Approximative metrische Suche mit Hashfunktionen in hochdimensionalen Datensätzen

Bachelorverteidigung von Jonas Köhler

Bauhaus-Universität Weimar, 11.7.2016

Bearbeitete Fragestellungen

1. Einordnung existierender Hashverfahren hinsichtlich ihrer Konstruktionsprinzipien

Bearbeitete Fragestellungen

1. Einordnung existierender Hashverfahren hinsichtlich ihrer Konstruktionsprinzipien

2. Abschätzungen einer oberen und einer unteren Schranke der Anzahl von benötigten Hashfunktionen für ideales Multi-Table-Hashing

Bearbeitete Fragestellungen

1. Einordnung existierender Hashverfahren hinsichtlich ihrer Konstruktionsprinzipien

2. Abschätzungen einer oberen und einer unteren Schranke der Anzahl von benötigten Hashfunktionen für ideales Multi-Table-Hashing

Struktur des Vortrags:

1. Was ist metrische Suche?

Struktur des Vortrags:

1. Was ist metrische Suche?

2. Was sind hochdimensionale Datensätze?

Struktur des Vortrags:

1. Was ist metrische Suche?

2. Was sind hochdimensionale Datensätze?

3. Wie sucht man mit Hashfunktionen?
   Was sind “Hashfunktionen”?

Struktur des Vortrags:

1. Was ist metrische Suche?

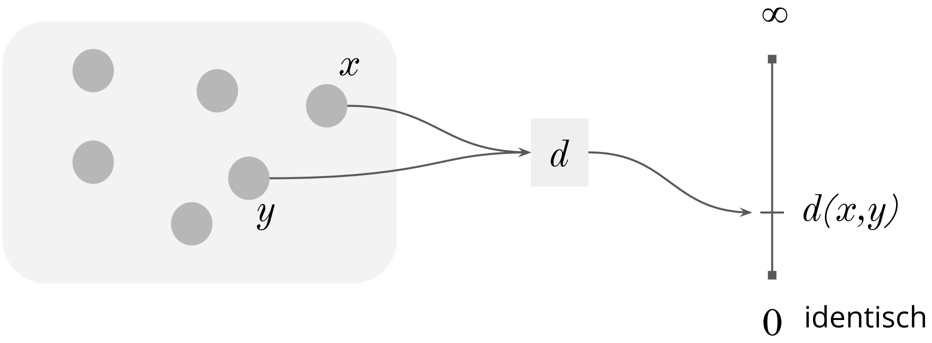
2. Was sind hochdimensionale Datensätze?

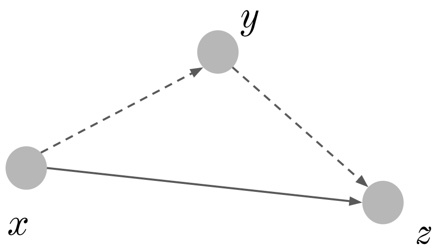
3. Wie sucht man mit Hashfunktionen?
   Was sind “Hashfunktionen”?

4. Was bedeuten die berechneten Grenzen?
   Wie wurden sie bestimmt?

Approximative metrische Suche mit Hashfunktionen in hochdimensionalen Datensätzen

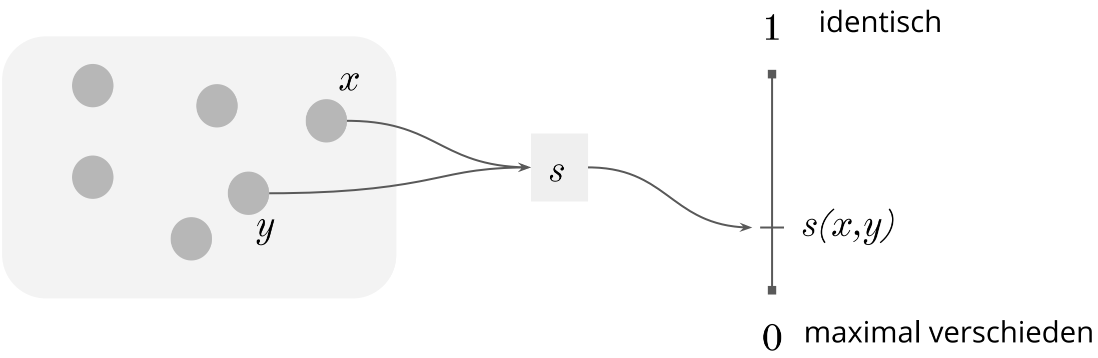
Metrik (Abstand)





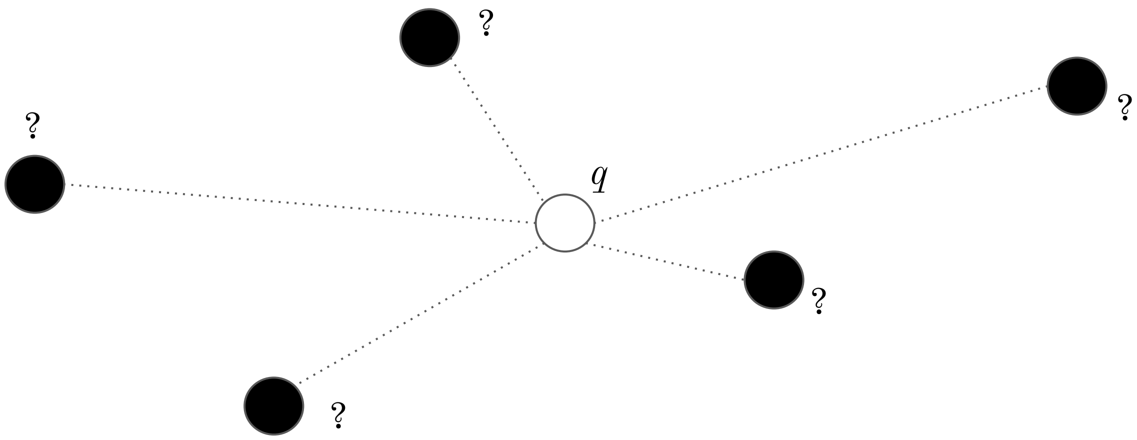
$$d(x,z) \leq d(x,y) + d(x,y)$$

Ähnlichkeit

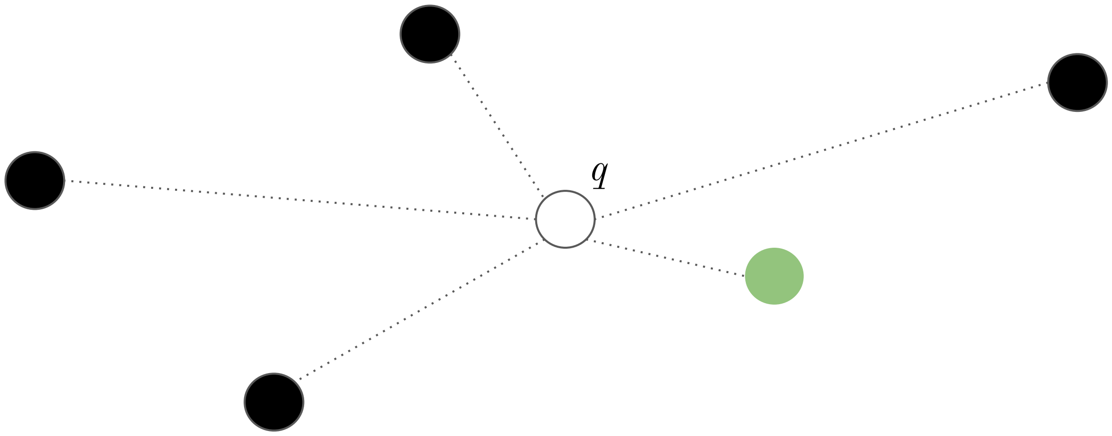


AMPEL	AMPEL	→	5/5 = 1
AMPEL	APFEL	→	3/5 = 0.6
AMPEL	ANGST	→	1/5 = 0.2
AMPEL	GURKE	→	0/5 = 0

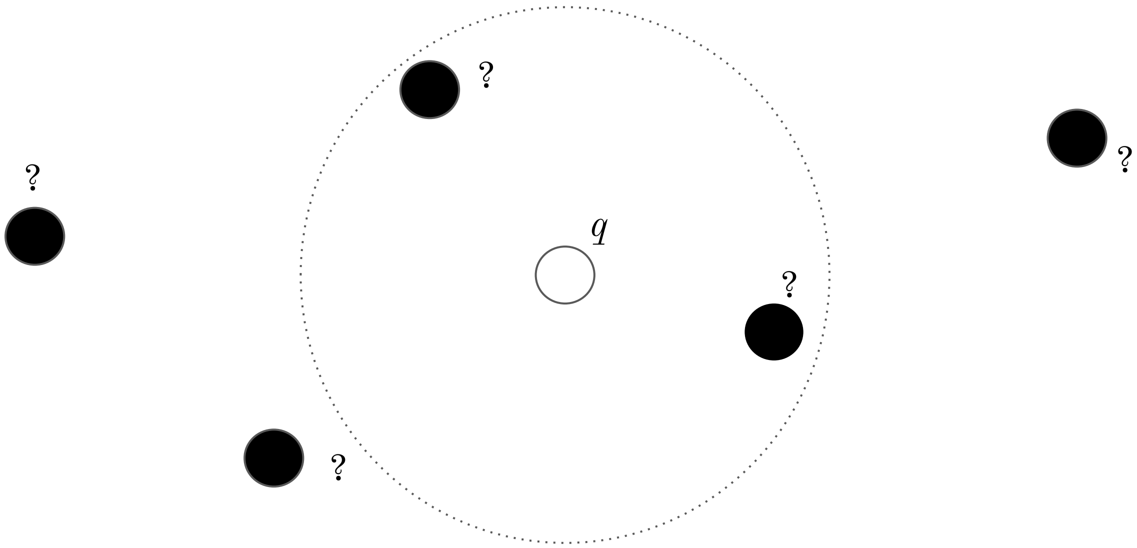
Suche nach dem nächsten Nachbarn



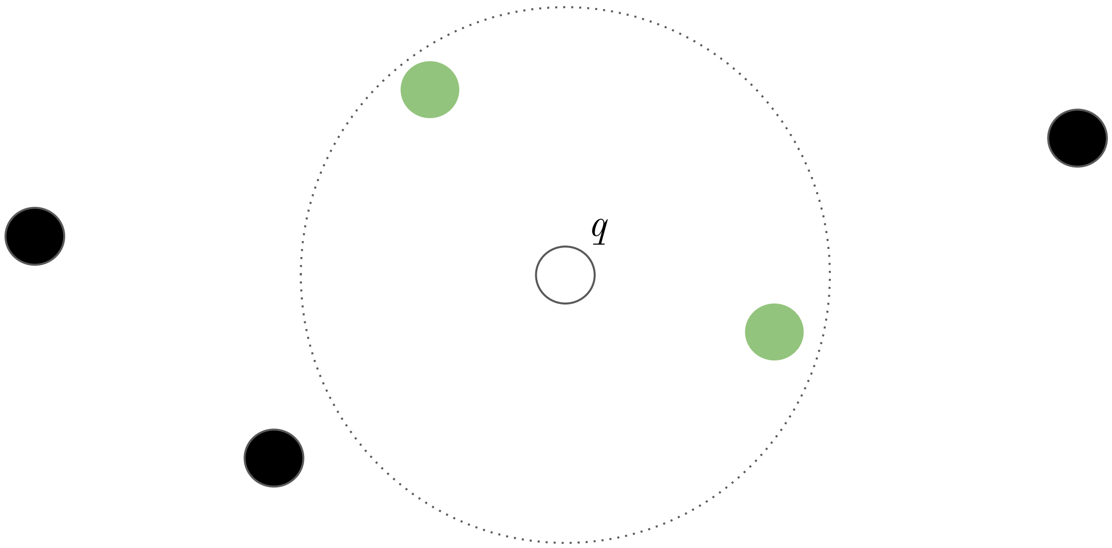
Suche nach dem nächsten Nachbarn



Umgebungssuche

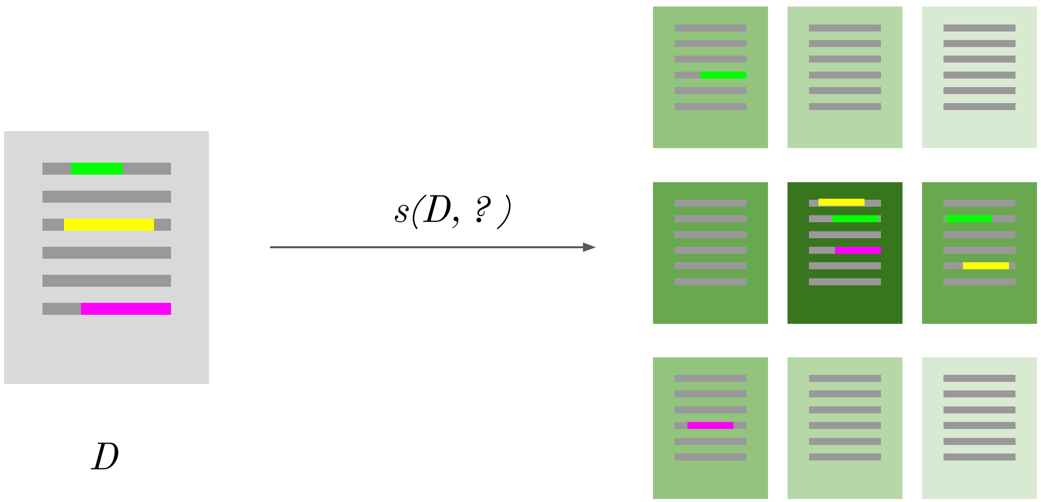


Umgebungssuche



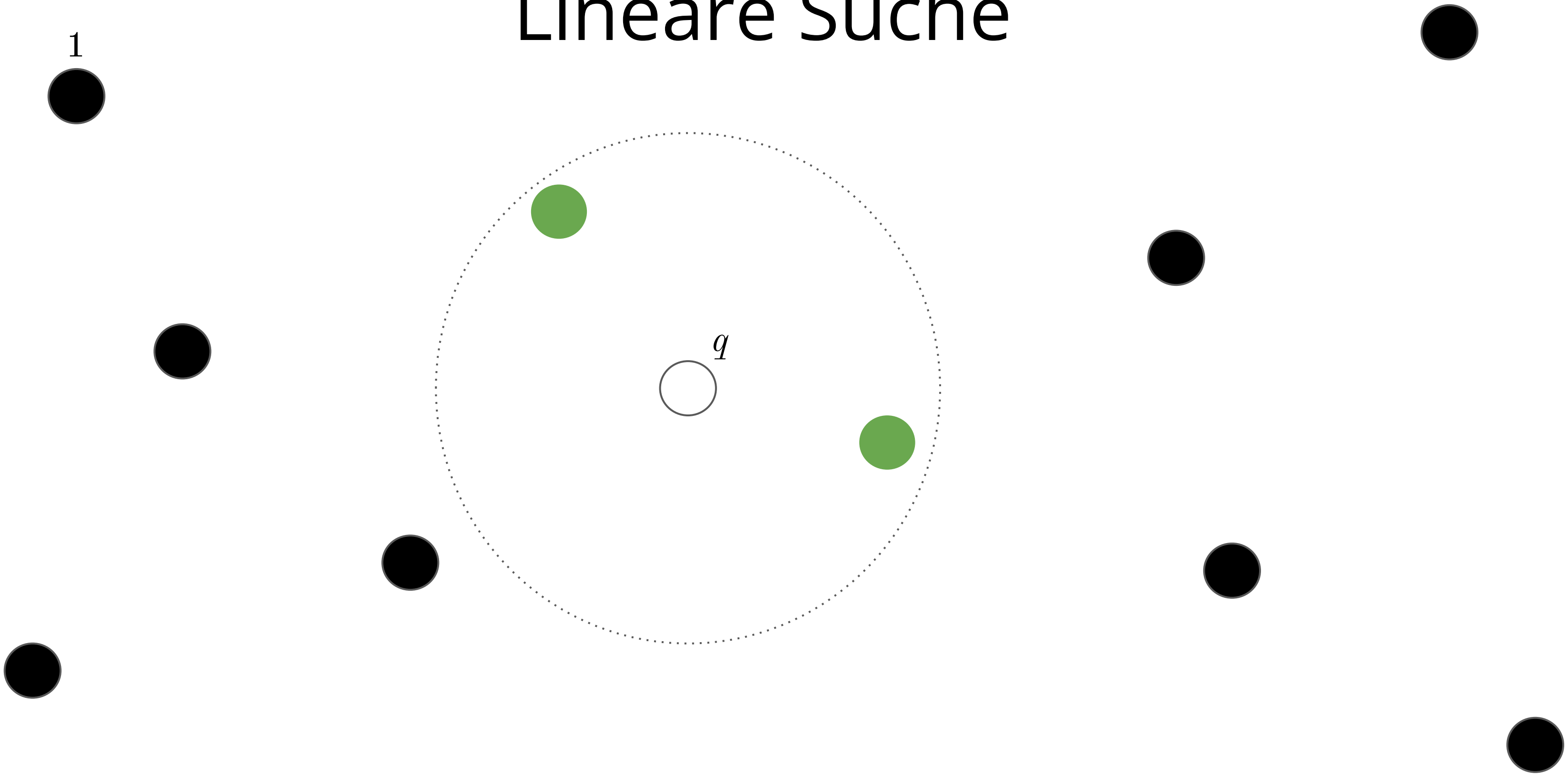
Motivation: automatisierte Suche nach Plagiaten

Motivation: automatisierte Suche nach Plagiaten

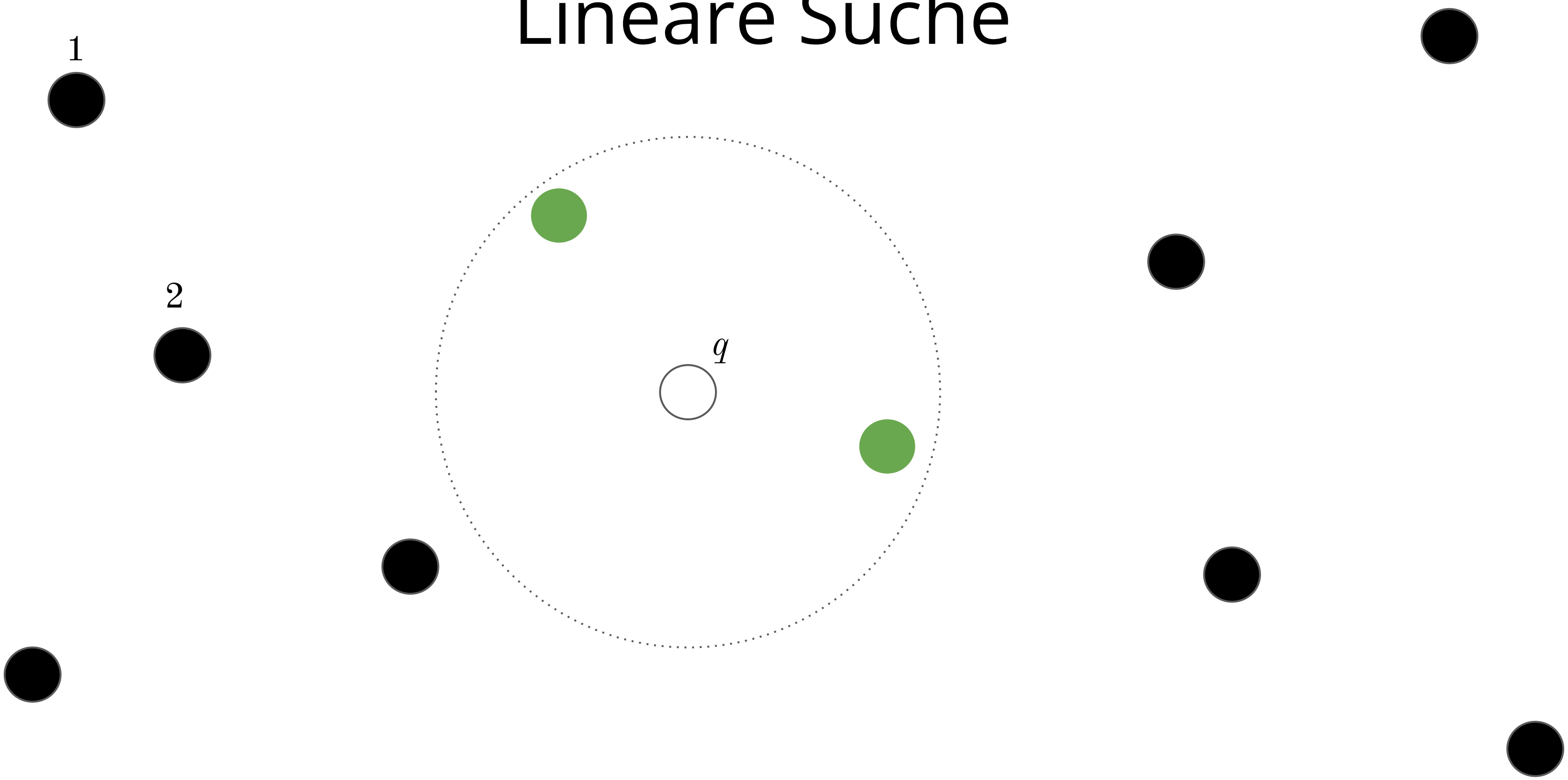


Motivation: automatisierte Suche nach Plagiaten

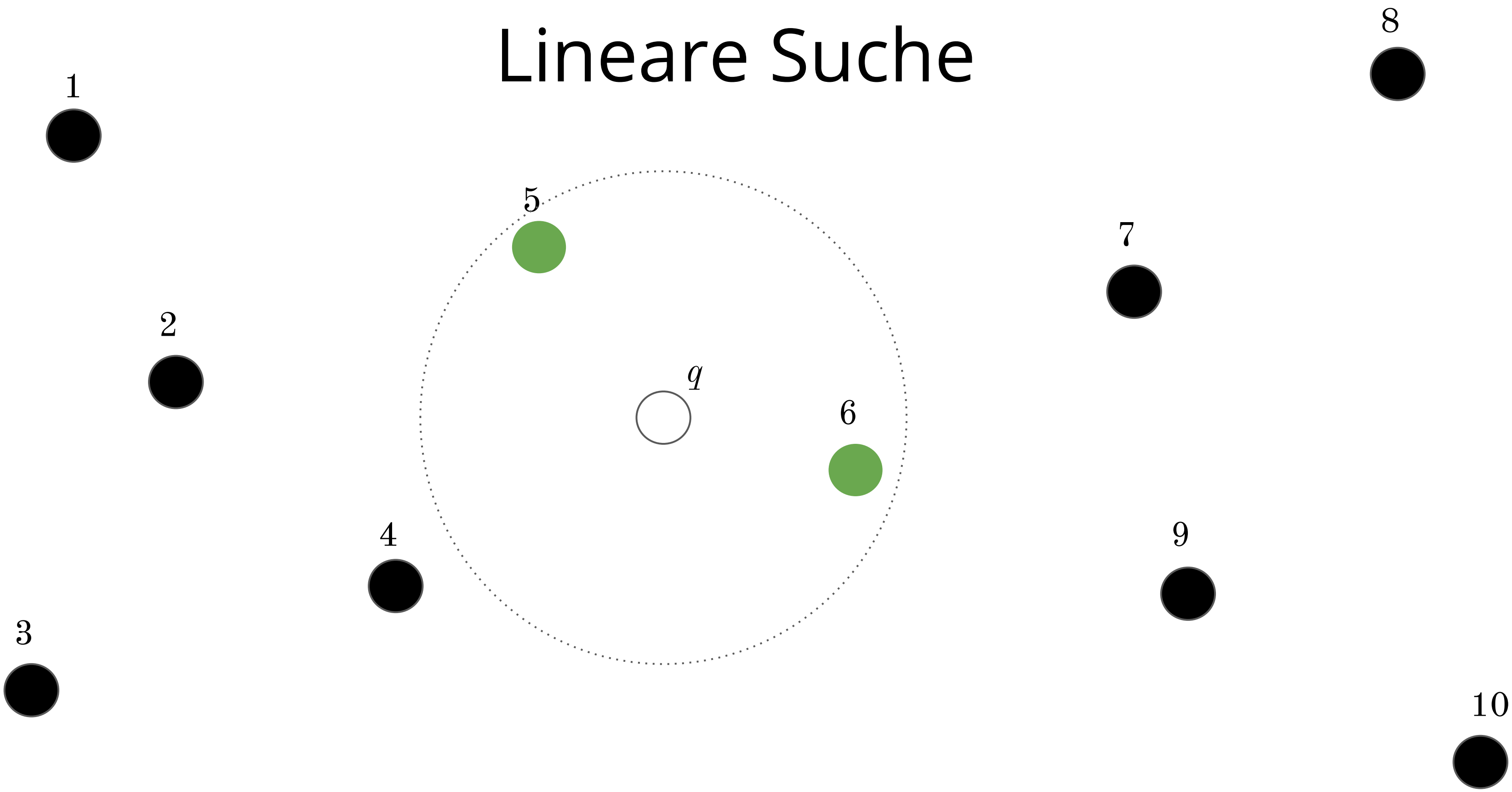
Lineare Suche



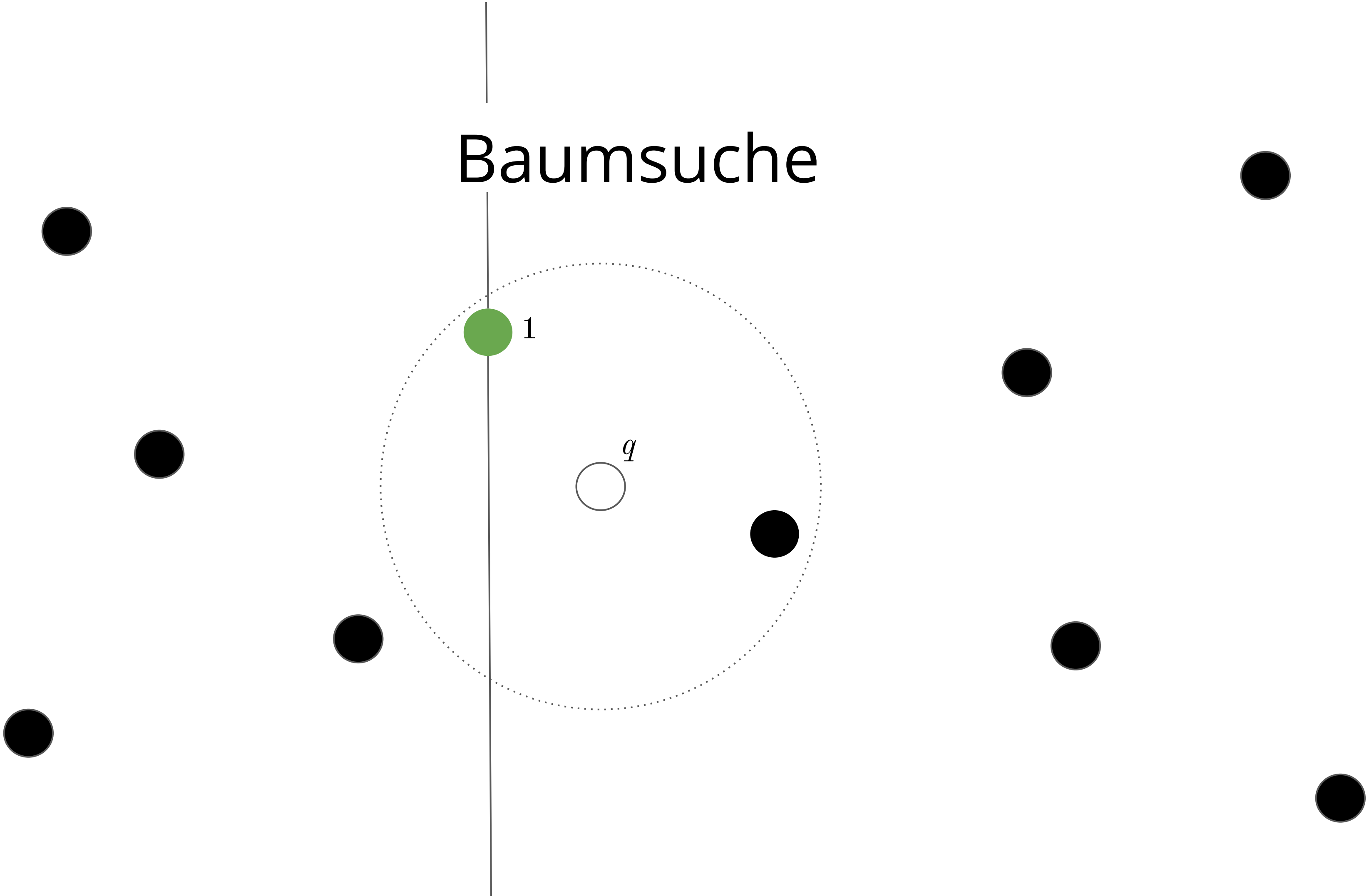
Lineare Suche



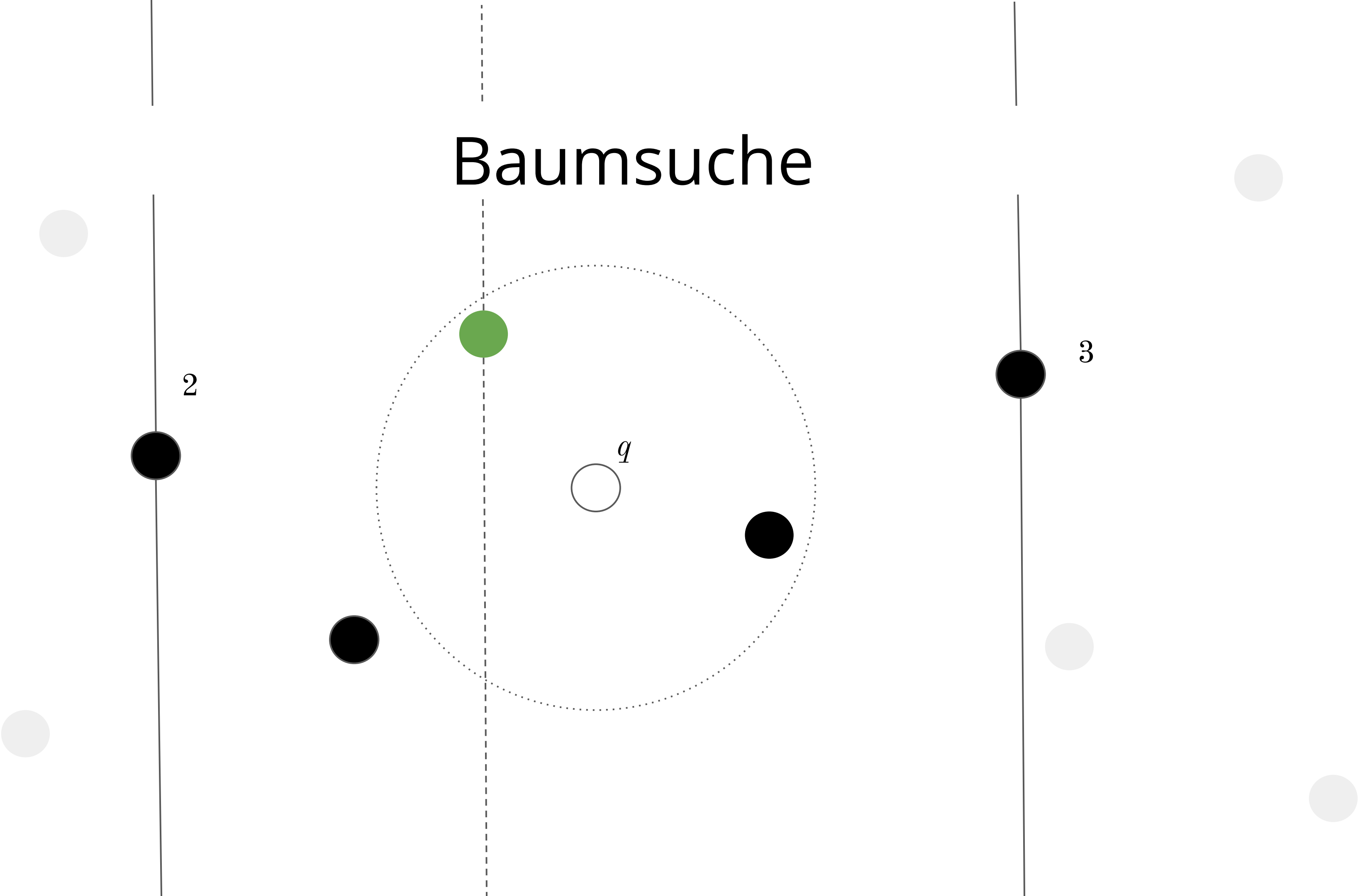
Lineare Suche



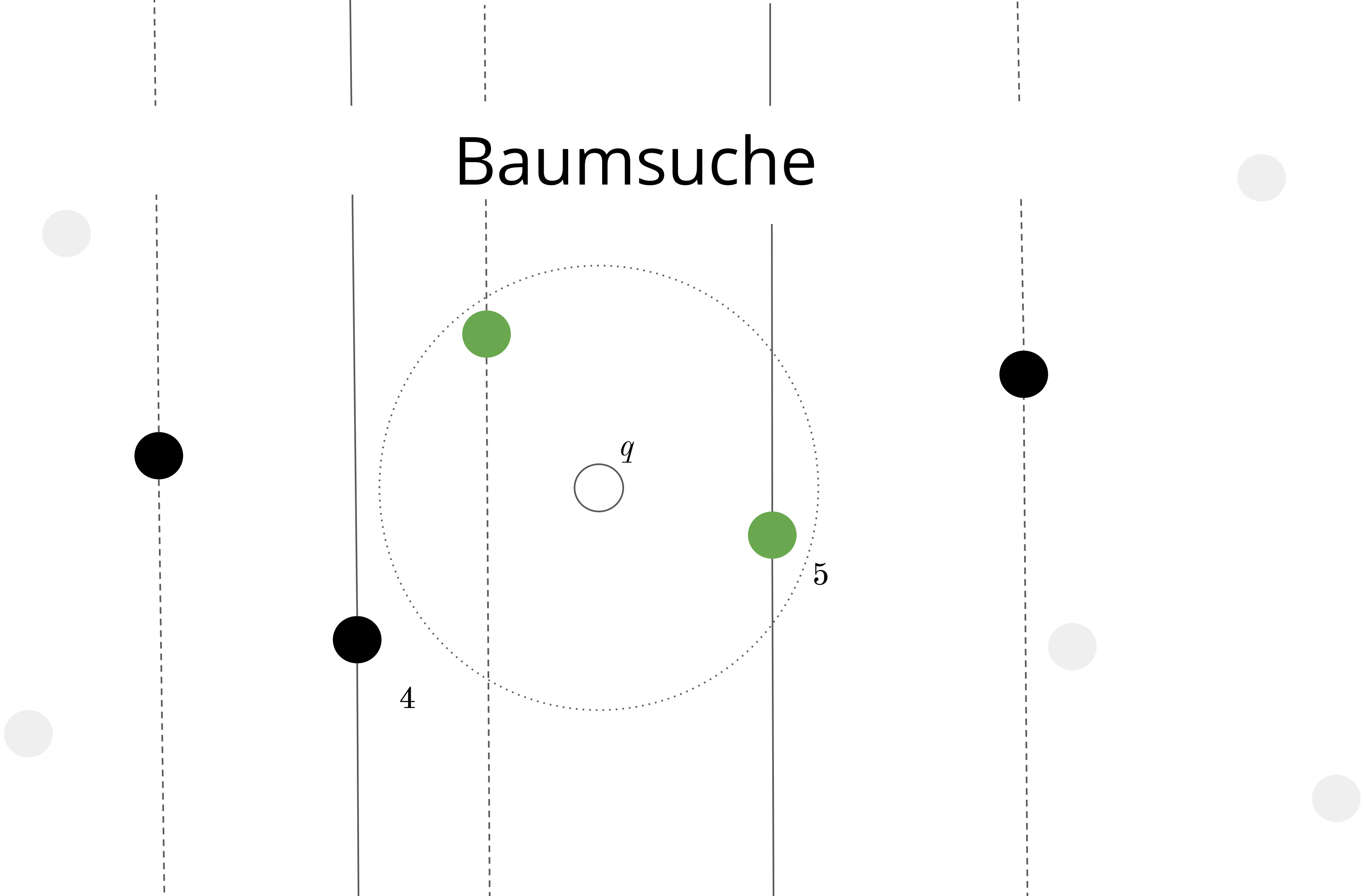
Baumsuche



Baumsuche



Baumsuche



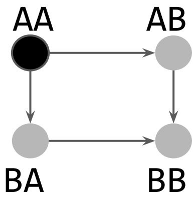
Approximative metrische Suche mit Hashfunktionen in hochdimensionalen Datensätzen

Dimension

A B

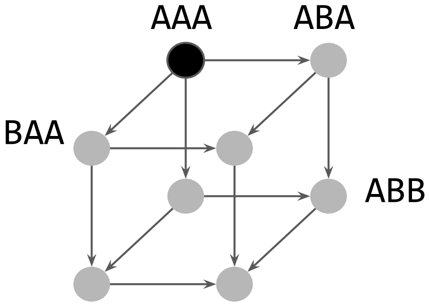
$D = 1$

AA AB
BA BB



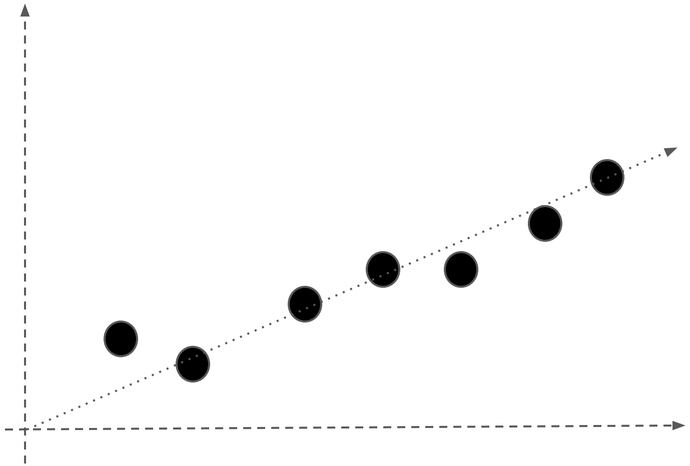
$D = 2$

AAA AAB
ABA ABB
BAA BAB
BBA BBB

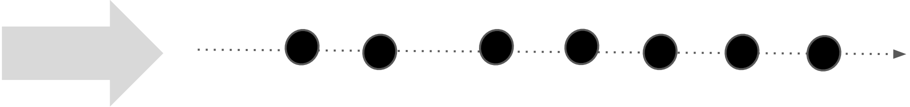


$D = 3$

Dimension einer Repräsentierung vs. intrinsische Dimension

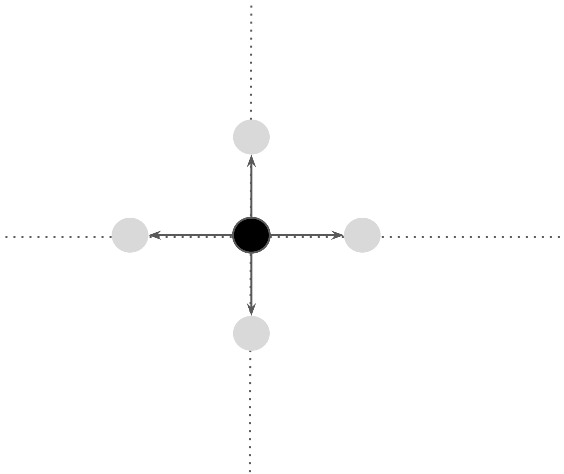


Repräsentierung: $D = 2$



Intrinsisch: $D = 1$

Niedrige und hohe Dimensionen



Reelle Ebene: $D = 2$

ACTCAAAGAAACGCCTCAAC...

