Verteilte Algorithmen und Datenstrukturen

Kapitel 2: Netzwerktheorie

Prof. Dr. Christian Scheideler

Institut für Informatik

Universität Paderborn

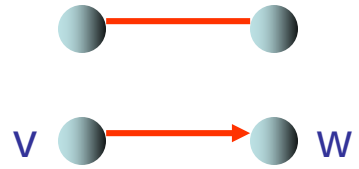
Übersicht

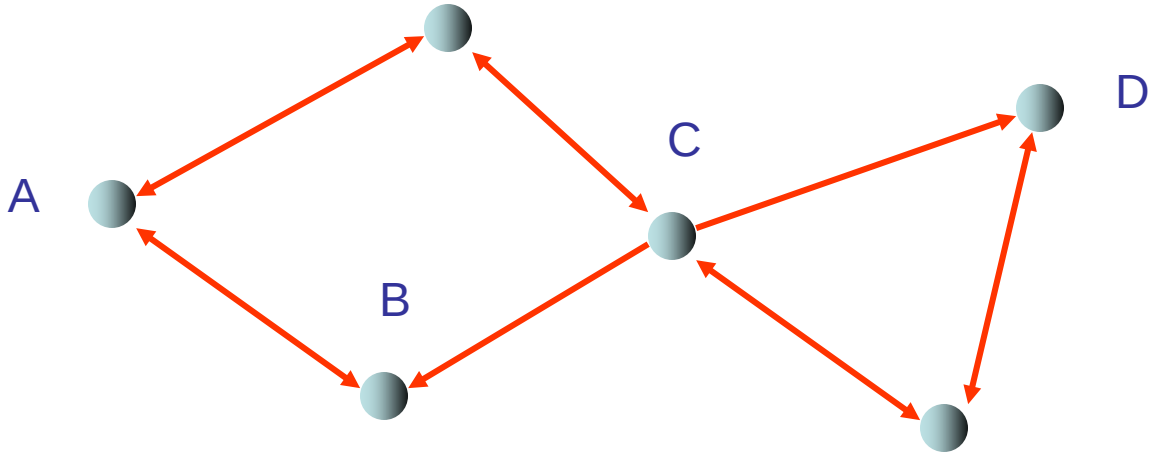
- Grundlagen
- Grundlegende Graphparameter
- Klassische Graphfamilien
- Skip Graphen
- Delaunay Graphen
- Routing

Grundlagen

Definition 2.1: Ein Graph G=(V,E) besteht aus einer Knotenmenge V und Kantenmenge E.

- G ungerichtet: $E \subseteq \{ \{v,w\} \mid v,w \in V\}$
- G gerichtet: $E \subseteq \{ (v,w) \mid v,w \in V\}$

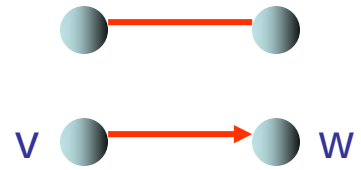




Grundlagen

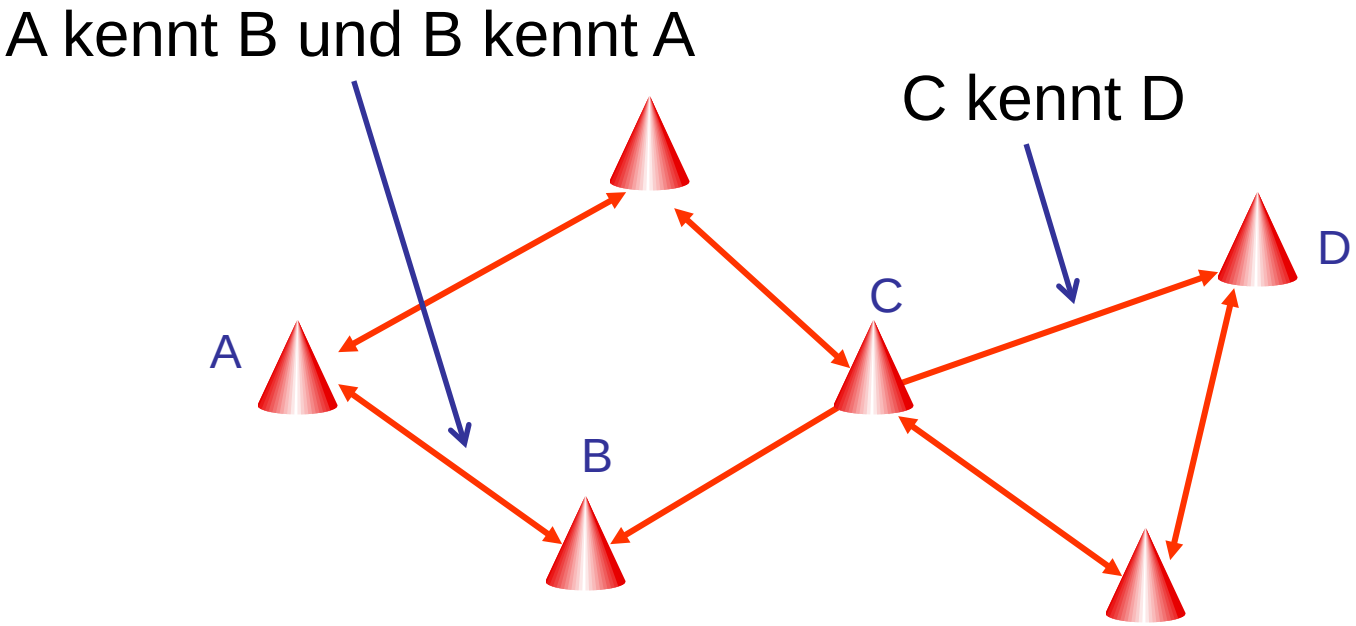
Definition 2.1: Ein Graph $G=(V,E)$ besteht aus einer Knotenmenge V und Kantenmenge E.

