} \min_{c \in C} ||x - c||^2 \\
&\leq \frac{12}{\epsilon} \Big(\frac{\epsilon}{16}\Big)^2 \cdot 10 \cdot OPT + 2\epsilon \sum_{x \in A} \min_{c \in C} ||x - c||^2 \\
&\leq 2\epsilon \cdot OPT + 2\epsilon \sum_{x \in A} \min_{c \in C} ||x - c||^2 \\
&\leq 4\epsilon \sum_{x \in A} \min_{c \in C} ||x - c||^2
\end{aligned}$$

gdzie ostatnia nierówność zachodzi ponieważ $OPT \leq \sum_{x \in A} \min_{c \in C} ||x - c||^2$ dla dowolnego zbioru center C.

Skalując ϵ przez $\frac{1}{4}$ kończymy dowód. Rozmiar coresetu S to $O(k\epsilon^{-d} \log n)$, ponieważ obliczamy $k \log 10n$ razy $(\frac{\epsilon}{64})$- pokrycie o rozmiarze $\epsilon^{-O(d)}$. □

5. Podsumowanie

W ramach naszej pracy przygotowaliśmy implementację opisanych konstrukcji budowania coresetów dla problemu k-means. Całość dostępna jest w repozytorium `https://github.com/piotrhm/coreset`, które jest podzielone następująco:

```
algorithm
├── geometric
│   └── coreset.py
└── lightweight
    └── coreset.py
dataset
└── s-set
```

Język programowania, który został wykorzystany do implementacji to python w wersji 3.8. Całość operacji na wejściowych zbiorach punktów bazuje na bibliotece numpy `https://numpy.org/`. Podczas lokalnego testowania działania konstrukcji korzystaliśmy z bazy danych s-set dostępnej `http://cs.joensuu.fi/sipu/datasets/`.

Zacznijmy od opisu implementcji konstrukcji przedstawionej w rozdziale 4, czyli geometrycznej dekompozycji. Wprowadzona w tym rozdziale konstrukcja jest wynikiem czysto teoretycznym, co implikuje kilka problemów implementacyjnych.

1. Nie umiemy w szybki sposób obliczać najbliższego sąsiada z danego zbioru dla zbioru punktu. Używanie naiwnego podejścia o złożoności $O(n^2 d)$ w praktyce, nawet dla niewielkich zbiorów o $n = 15000$ skutkuje zauważalnym czasem obliczeń. Problem jest szczególnie widoczny w implementacji heurystyki *single swap*, gdzie przy każdej iteracji algorytmu musimy obliczyć funkcję ϕ dla nowego zbioru kandydatów na rozwiązanie. Częściowym rozwiązaniem tego problemu było użycie biblioteki *sklearn.neighbors* `https://scikit-learn.org/stable/modules/neighbors.html`. Jej implementacja opiera się na *kd*-drzewach, dzięki czemu uzyskuje lepszą złożoność, która wynosi $O(dn \log n)$.

2. Nie umiemy obliczać ϵ-pokryć kul. Jest to bardzo bolesne ponieważ nie istnieje rozsądny zamiennik tego rozwiązania. W związku tym jedyną alternatywą jest użycie jakieś heurystyki, która w praktyce daje akceptowalne wyniki.

Nasza implementacja geometrycznej dekompozycji problemu k-means ma na-

stępujący schemat:

