Bottleneckdistanz auf Persistenzdiagrammen

Jacobus Leander Conradi
Vincent Rheinthal

24. Mai 2019

Lab Bericht

Betreuer: Prof. Dr. Rolf Klein

Zweitgutachter: Dr. Elmar Langetepe

Mathematisch-Naturwissenschaftliche Fakultät der
Rheinischen Friedrich-Wilhelms-Universität Bonn

Inhaltsverzeichnis

Inhaltsverzeichnis

Kapitel 1

Problemstellung und Definition

1.1 Einführung

In dieser Arbeit wollen wir uns dem Vergleich verschiedener Bilddaten widmen. Hierfür betrachten wir das Konzept der persistenten Homologie. Persistente Homologie lässt sich am besten beschreiben, als das "Erkennen von Formen mit zusammengekniffenen Augen". Wir wollen also einen komplex aussehenden Datensatz mit diesem Ansatz in grundlegende geometrische Formen zurückführen. Dieser Ansatz ist insbesondere auch sehr intuitiv und nah an dem unterbewussten Erkennen von Strukturen, die einen Menschen auszeichnet.
Wir wollen also zuerst ein Persistenzdiagramm zu einer gegebenen Punktmenge X berechnen und danach ein geeignetes Abstandsmaß für solche Diagramme finden.

1.2 Persistente Homologie

Um Homologien zu berechnen benötigen wir zunächst einmal eine Mannigfaltigkeit, oder im Fall eines Computers bzw. einer notwendigerweise diskreten Darstellung, einen Zell-Komplex. Sei also nun im allgemeinen eine Punktmenge $X \subset \mathbb{R}^d$ gegeben. Zuerst wollen wir den Čech-Komplex definieren.

Definition 1 (Čech-Komplex)**.** *Sei $X \subset \mathbb{R}^d$ eine endliche Menge und $\varepsilon > 0$ gegeben. Dann konstruieren wir $\check{\mathrm{C}}_\varepsilon(X)$ wie folgt. Das 0-Skellet von $\check{\mathrm{C}}_\varepsilon(X)$ ist X und für jede Menge $\sigma \subset X$ ist σ im $|\sigma|$-Skellet von $\check{\mathrm{C}}_\varepsilon(X)$, falls der Schnitt der Bälle mit Radius ε um alle Punkte in σ: $\bigcap_{x\in\sigma} B_\varepsilon(x)$ nicht leer ist.*

Man bemerke zunächst, dass für jedes endliche X ein $\mathcal{E} > 0$ existiert,

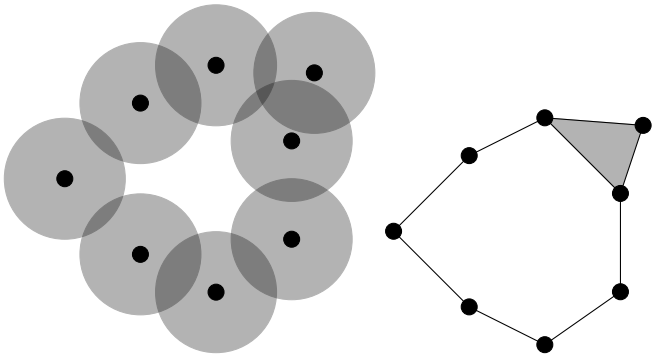

Abbildung 1.1: *Ein Čech-Komplex für ein gegebenes ε mit dem erzeugenden Datensatz.*

sodass für alle $\varepsilon \geq \mathcal{E}$ gilt, dass $\check{\mathrm{C}}_\varepsilon(X) = \check{\mathrm{C}}_\mathcal{E}(X)$. Nenne diesen Zellkomplex, an dem der Čech-Komplex stabilisiert $C(X) := \check{\mathrm{C}}_\mathcal{E}(X)$. Wir definieren nun eine Filtration $f : C(X) \to \mathbb{R}$,$\sigma \mapsto \min\{\varepsilon | \sigma \in \check{\mathrm{C}}_\varepsilon(X)\}$.
Weiterhin bemerke man, dass das Bild von f endlich ist. Sei $\mathrm{im}(f) = \{\varepsilon_i | 0 \leq i \leq N\}$. Dann erhalten wir eine Sequenz von Zellkomplexen:

$$\check{\mathrm{C}}_{\varepsilon_1}(X) \hookrightarrow \check{\mathrm{C}}_{\varepsilon_2}(X) \hookrightarrow \ldots \hookrightarrow \check{\mathrm{C}}_{\varepsilon_{N-1}}(X) \hookrightarrow \check{\mathrm{C}}_{\varepsilon_N}(X).$$

Mit einer solchen Kette können wir nun persistente Homologie definieren.

Definition 2 (persistente Homologie)**.** *Gegeben eine Sequenz von Zellkomplex Einbettungen:*

$$X_1 \xhookrightarrow{\iota_1} X_2 \xhookrightarrow{\iota_2} \ldots \xhookrightarrow{\iota_{n-2}} X_{n-1} \xhookrightarrow{\iota_{n-1}} X_n$$

Betrachten wir die implizierte Homologie Sequenz für jedes k:

$$H_k(X_1) \xrightarrow{\iota_1^*} H_k(X_2) \xrightarrow{\iota_2^*} \ldots \xrightarrow{\iota_{n-2}^*} H_k(X_{n-1}) \xrightarrow{\iota_{n-1}^*} H_k(X_n).$$

Für $0 \leq i < j \leq n$ sei $\mathcal{L}_i^j(X)$ die Menge der Erzeuger σ, die im Cokernel von ι_{i-1}^* liegen, sodass das Bild von σ in $H_{j-1}(X)$ nicht null ist, aber es im Kern von ι_{j-1} liegt. Diese Familie $\{\mathcal{L}_i^j(X) | 0 \leq i < j \leq n\}$ nennen wir die peristente Homologie von X.

Die persistente Homologie speichert also das Leben und Sterben verschiedener Erzeuger während dem Erhöhen der Grenze für Elemente in der Zellkomplexfiltration.
Da wir uns in dieser Arbeit auf 2-dimensionale Punktmengen X beschränken, genügt es, den Zellkomplex $C(X)$ auf sein 2-Komplex zu beschränken, da alle höheren Homologien verschwinden.

Man kann das Berechnen dieser Homologien für 2-dimensionale Daten wir folgt geometrisch definieren. Man lässt um jeden Punkt aus X gleichmäßig Bälle wachsen, vereinigt diese und berechnet die Homologie. Hierbei

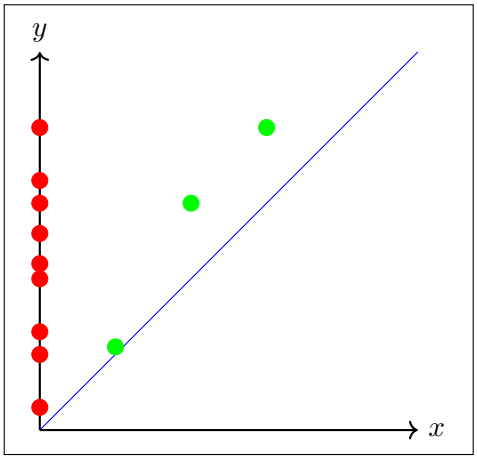

Abbildung 1.2: *Ein Persistenzdiagramm. Rote Punkte gehören zur nullten, und grüne Punkt zur ersten Homologie*

entstehen zwischendurch Kreise, also Erzeuger der ersten Homologie, die bei großem ε wieder verschwinden.

1.2.1 Persistenz Diagramme

Um jetzt mit persistenter Homologie arbeiten zu können wollen wir diese Visualisieren. Dafür kann man sich zum Beispiel Persistenz Diagramme angucken. Hierbei wollen wir für jeden Erzeuger der zu einem bestimmten Zeitpunkt t_1 geboren wird und zu einem Zeitpunkt t_2 stirbt, den Punkt (t_1, t_2) in $\mathbb{R}^2$ markieren.