- G ungerichtet: $E \subseteq \{ \{v,w\} \mid v,w \in V\}$
- G gerichtet: $E \subseteq \{ (v,w) \mid v,w \in V\}$
- G bigerichtet: für alle $(v,w) \in E$ ist auch $(w,v) \in E$



Grundlagen

Graph: repräsentiert Wissen über bzw. Verbindungen zwischen Prozessen

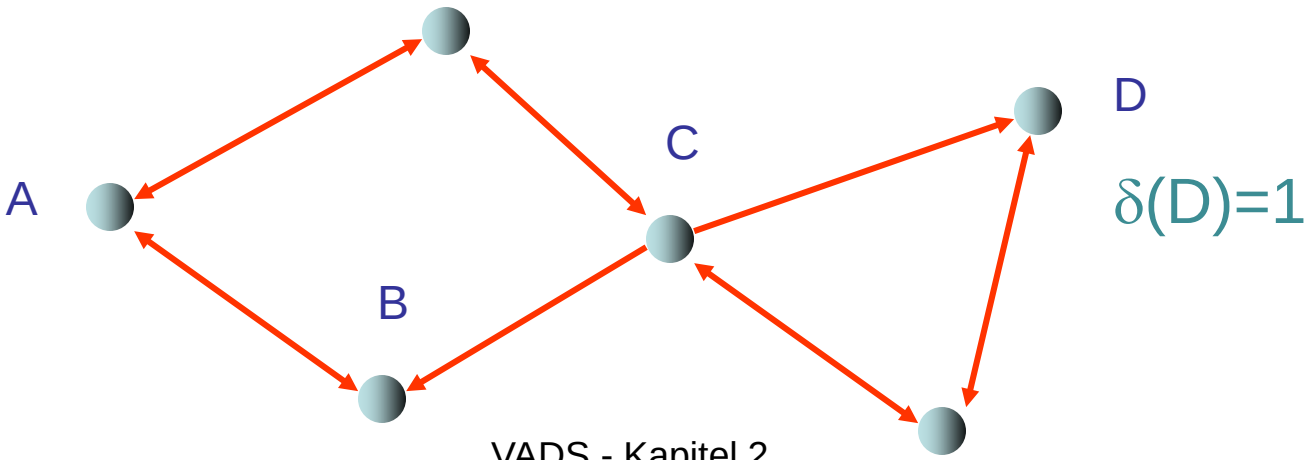


Grundlagen

Definition 2.2: Sei $G=(V,E)$ ein Graph.

- G ungerichtet: Grad von $v \in V$:
$$\delta(v)=|\{ w \in V \mid \{v,w\} \in E\}|$$
- G gerichtet: Grad von $v \in V$:
$$\delta(v)=|\{ w \in V \mid (v,w) \in E\}|$$

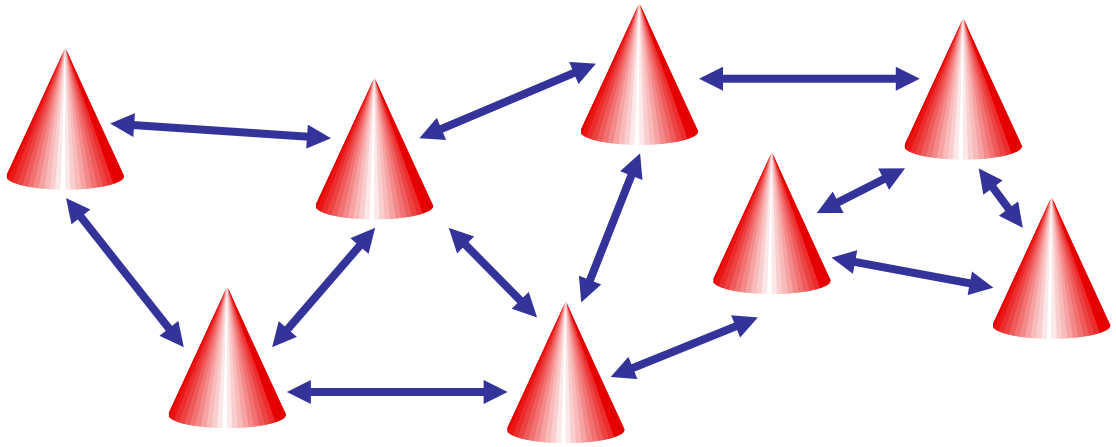
Grad von G: $\Delta = \max_{v \in V} \delta(v)$



Grundlagen

Grad:

- worst-case Aufwand pro Prozess für Kontrolle seiner Verbindungen
- in bigerichteten Graphen, worst-case update Kosten für Prozess, falls sich Menge der Prozesse verändert.

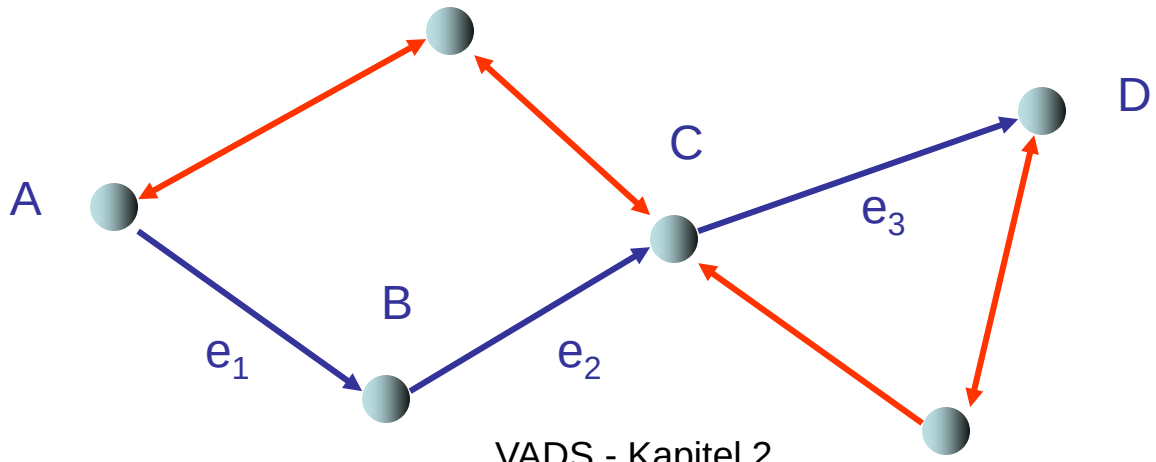


Grad sollte nicht zu hoch sein.