ACACAAAGAAACGCCTCAAC...

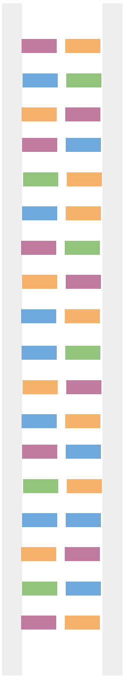
ACTGAAAGAAACGCCTCAAC...

CCTCAAAGAAACGCCTCAAC...

ACTCAAAGTAACGCCTCAAC...

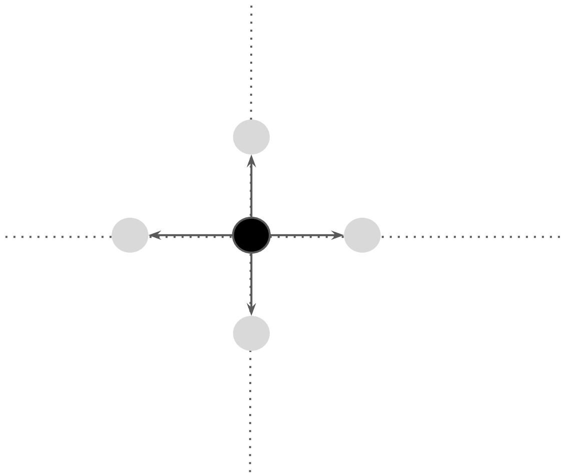
ACTCAAAGAAACCCCTCAAC...

...



Menschliches Genom: $\mathrm{D} = 3{,}2$ Mrd.

Niedrige und hohe Dimensionen

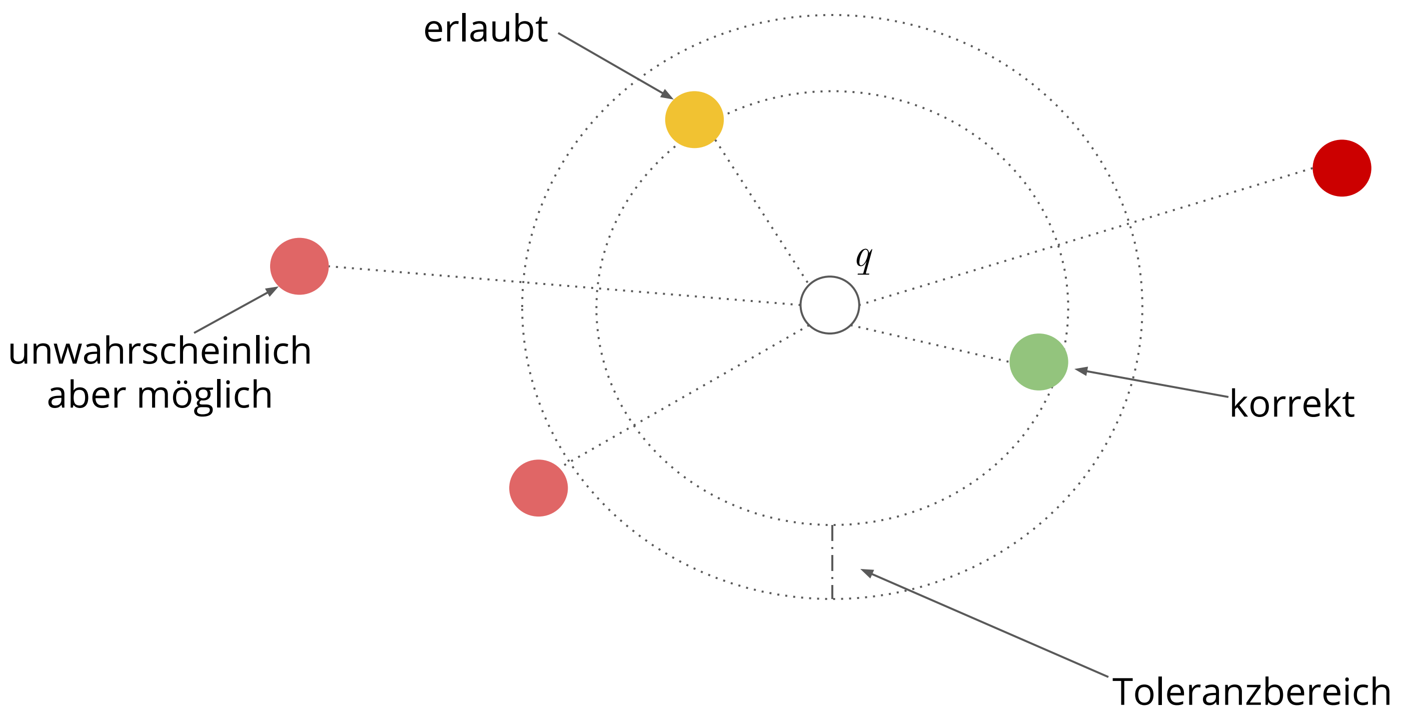


Reelle Ebene: $D = 2$

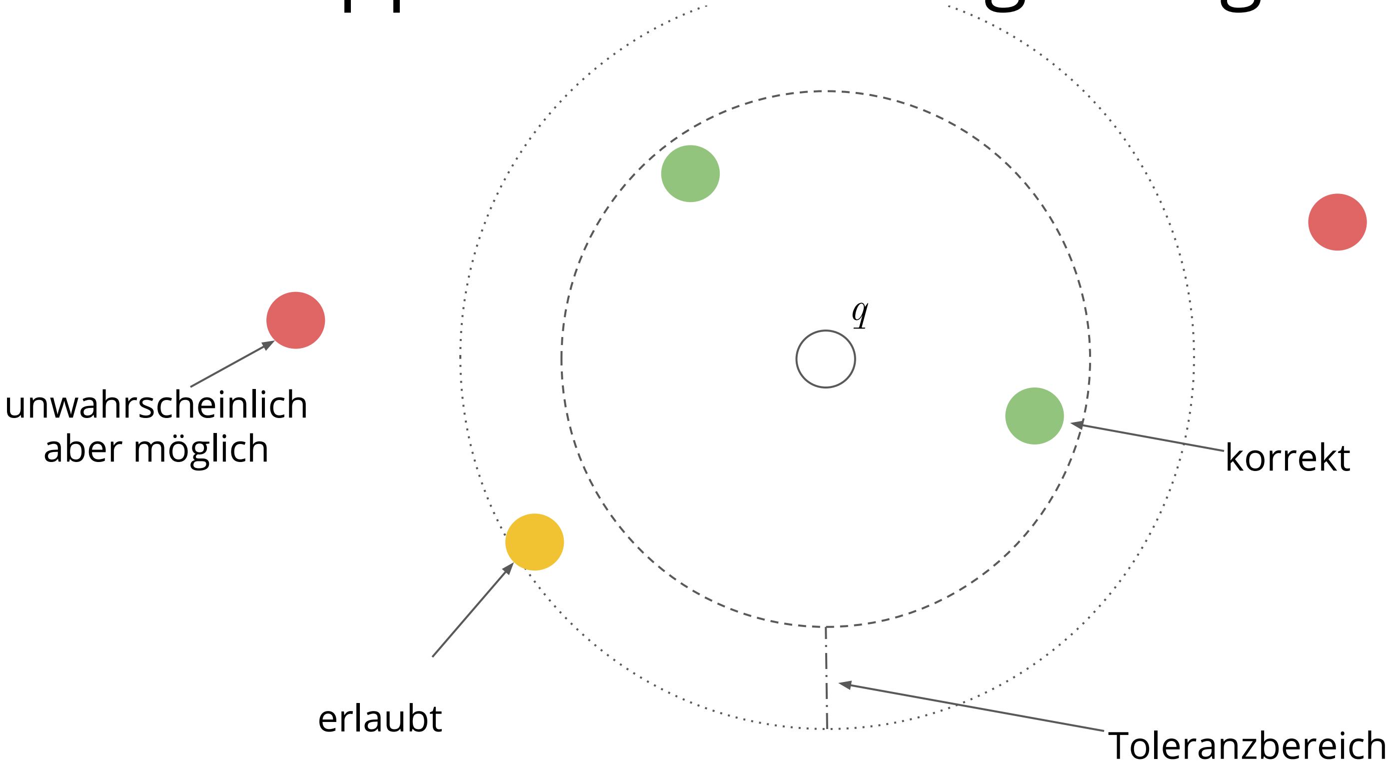
Menschliches Genom: $\mathrm{D} = 3{,}2$ Mrd.

Approximative metrische Suche mit Hashfunktionen in hochdimensionalen Datensätzen

Approximative Suche nach dem nächsten Nachbarn

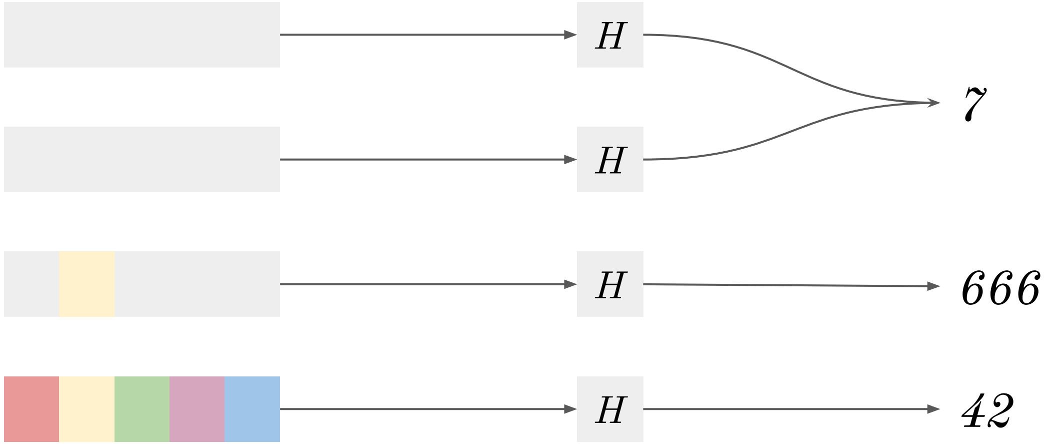


Approximative Umgebungssuche

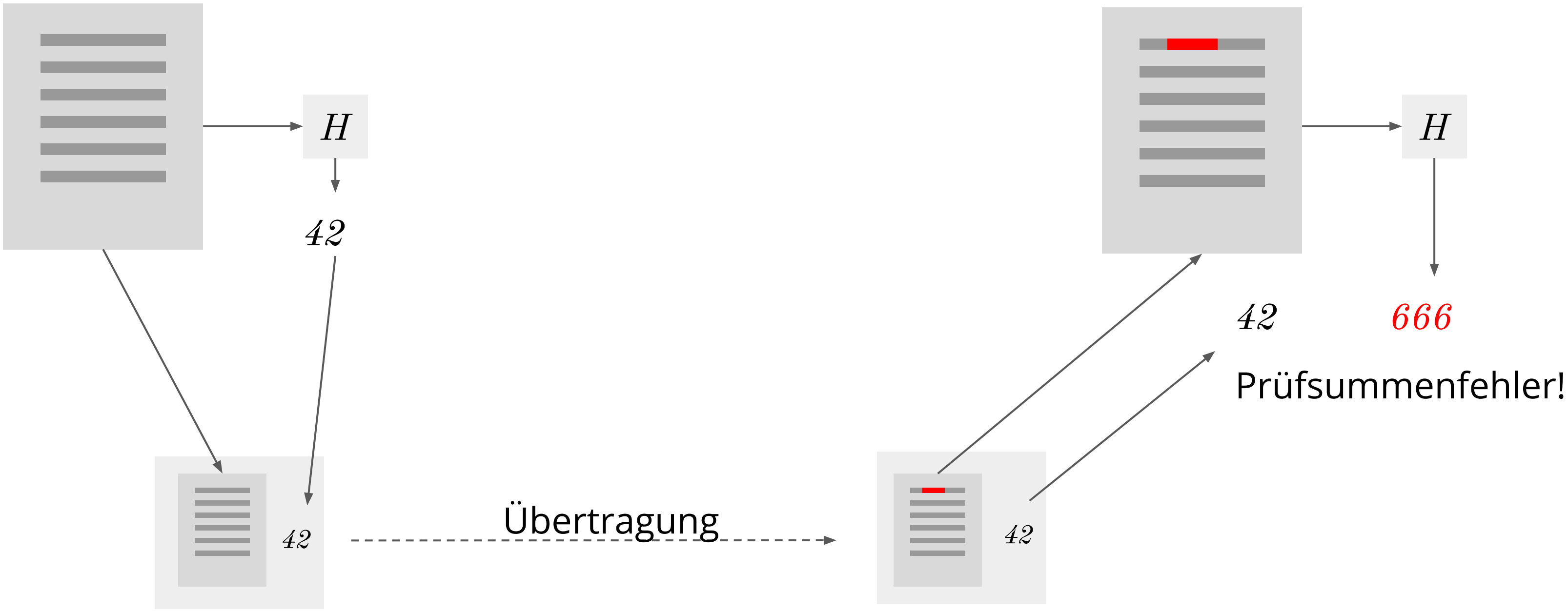


Approximative metrische Suche mit Hashfunktionen in hochdimensionalen Datensätzen