Man bemerke, dass die so markierten Punkte immer oberhalb der Diagonale $\Delta = \{(x, x) \in \mathbb{R}^2 | x \in \mathbb{R}\}$ liegen, wobei Punkte die direkt an der Diagonalen liegen, Erzeuger repräsentieren, die schnell nach der Geburt sterben, und Punkte die weit von der Diagonalen entfernt sind, Erzeuger repräsentieren, die sehr lange leben. In Abb. 1.2 sieht man ein Persistenzdiagramm, das zu drei unterschiedlichen Zeiten Kreise - also Erzeuger der ersten Homologie - aufweist. Des weiteren kann man sehen, dass die Erzeuger der nullten Homologie - also einer pro Zusammenhangskomponente zu jedem gegenen Zeitpunkt - alle zum Zeitpunkt 0 geboren werden.

1.3 Wasserstein Distanz

Seien also nun zwei solche Persistenzdiagramme gegeben. Um ein Abstandsmaß zu definieren, betrachten wir die durch die Wahrscheinlichkeitstheorie definierte Wassersteindistanz. Hierbei handelt es sich um den ”minimalen

Aufwand, eine Wahrscheinlichkeitsverteilung zu einer anderen zu bewegen". In unserer konkreten Anwendung definieren wir diese wie folgt.

Definition 3 (Wassersteindistanz)**.** *Seien X und Y zwei Punktmengen gegeben, mit $|X| = |Y|$. Dann definieren wir die Wassersteindistanz*

$$W_p(X,Y) = \min_{\varphi: X \to Y} \left(\sum_{x \in X} ||x - \varphi(x)||^p \right)^{\frac{1}{p}},$$

wobei über alle Bijektionen $\varphi : X \to Y$ minimiert wird und $||\cdot||$ die ℓ_2-Norm in $\mathbb{R}^2$ ist.

Ein Spezialfall $W_\infty(\cdot,\cdot)$ nennen wir die Bottleneckdistanz.
Man bemerke die starke Einschränkung, dass $|X| = |Y|$ sein muss. Diese wollen wir geeignet anpassen, sodass auch unterschiedliche große Mengen miteinander verglichen werden können und unterschiedliche Herangehensweisen qualitativ vergleichen.

1.4 Feature Proposals

Als letzten Schritt wollen wir Methoden untersuchen, um zu einem gegebenen Bild eine representierende Punktemenge X zu finden, um diese als Eingabe für das Berechnen der persistenten Homologie zu benutzten. Auch hier wollen wir unterschiedliche Methoden vergleichen und im Zusammenhang mit der persistenten Homologie vergleichen.

Kapitel 2

Persistente Homologie

In diesem Kapitel wollen wir die Implementierung zur Berechnung des Čech-Komplexes, sowie die Berechnung der persistenten Homologie widmen.