Grundlagen

Definition 2.3: Sei $G=(V,E)$ ein Graph. Eine Kantenfolge $p=(e_1,e_2,\ldots,e_k)$ in G heißt Weg/Pfad, falls es eine Knotenfolge $(v_0,\ldots,v_k)$ gibt mit

- G ungerichtet: $e_i=\{v_{i-1},v_i\}$ für alle $i\in\{1,\ldots,k\}$
- G gerichtet: $e_i=(v_{i-1},v_i)$ für alle $i\in\{1,\ldots,k\}$



Grundlagen

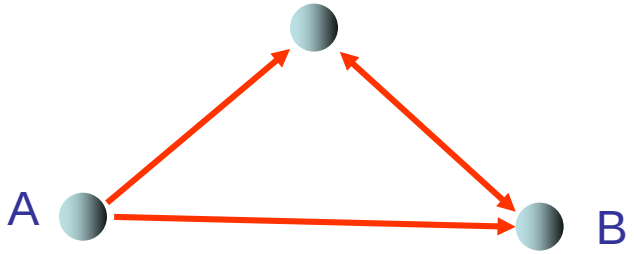
Definition 2.4: Ein Graph $G=(V,E)$ heißt

- zusammenhängend, wenn G ungerichtet ist und für jedes Knotenpaar $v,w \in V$ ein Pfad von v nach w in G existiert.
- schwach zusammenhängend, wenn G gerichtet ist und für jedes Knotenpaar $v,w \in V$ ein Pfad von v nach w in der ungerichteten Version von G existiert
- stark zusammenhängend, wenn G gerichtet ist und für jedes Knotenpaar $v,w \in V$ ein Pfad von v nach w in G existiert

Grundlagen

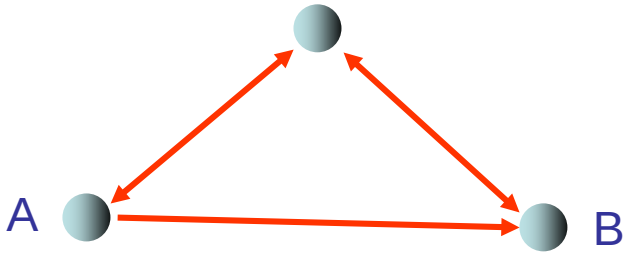
Beispiele:

(1) Graph nur schwach zusammenhängend



kein gerichteter Weg
von B nach A

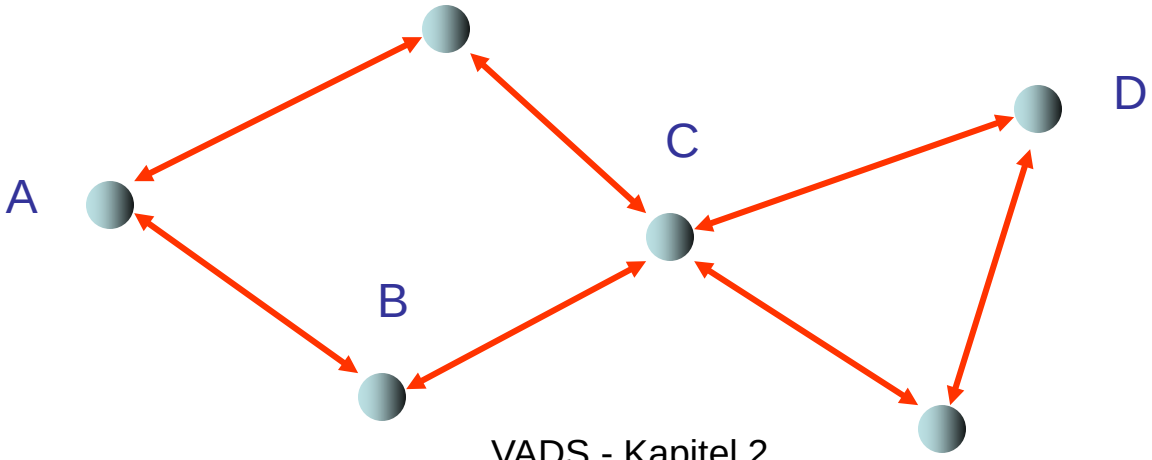
(2) Graph stark zusammenhängend



Grundlegende Graphparameter

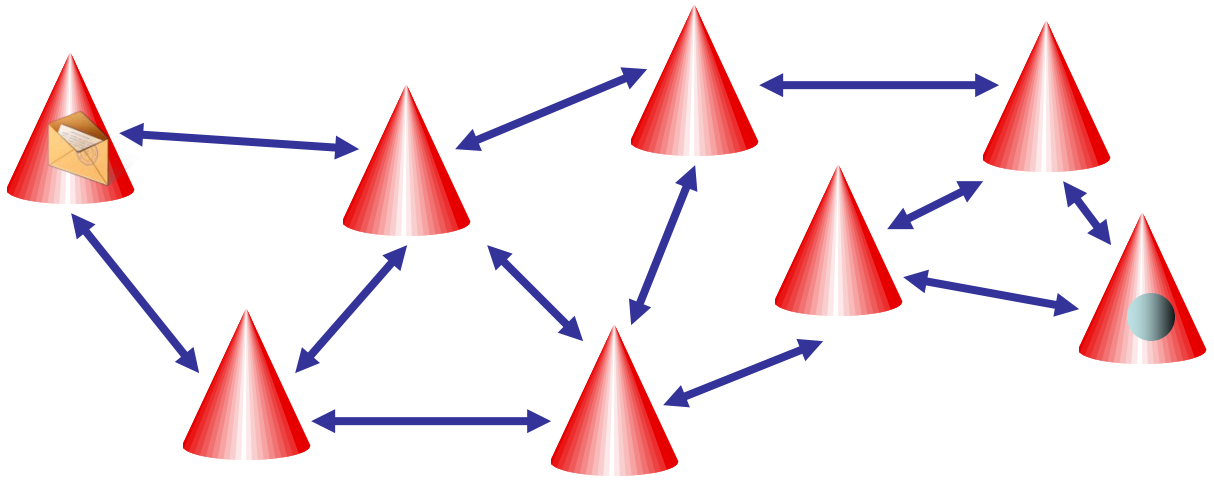
Definition 2.5: Sei $G=(V,E)$ ein Graph und $p=(e_1,e_2,\ldots,e_k)$ ein Weg von v nach w in G.