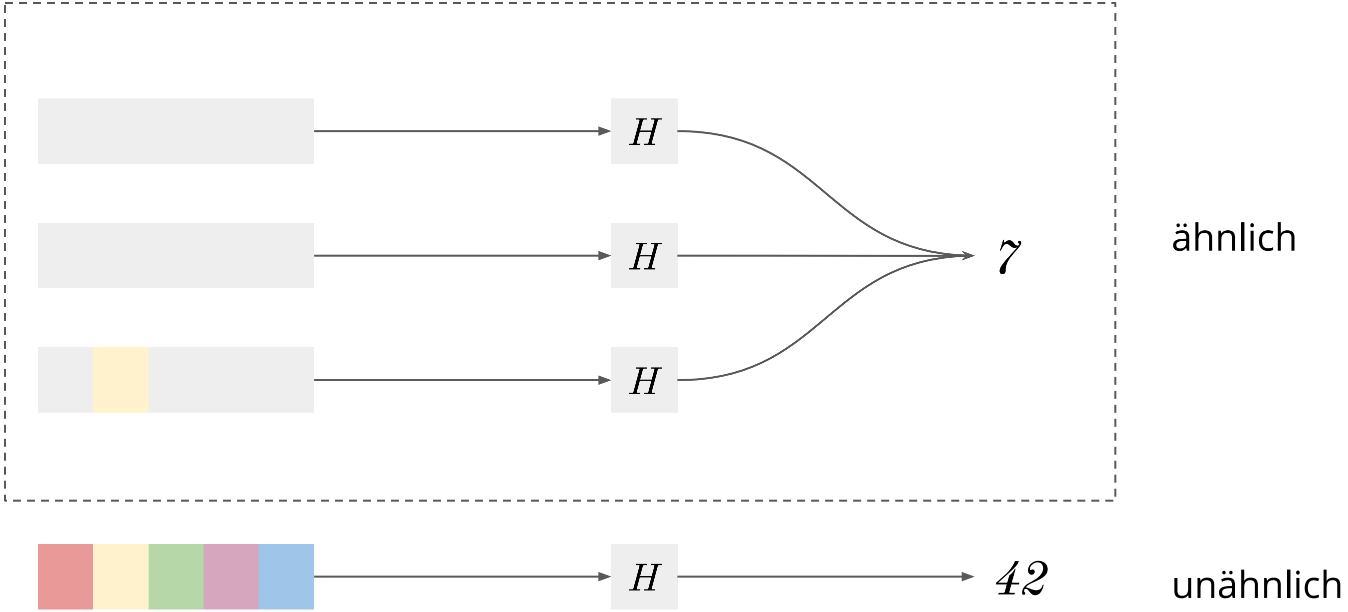
Klassische Hashfunktionen



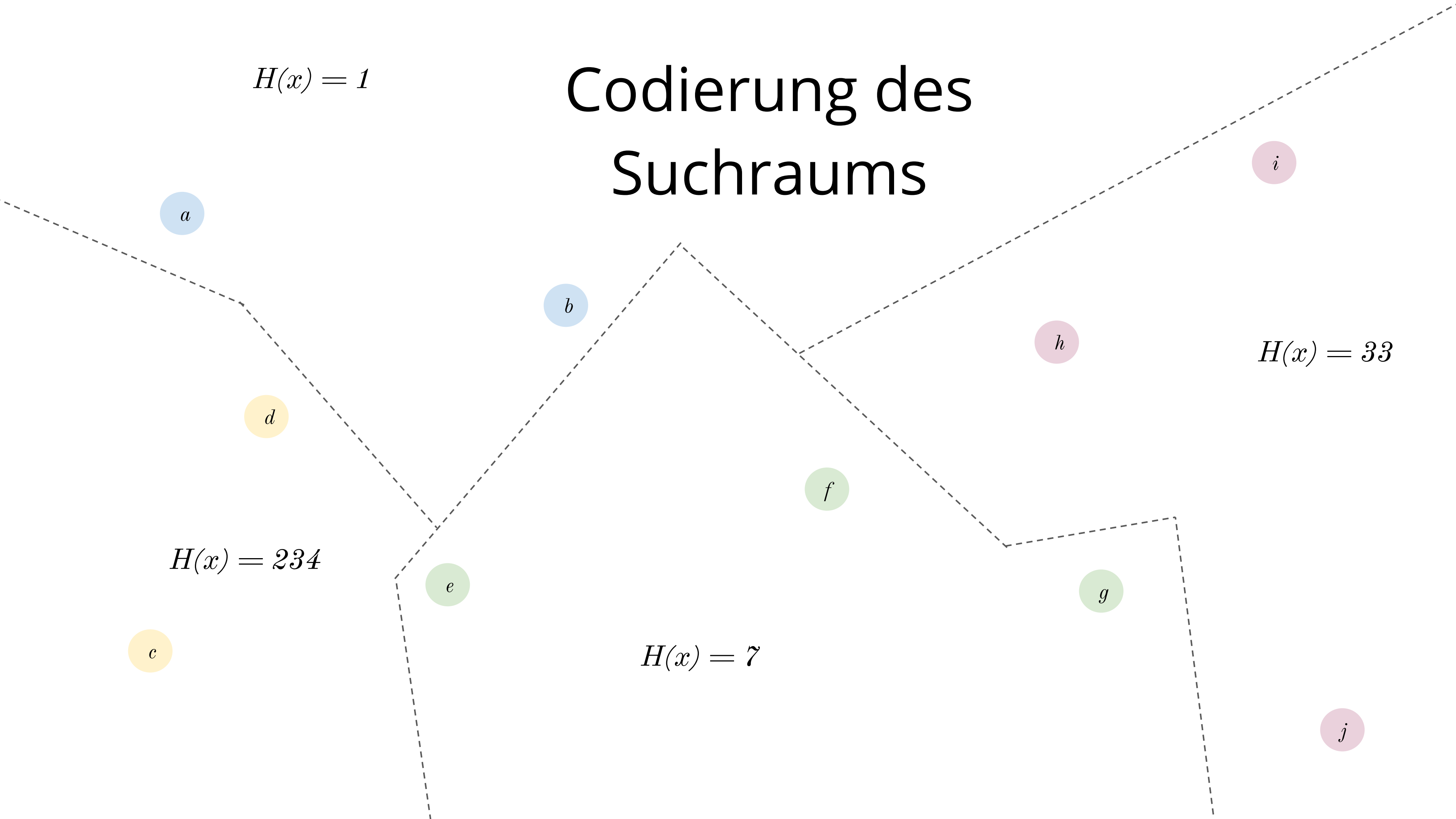
Klassische Hashfunktionen



lokal-sensitive Hashfunktionen



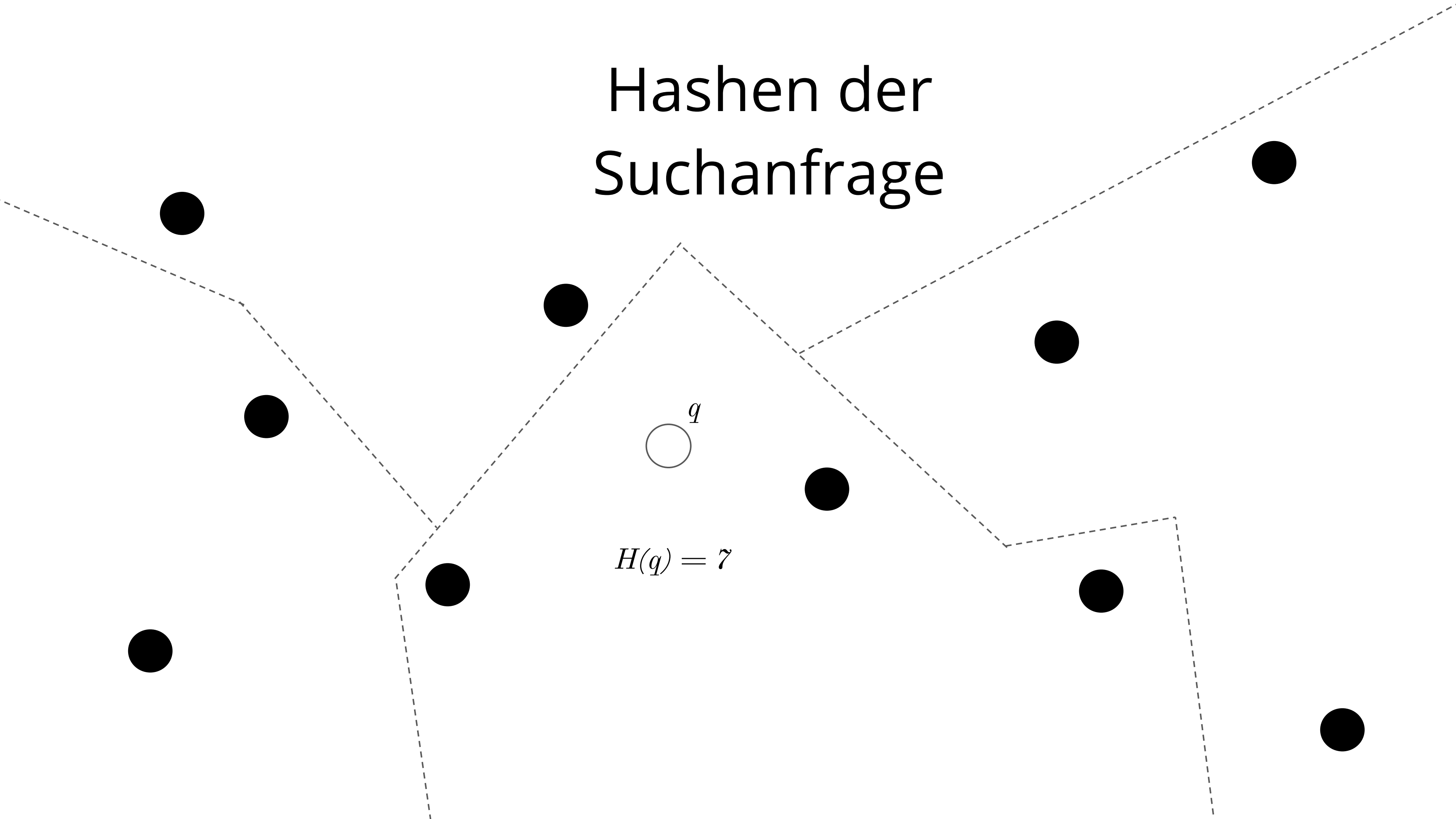
Codierung des Suchraums



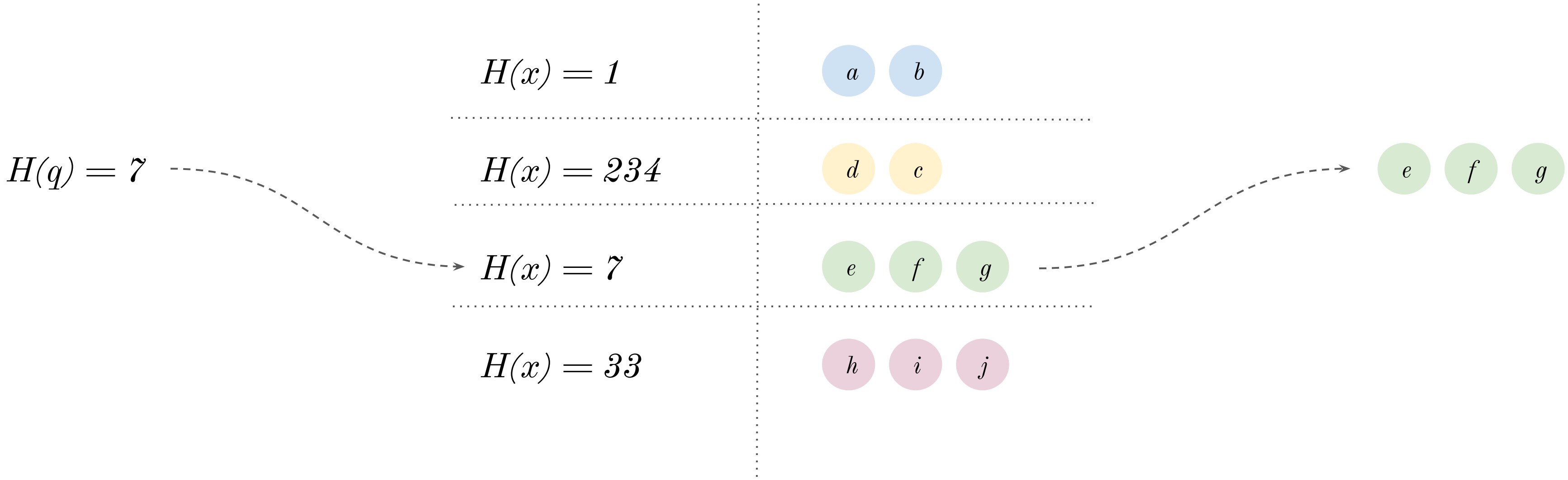
Anlegen einer Codetabelle

$H(x) = 1$	a b
$H(x) = 234$	d c
$H(x) = 7$	e f g
$H(x) = 33$	h i j

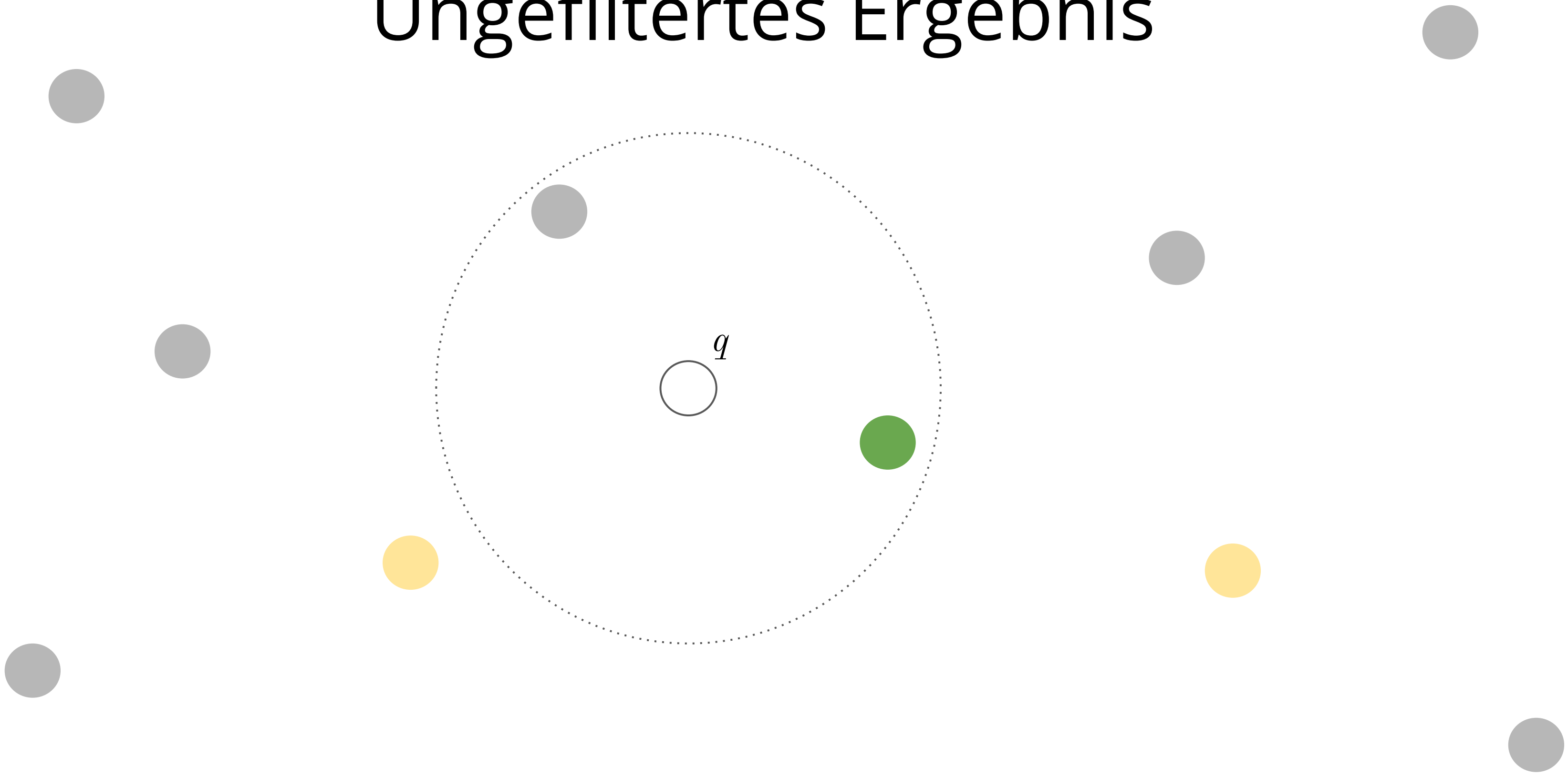
Hashen der Suchanfrage



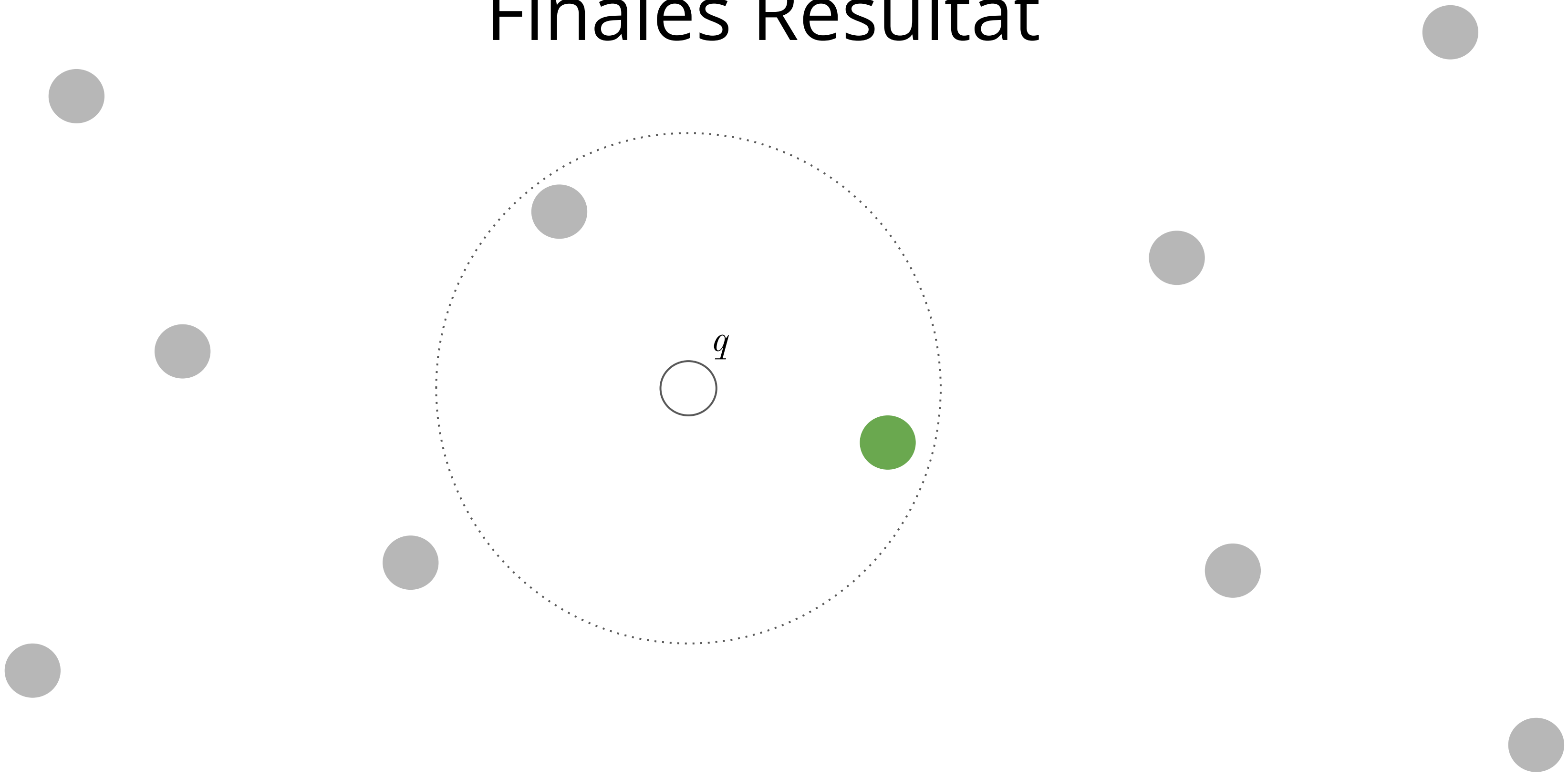
Nachschauen in der Codetabelle



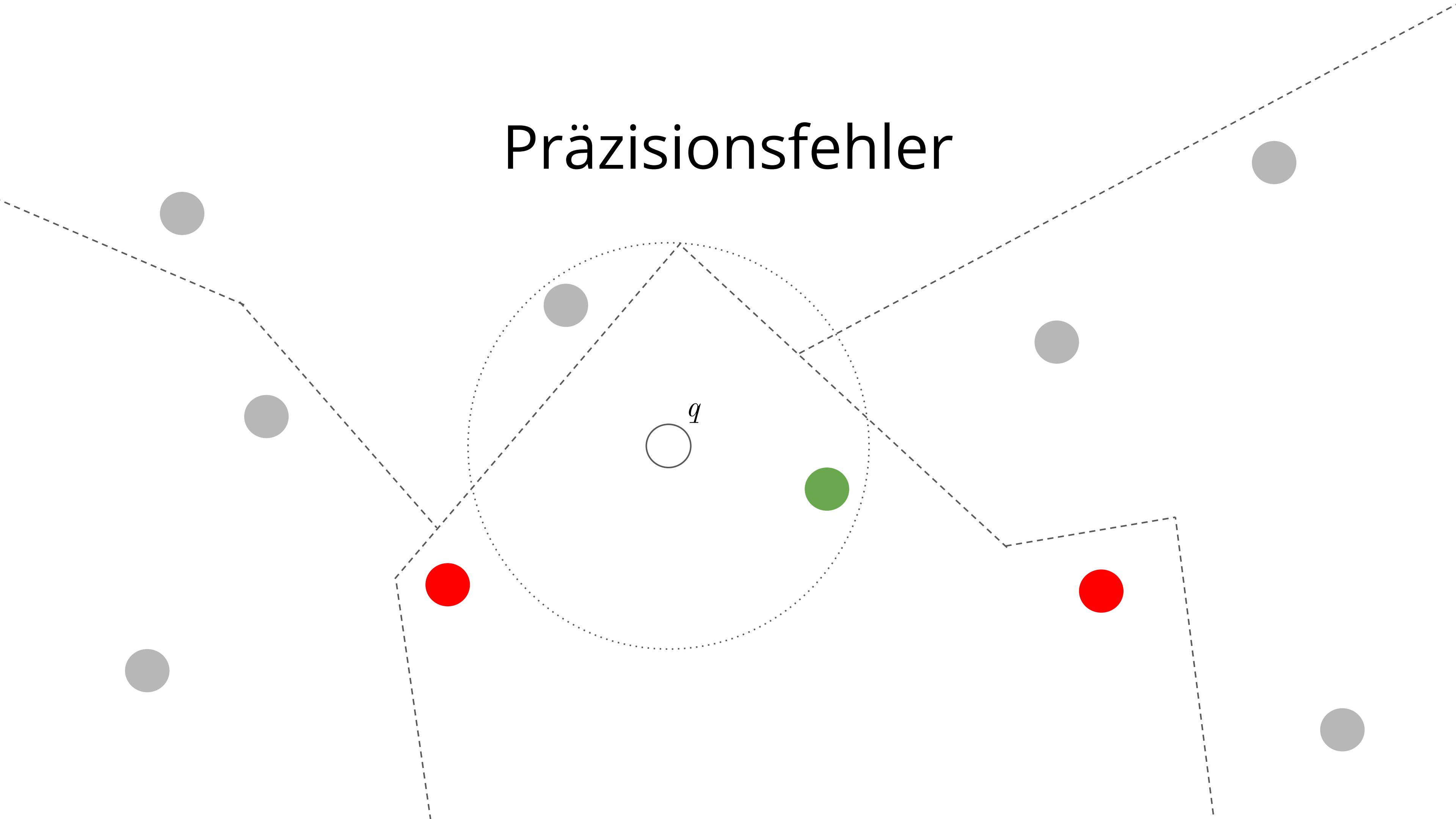
Ungefiltertes Ergebnis



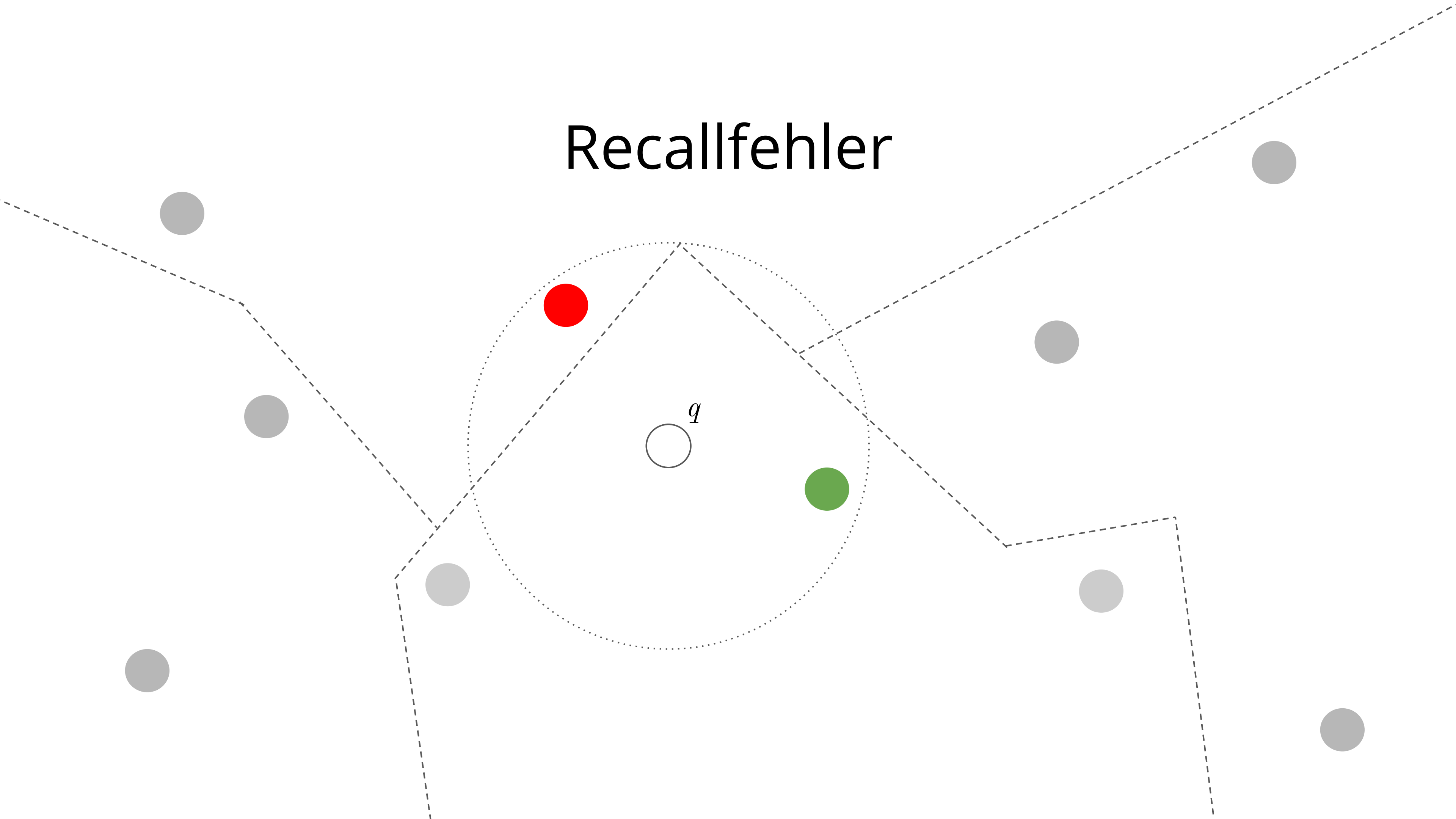
Finales Resultat



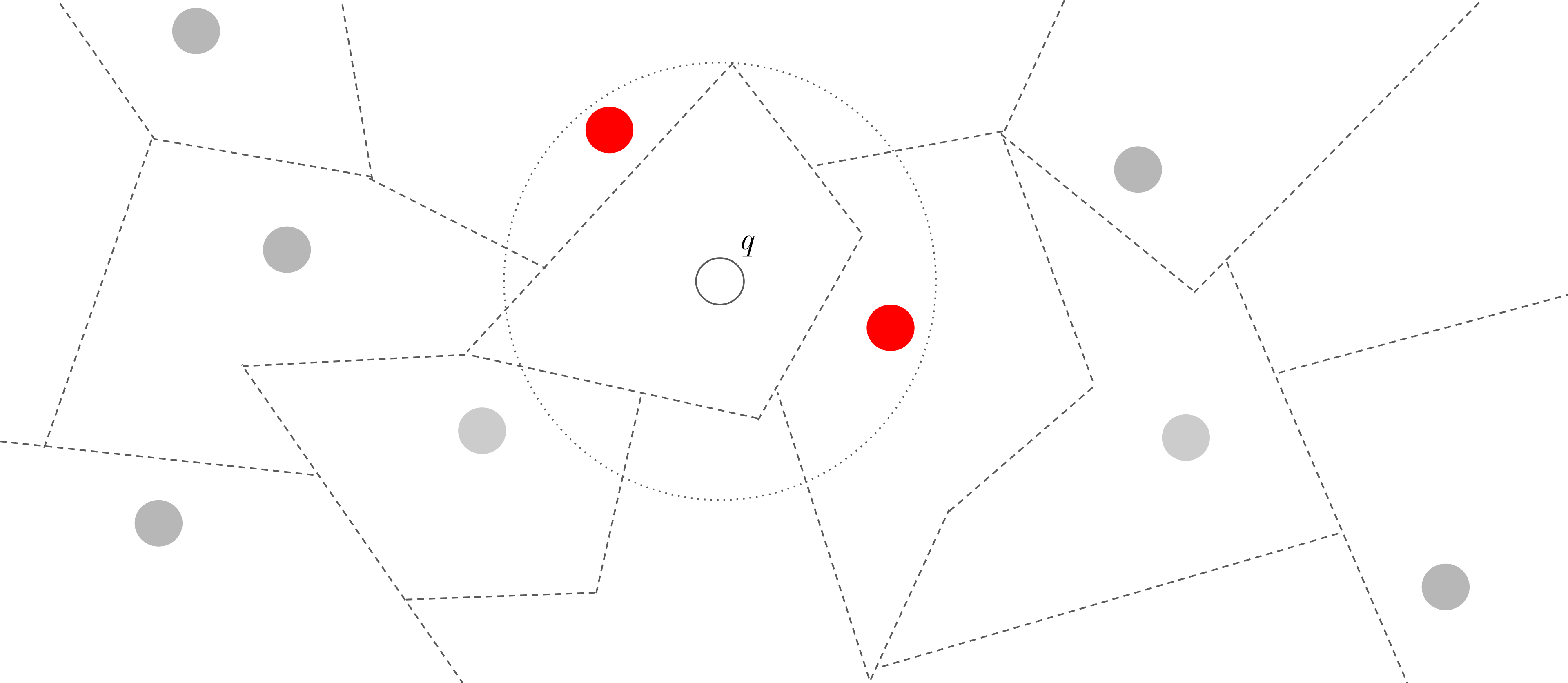
Präzisionsfehler



Recallfehler

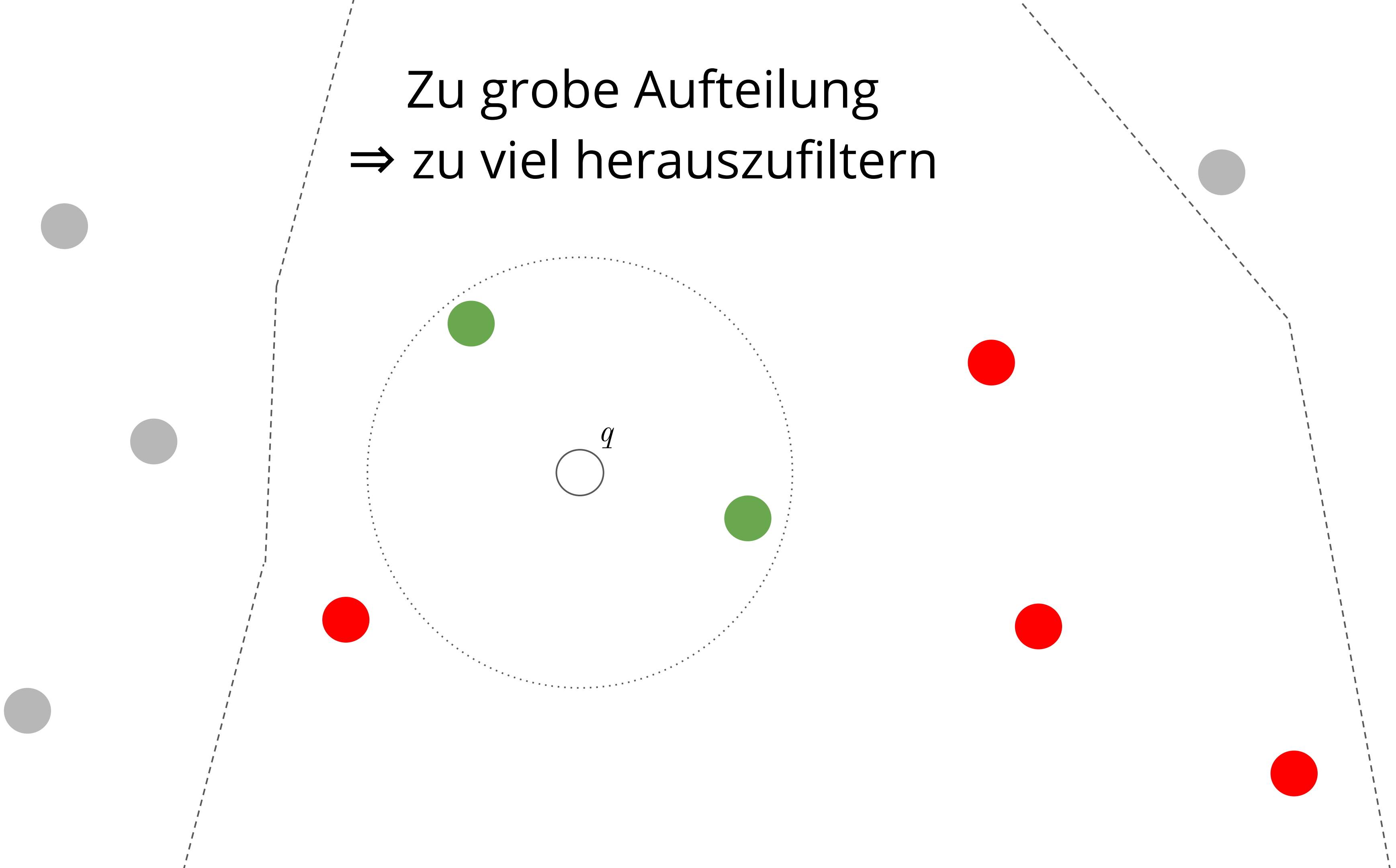


Zu feine Aufteilung
⇒ Zu schlechter Recall

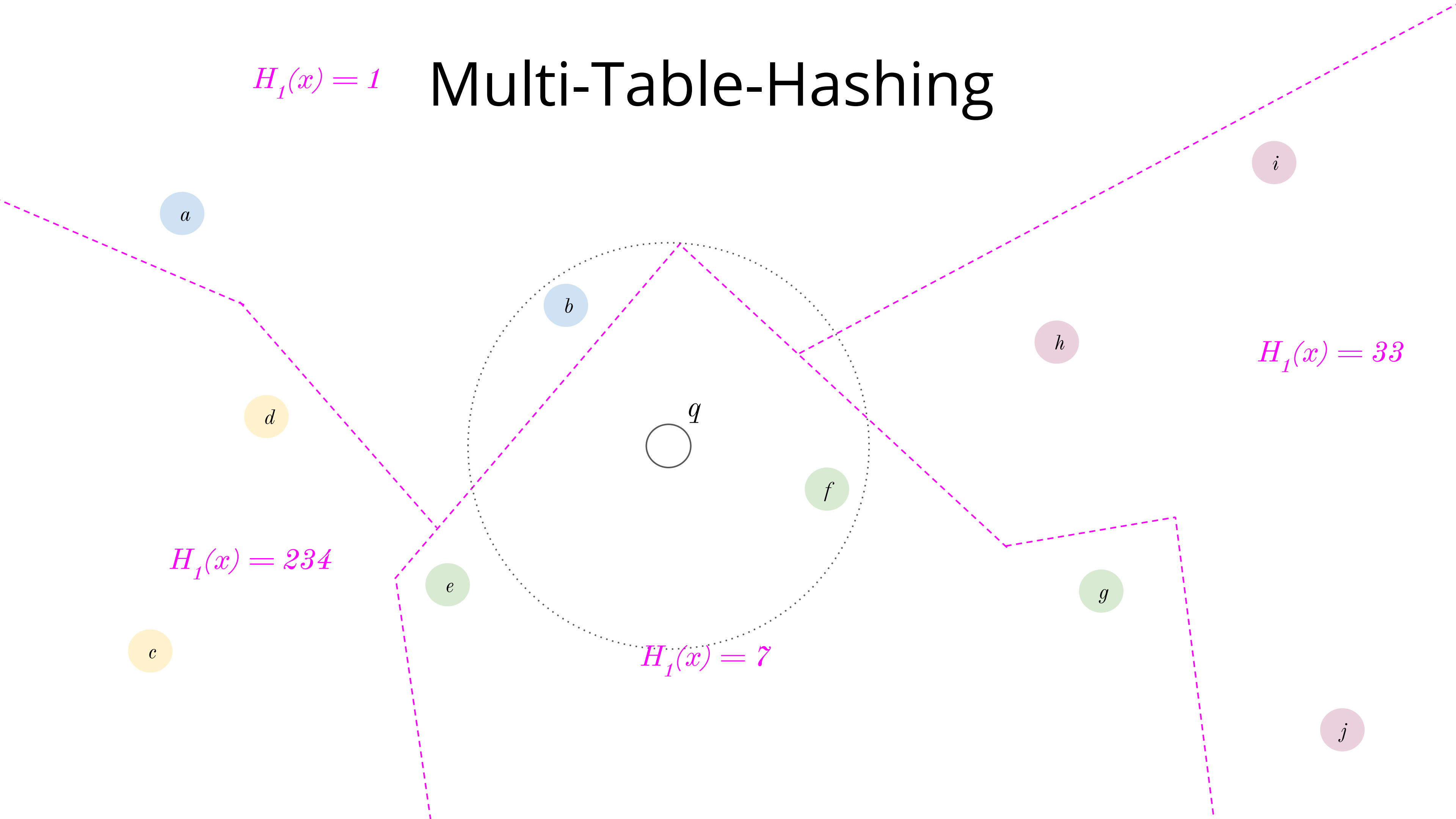


Zu grobe Aufteilung

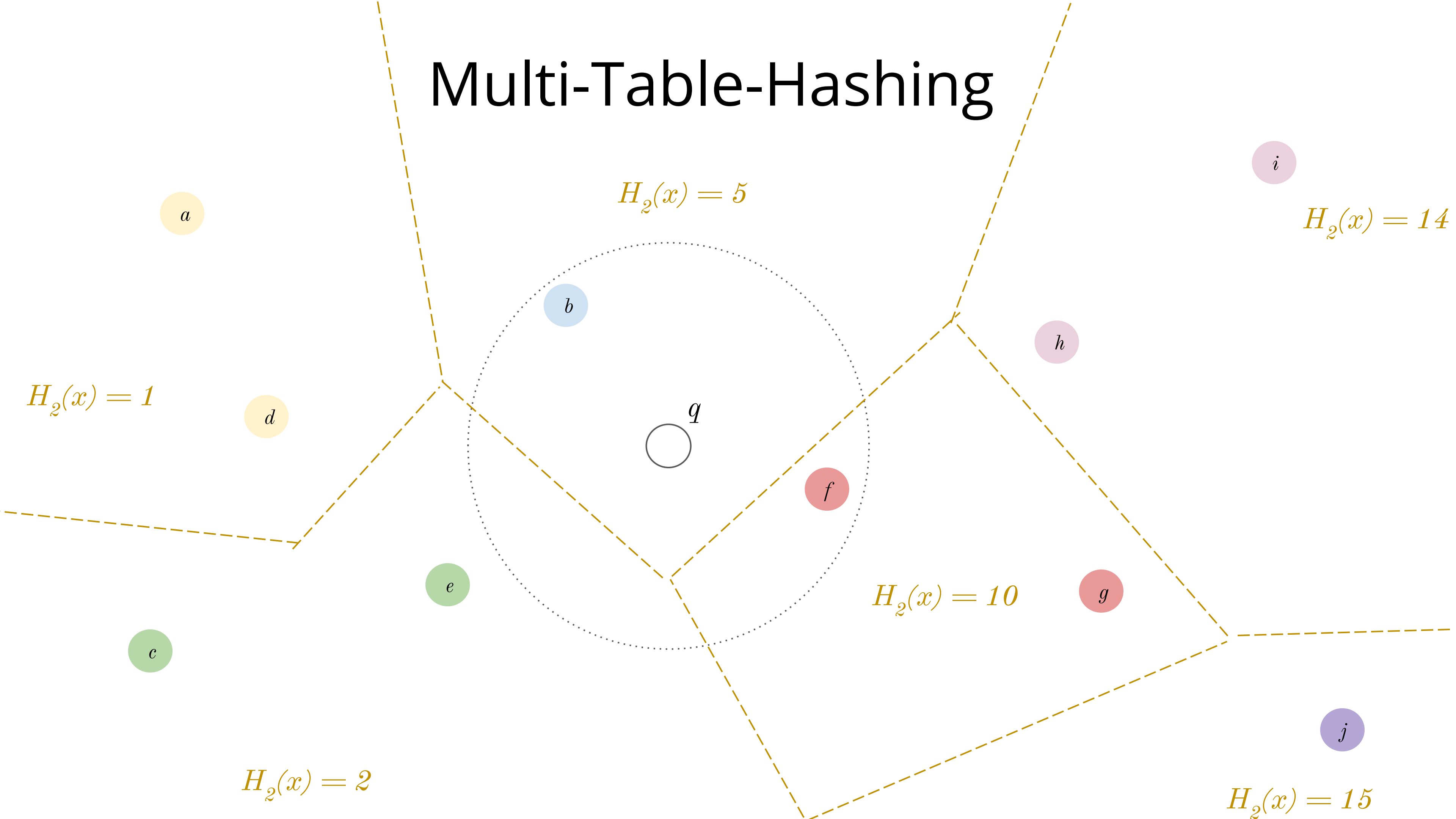
⇒ zu viel herauszufiltern



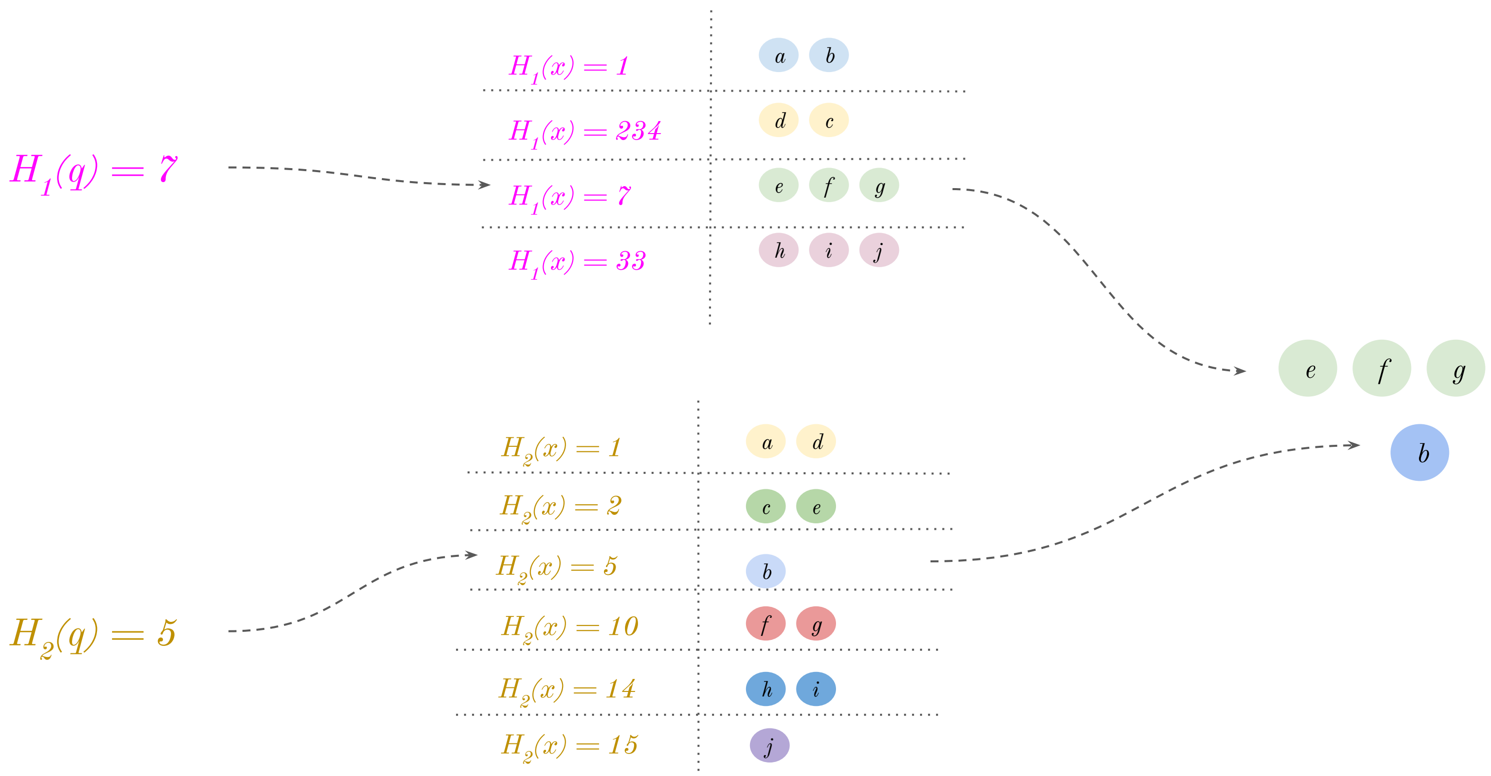
Multi-Table-Hashing



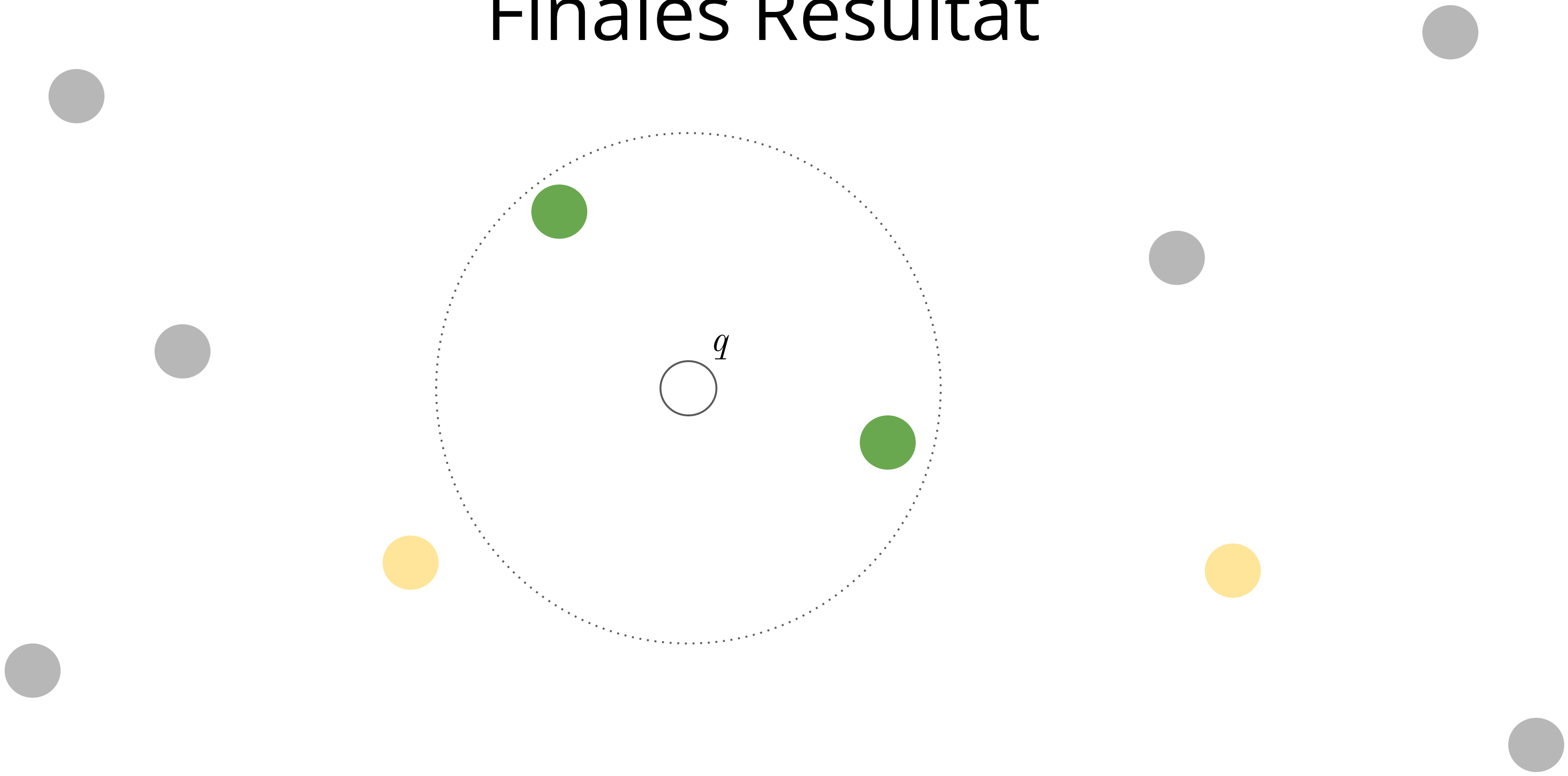
Multi-Table-Hashing



Redundantes Speichern und Abrufen in zwei Tabellen



Finales Resultat



Bearbeitete Fragestellungen

1. Einordnung existierender Hashverfahren hinsichtlich ihrer Konstruktionsprinzipien

2. Abschätzungen einer oberen und einer unteren Schranke der Anzahl von benötigten Hashfunktionen für ideales Multi-Table-Hashing

Bearbeitete Fragestellungen

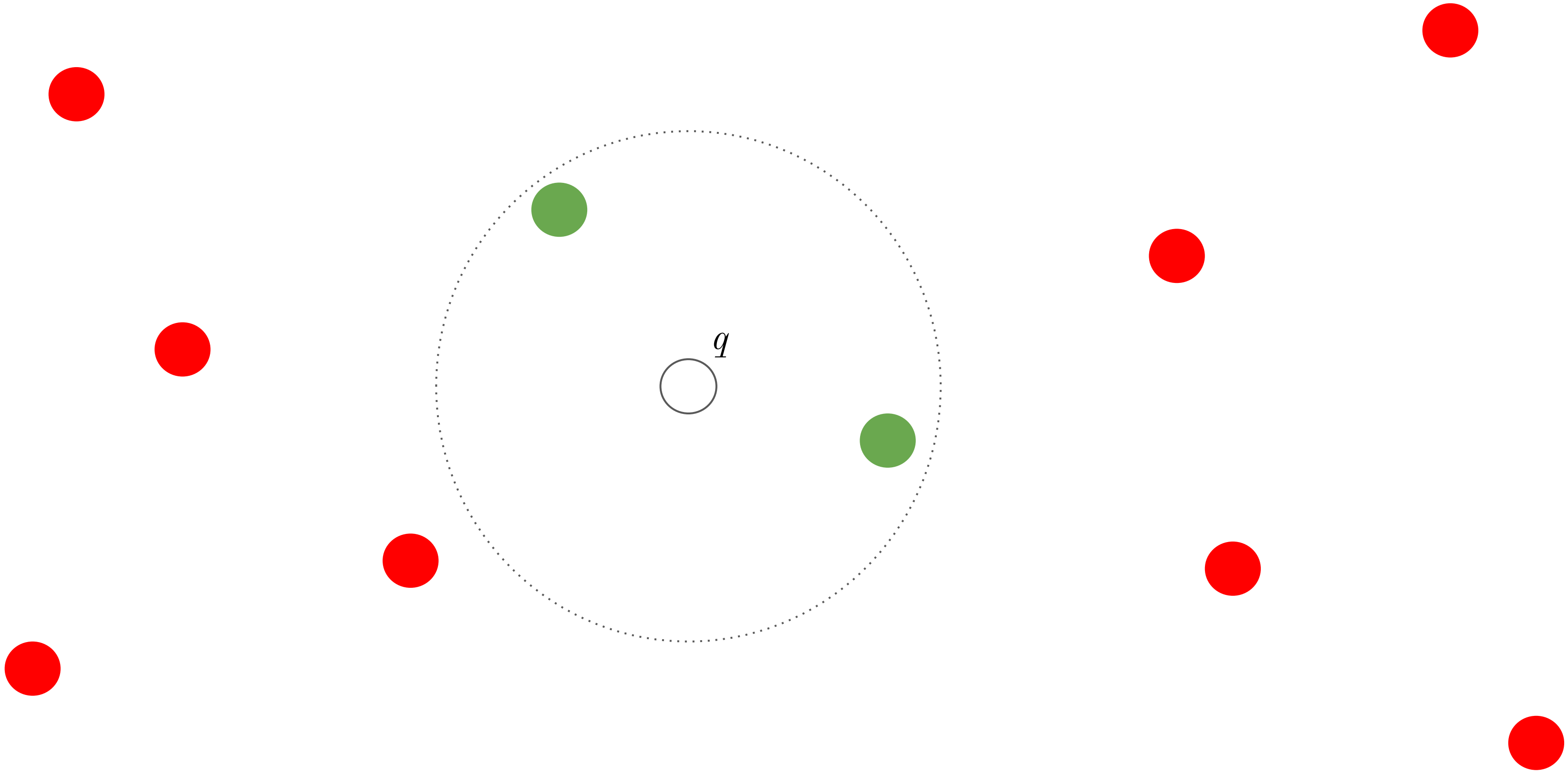
1. Einordnung existierender Hashverfahren hinsichtlich ihrer Konstruktionsprinzipien

2. Abschätzungen einer oberen und einer unteren Schranke der Anzahl von benötigten Hashfunktionen für ideales Multi-Table-Hashing

⇒ Wieviele Tabellen/Hashfunktionen sind notwendig für vollen Recall?

Trivial: Eine

$H(x) = 0$



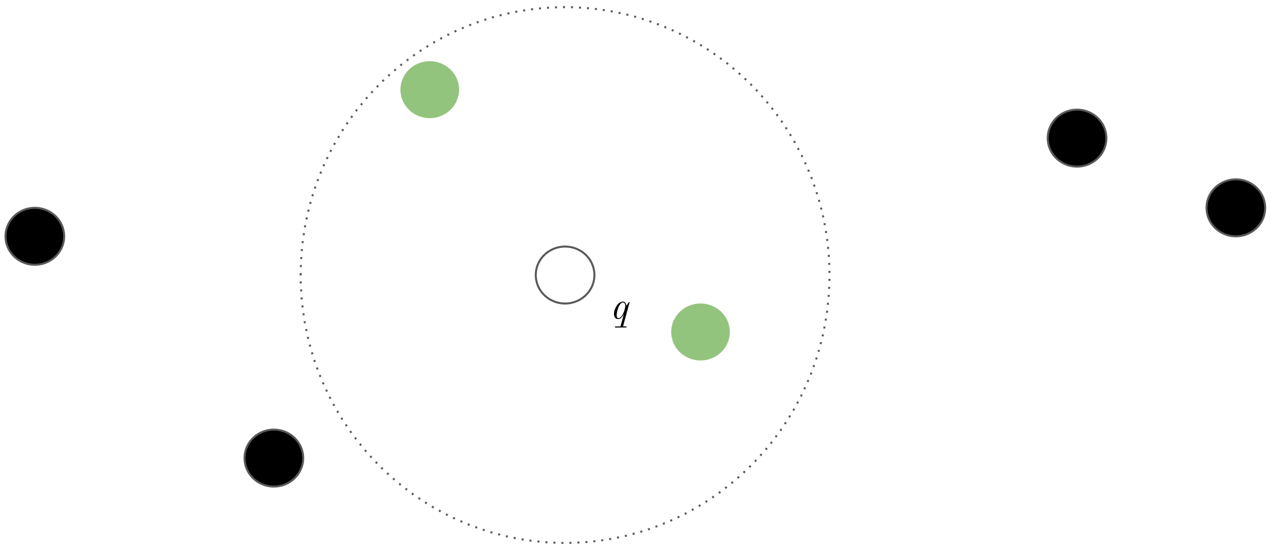
Bearbeitete Fragestellungen

1. Einordnung existierender Hashverfahren hinsichtlich ihrer Konstruktionsprinzipien

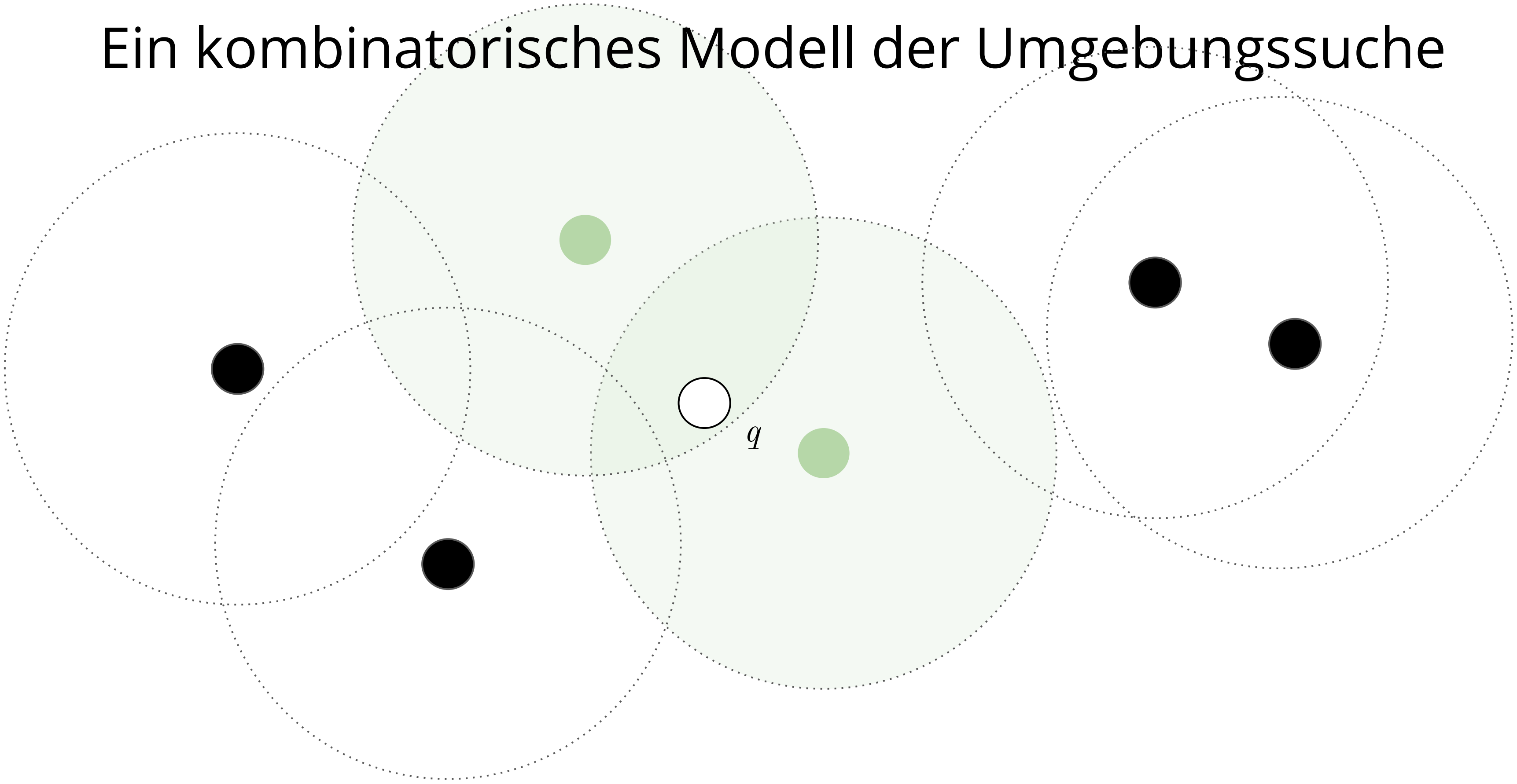
2. Abschätzungen einer oberen und einer unteren Schranke der Anzahl von benötigten Hashfunktionen für ideales Multi-Table-Hashing

⇒ Wieviele Tabellen/Hashfunktionen sind notwendig für vollen Recall, wenn alle Hashfunktionen keine Präzisionsfehler machen dürfen?