2.1 Čech-Komplex

Zum Berechnen des Čech-Komplexes betrachten wir zunächst einmal das Voronoi Diagramm für gegebene Punkte X im 2-dimensionalen Raum. Hierbei wird $\mathbb{R}^2$ in Regionen unterteilt, sodass in jeder Region genau ein Punkt x aus X liegt, und für jeden anderen Punkt y aus dieser Region ist x der näheste Punkt aus X an y.
Aus dem Voronoi Diagramm können wir den Čech-Komplex extrahieren. Hierbei fügen wir zu dem Zellkomplex eine Kante zwischen x und y aus X hinzu, wenn die Voronoi-Regionen von x und y sich berühren. Die Distanz von x und y gibt uns außerdem den Filtrationswert $f(\{x, y\}) = \frac{||x-y||}{2}$ der Kante. Fügt man alle diese Kanten hinzu, erhält man den planaren Dual-Graph, die DeLauney-Triangulierung der Punkte aus X.
Komplizierter wird es bei Flächen, also 2-Zellen. Hierbei müssen wir eine wichtige Fallunterscheidung treffen. Falls sich mehr als 2 Voronoi-Regionen in einem Punkt berühren, korrespondiert dies zu einem Kreis und des weiteren einer 2-Zelle im Čech-Komplex. Doch was genau ist der Filtrationswert. Hier kommt die folgende Fallunterscheidung ins Spiel. Betrachtet man die konvexe Hülle der Punkte $x_1, \ldots, x_k \in X$, deren Voronoi Regionen sich berühren, kann es passieren, dass der Punkt, indem sich die Voronoi-Regionen sich berühren, innerhalb oder außerhalb des Polytops liegen. Falls der Punkt v innerhalb des Polytops liegt, muss der Filtrationswert als der Abstand von v zu allen x_i, die alle gleich sind, gewählt werden. Denn sobald dieser Wert überschritten wird, "füllt" sich das Loch in der geometrischen Präsentation mit den Bällen, also muss die Fläche eingefügt werden.
Im anderen Fall gibt es keinen Zeitpunkt, wo der durch die Punkte $x_1, \ldots, x_k$ gegebene Kreis im in der geometrischen Repräsentierung "um ein Loch" ist,

```
public class Voronoi {
   //stores voronoi vertices, where more than 2 regions touch
   private PointD[] vertices = null;
   private VEdge[] edges = null;
   ...
   private void compute(int width, int height) {
      VoronoiResults results =
          org.kynosarges.tektosyne.geometry.Voronoi.findAll(
      sites, new RectD(0, 0, width, height));
      vertices = results.voronoiVertices;
      //transform output of library to our own data types
      ...
   }
   ...
}
public class ActionGenerator {
   @NotNull
   public List<Action> generate() {
      //generates the list of elements added to the cell
          complex sorted by their filtration values
      ...
      voronoi.forEachVertex(this::computeVertex);
      voronoi.forEachEdge(this::computeEdge);
      actions.sort(Action::compareTo);
      return actions;
   }
   private void computeVertex(@NotNull PointD vertex, int
       index) {
      //create list of actions given the Voronoi Diagram
      ...
      VEdge[] edges = voronoi.getEdges(edgeIndices);
      PointD[] sites = getSites(edges);
      if (Util.isInside(vertex, sites)) {
         actions.add(new FaceAction(...));
         return;
      }
      ...
      actions.add(new EdgeFaceAction(...));
   }
}
```

Abbildung 2.1: *Codesnippet of the generation ofzur Generierung von* $C(X)$

sodass die Fläche zu dem Zeitpunkt eingefügt werden muss, wo die letzte Kante den Kreis in $x_1, \ldots, x_k$ schließt. Mit anderen Worten, der Filtrationswert ist $max_{\{x_i,x_y\}\in C(X)}||x_i - x_j||$. Diese 2-Zellen wollen wir *degeneriert* nennen.
Für die Implementierung in Java haben wir uns für eine Library entschieden, die uns das Voronoi Diagramm berechnet. Den planaren Dualgraph berechnen wir aufbauend auf der Library, wie mn in Abb. 2.1 sehen kann. Des weiteren sieht man, dass wir im Fall von degenerierten 2-Flächen eine `EdgeFaceAction` benutzten, um die längste Kante aus dem Kreis sowie die 2-Zelle direkt im selben Zeitschritt einfügen.

2.2 Berechnung der persistenten Homologie

Ein wichtiges Konzept zur Berechnung der persistenten Homologie ist die sogenannte "elder rule". Wenn zwei Erzeuger zu einem Zeitpunkt miteinander identifiziert werden, muss man eine Wahl treffen, welche der Erzeuger noch lebt und welcher nicht. Wir wollen in dieser Arbeit hier den Älteren überleben lassen. Dies bedeutet insbesondere, dass wir für 0-Zellen eine Wahl treffen müssen, welche Knoten älter als andere Knoten sind. Da die Wahl aber für das Persistenzdiagramm der nullten Homologie kein Einfluss hat, nummerieren wir die Knoten zufällig durch. Diese Nummerierung benutzten wir beim berechnen der ersten Homologie als eine Orierntierung der Kanten.