- Länge von p: $|p|=k$
- Distanz von w zu v: $d(v,w)$ = min. Weglänge von v nach w ($d(v,w) = \infty$ falls kein Weg von v nach w existiert)
- Durchmesser von G: $D(G)=\max_{v,w\in V} d(v,w)$



Grundlegende Graphparameter

Durchmesser: untere Schranke für worst-case Zeit (gemessen in Kommunikationsrunden) für Zugriff auf anderen Prozess

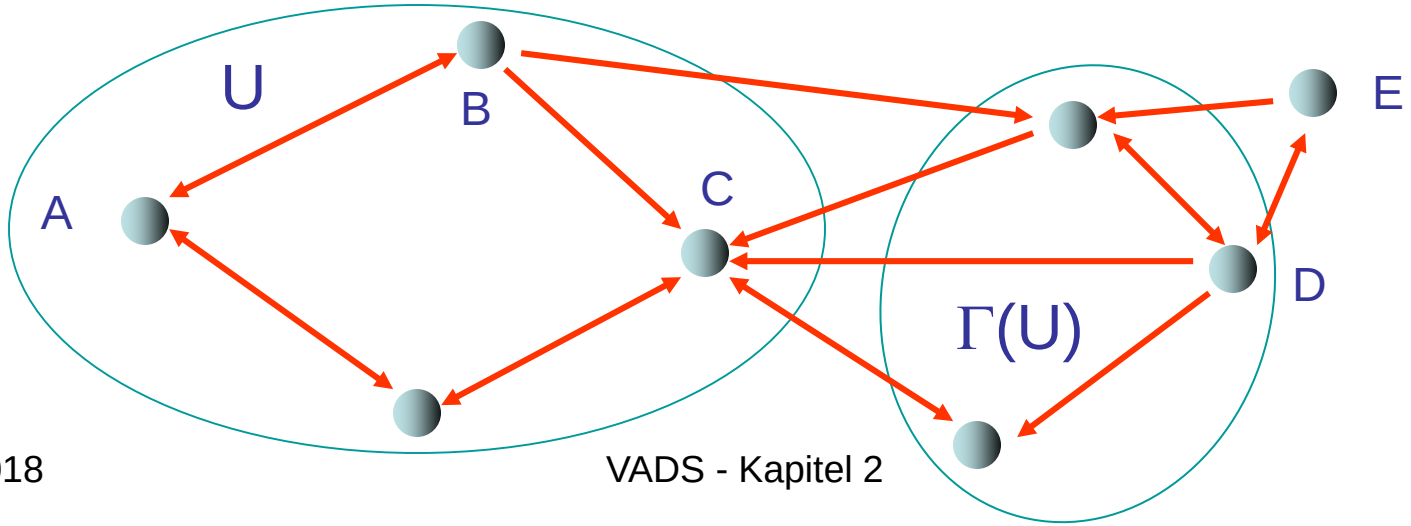


Durchmesser sollte nicht zu hoch sein.

Grundlegende Graphparameter

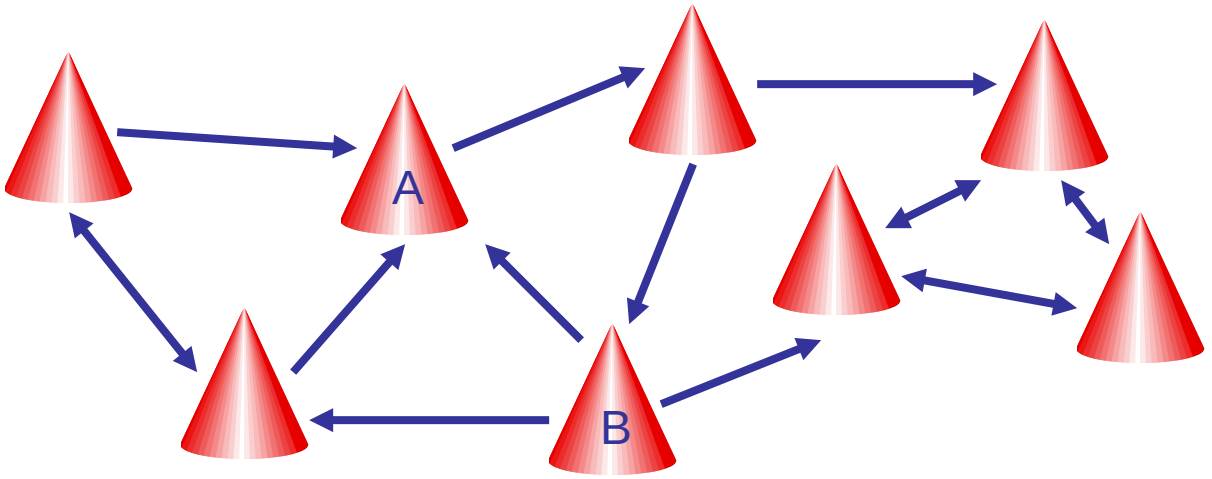
Definition 2.6: Sei $G=(V,E)$ ein Graph.

- $\Gamma(U)$: Nachbarmenge einer Knotenmenge $U \subseteq V$, d.h.
  $\Gamma(U) = \{ w \in V \setminus U \mid$ es gibt $v \in U$ mit $\{v,w\} \in E$ (bzw. $(v,w) \in E$ oder $(w,v) \in E$ im gerichteten Fall) $\}$
- $\alpha(U) = |\Gamma(U)| / |U|$: Expansion von U
- $\alpha(G) = \min_{U,|U| \leq \lceil |V|/2 \rceil} \alpha(U)$: Expansion von G