1. Oblicz wielomianową aproksymację problemu k-means:
   (a) Oblicz 2-aproksymację dla problemu k^*-center korzystając z algorytmu Gonzaleza opisanego w 4.1, gdzie $k^* = O(k \log^2 n)$.
   (b) Obliczoną 2-aproksymację dla problemu k^*-center zamień na 20 - aproksymację dla problemu k^{**}-means, korzystając z konstrukcji opisanej w 4.2.1, gdzie $k^{**} = O(k \log^3 n)$.
   (c) Korzystając z konstrukcji kraty wykładniczej opisanej w 4.2.3 zamień 20-aproksymację dla problemu k^{**}-means na (ϵ, k^{**})-coreset.
   (d) Oblicz 25-aproksymację dla problemu k-means korzystając z heurystyki single swap, gdzie obliczony (ϵ, k^{**})-coreset jest zbiorem kandydatów na centra.
2. Oblicz ϵ-pokrycie dla kul o środkach należących do zbioru C, gdzie C to obliczona 25-aproksymacja dla problemu k-means.
   (a) Oblicz kule o środkach w C zgodnie z konstrukcją opisaną w 4.4.
   (b) Dla każdej kuli B_i o promieniu równym R oraz N punktach wykonaj:

```
levels = int(np.ceil(np.log(N)))
last_radius = 0
current_radius = R/2
sample_size_current = N*0.5
for i in range(levels):
    index = np.where((last_radius<B_i)&(B_i<current_radius))
    sample = points[index[0]]

    index_coreset = np.random.choice(sample.shape[0],
        int(sample_size_current), replace=False)
    coreset = np.append(coreset, points[index_coreset], axis=0)

    last_radius = current_radius
    current_radius += radius/np.power(2,i+2)
    sample_size_current /= 2
```

Krok 2.(b) jest autorską konstrukcją. Rozwiązanie bazuje na losowym próbkowaniu. Ideą algorytmu jest podzielenie kuli B_i na $\log n$ poziomów. Pierwszym poziomem jest kula B_i^0 o tym samym środku co B_i oraz promieniu równym $\frac{R}{2}$. Kolejnymi poziomami są zbiory punktów $B_i^{j+1} \setminus B_i^j$, gdzie B_i^j to kula o tym samym środku co B_i oraz promieniu równym $R_i = \frac{R}{2} + \sum_{j=0}^{i} \frac{R}{2^{j+2}}$. Z każdego poziomu losowo wybieramy sample_size_current punktów, gdzie z każdą iteracją zmniejszamy zmienną sample_size_current o połowę. Intuicyjnie taki wybór punktów powinien wybrać odpowiednio dużo punktów blisko środka kuli B_i, która daje nam *dobrą* aproksymację tego, gdzie leży środek klastra optymalnego rozwiązania problemu k-means. Wybieramy też odpowiednio dużo punktów z dalszych poziomów, które mają większy wpływ na wartość funkcji ϕ niż punkty z bliskiego otoczenia środka klastra.

W ogólności, większość algorytmów budowy coresetów bazujących na geometrycznym podejściu cechuje skomplikowana konstrukcja oraz mało atrakcyjna złożoność. Nasza konstrukcja nie jest w tej kwestii wyjątkiem, jednak tym co ją wyrożnia to kroki 1.(a)-1.(c) wyżej opisanego schematu, które bazują na pracy [10].

Autorzy pracy [10] zauważyli, że dla odpowiednio dużego k jesteśmy w stanie szybko obliczać najbliższego sąsiada z danego zbioru dla punktu. Jest to kluczowe w kontekście analizy gwarancji teoretycznych jednak w praktyce zaproponowne rozwiązania są zupełnie nieużyteczne. Szkic konstrukcji znajduje się w [10]. W naszej implementacji zastosowaliśmy gotowe rozwiązanie z biblioteki *sklearn.neighbors*. Dodatkowo zamieniliśmy wykorzystany w pracy [10] algorytm 2-aproksymacyjny dla problemu k-center o liniowej złożoności względem rozmiaru dancyh [9]. Zamiast niego wykorzystaliśmy algorytm Gonzaleza, który został opisany w rozdziale 4.1.

Kolejną optymalizacją jest zainicjalizowanie początkowego rozwiązania S w heurystyce single swap korzystając z algorytmu Gonzaleza. Pomysłodawcami są autorzy [10], którzy nie dowodzą w żaden sposób faktycznej poprawy jakości wyników czy czasu działania.