Da das 1-Skellet das selbe wie ein Graph ist, speichern wir den Komplex als Graph, und merken uns, welche Relationen durch 2-Zellen erzeugt werden.

Die Hauptarbeit in unserer Implementierung ist das Berechnen der ersten Homologie. Für die nullte muss man sich lediglich beim Hinzufügen einer Kante merken, ob die zwei Knoten in der selben, oder in unterschiedlichen Zusammenhangskomponenten sind. Um gleichzeitig noch die "elder rule" für Erzeuger aufrecht zu erhalten merken wir uns für jeden Knoten, welche der älteste ist, der in der Zusammenhangskomponente ist. Falls die hinzugefügte Kante zwischen zwei unterschiedlichen Zusammenhangskomponenten ist, updaten wir für die eine der Zusammenhangskomponenten welche der neue älteste Knoten ist und merken uns, dass ein Erzeuger der nullten Homologie gestorben ist.

Da Erzeuger der ersten Homologie im Kern des Randoperators $H_1(C(X)) \rightarrow H_0(C(X))$ leben, und diese gerade beschrieben werden durch alle Kreise, müssen wir alle möglichen Kreise im Graphen finden, und dann über Linearkombinationen dieser herausfinden, welche Kreise sterben. Um Kreise zu finden, gucken wir beim Hinzufügen einer Kante $e = (v, w)$, ob v und w in der selben Zusammenhangskomponente sind. Ist dies der Fall, finden wir einen v, w-Pfad P in $G \setminus \{e\}$. Dann merken wir uns $P \cup \{e\}$ als einen neuen Erzeuger. zum Zeitpunkt, wo dieser Erzeuger hinzugefügt wird, ist e ganz

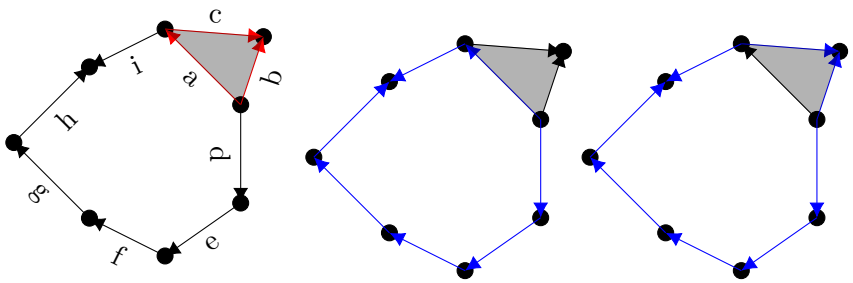

Abbildung 2.2: Beispiel der Kreisvektoren. In Rot der Kreis der Relation, in Blau zwei Kreise, die durch die rote Relation miteinander identifiziert werden

neu und noch in keinem Rand einer 2-Zelle oder einem andern Kreis enthalten ist, sodass dieser Kreis auf jeden Fall einen Schritt lang ein lebender Erzeuger ist.

Falls eine 2-Zelle dazu kommt merkt man sich eine Relation, gerade der eindeutige Kreis, der den Rand der 2-Zelle beschreibt. Jedes mal, das eine 2-Zelle hinzugefügt wird, muss man überprüfen, ob ein Kreis dadurch stirbt. Hierfür lösen wir die Matrix:

$$\begin{Bmatrix} c_{11} & c_{21} & \dots & c_{n1} & r_{11} & \dots & r_{k1} \\ c_{12} & c_{22} & \dots & c_{n2} & r_{12} & \dots & r_{k2} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ c_{1m} & c_{2m} & \dots & c_{nm} & r_{1m} & \dots & r_{km} \end{Bmatrix}$$

Wobei $c_{ij} = \sum_{e_j \in P_i} 1 - \sum_{e_j * \in P_i} 1$ und $r_{ij} = \sum_{e_j \in R_i} 1 - \sum_{e_j^* \in R_i} 1$, wobei P_i die Erzeuger-Kreise und R_i die Relationskreise sind, und e_j eine Kante sowie e_j^* die umgedrehte Kante ist.