Grundlegende Graphparameter

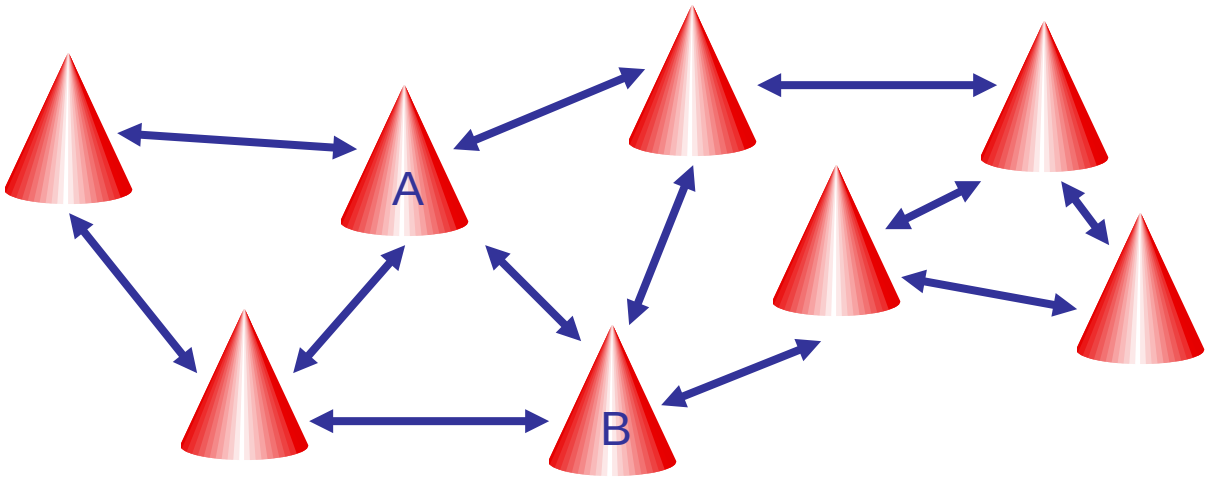
Expansion: k Ausfälle $\Rightarrow$maximal $k/\alpha(G)$ Knoten werden vom Graphen abgetrennt



Beweis: Sei U eine Menge nicht augefallener Knoten, die durch Ausfälle abgetrennt werden. Dann müssen alle Knoten in $\Gamma(U)$ ausfallen, d.h. $|\Gamma(U)| \leq k$. Weiterhin ist $\alpha(U) \geq \alpha(G)$ und $\alpha(U)=|\Gamma(U)|/|U|$, also $|U| \leq k/\alpha(G)$.

Grundlegende Graphparameter

Expansion: k Ausfälle $\Rightarrow$maximal $k/\alpha(G)$ Knoten werden vom Graphen abgetrennt

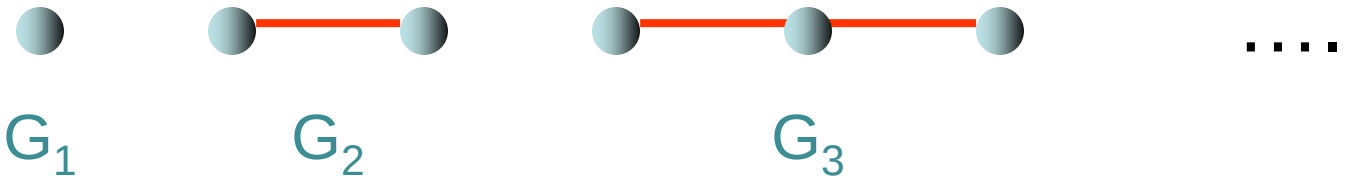


Expansion sollte möglichst groß sein

Klassische Graphfamilien

Im folgenden betrachten wir klassische Familien ungerichteter Graphen $\mathcal{G}=\{G_1, G_2, \ldots\}$.

Beispiel: Familie der linearen Listen

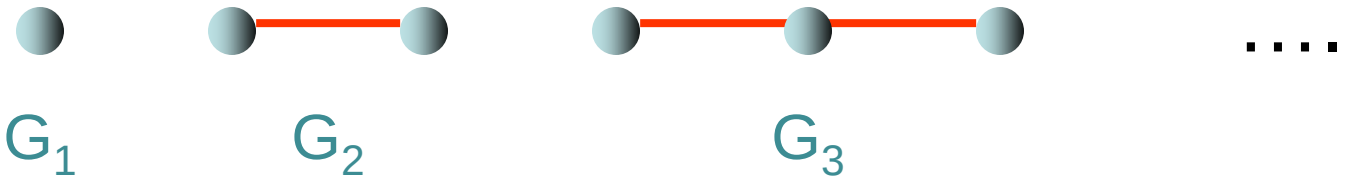


Wir sagen: Graph G aus einer Familie $\mathcal{G}$ hat konstanten Grad, falls der Grad aller Graphen in $\mathcal{G}$ durch eine Konstante beschränkt ist.

Klassische Graphfamilien

Im folgenden betrachten wir klassische Familien ungerichteter Graphen $\mathcal{G}=\{G_1, G_2, \ldots\}$.