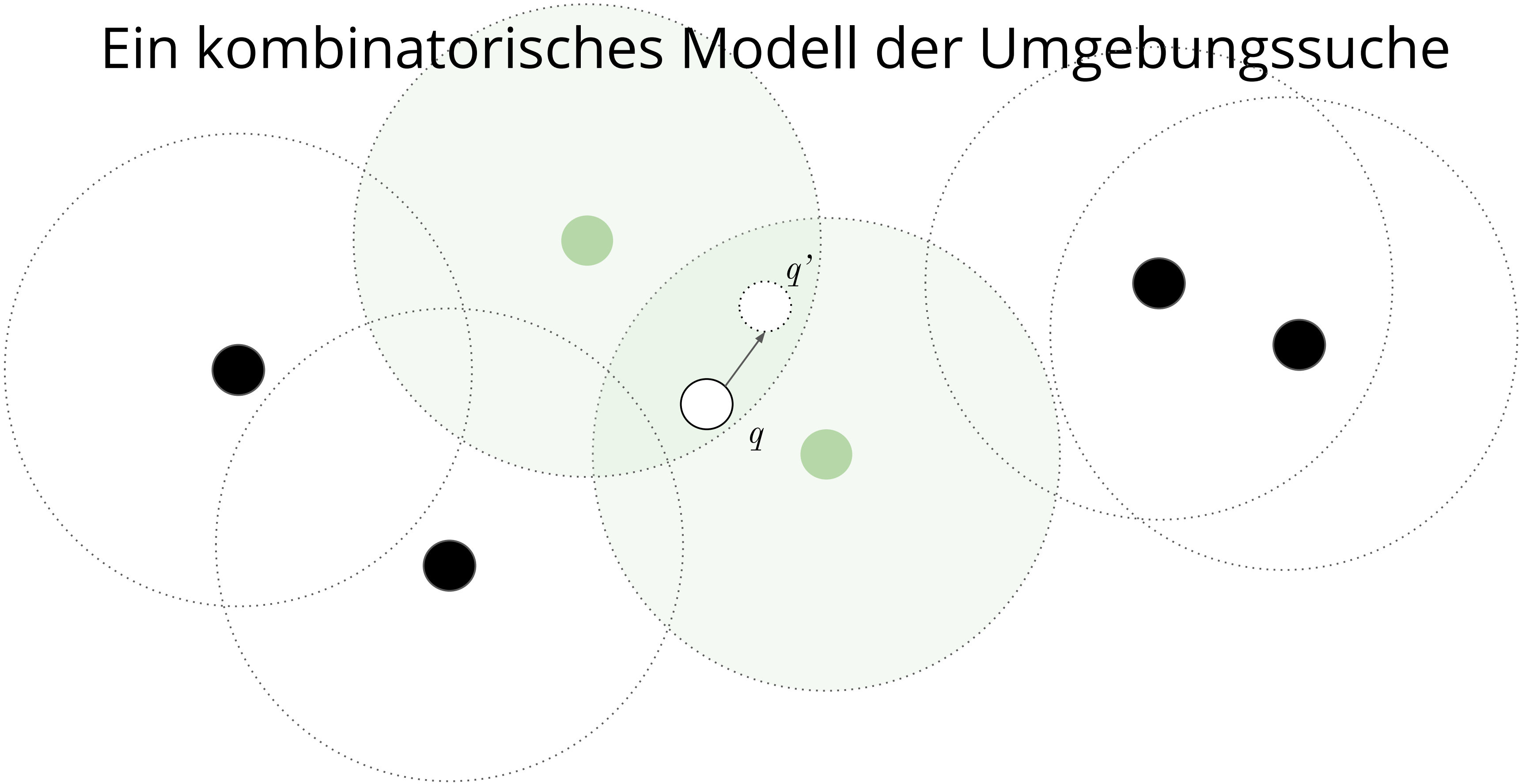
Ein kombinatorisches Modell der Umgebungssuche



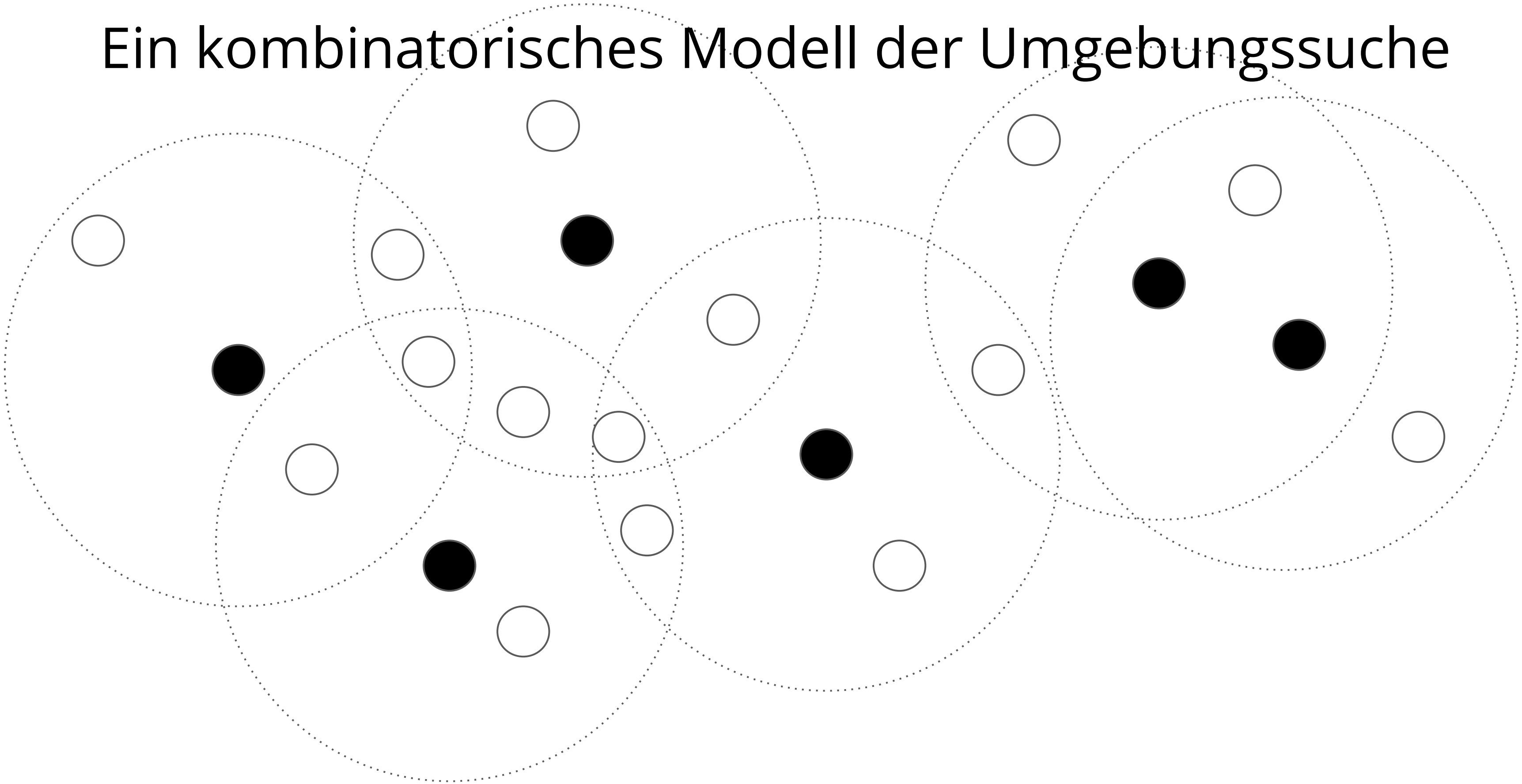
Ein kombinatorisches Modell der Umgebungssuche



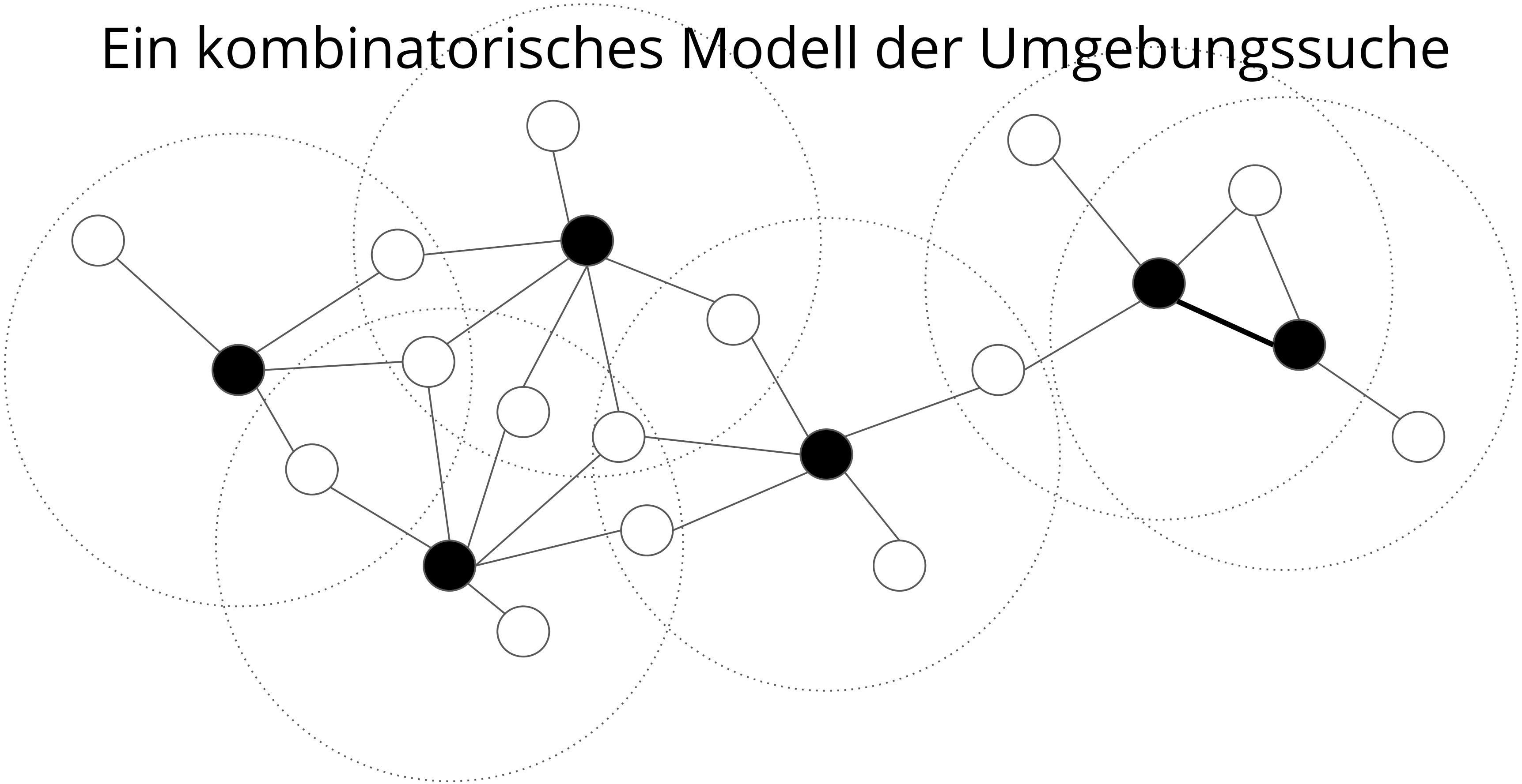
Ein kombinatorisches Modell der Umgebungssuche



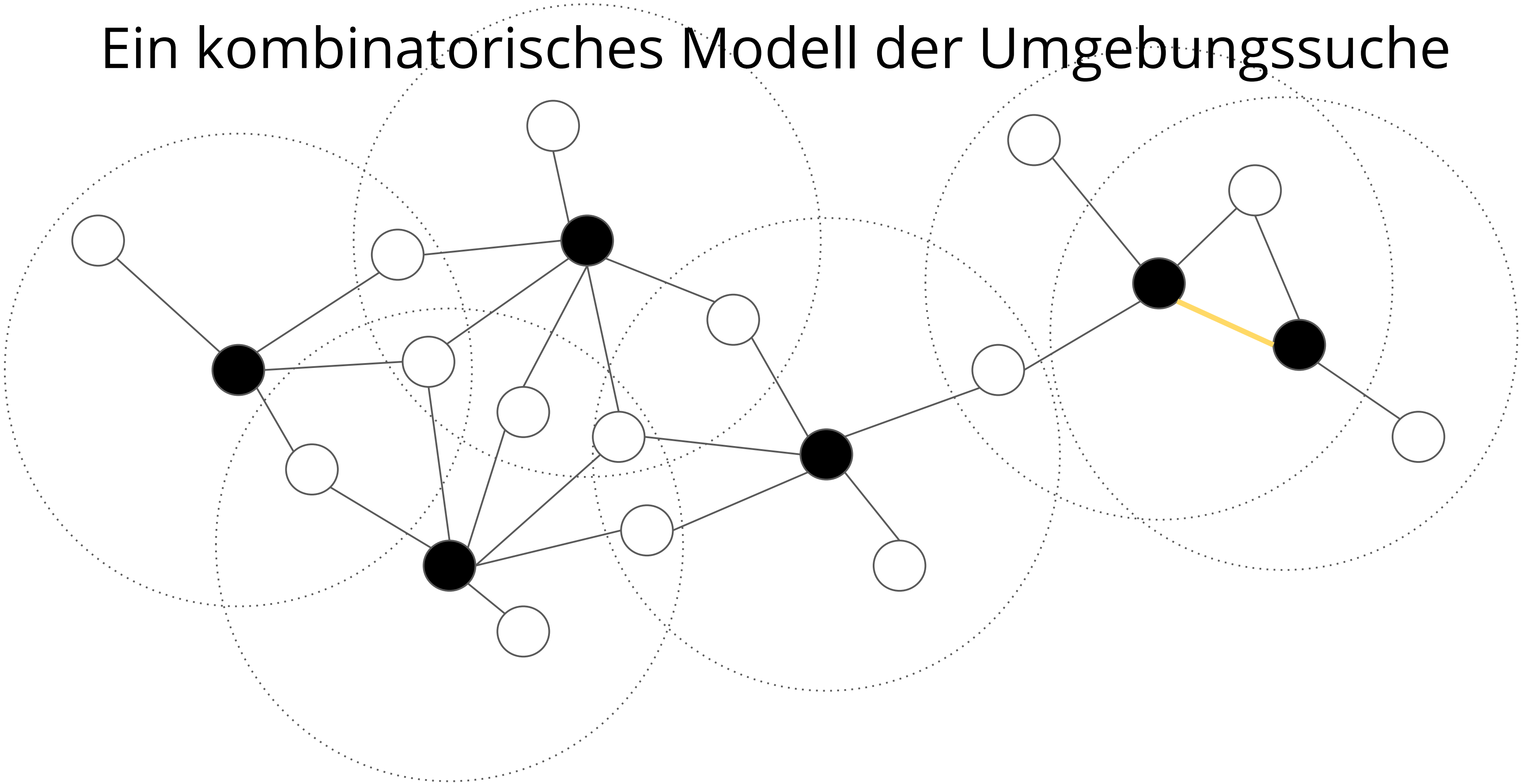
Ein kombinatorisches Modell der Umgebungssuche



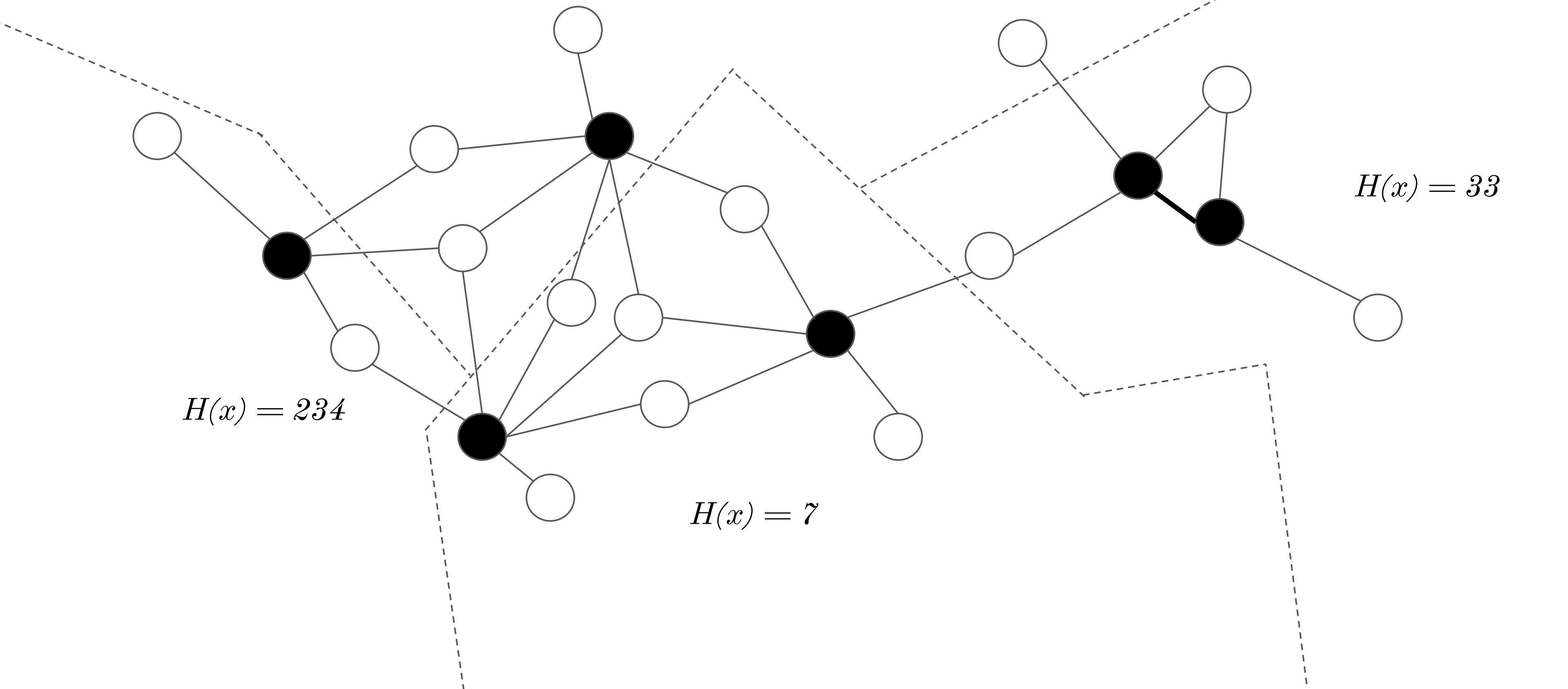
Ein kombinatorisches Modell der Umgebungssuche



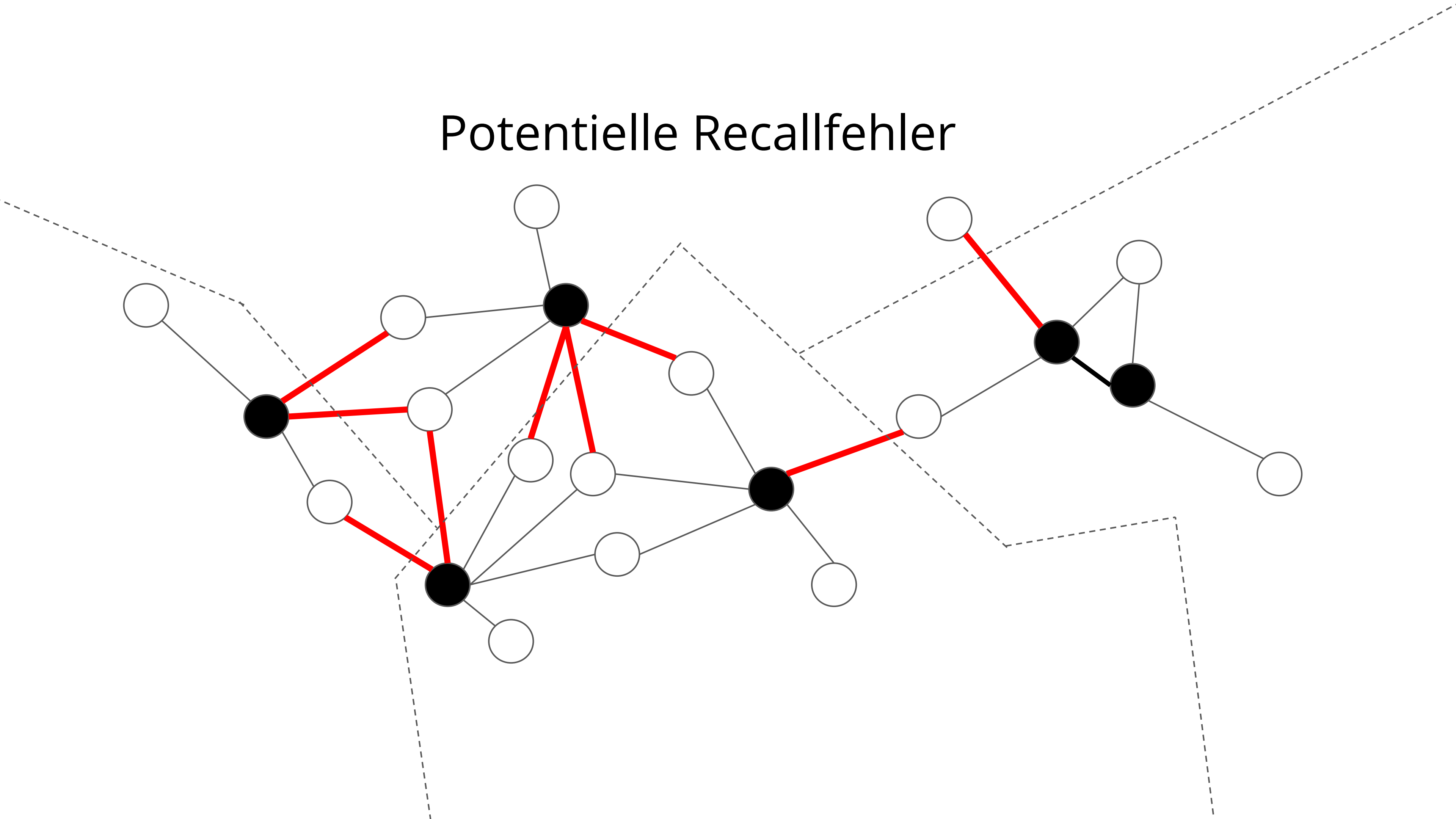
Ein kombinatorisches Modell der Umgebungssuche



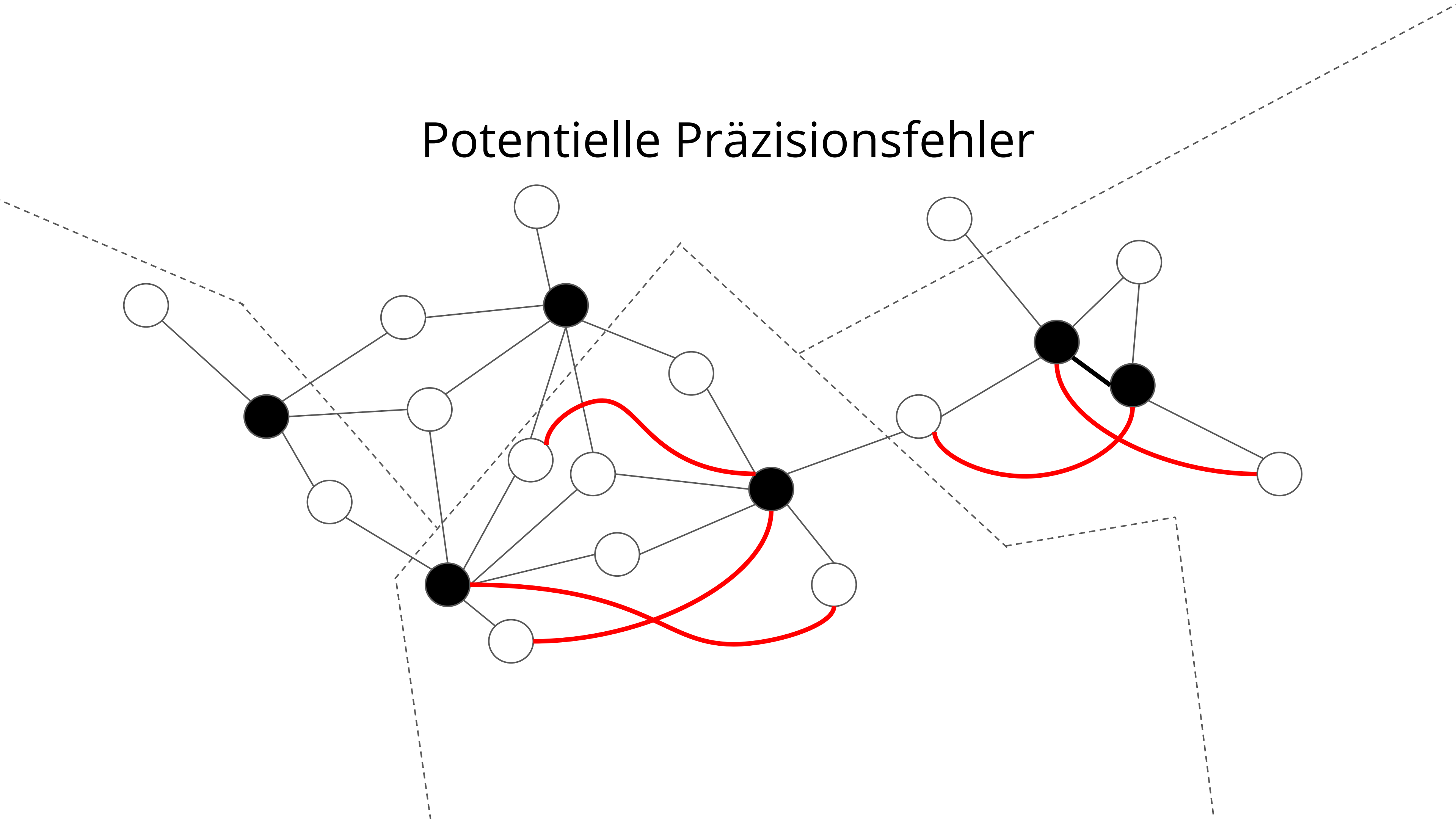
Was macht eine Hashfunktion?



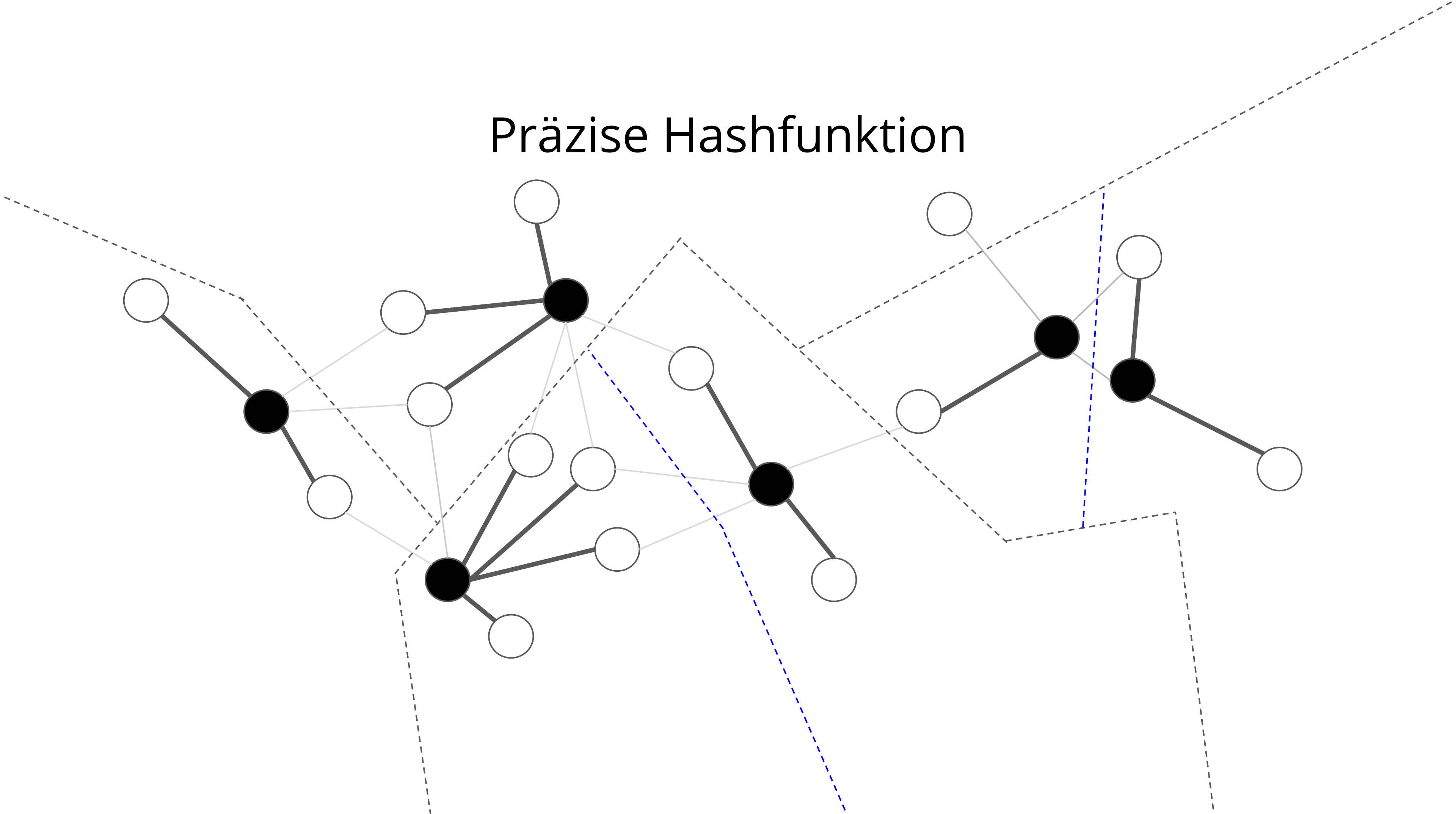
Potentielle Recallfehler



Potentielle Präzisionsfehler

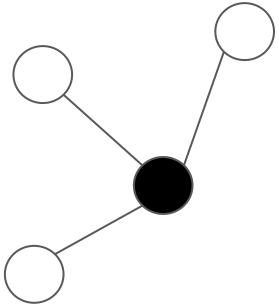


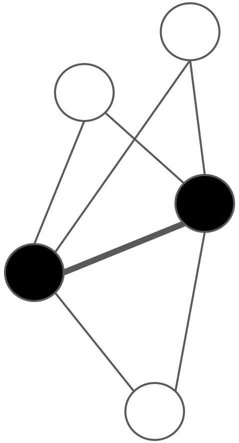
Präzise Hashfunktion



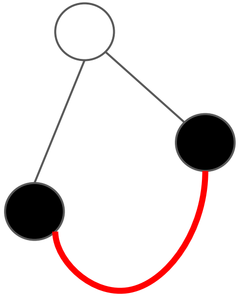
Beobachtung 1a:

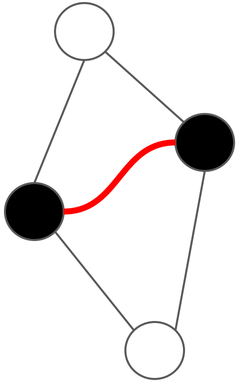
Datenpunkte im Hashgebiet einer präzisen Funktion müssen mit **allen** anderen darin enthaltenen Knoten zusammenhängen





Erlaubt

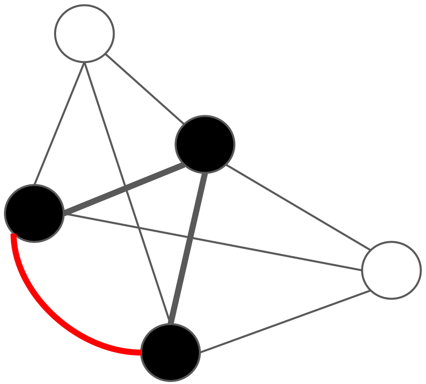




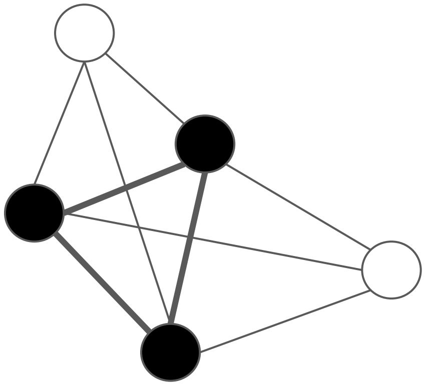
Nicht erlaubt

Beobachtung 1a:

Datenpunkte im gleichen Hashgebiet einer präzisen Funktion bilden eine Clique

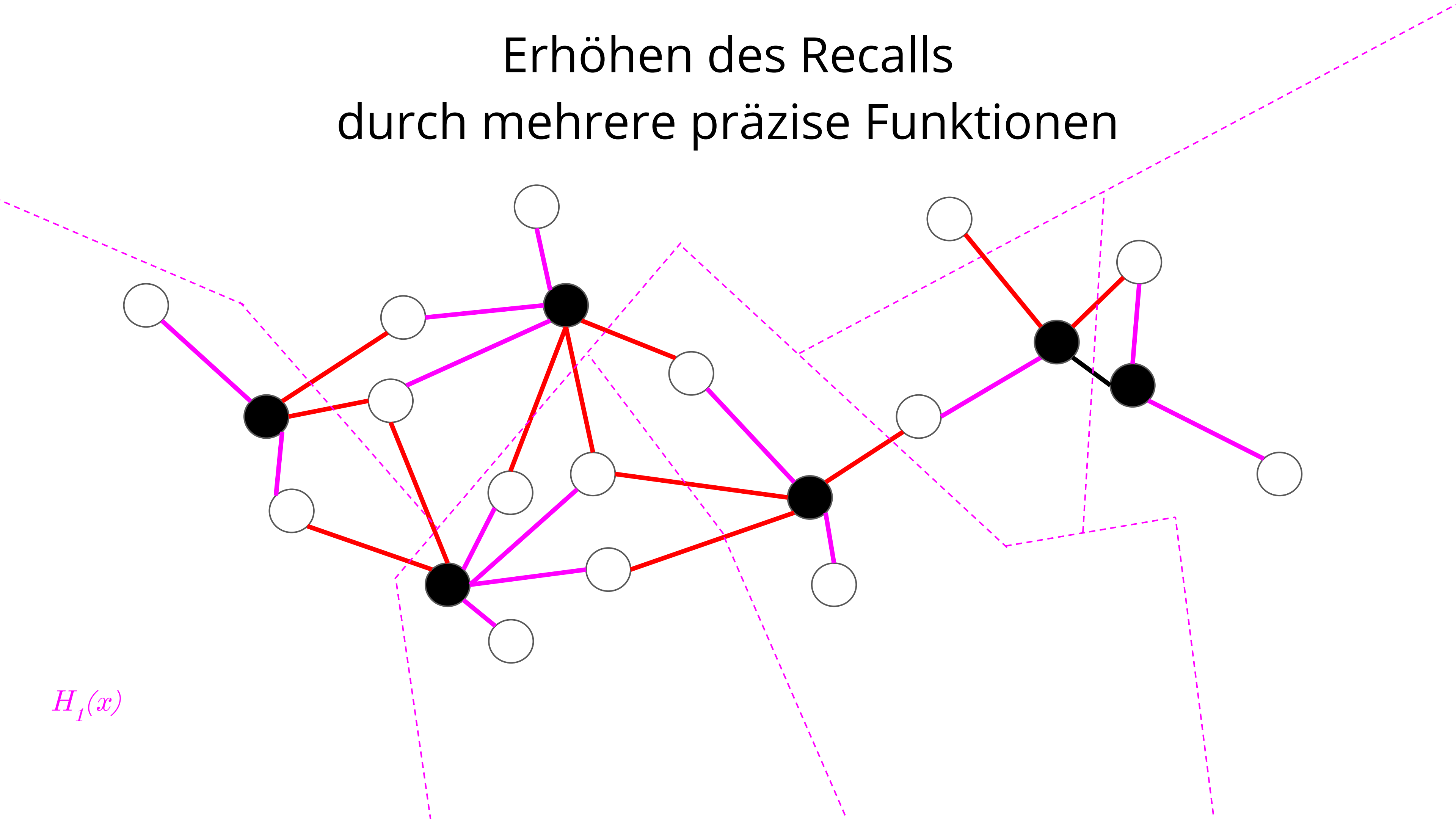


Nicht präzise

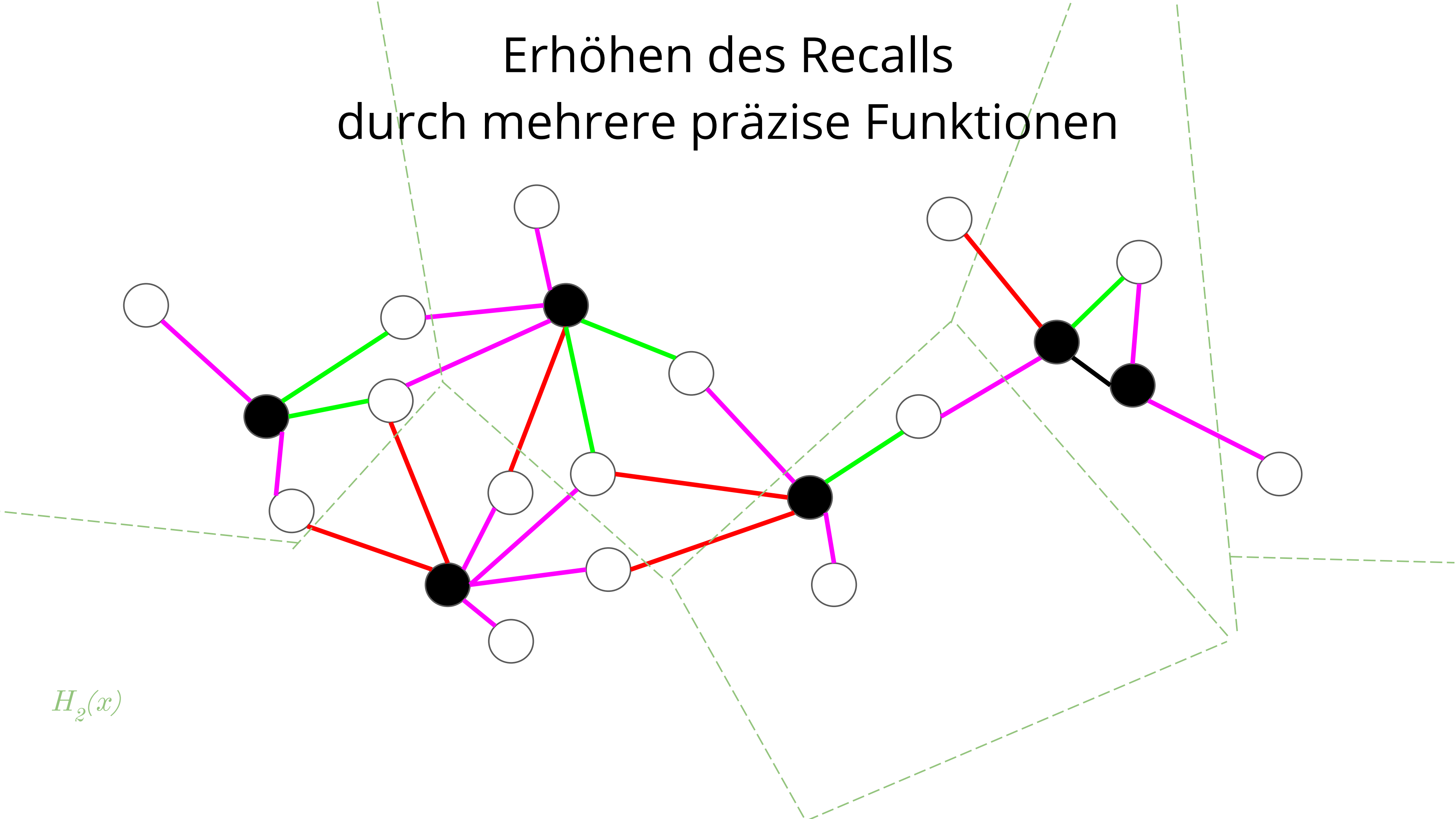


präzise

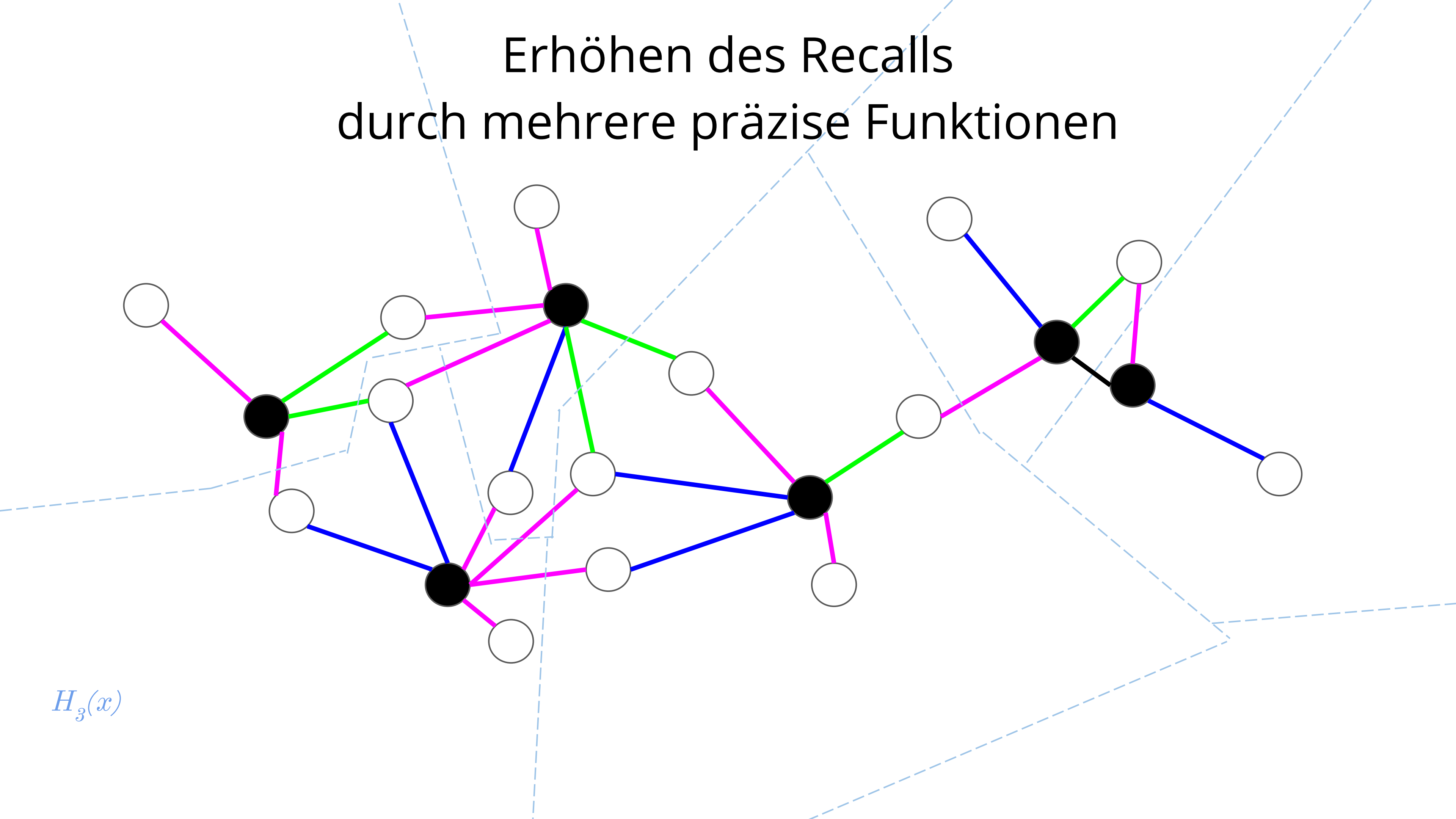
Erhöhen des Recalls durch mehrere präzise Funktionen



Erhöhen des Recalls durch mehrere präzise Funktionen

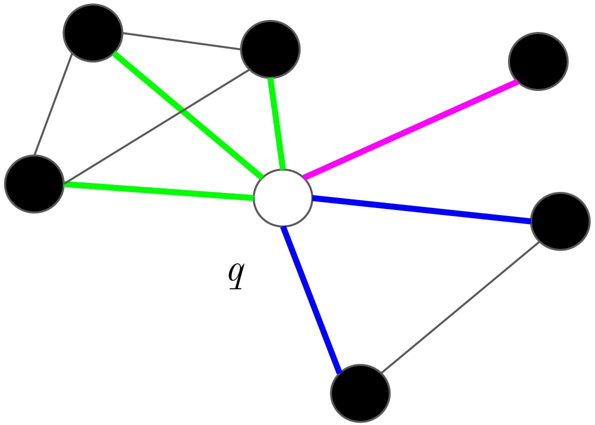


Erhöhen des Recalls durch mehrere präzise Funktionen



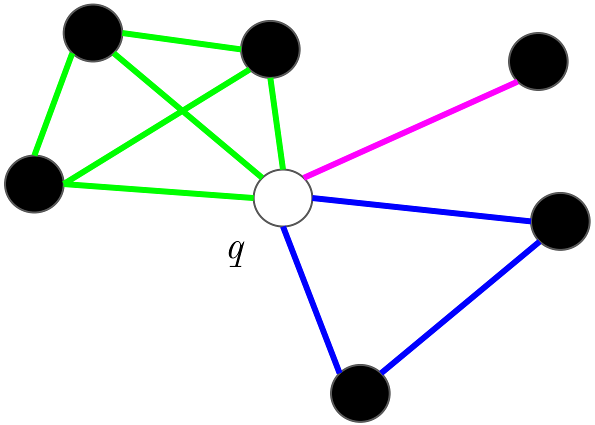
Beobachtung 2:

Alle Kanten an einem Anfragepunkt müssen
Durch mindestens eine Funktion erwischt werden,
sonst geht Recall verloren.



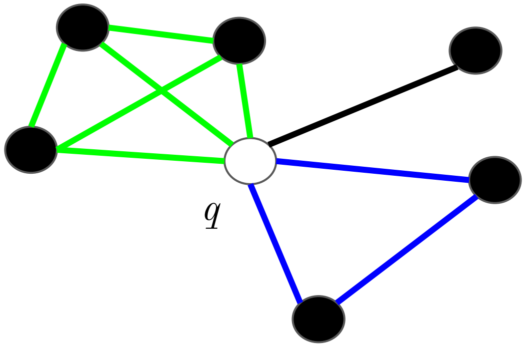
Beobachtung 3:

Totaler Recall mit präzisen Hashfunktionen
ist eine Überdeckung der Nachbarschaft von q mit Cliquen

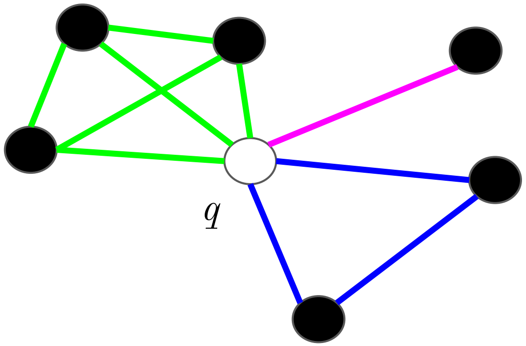


Theorem 1:

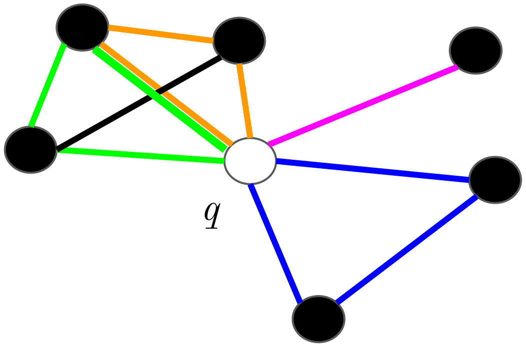
Für perfekten Recall braucht man mindestens so viele Hashfunktionen, wie Cliquen, um alle Kanten der Nachbarschaft von q zu überdecken



Kante geht verloren



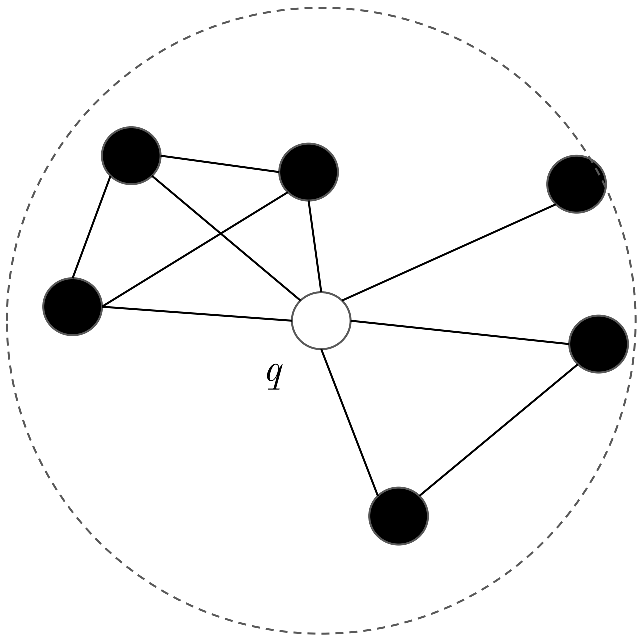
Minimale Lösung

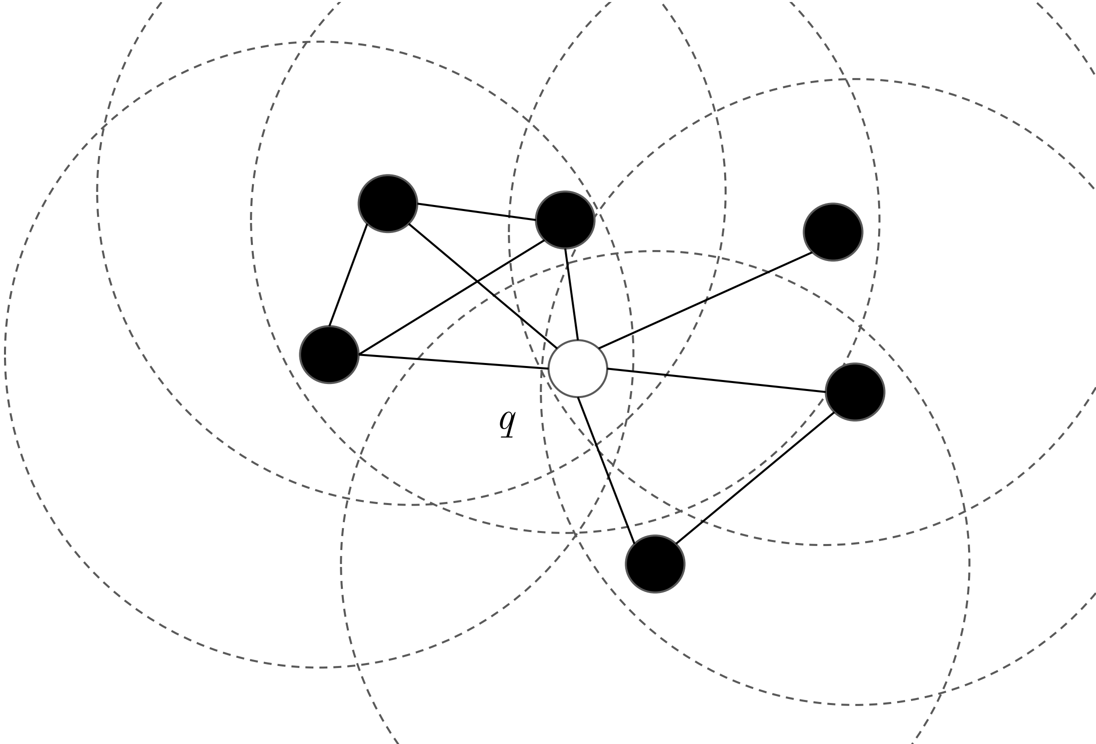


Nicht minimale Lösung

Beobachtung 4:

Der Grad (=Anzahl der Nachbarn) eines Anfragepunkts q ist die Anzahl an Datenpunkten in seiner Umgebung, bzw. die Anzahl der sich überlappenden Umgebungen.



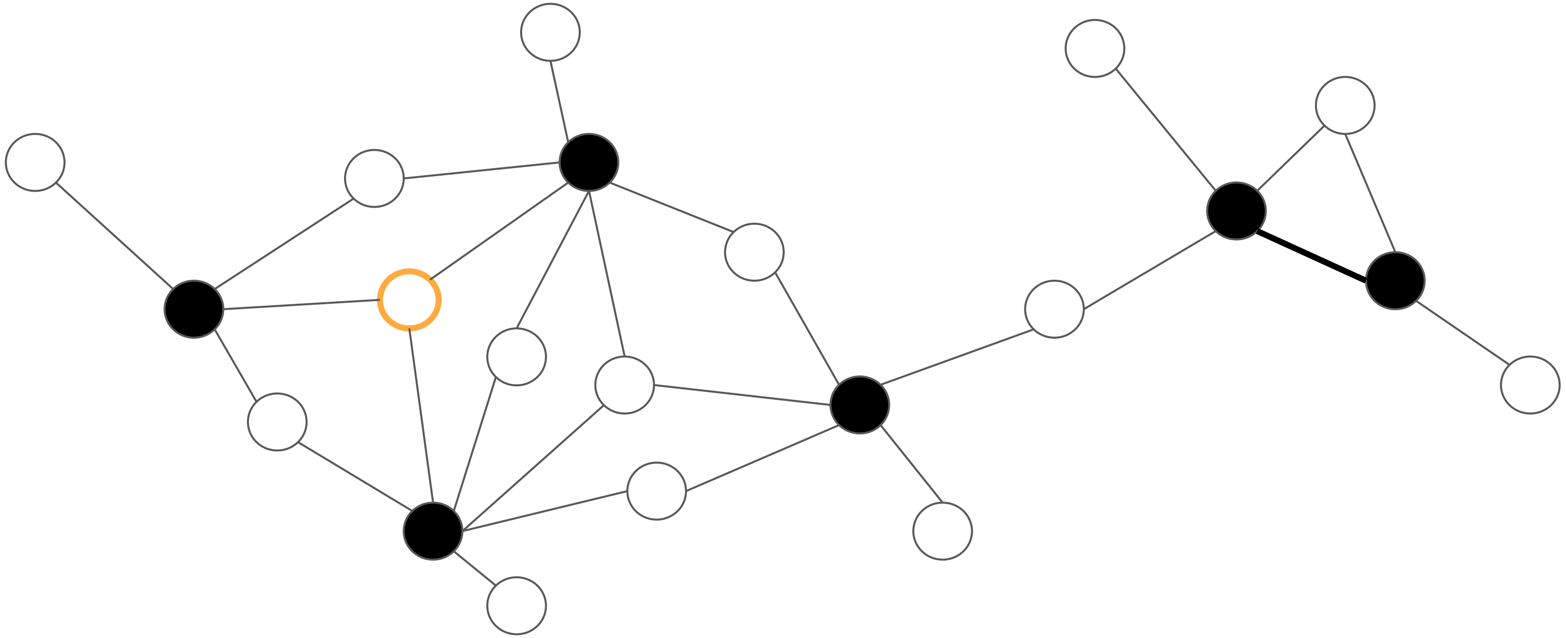


Theorem 2:

Ist K die maximale Anzahl an Überlappungen, die Datenpunkte bilden können, dann reichen K präzise Hashfunktionen aus.

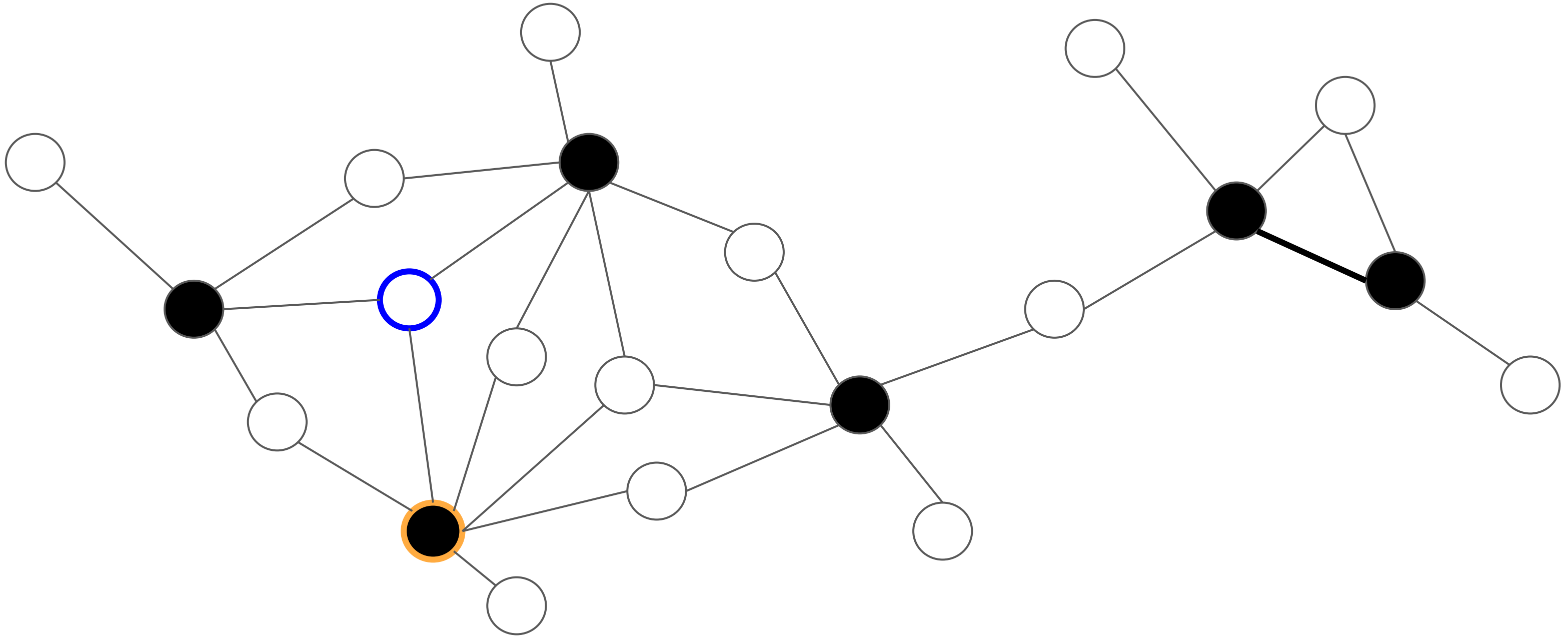
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



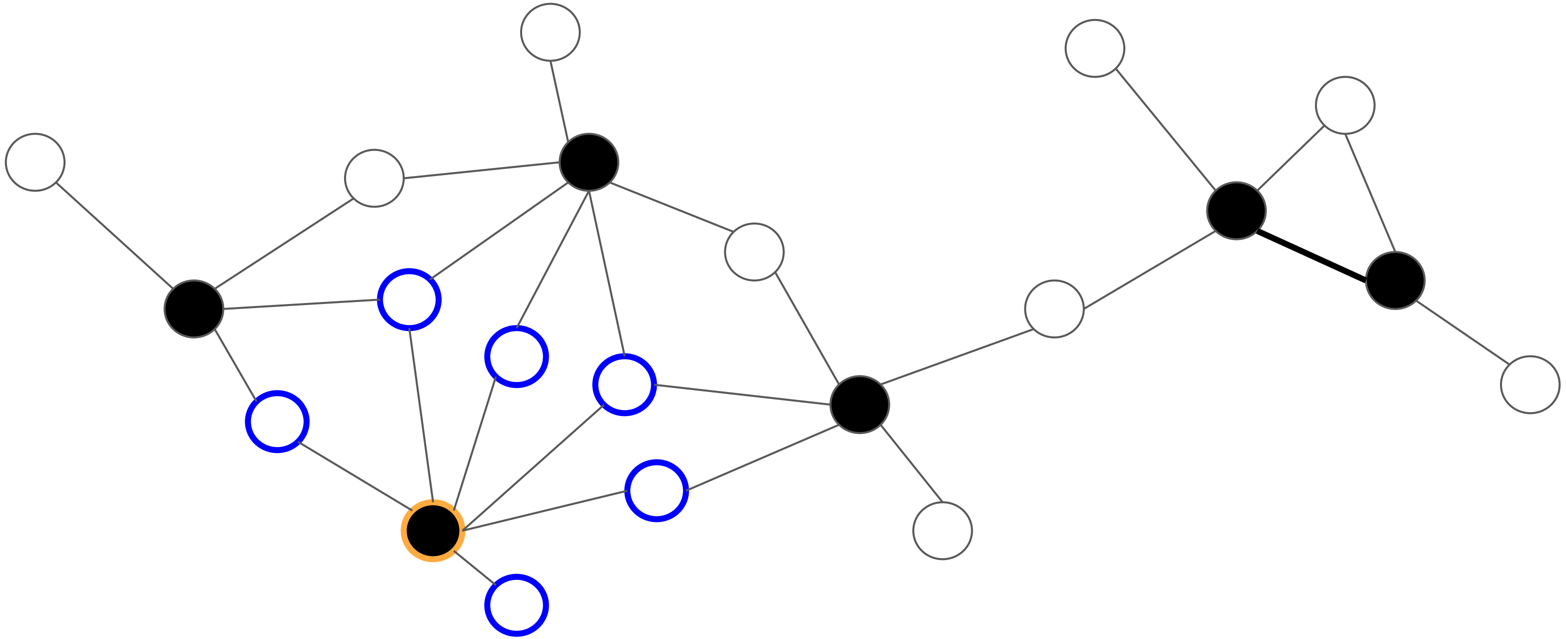
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



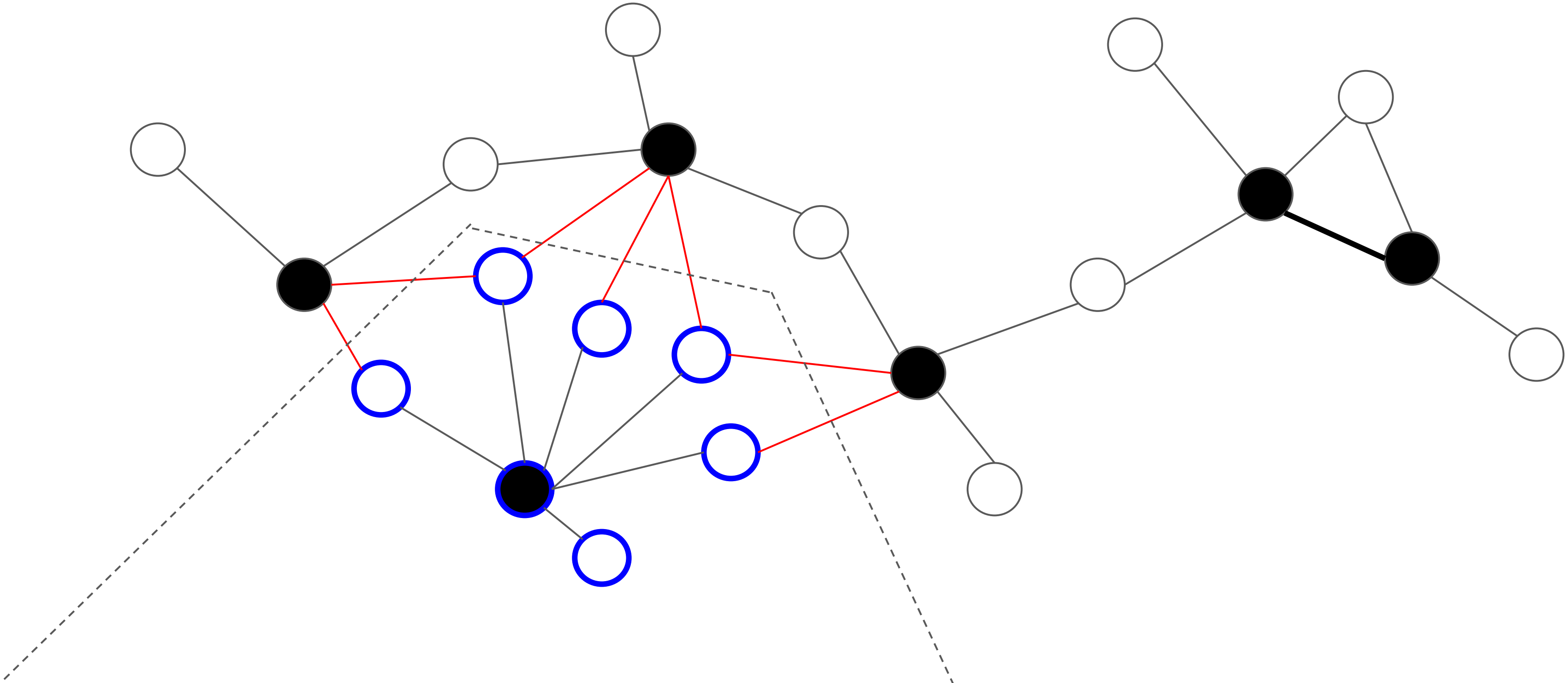
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



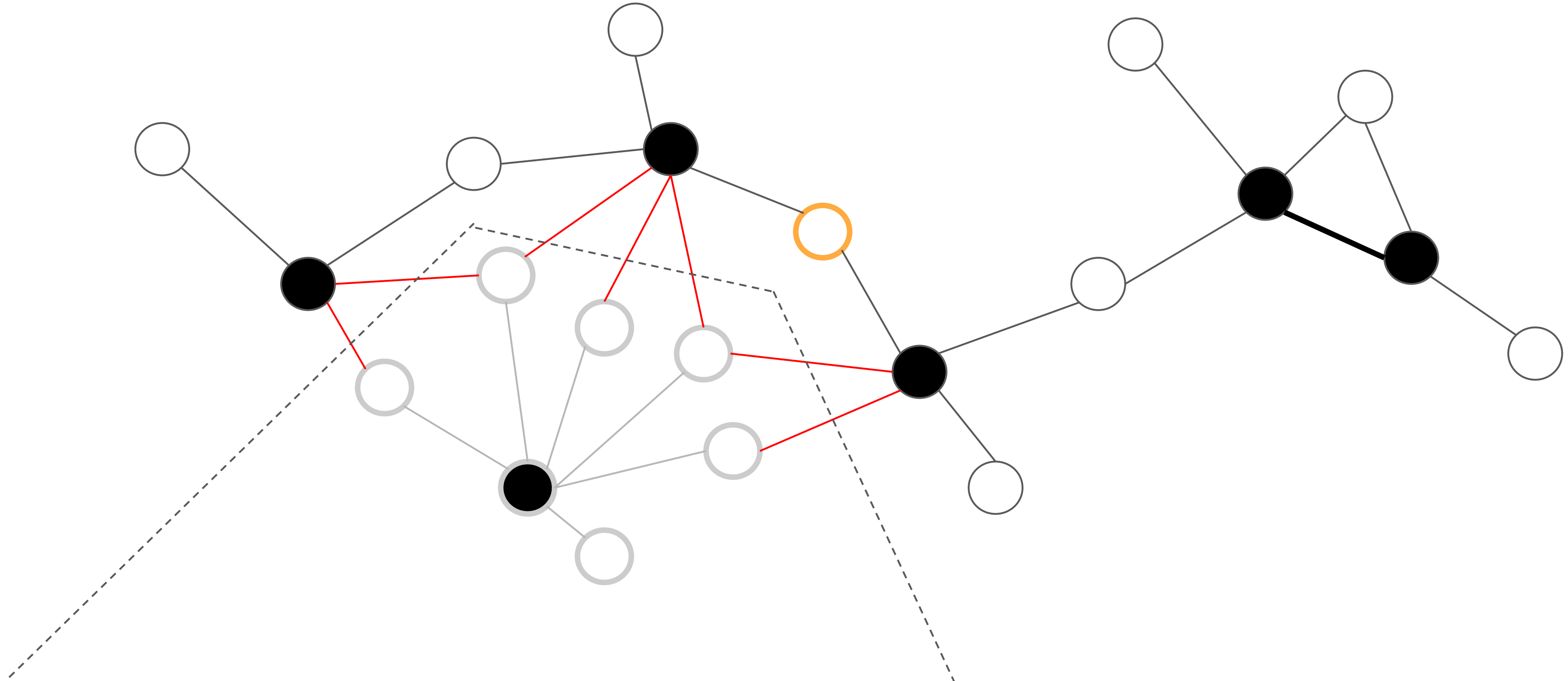
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



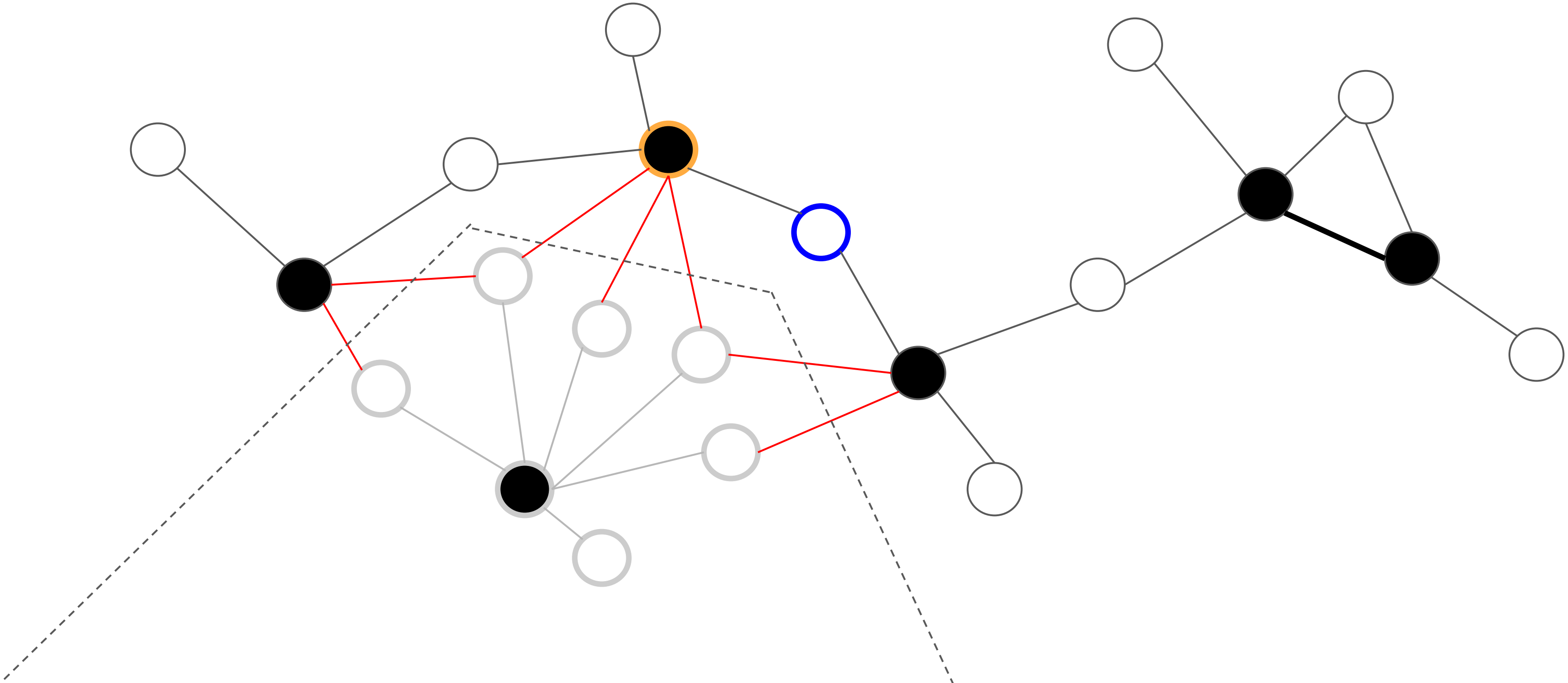
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