Ein Beispiel in Abb. 2.2. Wir haben zwei Kreise in Blau gegeben, und eine Relation in Rot. Nummeriert man die Kanten wie im ersten Bild und wähle alle Kreisorientierungen gegen den Uhrzeigersinn erhält man folgende Matrix:

$$A = \begin{Bmatrix} 1 & 0 & -1 \\ 0 & 1 & 1 \\ 0 & -1 & -1 \\ -1 & -1 & 0 \\ -1 & -1 & 0 \\ -1 & -1 & 0 \\ -1 & -1 & 0 \\ -1 & -1 & 0 \\ 1 & 1 & 0 \end{Bmatrix}$$

Und erhält somit einen nichtleeren Kern, $A(1, 1, -1)^T = 0$ Damit wissen wir, das die beiden Erzeuger miteinander identifiziert werden und können

den jüngeren aus der Erzeugerliste schmeißen.

Wir wissen nun, wann wir Kanten und wann, sowie wie wir 2-Zellen zu unserem Komplex hinzufügen und wie wir alle möglichen Erzeuger finden. Fügt man dies zusammen ist es möglich die erste persistente Homologie zu berechnen.

2.2.1 Optimierung

Da uns dieses Verfahren zu langsam war (5 Sekunden für 100 Knoten, 20 Sekunden für 200 Knoten) haben wir das ganze optimiert, indem wir den Graphen zu geschickten Zeiten kontrahieren, um die Berechnungskomplexität niedrig zu halten. So erhalten wir fast lineare Laufzeit, mit 10 Sekunden für 1000 Knoten.

Probably better if Vince LaTeXthis.

Kapitel 3

Distanzmaße

Als erstes wollen wir die Wassersteindistanz auf beliebig Große Mengen erweitern. Eine erste Idee war für $|X| < |Y|$ über Injektionen $\varphi : X \to Y$ zu minimieren, und einen Fehlerterm für alle nicht getroffenen Y einzuführen. Dies schien vielversprechend.
Definiere $\gamma : \mathbb{R}^2 \to \mathbb{R}$, mit $(x, y) \mapsto y - x$. Die Motivation hinter dieser Definition ist, dass dies genau der vertikale Abstand von einem Punkt (x, y) zu der Diagonalen ist. Dann defineren wir

$$W'_p(X,Y) := \min_{\varphi: X \to Y} \left(\sum_{x \in X} ||x - \varphi(x)||^p + \sum_{y \in Y \setminus \mathrm{im}(\varphi)} \gamma(y)^p \right)^{\frac{1}{p}}.$$

Dieser Ansatz erschien zuerst vielversprechend, aber der Fehlerterm war zu groß. Das Problem ist, dass $||x-\varphi(x)||$ die ℓ_2-Norm von dem Abstand von x und $\varphi(x)$ ist, wohingegen $\gamma(y)$ die ℓ_1-Norm von dem Abstand von y und Δ ist. Definiert man stattdessen $\gamma(x, y) = \frac{y-x}{\sqrt{2}}$ erhält man eine faire Verteilung. Allerdings ist auch dieser Abstand zu restriktiv. Wir wollen Punkten aus X auch erlauben, nicht gemapped zu werden. Also definieren wir nun die finale Version unserer modifizierten Wassersteindistanz

$$V_p(X,Y) := \min_{Z \subset X} \min_{\varphi: Z \to Y} \left(\sum_{z \in Z} ||z - \varphi(z)||^p + \sum_{y \in (Y \setminus \mathrm{im}(\varphi)) \cup X \setminus Z} \gamma(y)^p \right)^{\frac{1}{p}}.$$

Wir nennen die Bottleneckdistanz $B(X, Y) := V_\infty(X, Y)$.