Beispiel: Familie der linearen Listen

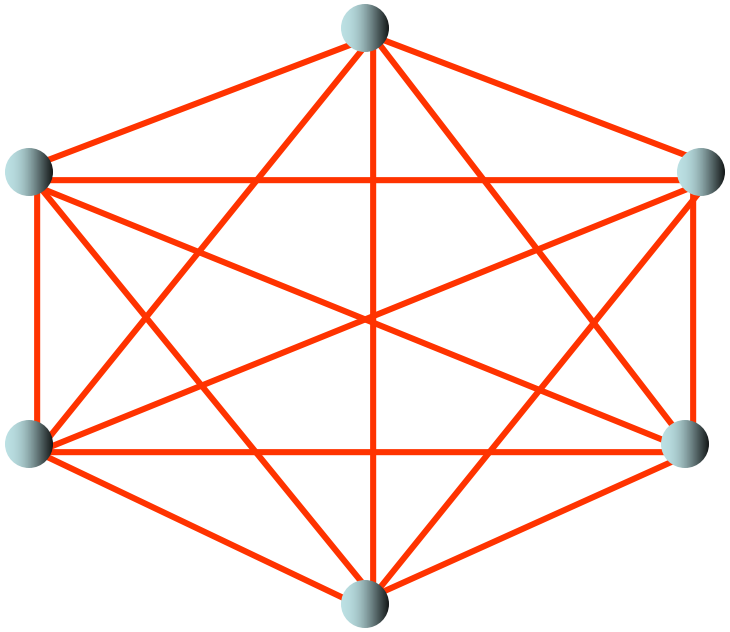


Für einen Graphen G aus $\mathcal{G}$ bezeichnen wir mit

- n: Anzahl der Knoten (bzw. Größe) von G
- m: Anzahl der Kanten von G

Clique

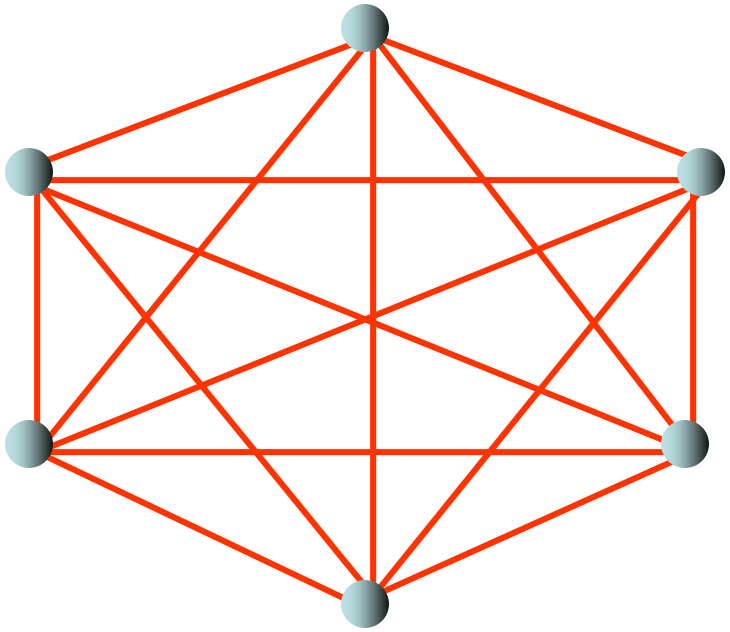
Vollständiger Graph / Clique: jeder Knoten ist mit jedem anderen verbunden



Vorteil: niedriger Durchmesser, hohe Expansion

Clique

Vollständiger Graph / Clique: jeder Knoten ist mit jedem anderen verbunden



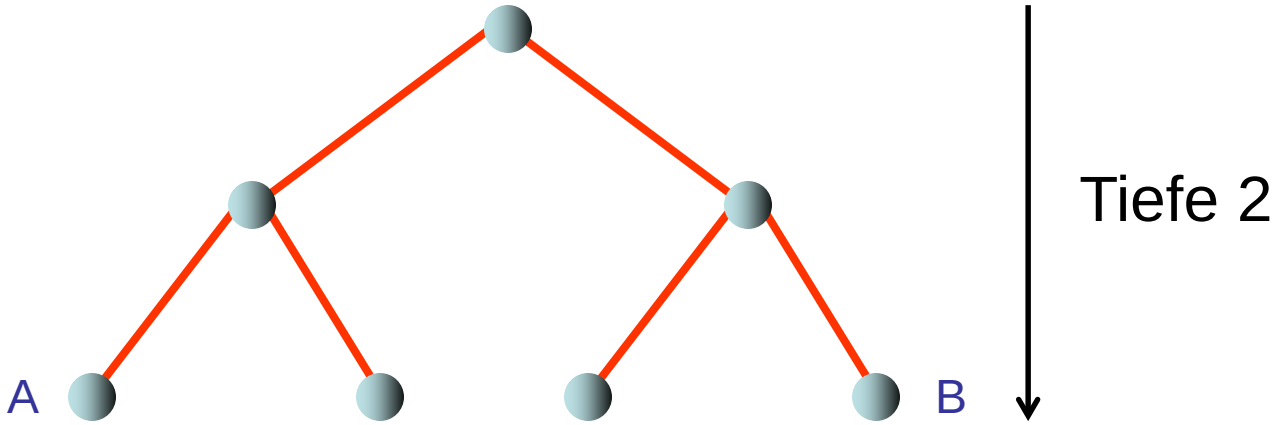
Problem: hoher Grad! ($\delta(v)=n-1$ für alle v)

Lineare Liste

- Grad 2 (minimal für Zusammenhang), ABER
- Durchmesser schlecht (D(Liste)=n-1)
- Expansion schlecht (α(Liste)$\approx$2/n)

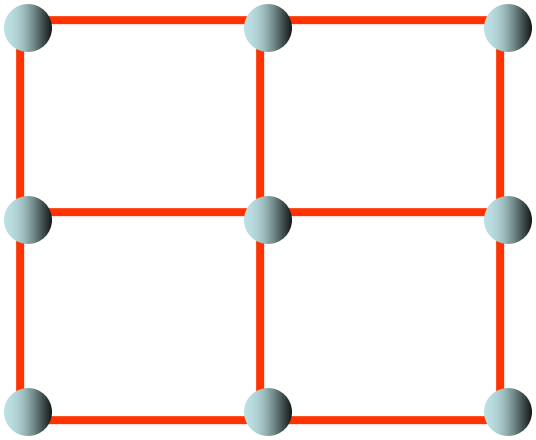
Wie erhält man kleinen Grad und Durchmesser?

Vollständiger binärer Baum



- $n=2^{k+1}-1$ Knoten bei Tiefe $k \in \mathbb{N}_0$, Grad 3
- Durchmesser ist $2k \approx 2 \log_2 n$, ABER
- Expansion schlecht ($\alpha(\text{Baum}) \approx 2/n$)

2-dimensionales Gitter



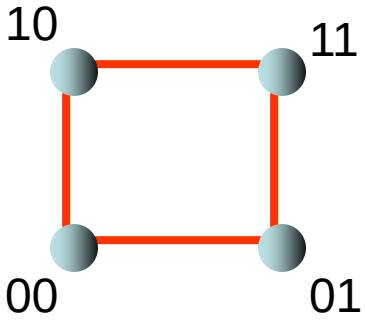
- $n = k^2$ Knoten bei k Knoten pro Seite, maximaler Grad 4
- Durchmesser ist $2(k-1) \approx 2\sqrt{n}$
- Expansion ist $\approx 2/\sqrt{n}$
- Nicht schlecht, aber geht es besser?