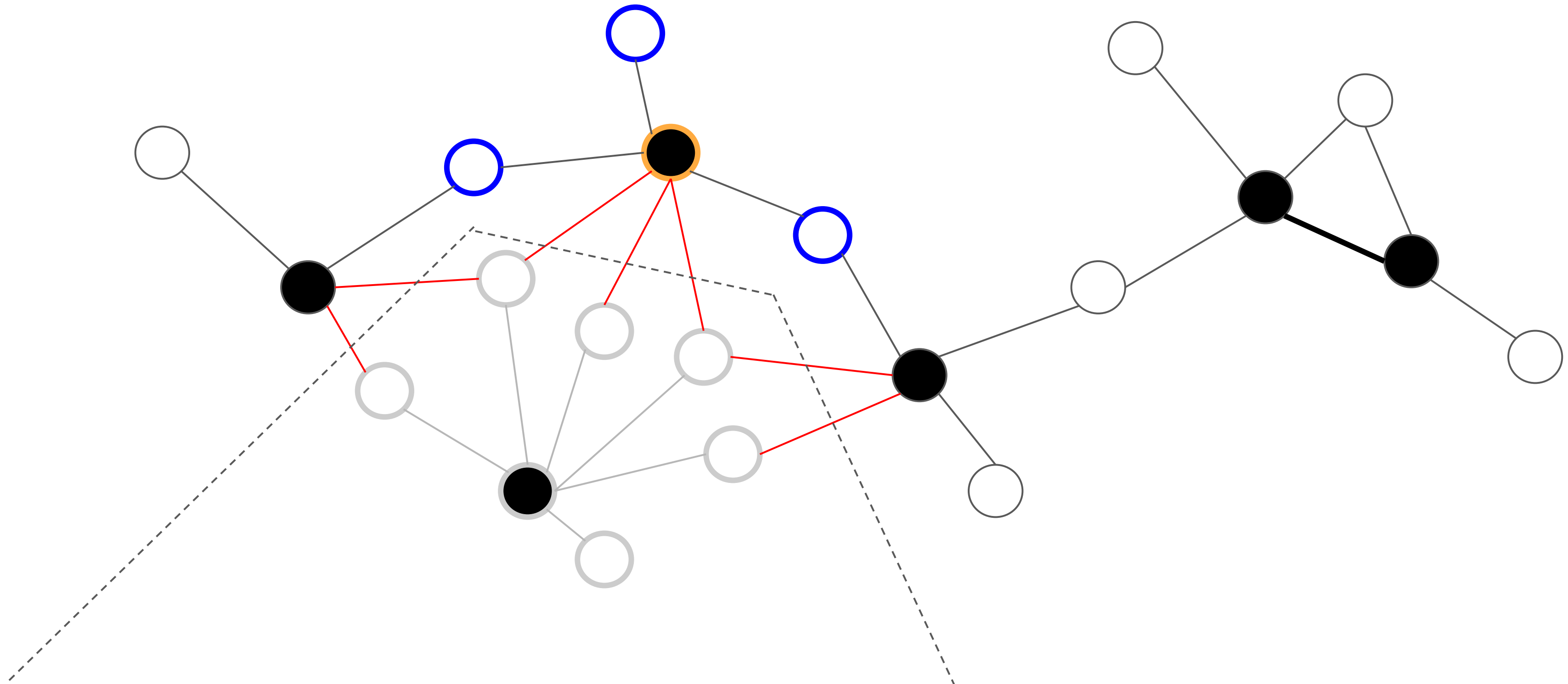
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



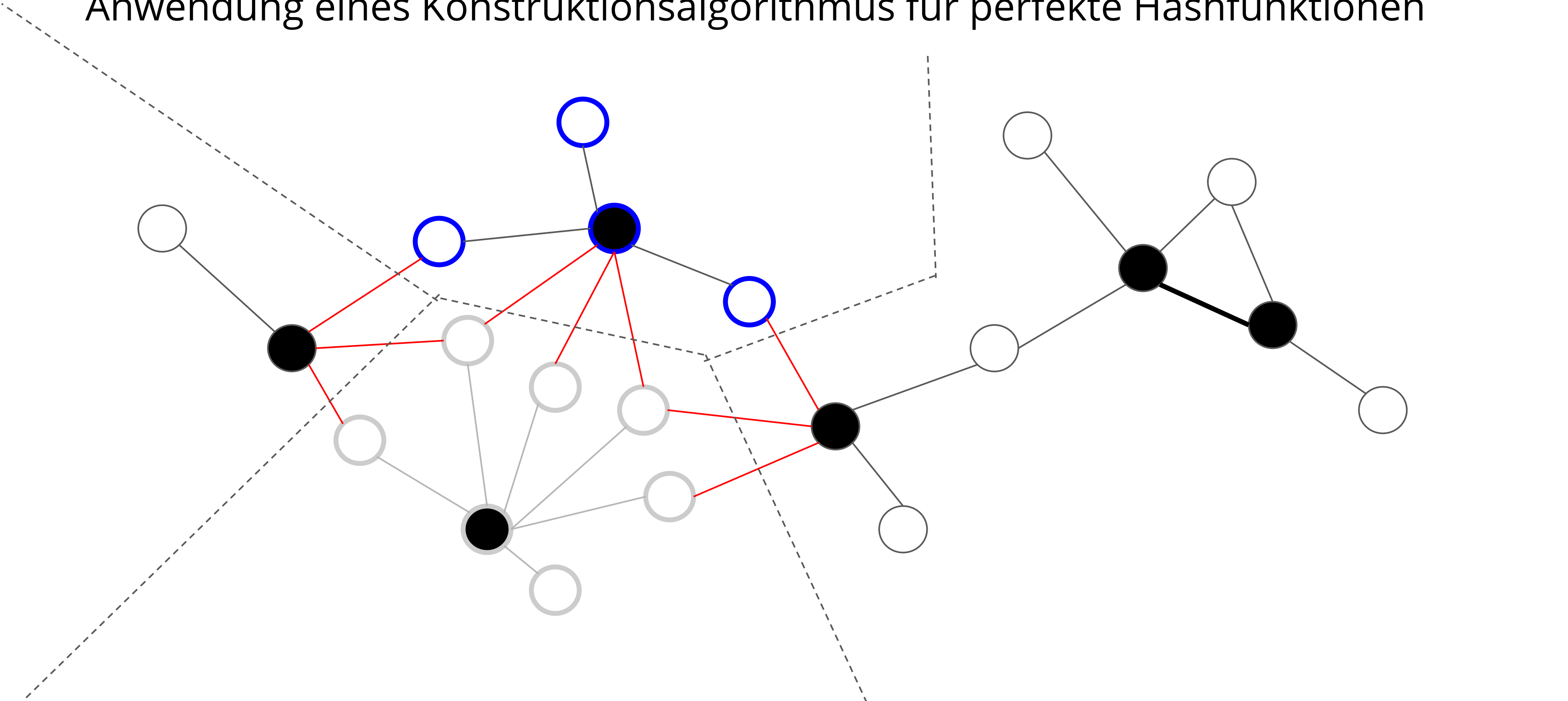
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



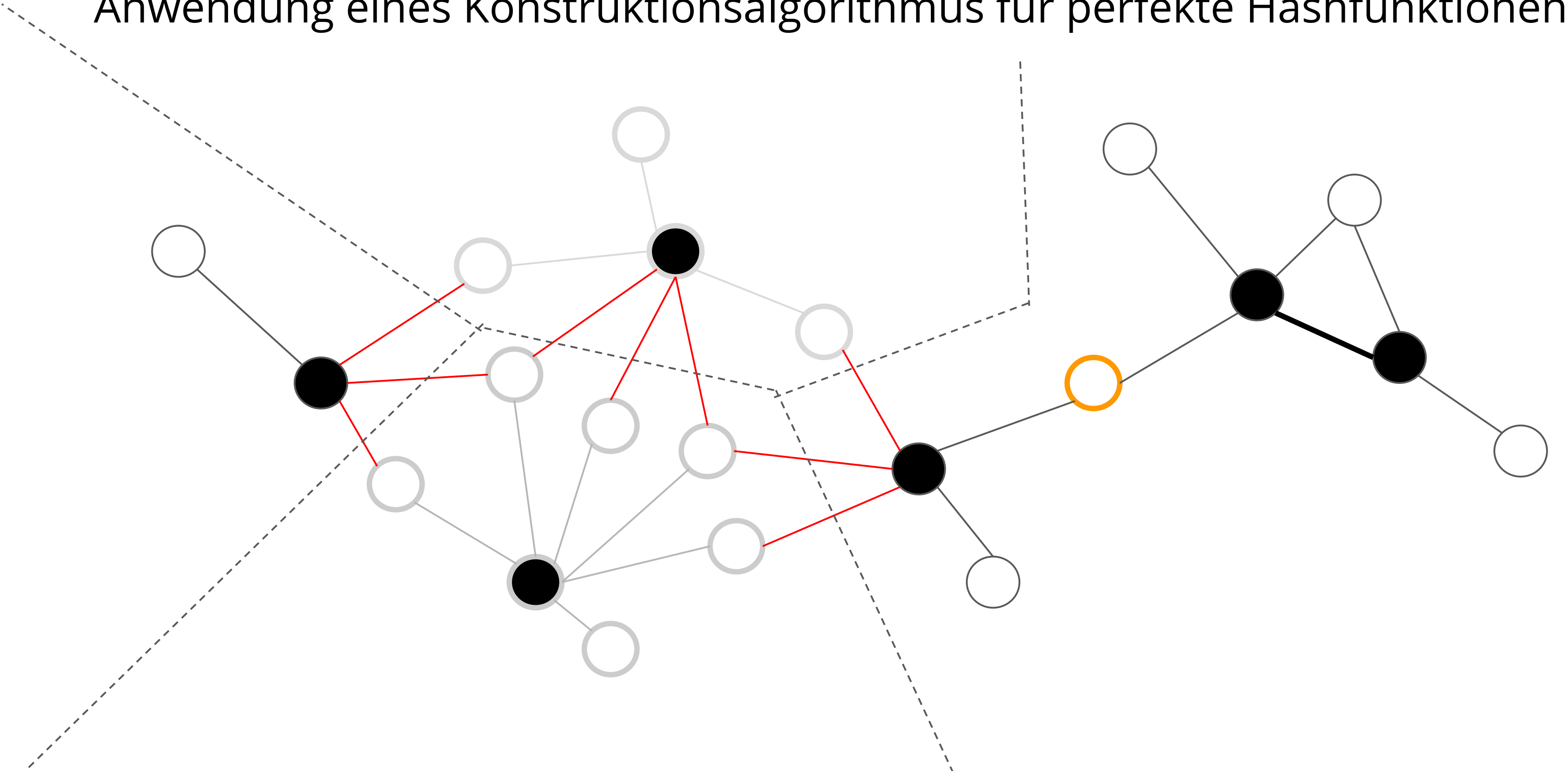
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



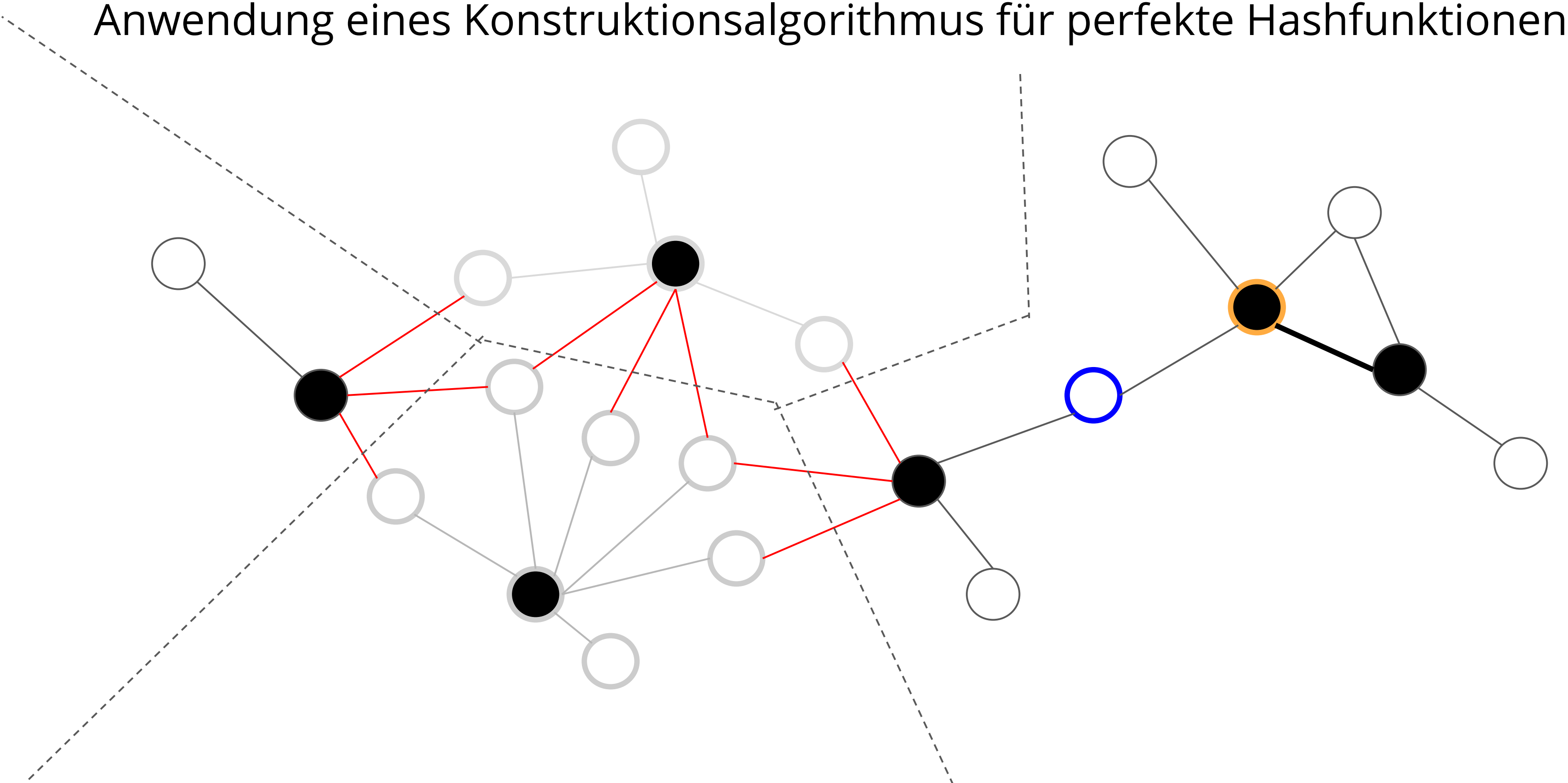
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



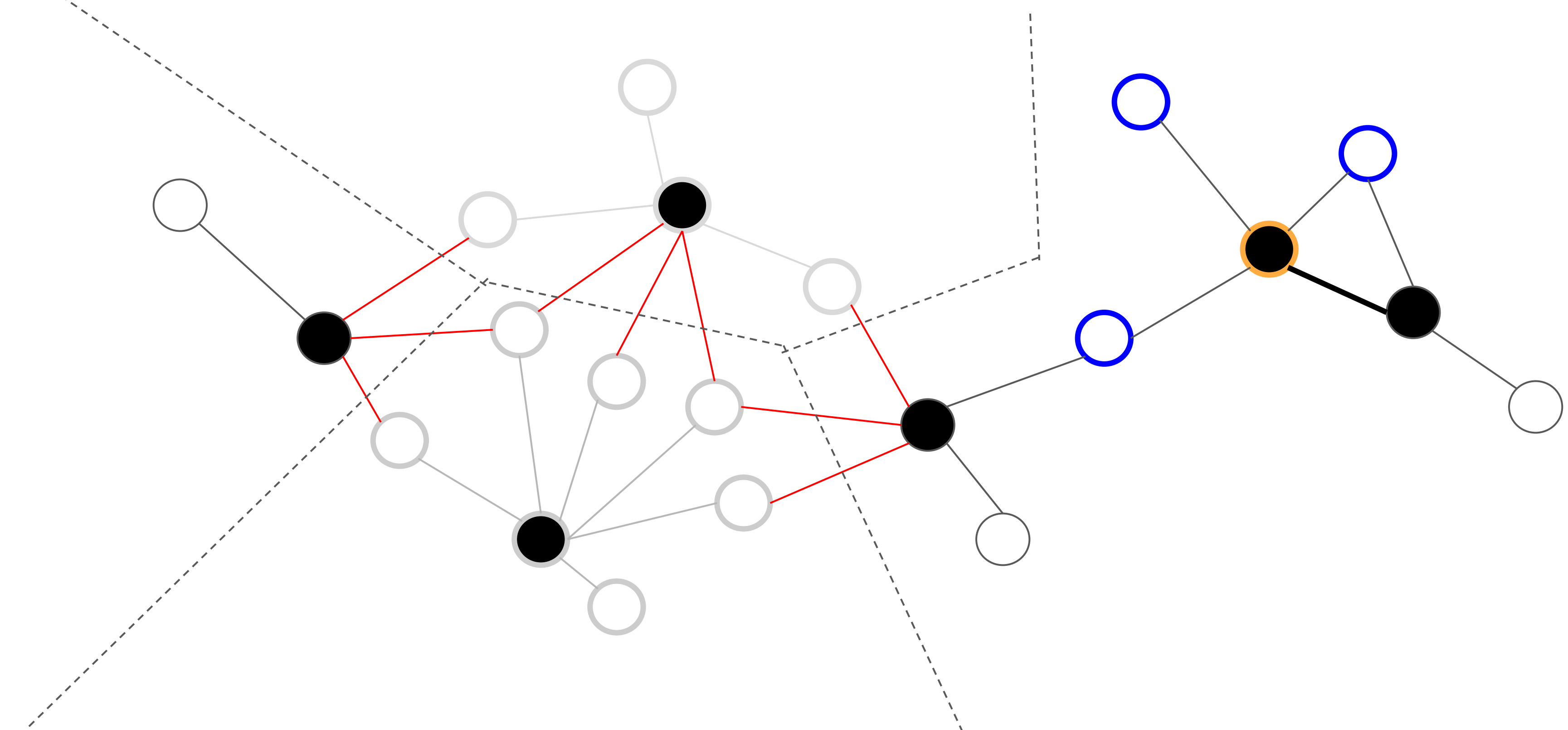
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



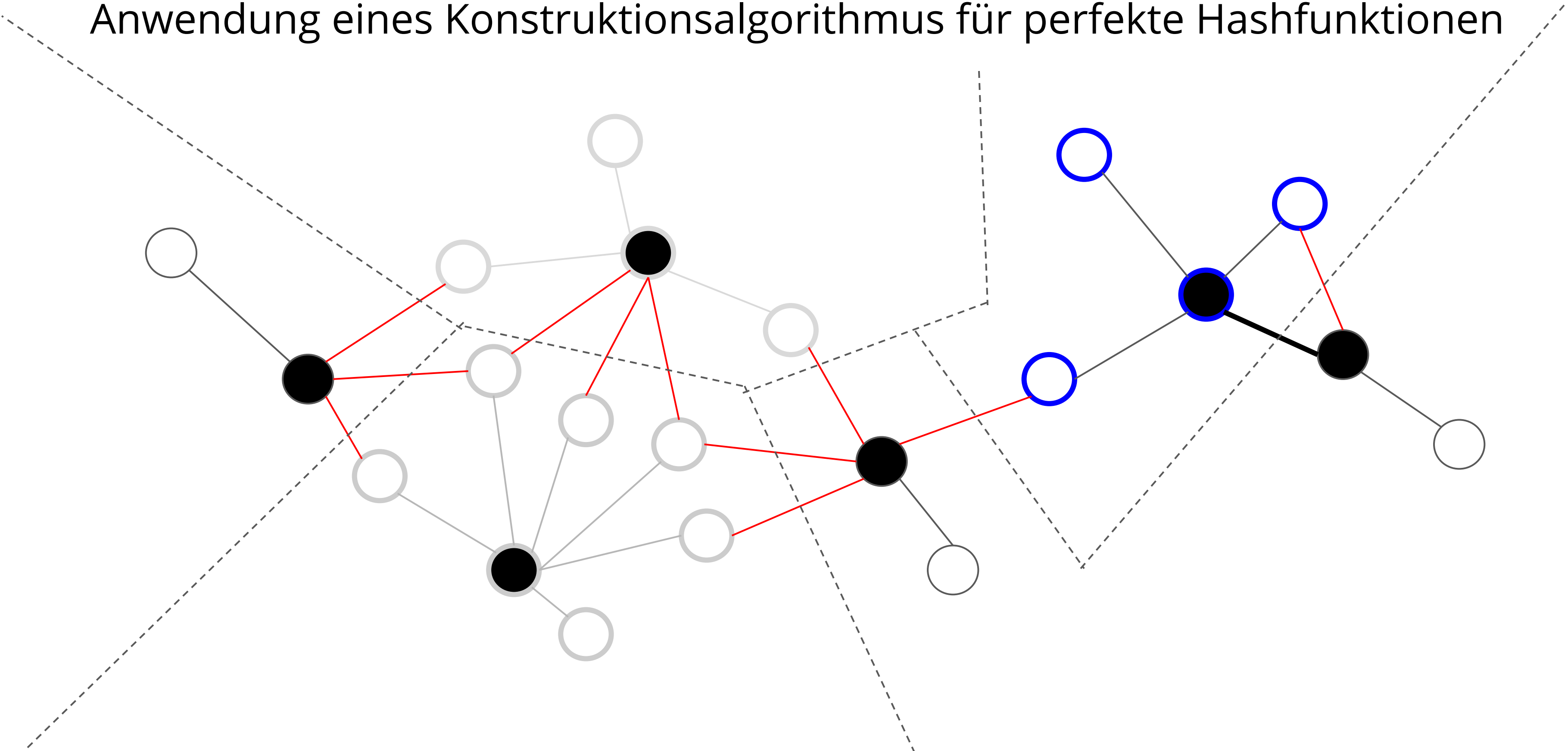
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



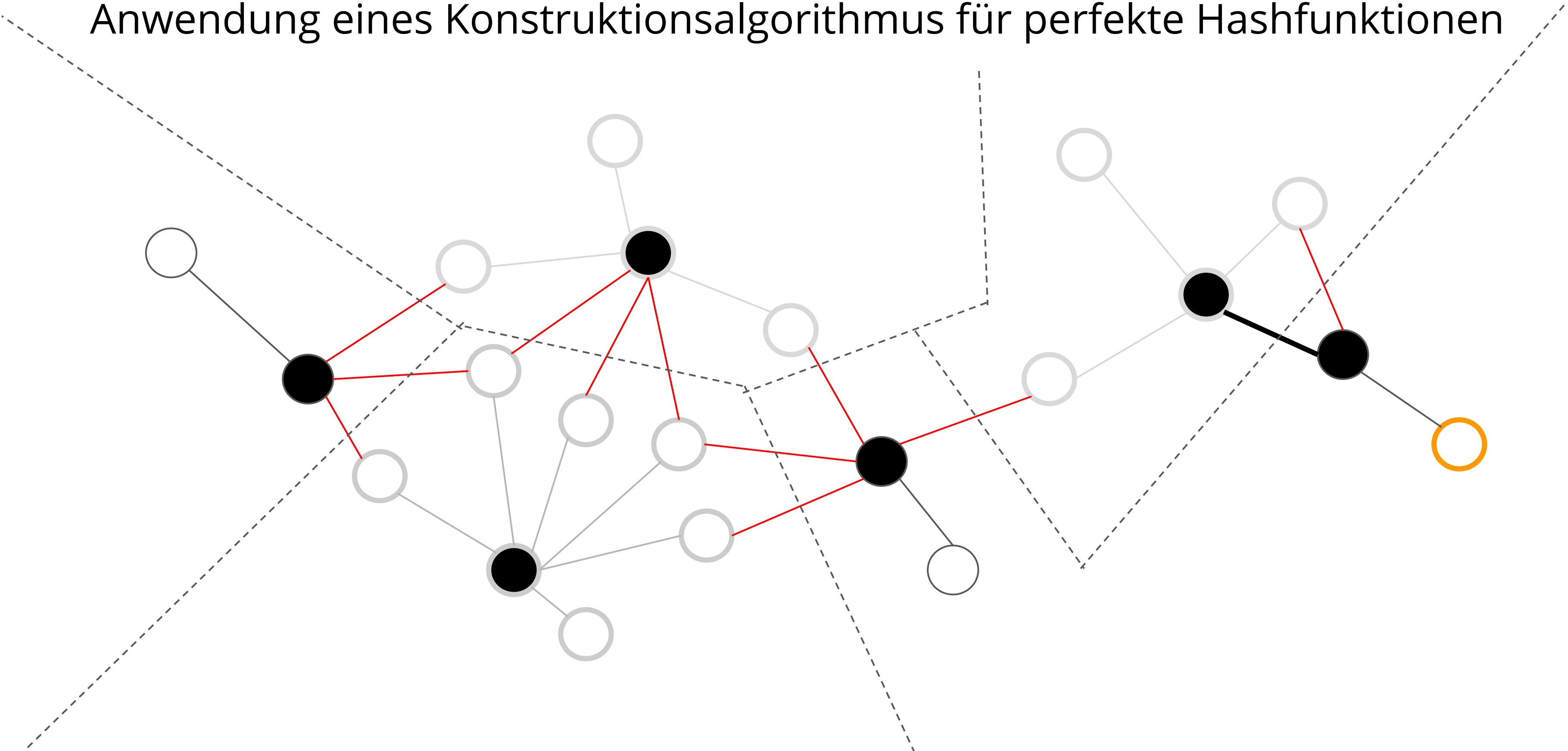
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



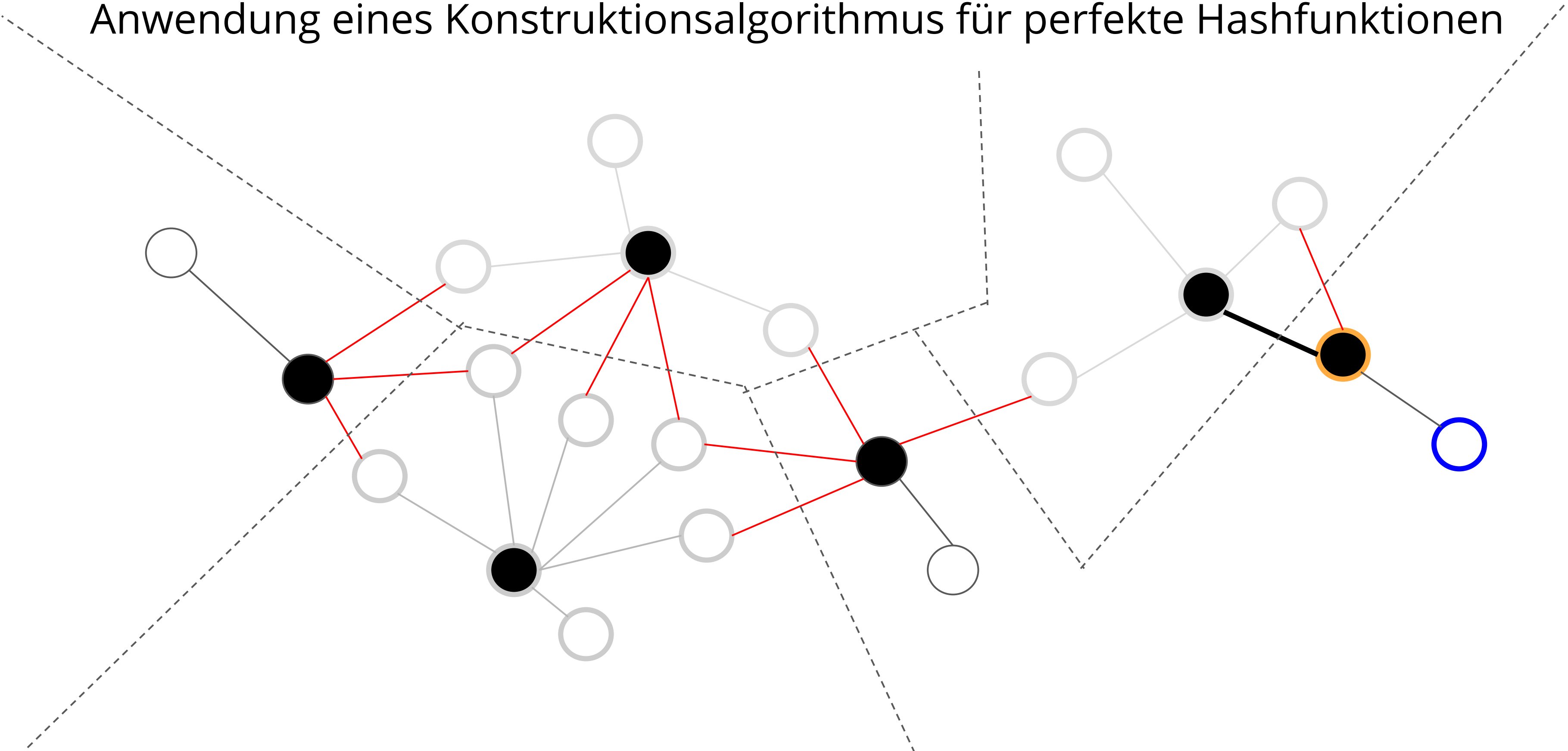
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



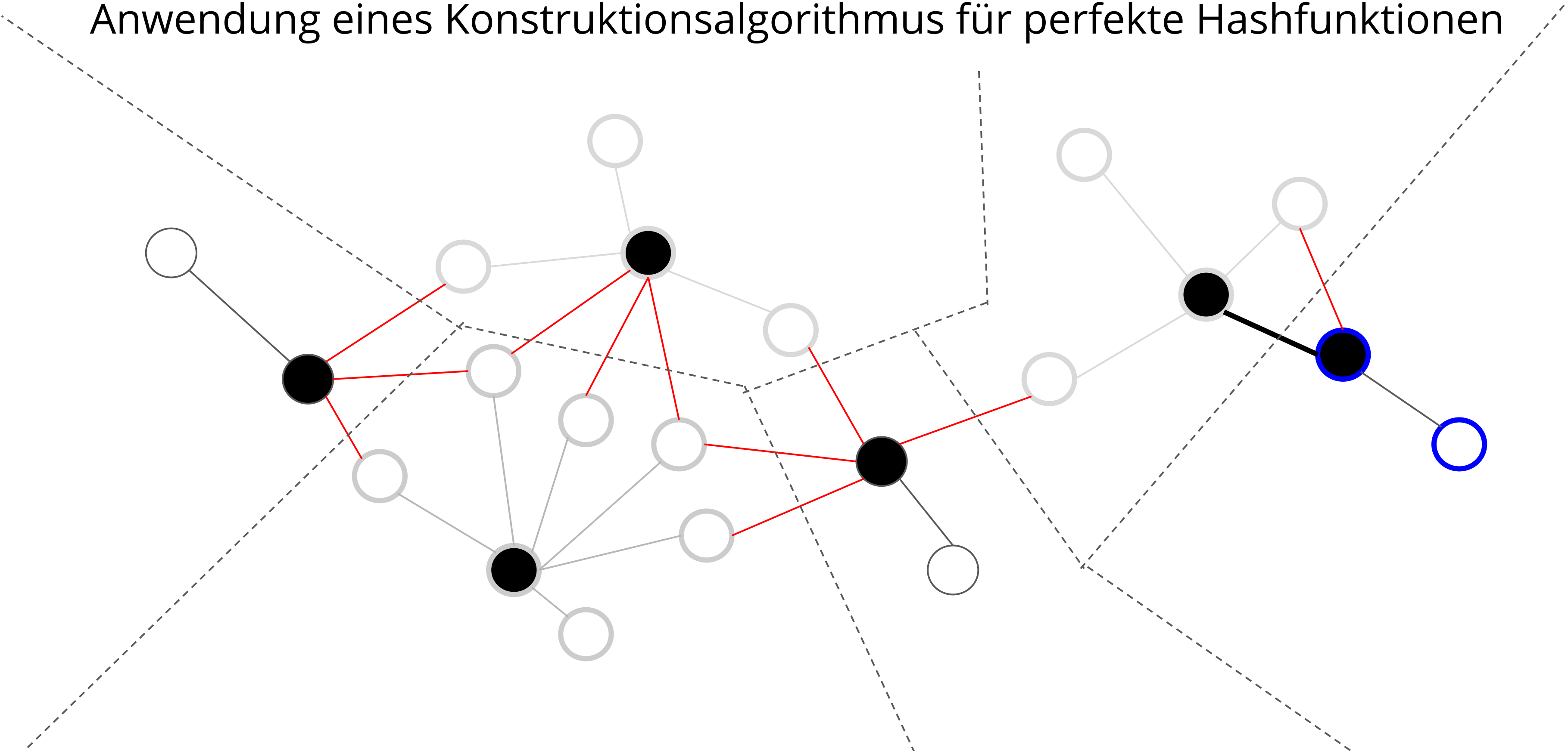
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



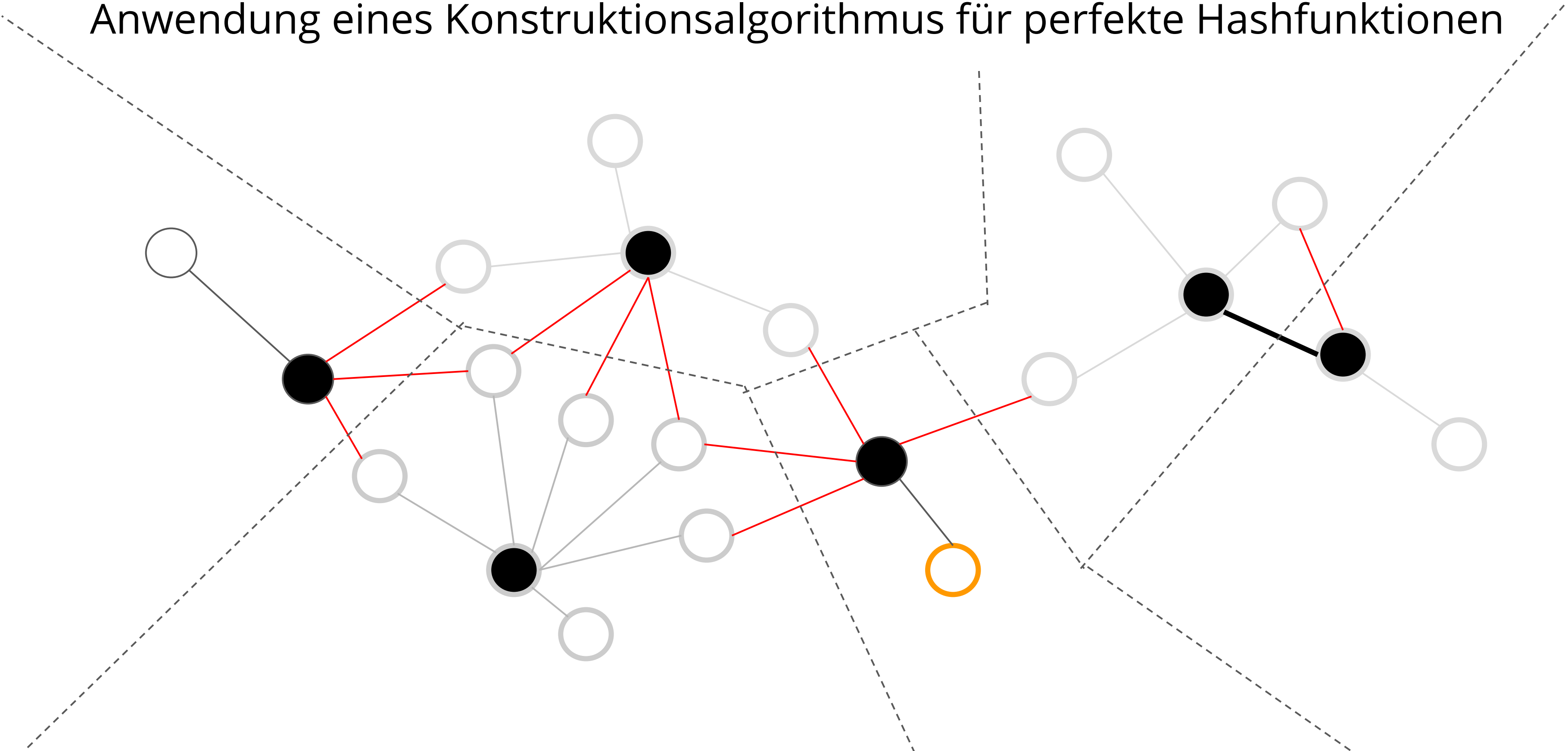
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



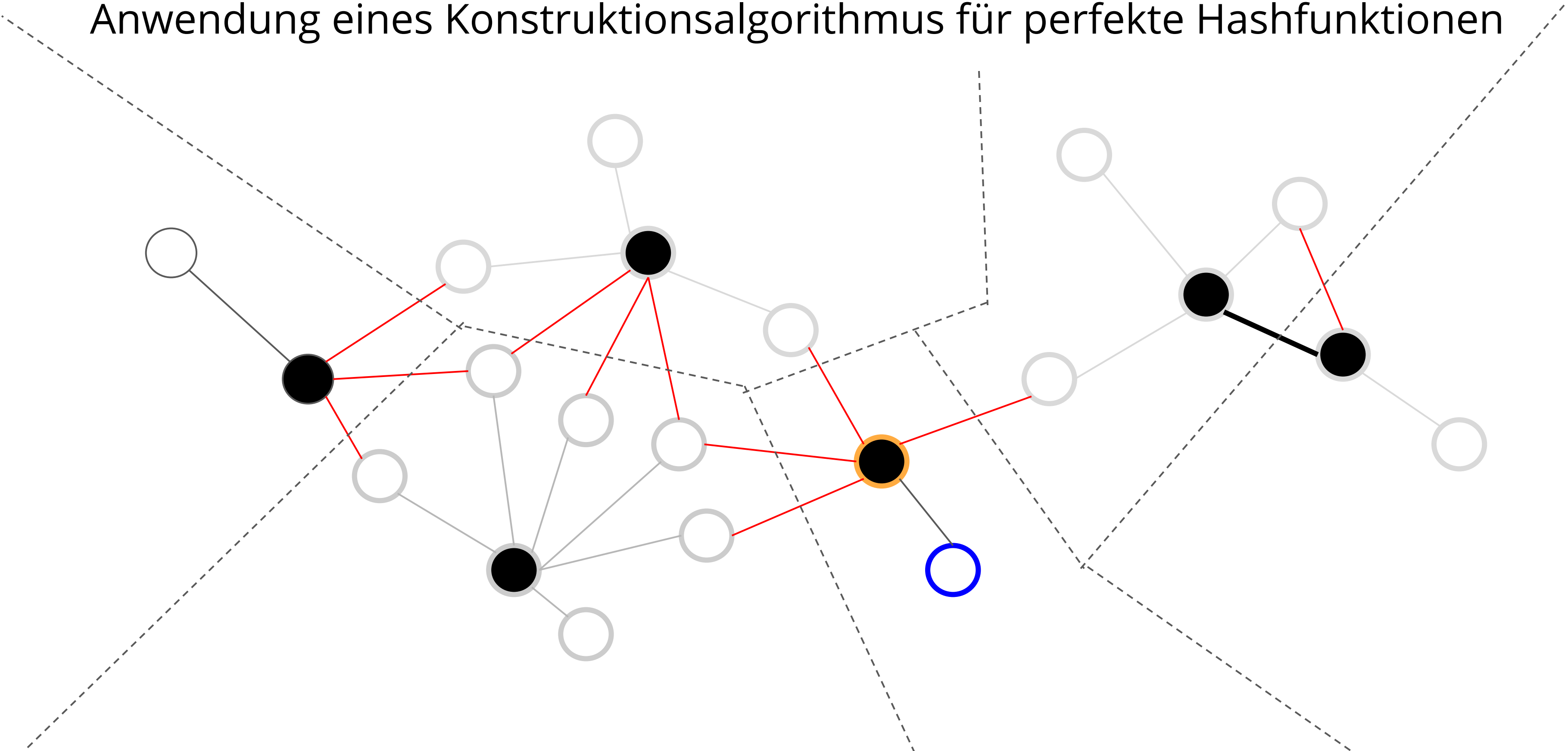
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