3.1 Effizientes Berechnen

Man bemerkt schnell, dass das ganze Problem sehr verwandt mit dem "Optimal Mapping"-Problem ist. Wir wollen uns das wissen vom "Optimal Mapping" Problem zu nutzen machen und das Problem des Miniierens über exponentiell viele Abbildungen als Flow-Problem darstellen. Für $V_p(X, Y)$

definieren wir also folgenden Graphen. $V := X \cup Y \cup s, t, h, h'$ und $E := (X \cup h) \times (Y \cup h') \cup s \times (X \cup h) \cup (Y \cup h') \times t$.
Gewichte wählen wir für Kanten in $s \times (X \cup h) \cup (Y \cup h') \times t \cup \{(h, h')\}$ immer 0, für Kanten der Form $(x, y) \in X \times Y$ sind die Kosten genau $||x-y||$, und für Kanten der Form $(x, h') \in X \times \{h'\}$ resp. $(h, y) \in \{h\} \times Y$ wählen wir $\gamma(x)$ resp. $\gamma(y)$. Als Kapazitäten wählen wir für die Kante (s, h) als $|Y|$, für die Kante (h', t) als $|X|$, für die Kante (h, h') als $\min(|X|, |Y|)$ und für alle anderen Kanten als 1. Und als letztes die Fluss-Requirements $b(s) = b(-t) = |X| + |Y|$, und $b(v) = 0$ für alle anderen. Um die Wassersteindistanz über eine min-cost-Flow Instanz zu lösen, müssen wir vorher beweisen, dass sich jeder zulässige ganzzahlige Fluss als eine solche Abbildung φ für den Kosten gleich dem der Kostenterm für φ und andersherum darstellen lässt. Daraus folgt dann, dass die Kosten eines min-cost-flows gleich der Wassersteindistanz ist. Man bemerke, dass diese Kosten nur für $p = 1$ gilt. Für $1 < p < \infty$ exponentiere die Kosten aller Kanten um p und ziehe am Ende die $p$te Wurzel der Kosten.

Lemma 1. *Die Kosten eines min-cost-flows für den obigen Graphen $G = (V, E)$ ist gleich der Wassersteindistanz $V_p(X, Y)$.*

Beweis. Sei $f : E \to \mathbb{Z}_{\geq 0}$ also ein zulässiger Fluss. Wähle als Z alle Knoten x aus X, sodass $f((x, y)) = 1$ für ein $y \in Y$ und $\varphi(x) = y$ für dieses y. In Worten, φ ist gegeben durch die gewählten Flusskanten von f. Da die Kosten der vom Fluss gewählten Kanten der Form (x, h') oder (h, y) genau $\gamma(x)$ und $\gamma(y)$ sind, gilt also, dass $V_p(X, Y)$ eine untere Schranke für den min-cost-flow Wert sind.
Für die andere Richtung betrachtet man für gegebenes Z, φ den Fluss, der gegeben ist, durch $f(x, y) = 1$ genau denn wenn $x \in Z$ und$\varphi(x) = y$. Für alle $x \in \backslash Z$ wählt man $f((x, h')) = 1$ und für $y \in Y \setminus \mathrm{im}(\varphi)$ wählt man respektive $f((h, y)) = 1$. Somit erhält man einen zulässigen Fluss, mit den selben Kosten. □

Und wie wir wissen, kann man das min-cost-flow Problem in polynomieller Zeit z.B. mit einem Algorithmus wie Cycle Canceling berechnen.
Um die Bottleneckdistanz zu berechnen betrachten wir den Subgraphen, der gegeben ist die Alle Kanten die kleiner als ein bestimmter Wert sind. Da die Bottleneckdistanz gegeben ist durch die billigste Kante, sodass mit allen billigerern Kanten noch ein zulässiger Fluss möglich ist, kann man mit logarithmisch vielen Aufrufen eines min-cut Algorithmuses mit einer Methode wie Binary-Search die Bottleneck-Distanz bestimmen.