d-dimensionaler Hypercube

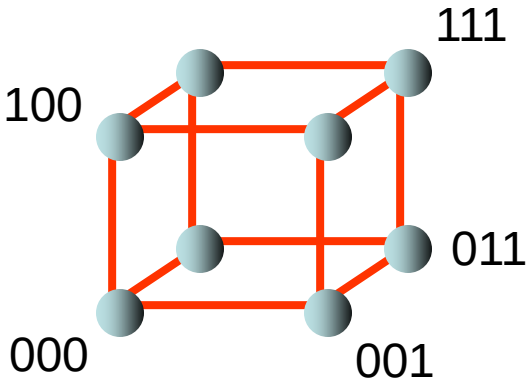
- Knoten: $(x_1,\ldots,x_d) \in \{0,1\}^d$
- Kanten: $\forall i: (x_1,\ldots,x_d) \rightarrow (x_1,..,1\text{-}x_i,..,x_d)$
- ungerichtete Hypercubes:

nur Bit i gedreht

d=1



d=2

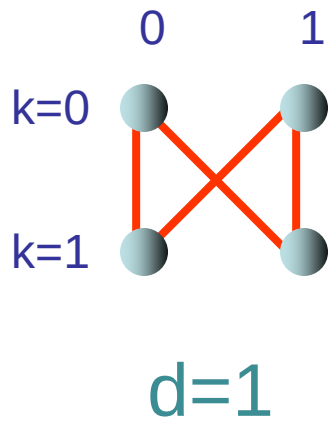


d=3

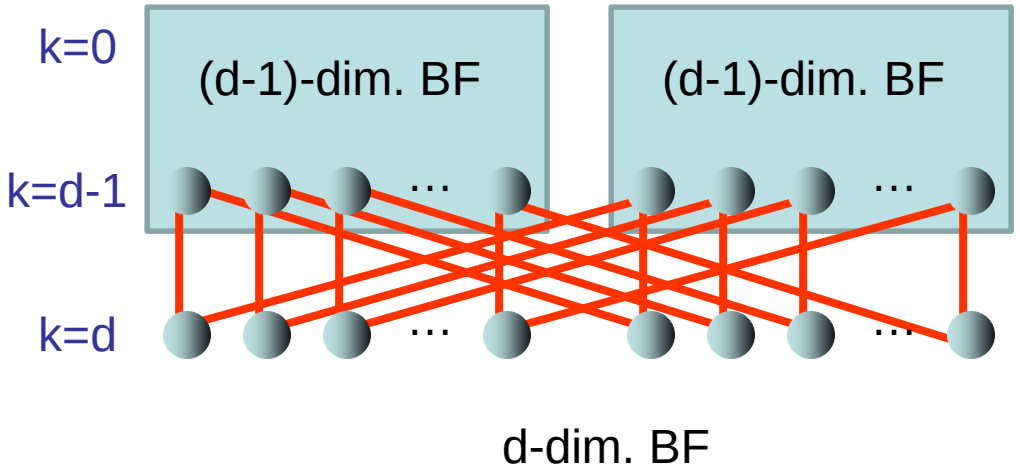
Grad d, Durchmesser d, Expansion $\approx 1/\sqrt{d}$

d-dimensionales Butterfly

- Knoten: $(k,(x_d,\ldots,x_1)) \in \{0,..,d\} \times \{0,1\}^d$
- Kanten: $(k,(x_d,\ldots,x_1)) \rightarrow (k+1,(x_d,..,x_{k+1},..,x_1)),$
  $(k+1,(x_d,..,1\text{-}x_{k+1},..,x_1))$
- ungerichtetes Butterfly:



d=1

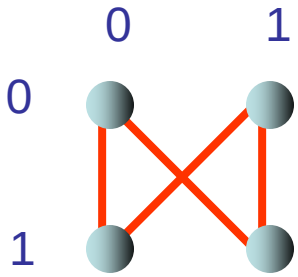


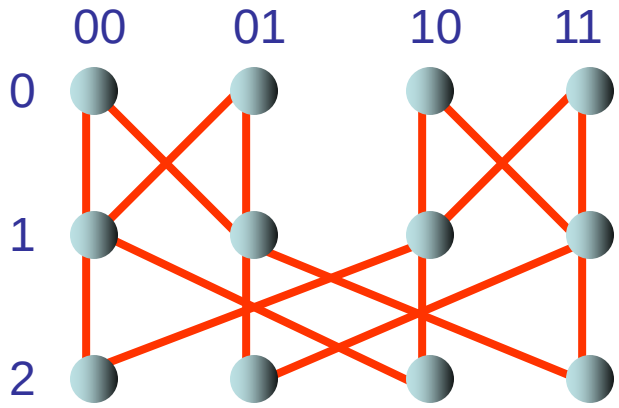
d-dim. BF

d-dimensionales Butterfly

- Knoten: $(k,(x_d,\ldots,x_1)) \in \{0,..,d\} \times \{0,1\}^d$
- Kanten: $(k,(x_d,\ldots,x_1)) \rightarrow (k+1,(x_d,..,x_{k+1}..,x_1)),$
  $(k+1,(x_d,..,1\text{-}x_{k+1},..,x_1))$