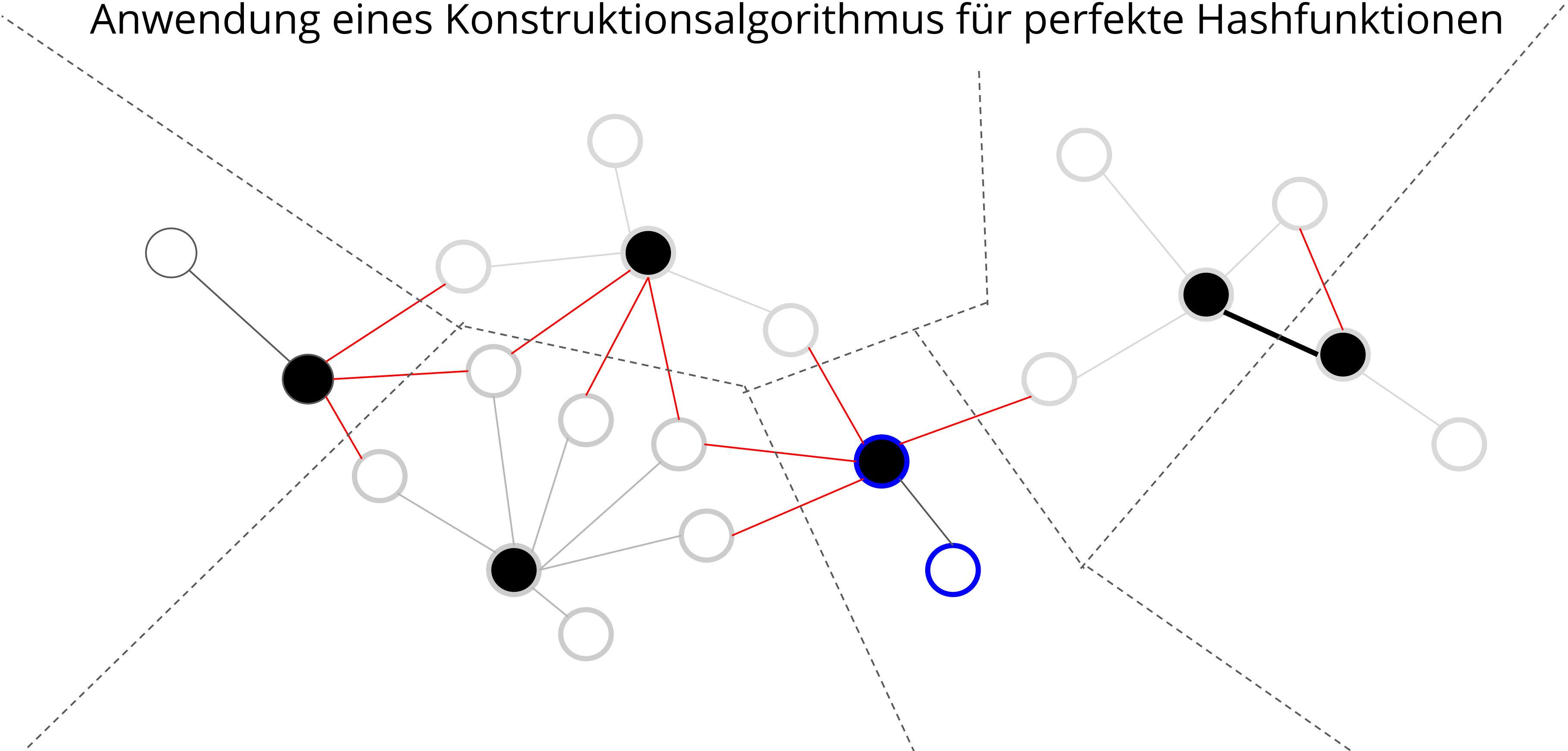
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



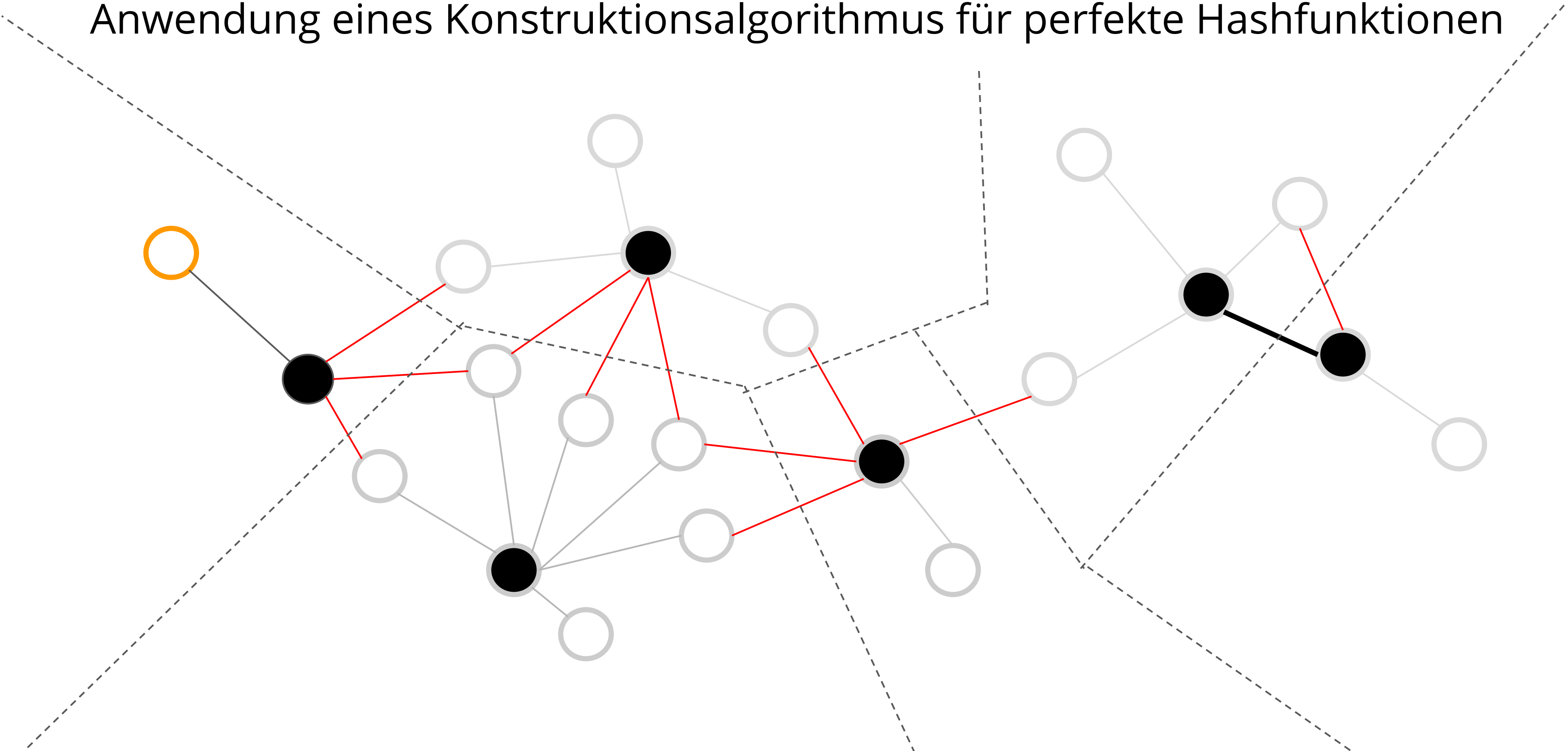
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



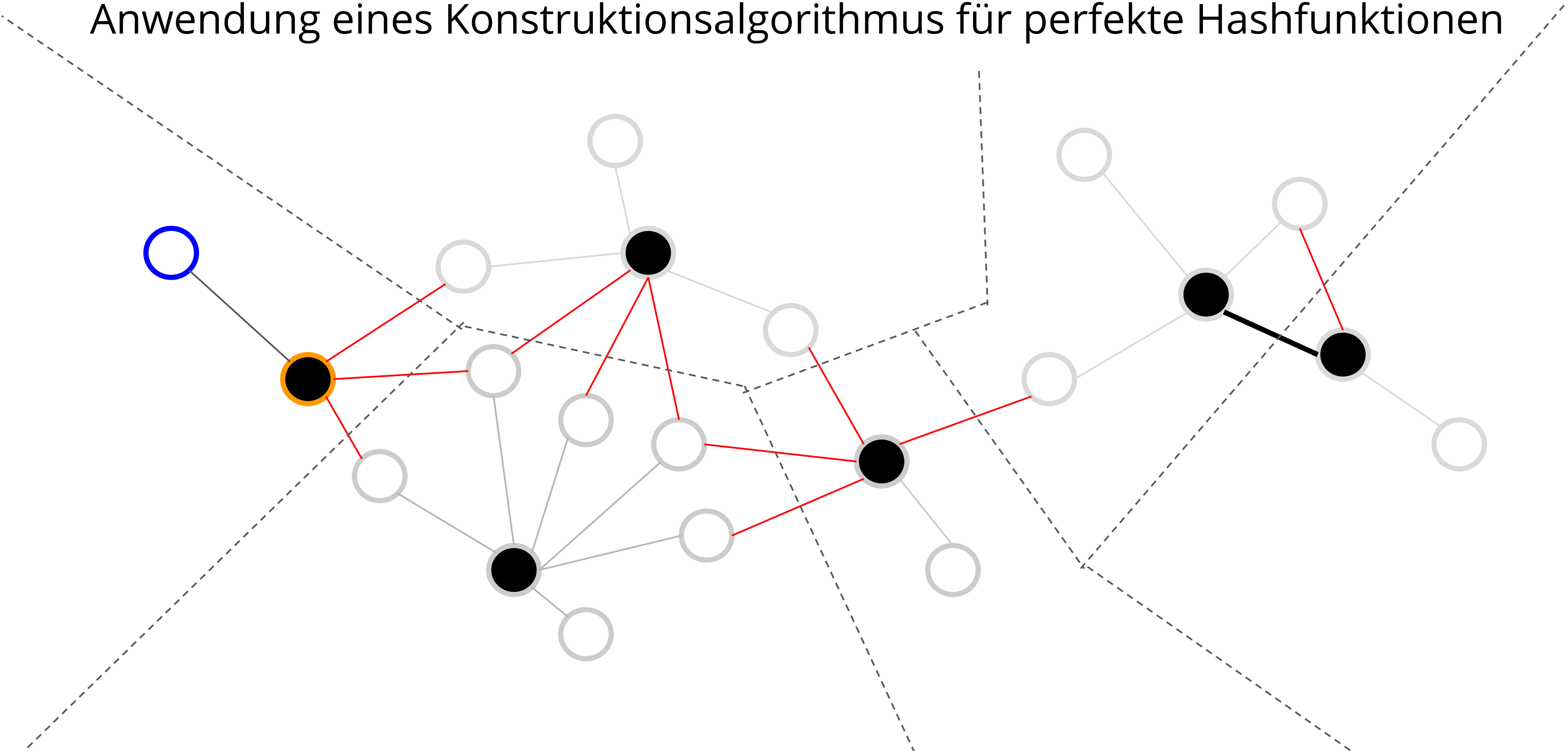
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



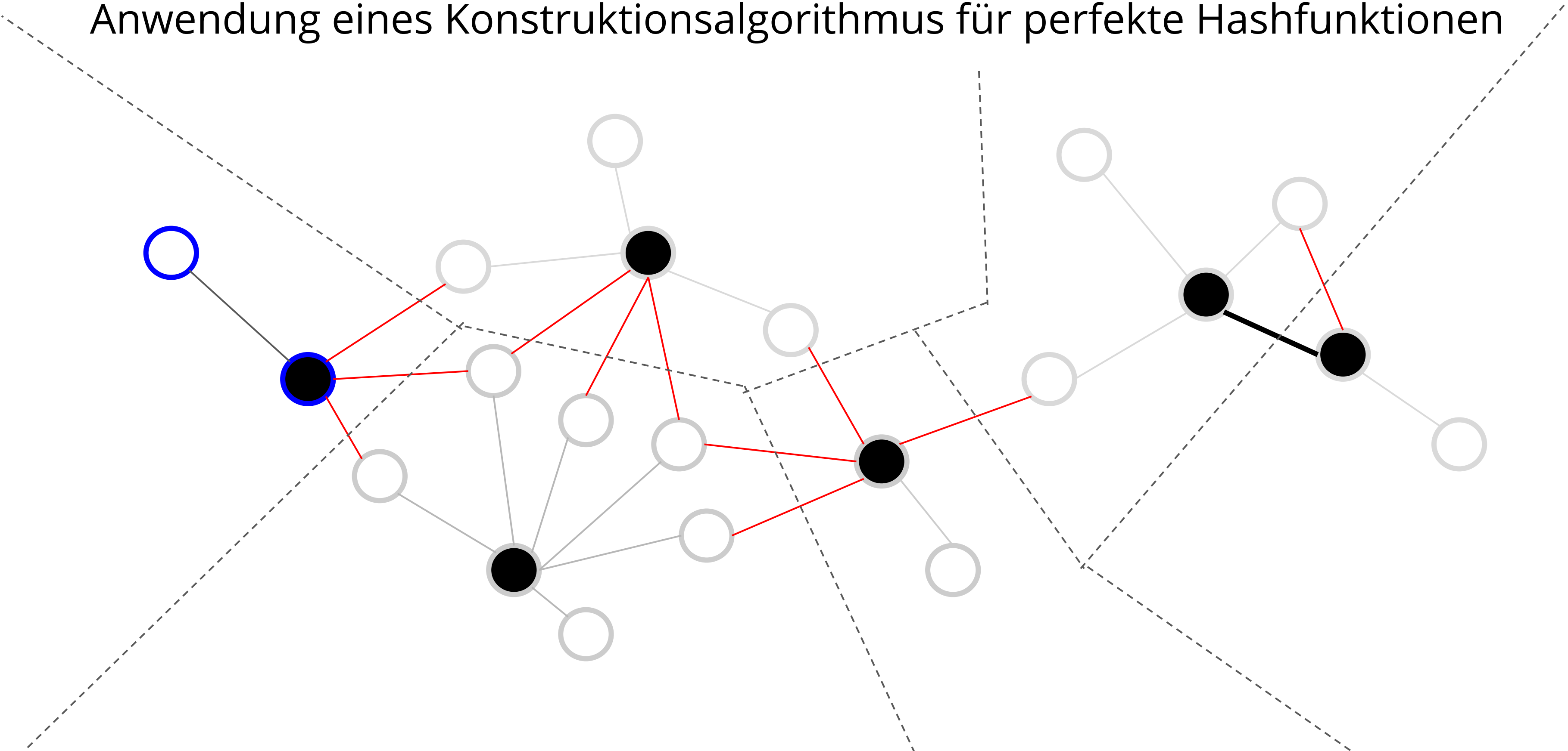
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



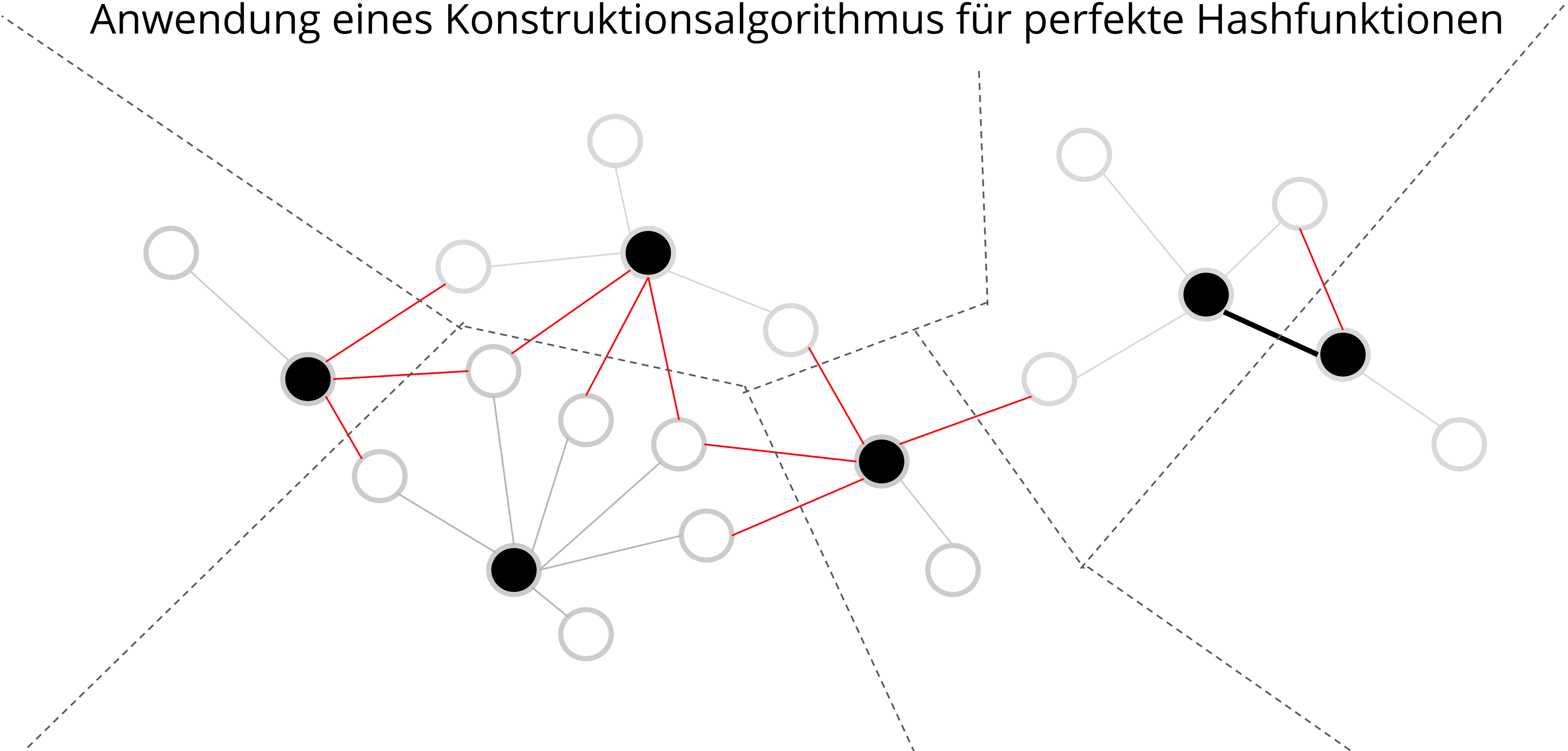
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



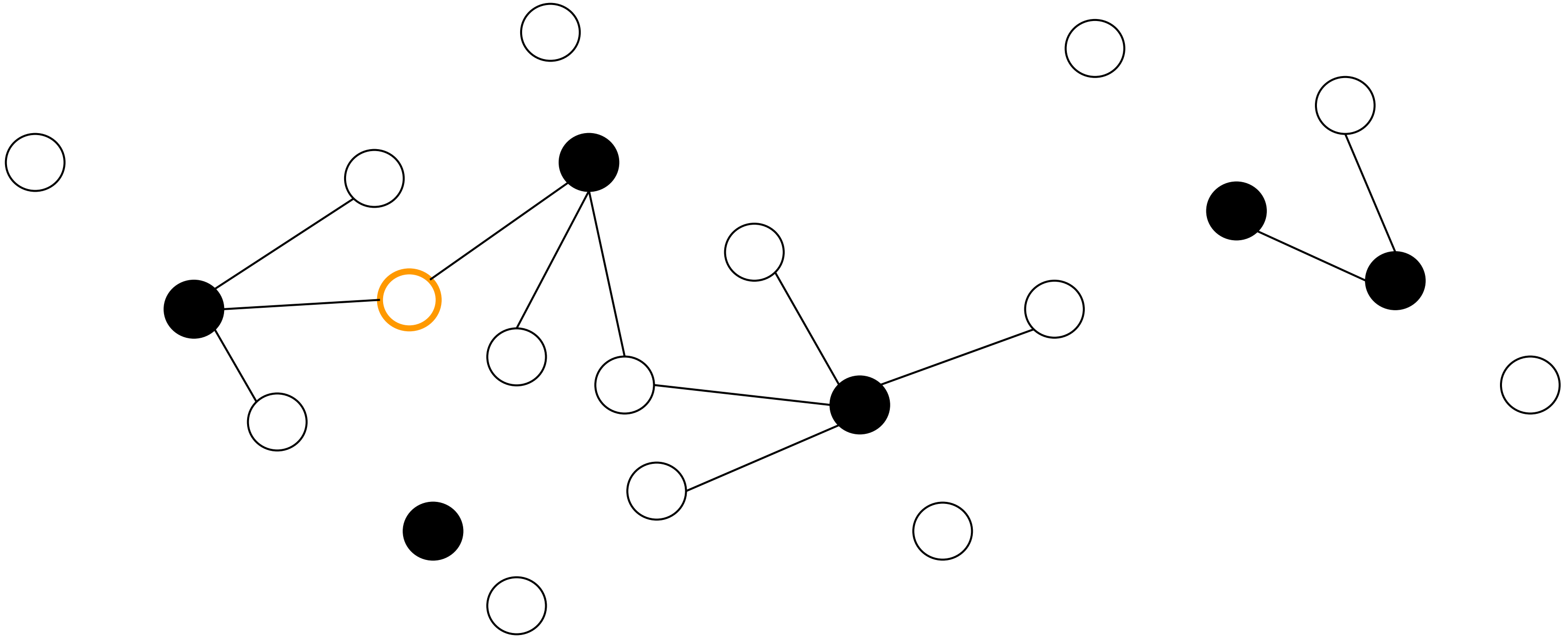
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



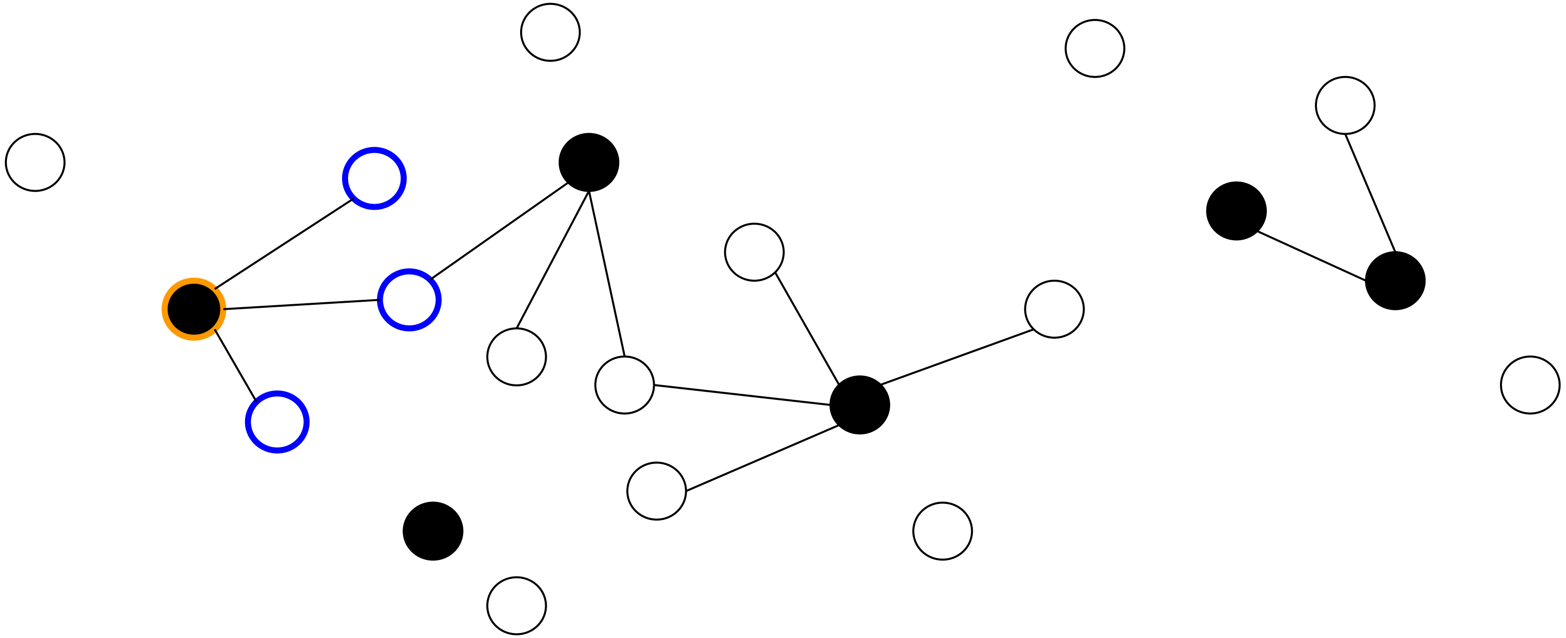
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



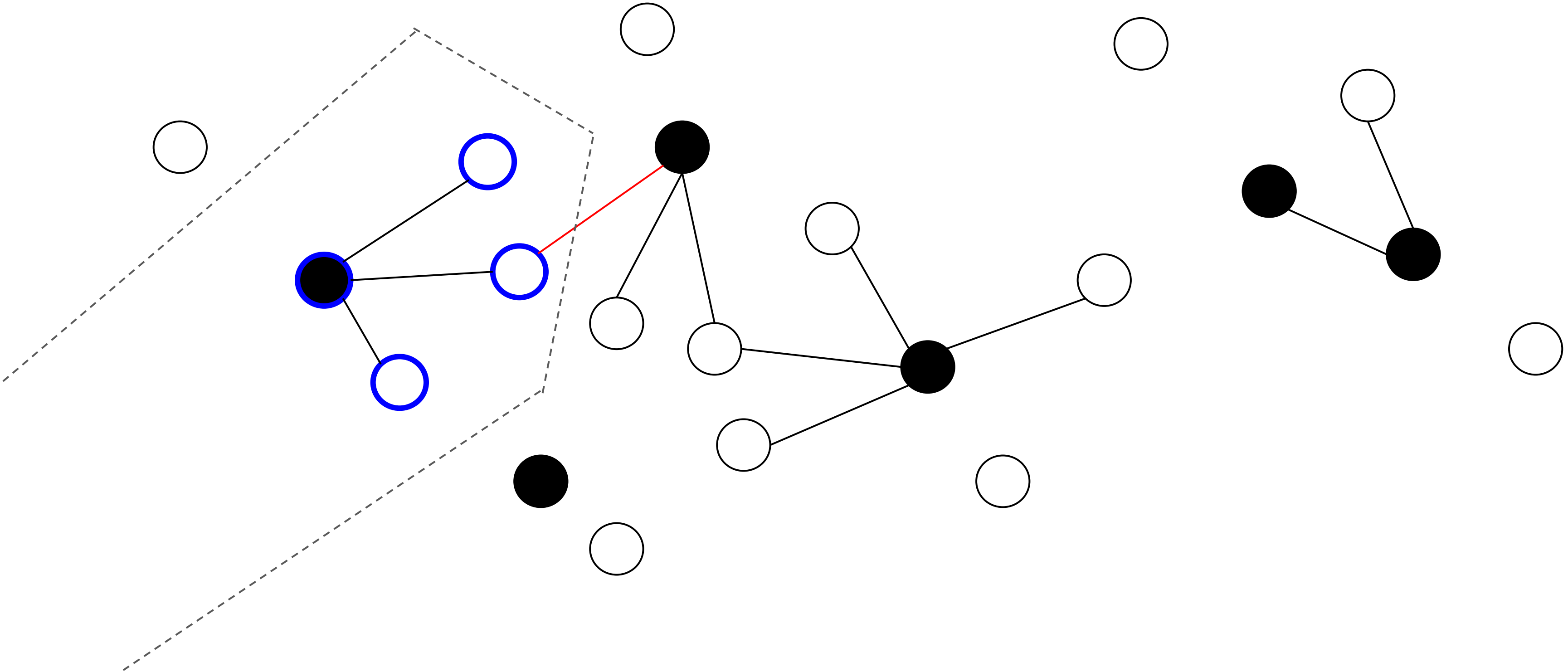
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



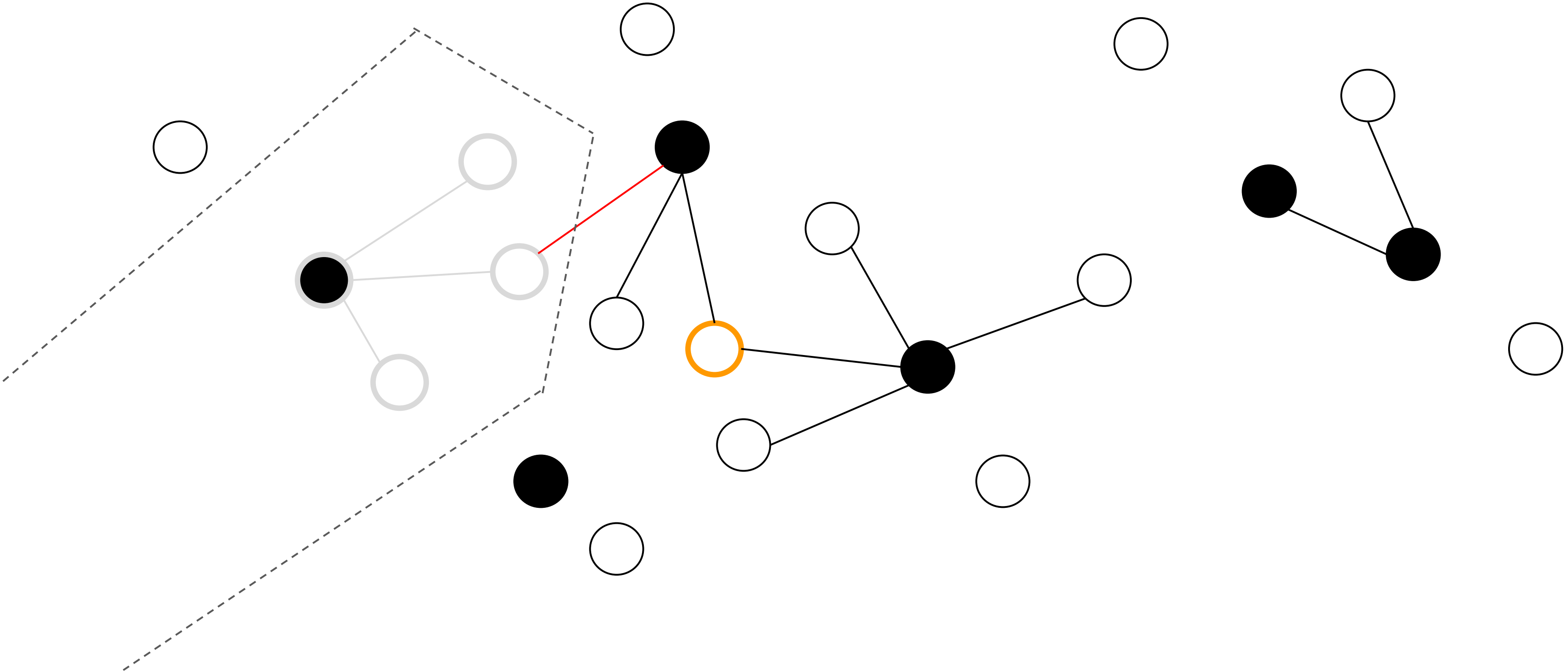
Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen



Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen

Teilschritte:

1. Zeige, dass der Algorithmus terminiert.

Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen

Teilschritte:

1. Zeige, dass der Algorithmus terminiert.
2. Zeige, dass sich alle Kanten des Ursprungsgraphen, in einer der ausgewählten Nachbarschaften befinden.

Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen

Teilschritte:

1. Zeige, dass der Algorithmus terminiert.
2. Zeige, dass sich alle Kanten des Ursprungsgraphen, in einer der ausgewählten Nachbarschaften befinden.
3. Zeige, dass jede so enstehende Knotenzerlegung zu einer präzisen Hashfunktion korrespondiert.

Beweisskizze:

Anwendung eines Konstruktionsalgorithmus für perfekte Hashfunktionen

Teilschritte:

1. Zeige, dass der Algorithmus terminiert.
2. Zeige, dass sich alle Kanten des Ursprungsgraphen, in einer der ausgewählten Nachbarschaften befinden.
3. Zeige, dass jede so enstehende Knotenzerlegung zu einer präzisen Hashfunktion korrespondiert.

q.e.d.

Diskussion des Ergebnisses

Theorem 1:

- Berechne zu jedem Knoten v die minimale Anzahl an Cliquen, die ausreichen um alle Kanten der Nachbarschaft von v zu überdecken

Diskussion des Ergebnisses

Theorem 1:

- Berechne zu jedem Knoten v die minimale Anzahl an Cliquen, die ausreichen um alle Kanten der Nachbarschaft von v zu überdecken

- Das Maximum dieser Zahlen ist eine Untergrenze für die gesuchte Anzahl - weniger geht nicht

Diskussion des Ergebnisses

Theorem 1:

- Berechne zu jedem Knoten v die minimale Anzahl an Cliquen, die ausreichen um alle Kanten der Nachbarschaft von v zu überdecken
- Das Maximum dieser Zahlen ist eine Untergrenze für die gesuchte Anzahl - weniger geht nicht
- Das Bestimmen der minimalen Anzahl an Cliquen, die ausreichen einen Graphen zu überdecken ist ein *NP*-vollständiges Problem!

Diskussion des Ergebnisses

Theorem 2:

- Der maximale Grad eines Anfrageknotens ist eine Obergrenze der gesuchten Anzahl

Diskussion des Ergebnisses

Theorem 2:

- Der maximale Grad eines Anfrageknotens ist eine Obergrenze der gesuchten Anzahl

- Dies entspricht der maximalen Anzahl an Datenpunkten, die ein Ball von festem Radius überdecken kann

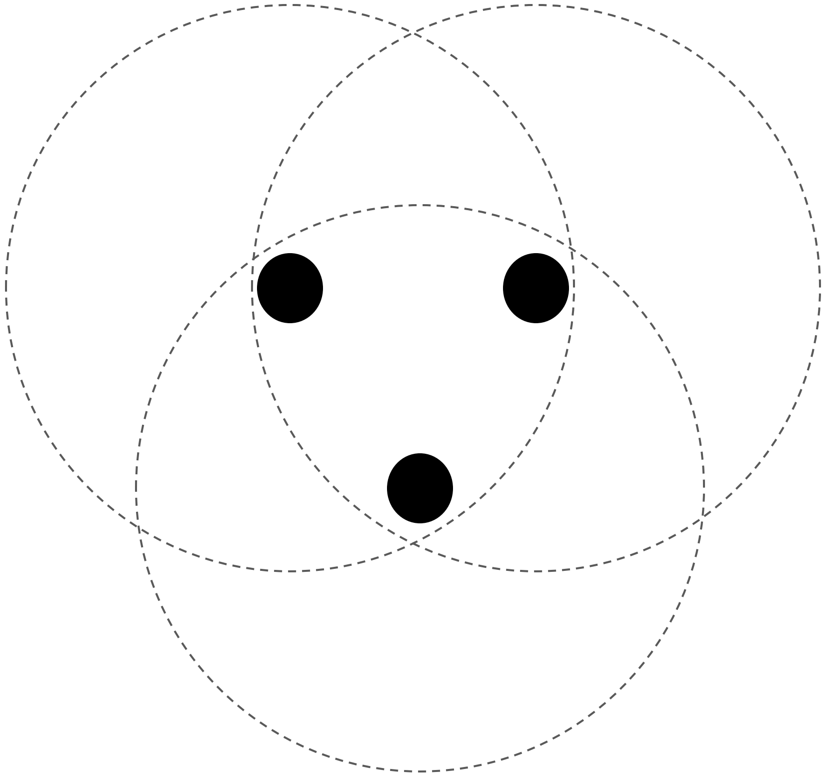
Diskussion des Ergebnisses

Theorem 2:

- Der maximale Grad eines Anfrageknotens ist eine Obergrenze der gesuchten Anzahl
- Dies entspricht der maximalen Anzahl an Datenpunkten, die ein Ball von festem Radius überdecken kann
- Das Bestimmen dieser Zahl ist sogar exponentiell schwer (abhängig von der Dimension)!