Zaimplementowana przez nas konstrukcja geometrycznej dekompozycji problemu k-means miała być próbą zaproponowania konkurencyjnego rozwiązania dla lightweight coresetu, czyli konstrukcji omówionej w rozdziale 3. Niestety, z uwagi na wyżej opisane problemy nie udało się dostarczyć rozwiązania, które byłoby zgodne z udowodnionymi gwarancjami teoretycznymi. Przeprowadziliśmy kilka wstępnych testów konstrukcji, które pokazują, że implementacja jest obiecująca. Jej aktualny stan przedstawia mocną bazę do dalszych optymalizacji.

Naturalnym pytaniem jest to czy istnieje *praktyczny* i *łatwy* w implementacji algorytm budujący coreset dla problemu k-means. Podobne pytanie zadali sobie autorzy pracy [4], którzy zaproponowali konstrukcję o nazwie lightweight coreset. Zauważyli, że większość dostępnych algorytmów jest bardzo skomplikowana w implementacji albo nie daje żadnych gwarancji teoretycznych. Całą konstrukcję można streścić w kilku linijach kodu widocznego poniżej.

```
def _compute_coreset(self):
    #Algorithm 1 Lightweight coreset construction
    dist = np.power(self.X-self.X.mean(axis=0), 2).sum(axis=1)
    q = 0.5/self.X.shape[0] + 0.5*dist/dist.sum()
    indices = np.random.choice(self.X.shape[0], size=self.m,
        replace=True)
    X_cs = self.X[indices]
    w_cs = 1.0/(self.m*q[indices])
    return X_cs, w_cs
```

6. Bibliografia

[1] Aloise, D., Deshpande, A., Hansen, P., and Popat, P. Np-hardness of euclidean sum-of-squares clustering. *Machine Learning 75* (05 2009), 245–248.

[2] Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, A. Y. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM 45*, 6 (Nov. 1998), 891–923.

[3] Arya, V., Garg, N., Khandekar, R., Meyerson, A., Munagala, K., and Pandit, V. Local search heuristic for k-median and facility location problems. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 2001), STOC '01, Association for Computing Machinery, p. 21–29.

[4] Bachem, O., Lucic, M., and Krause, A. Scalable k -means clustering via lightweight coresets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2018), KDD '18, Association for Computing Machinery, p. 1119–1127.

[5] Chazelle, B. *The Discrepancy Method: Randomness and Complexity.* Cambridge University Press, 2000.

[6] Du, Q., Faber, V., and Gunzburger, M. Centroidal voronoi tessellations: Applications and algorithms. *SIAM Review 41*, 4 (1999), 637–676.

[7] Feder, T., and Greene, D. Optimal algorithms for approximate clustering. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 1988), STOC '88, Association for Computing Machinery, p. 434–444.

[8] Gonzalez, T. F. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci. 38* (1985), 293–306.

[9] Har-Peled, S. Clustering motion. *Discrete Comput. Geom. 31*, 4 (Mar. 2004), 545–565.

[10] Har-Peled, S., and Mazumdar, S. On coresets for k-means and k-median clustering. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 2004), STOC '04, Association for Computing Machinery, p. 291–300.

[11] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y. A local search approximation algorithm for k-means clustering. *Computational Geometry 28*, 2 (2004), 89 – 112. Special Issue on the 18th Annual Symposium on Computational Geometry.

[12] Li, Y., Long, P. M., and Srinivasan, A. Improved bounds on the sample complexity of learning. *Journal of Computer and System Sciences 62*, 3 (2001), 516 – 527.

[13] Lloyd, S. Least squares quantization in pcm. *IEEE Transactions on Information Theory 28*, 2 (1982), 129–137.

[14] Matousek, J. On approximate geometric k-clustering. *Discrete and Computational Geometry 24* (2000), 61–84.

[15] Munteanu, A., and Schwiegelshohn, C. Coresets-methods and history: A theoreticians design pattern for approximation and streaming algorithms. *Künstliche Intell. 32*, 1 (2018), 37–53.

[16] Pisier, G. *The Volume of Convex Bodies and Banach Space Geometry.* Cambridge Tracts in Mathematics. Cambridge University Press, 1989.