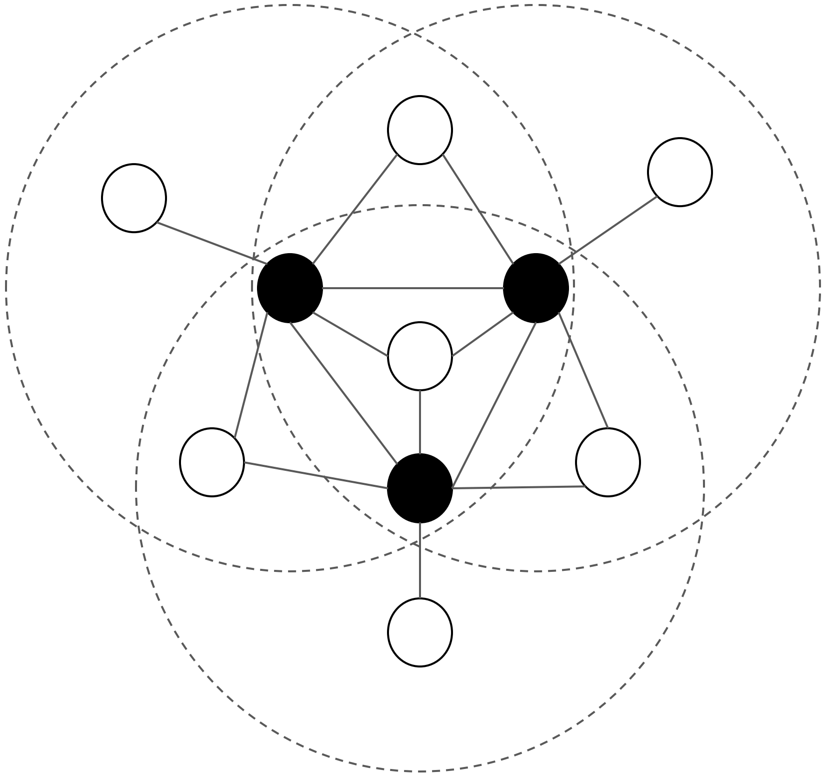
Diskussion des Ergebnisses

Die Grenzen sind echte Über- und Unterschätzungen:



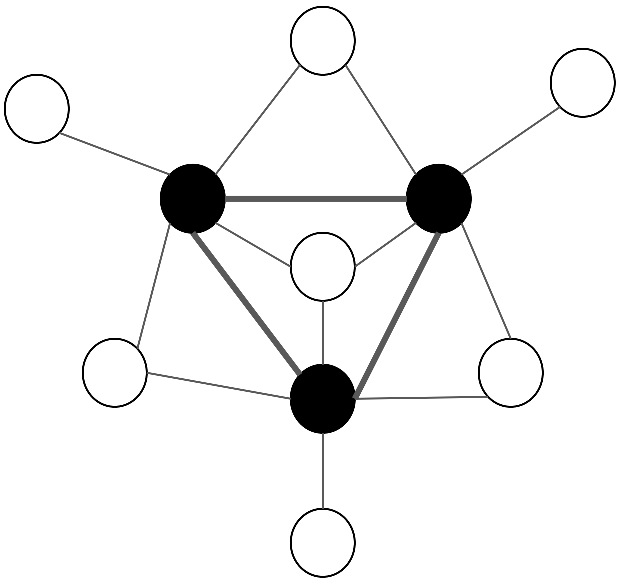
Diskussion des Ergebnisses

Die Grenzen sind echte Über- und Unterschätzungen:



Diskussion des Ergebnisses

Die Grenzen sind echte Über- und Unterschätzungen:



- Jeder Anfragepunkt ist nur in einer Clique enthalten

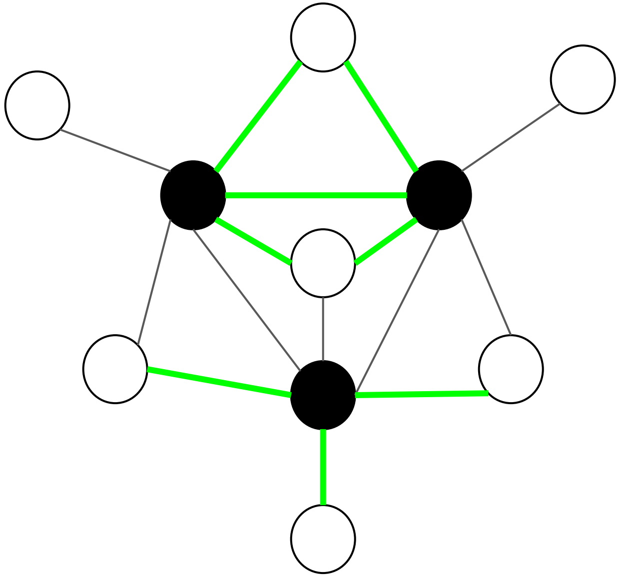
  ⇒ Untergrenze: 1

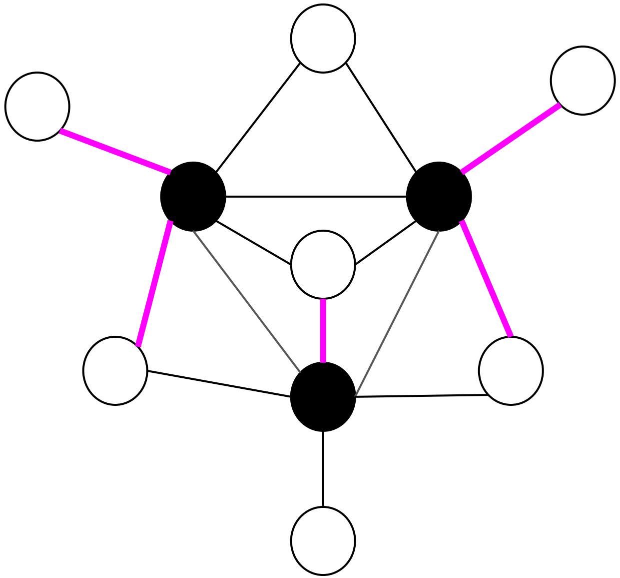
- Der Maximalgrad eines Anfragepunkts ist 3

  ⇒ Obergrenze: 3

Diskussion des Ergebnisses

Die Grenzen sind echte Über- und Unterschätzungen:

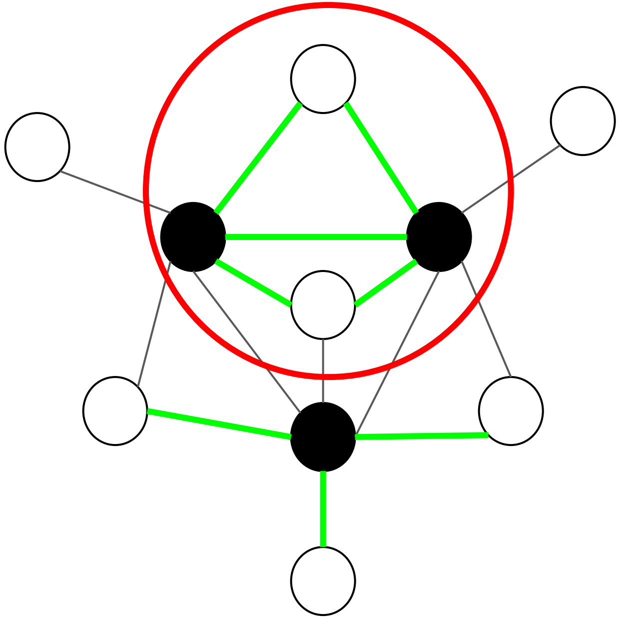


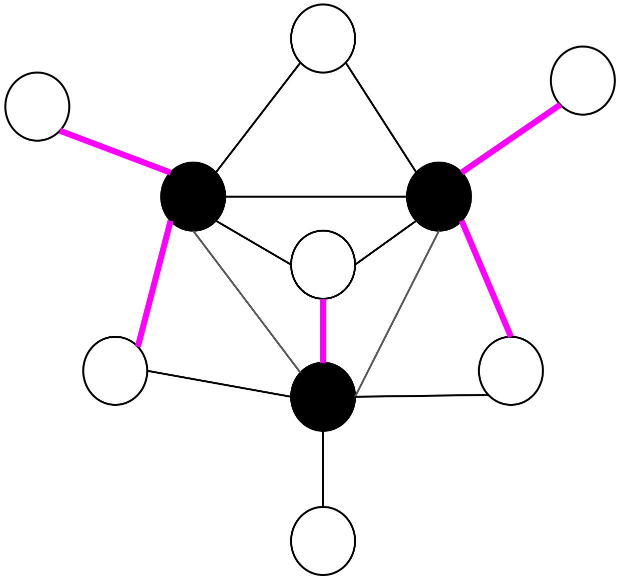


$\Rightarrow 2$ Hashfunktionen reichen aus!

Diskussion des Ergebnisses

Die Grenzen sind echte Über- und Unterschätzungen:





$\Rightarrow$ 2 Hashfunktionen reichen aus!

Bearbeitete Fragestellungen

1. Einordnung existierender Hashverfahren hinsichtlich ihrer Konstruktionsprinzipien

2. Abschätzungen einer oberen und einer unteren Schranke der Anzahl von benötigten Hashfunktionen für ideales Multi-Table-Hashing

Bearbeitete Fragestellungen

1. Einordnung existierender Hashverfahren hinsichtlich ihrer Konstruktionsprinzipien

2. Abschätzungen einer oberen und einer unteren Schranke der Anzahl von benötigten Hashfunktionen für ideales Multi-Table-Hashing

✓ Obergrenze gefunden

Bearbeitete Fragestellungen

1. Einordnung existierender Hashverfahren hinsichtlich ihrer Konstruktionsprinzipien

2. Abschätzungen einer oberen und einer unteren Schranke der Anzahl von benötigten Hashfunktionen für ideales Multi-Table-Hashing

✓ Obergrenze gefunden
✓ Untergrenze gefunden

Bearbeitete Fragestellungen

1. Einordnung existierender Hashverfahren hinsichtlich ihrer Konstruktionsprinzipien

2. Abschätzungen einer oberen und einer unteren Schranke der Anzahl von benötigten Hashfunktionen für ideales Multi-Table-Hashing

✓ Obergrenze gefunden
✓ Untergrenze gefunden

✓ Komplexität der Berechnung dieser Grenzen eingeordnet

Ausblick und Folgefragen

1. Lassen sich engere Ober- und Untergrenzen bzw. lässt sich die Anzahl exakt bestimmen?

Ausblick und Folgefragen

1. Lassen sich engere Ober- und Untergrenzen bzw. lässt sich die Anzahl exakt bestimmen?

2. Lassen sich besser berechenbare Ober- und Untergrenzen finden?

Ausblick und Folgefragen

1. Lassen sich engere Ober- und Untergrenzen bzw. lässt sich die Anzahl exakt bestimmen?

2. Lassen sich besser berechenbare Ober- und Untergrenzen finden?

3. Lassen sich die Grenzen konkret für reale Datensätze ausrechnen/approximieren?

Ausblick und Folgefragen

1. Lassen sich engere Ober- und Untergrenzen bzw. lässt sich die Anzahl exakt bestimmen?

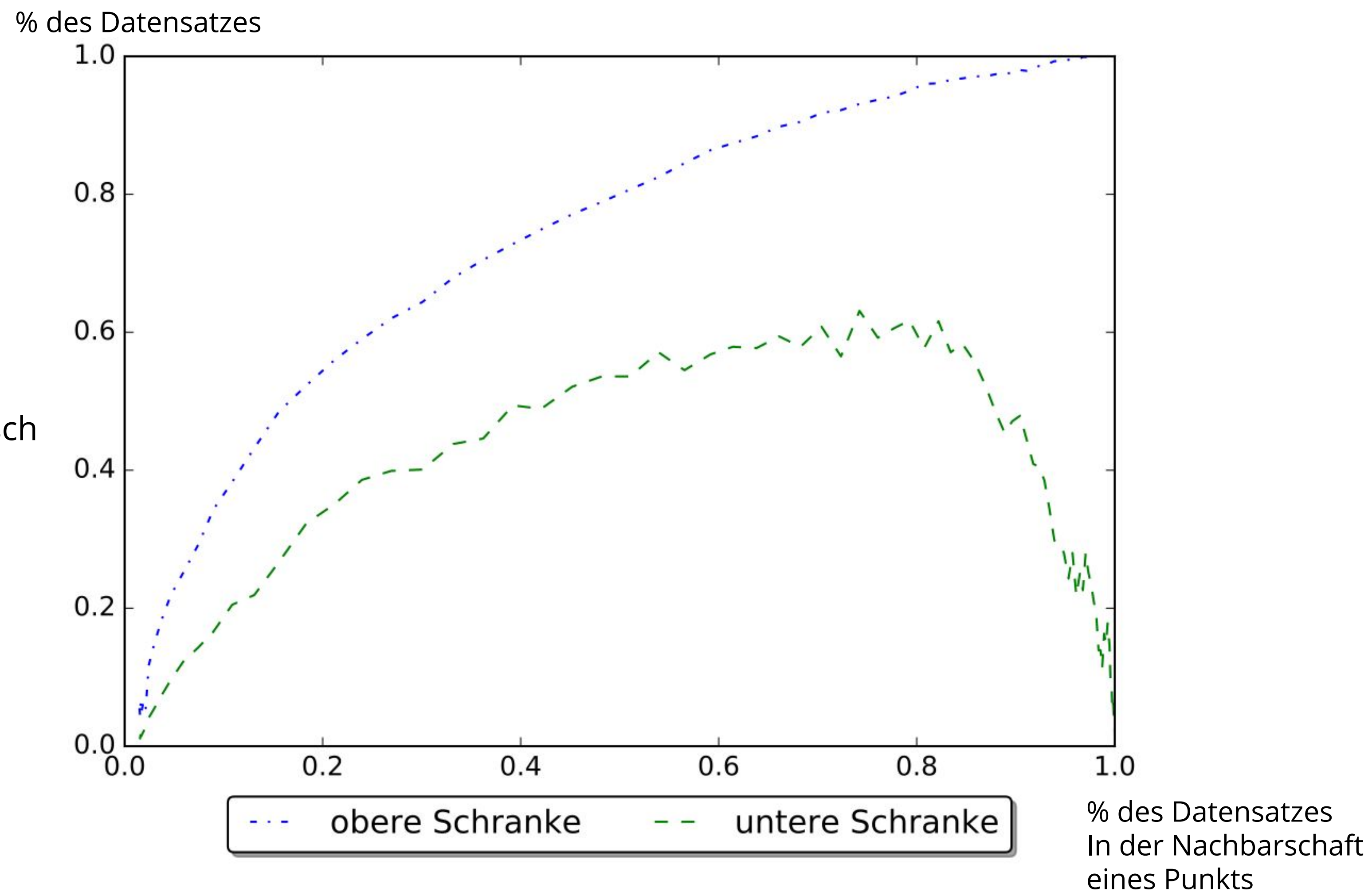
2. Lassen sich besser berechenbare Ober- und Untergrenzen finden?

3. Lassen sich die Grenzen konkret für reale Datensätze ausrechnen/approximieren?

4. Lassen sich mit dem Ergebnis bessere Hashfunktionen konstruieren?

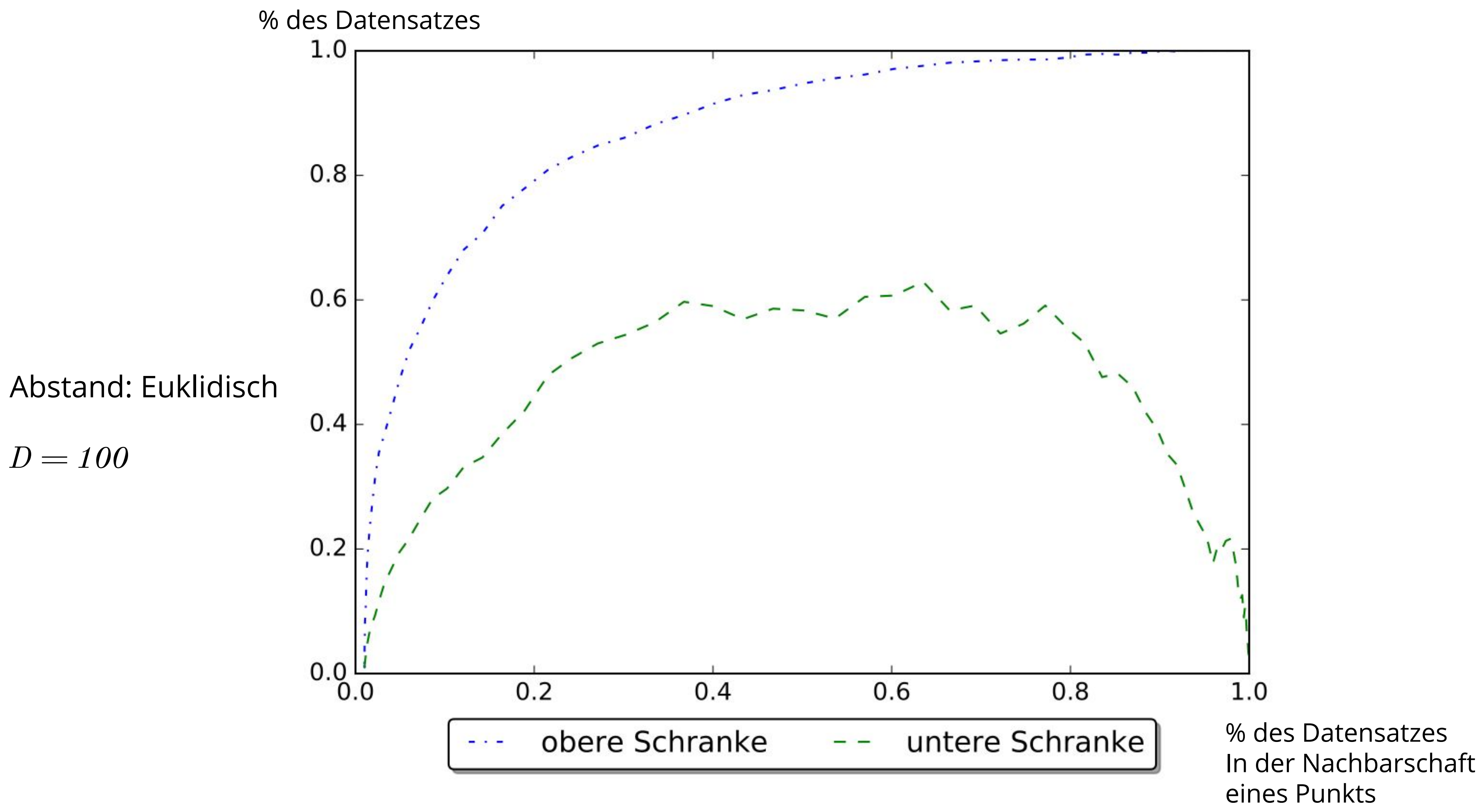
Danke! :-)

Fragen?



Abstand: Euklidisch

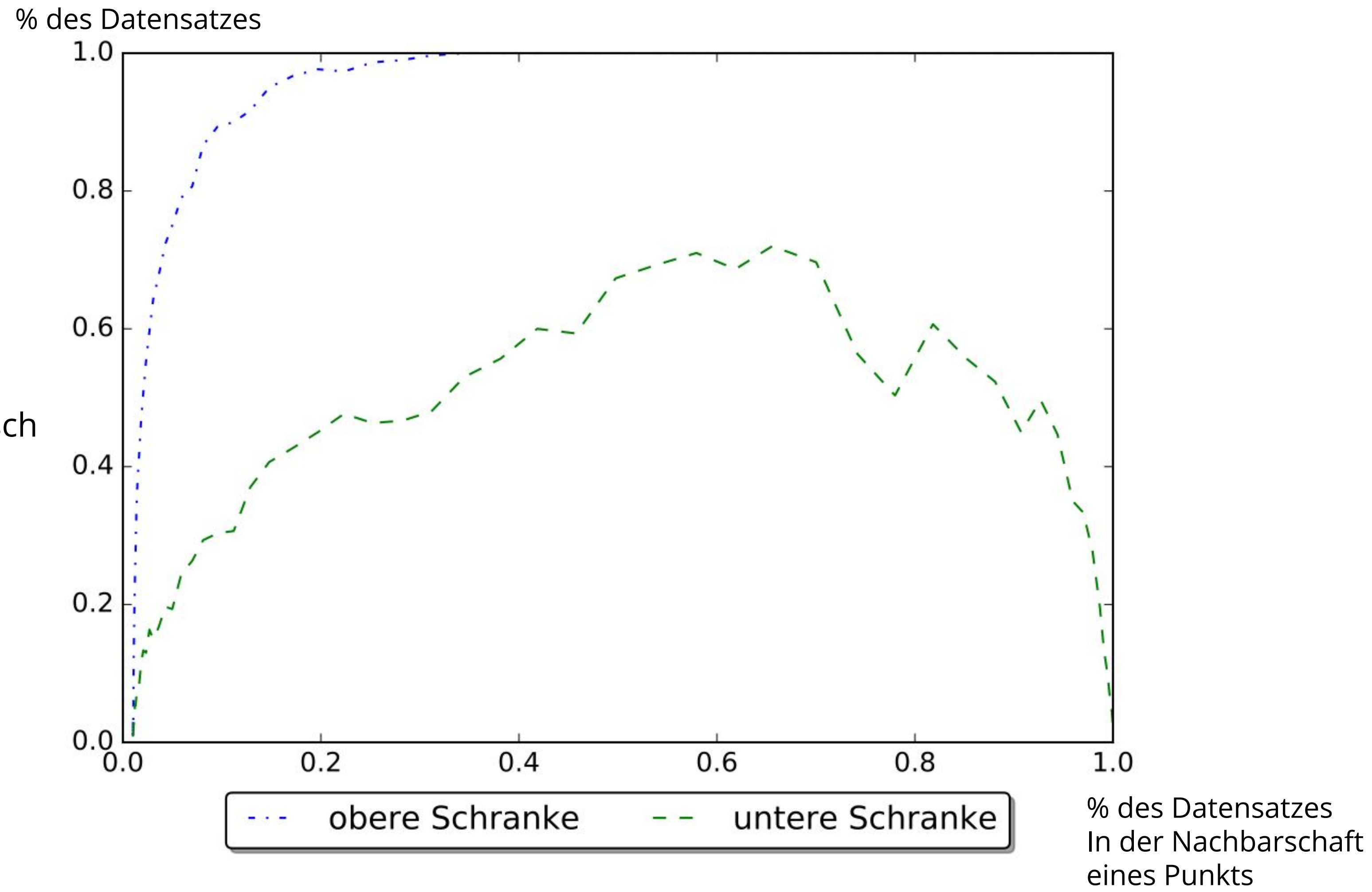
$D = 10$



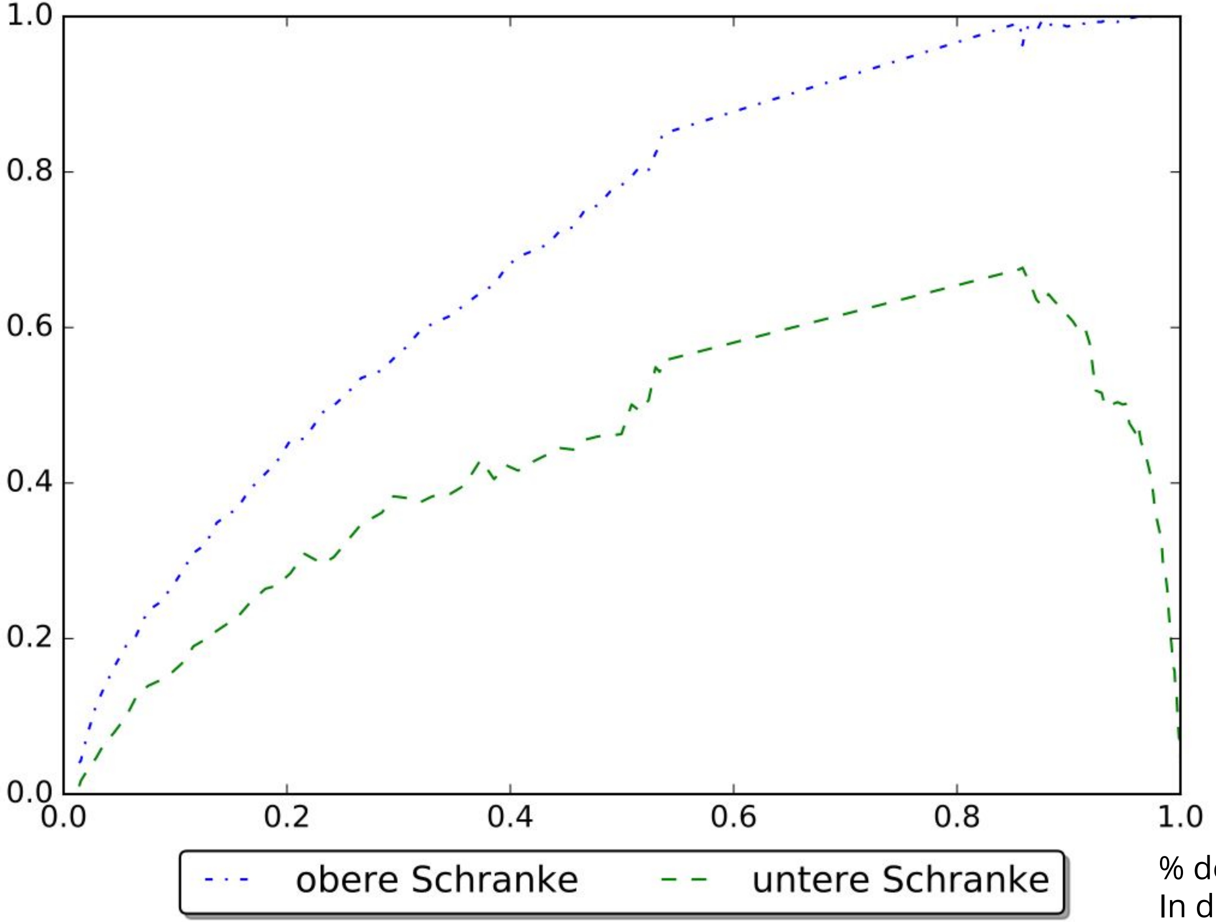
% des Datensatzes
Abstand: Euklidisch
$D = 100$
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
obere Schranke
untere Schranke
% des Datensatzes
In der Nachbarschaft
eines Punkts

Abstand: Euklidisch

$D = 1000$



% des Datensatzes



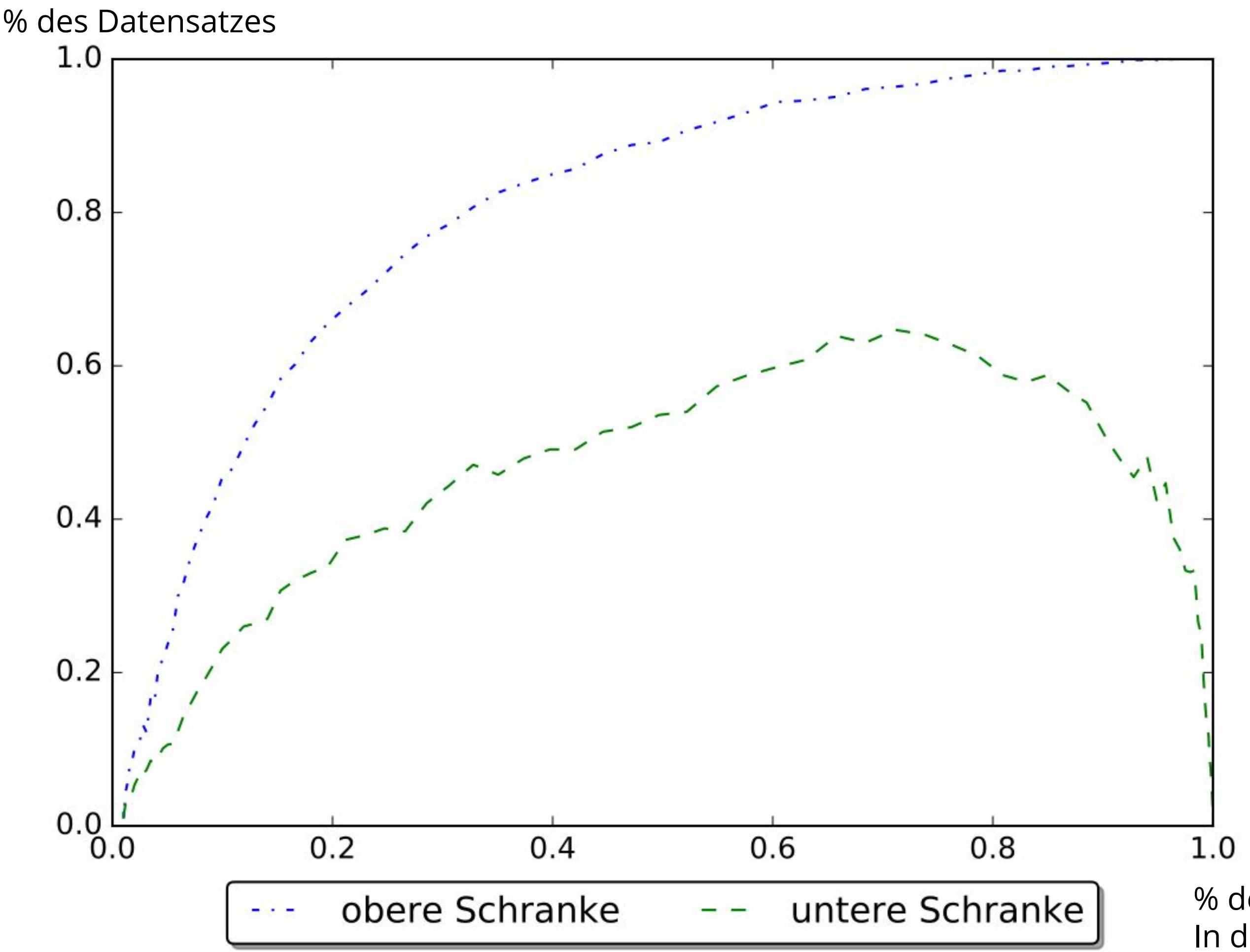
% des Datensatzes
In der Nachbarschaft
eines Punkts

Abstand: Kosinus

$D = 10$

Abstand: Kosinus

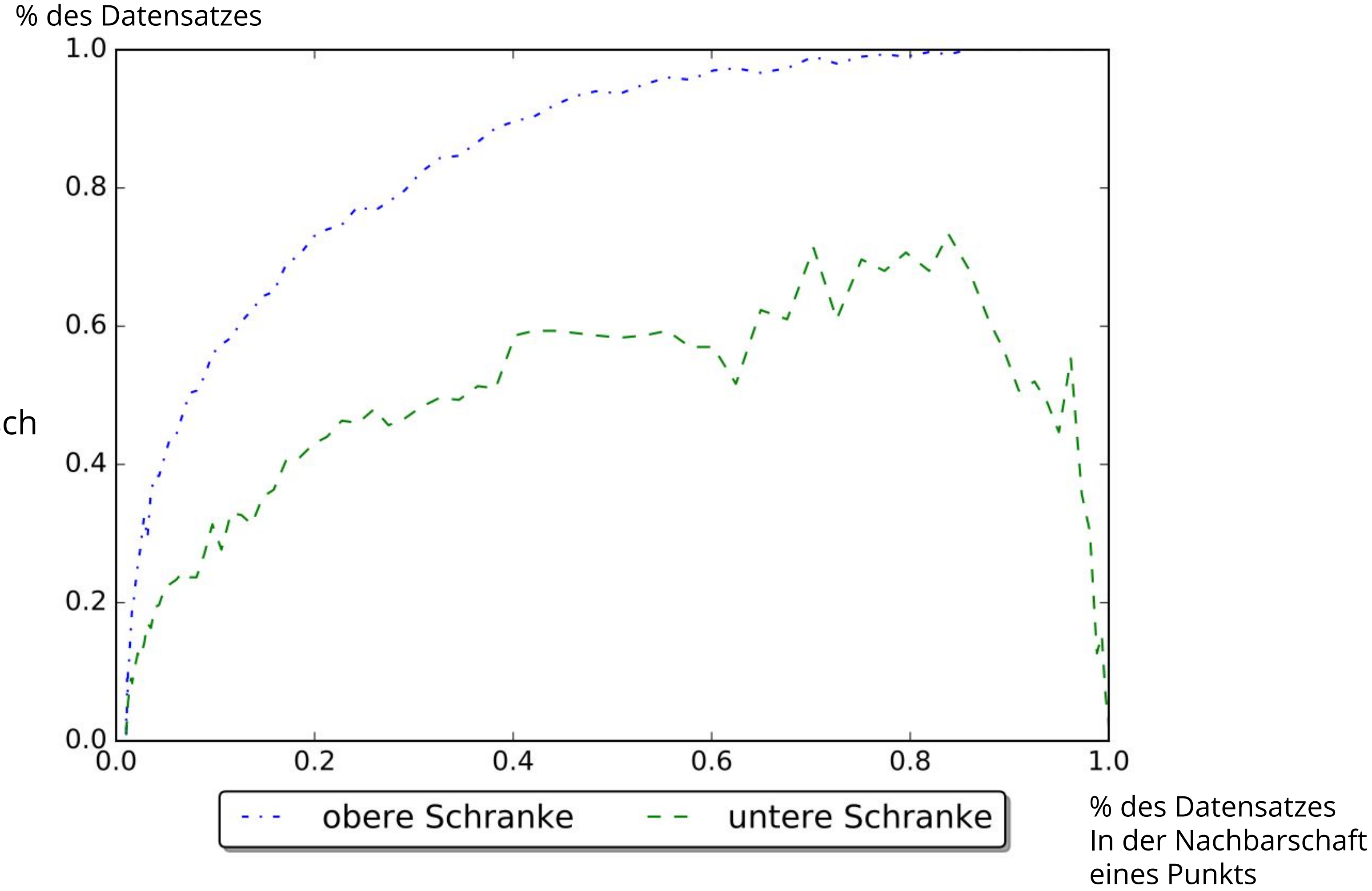
$D = 100$

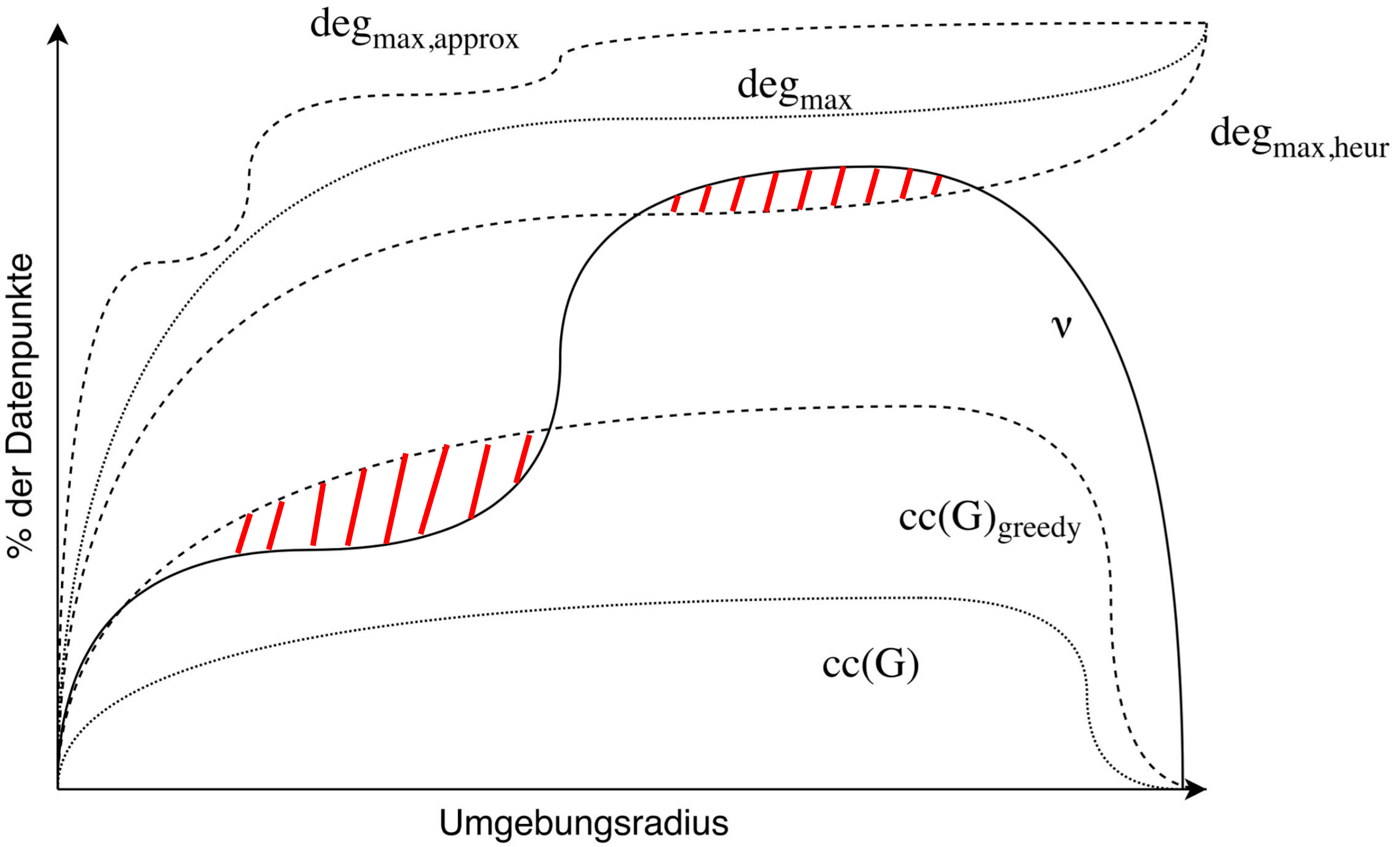


% des Datensatzes
In der Nachbarschaft
eines Punkts

Abstand: Euklidisch

$D = 1000$





$deg_{max,approx}$
deg_{max}
$deg_{max,heur}$
ν
$cc(G)_{greedy}$
$cc(G)$
% der Datenpunkte
Umgebungsradius