nur Bit k+1 gedreht

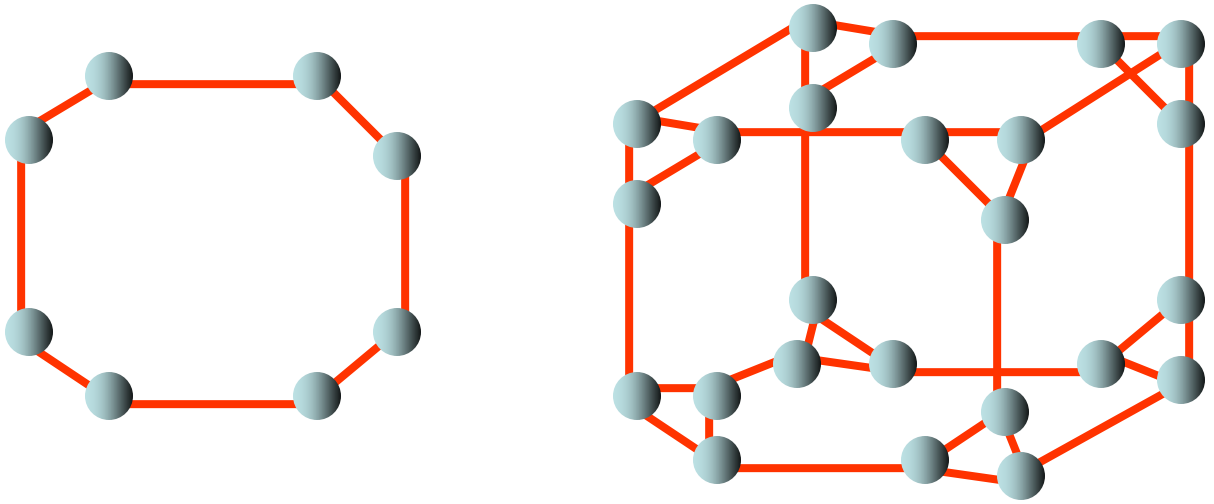




Grad 4, Durchmesser 2d, Expansion ~1/d

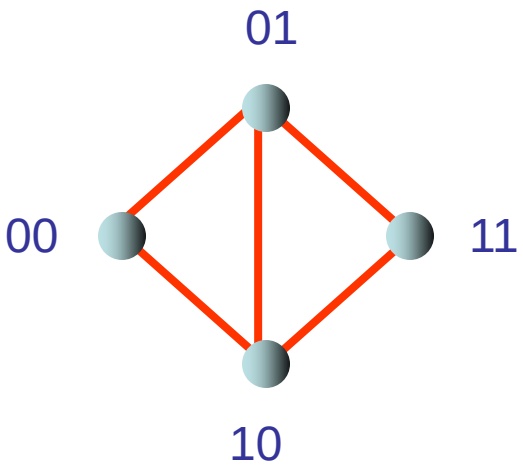
Cube-Connected-Cycles

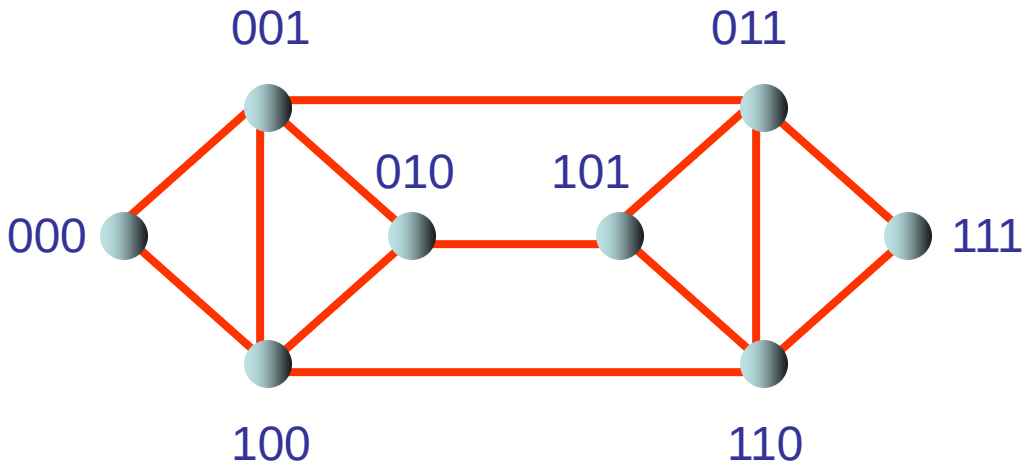
- Knoten: $(k,(x_1,\ldots,x_d)) \in \{0,..,d\text{-}1\} \times \{0,1\}^d$
- Kanten: $(k,(x_1,\ldots,x_d)) \rightarrow (k\text{-}1 \pmod d,(x_1,..,x_d)),$
  $(k\text{+}1 \pmod d,(x_1,..,x_d)),$
  $(k,(x_1,..,1\text{-}x_{k+1},..,x_d))$



d-dimensionaler De Bruijn Graph

- Knoten: $(x_1,\ldots,x_d) \in \{0,1\}^d$
- Kanten: $(x_1,\ldots,x_d) \rightarrow (0,x_1,x_2,\ldots,x_{d-1})$
  $(1,x_1,x_2,\ldots,x_{d-1})$
- ungerichteter de Bruijn Graph:





Durchmesser

Satz 2.7: Jeder Graph mit maximalem Grad $\delta \geq 4$ und Größe n muss einen Durchmesser von mindestens $(\log n)/(\log(\delta-1))-1$ haben.

Beweis: Übung

Satz 2.8: Für jedes gerade $\delta \geq 4$ gibt es eine Familie von Graphen mit maximalem Grad δ und Größe n mit Durchmesser höchstens $(\log n) / (\log \delta - 1)$.

Beweis: Übung

Expansion

Satz 2.9: Für jeden Graph G ist die Expansion $\alpha(G) \in [0,1]$.

Beweis: siehe Definition von $\alpha(G)$.

Definition 2.10: Ein Graph heißt Expander, wenn er eine konstante Expansion besitzt.

Satz 2.11: Es gibt Familien von Graphen mit konstantem Grad und konstanter Expansion.

Beispiel: Gabber-Galil Graph

- Knotenmenge: $(x,y) \in \{0,\ldots,k-1\}^2$
- $(x,y) \rightarrow (x,x+y),(x,x+y+1), (x+y,y), (x+y+1,y) \pmod k$

Noch bessere Expander bekannt als Ramanujan Graphen.

Übersicht

- Grundlagen
- Grundlegende Graphparameter
- Klassische Graphfamilien
- Skip Graphen
- Delaunay Graphen
- Routing

Skip Graphen

Betrachte eine beliebige Menge V an Knoten mit totaler Ordnung (d.h. die Knoten können bzgl. einer Ordnung ´<´ sortiert werden).

- Jeder Knoten v sei assoziiert mit einer genügend langen zufälligen Bitfolge $r(v)$ (so dass $r(v) \neq r(w)$ für alle $v,w \in V$).
- $prefix_i(v)$: erste i Bits von $r(v)$
- $succ_i(v)$: nächster Nachfolger w von v (bzgl. der Ordnung ´<´) mit $prefix_i(w)=prefix_i(v)$.
- $pred_i(v)$: nächster Vorgänger w von v mit $prefix_i(w)=prefix_i(v)$.

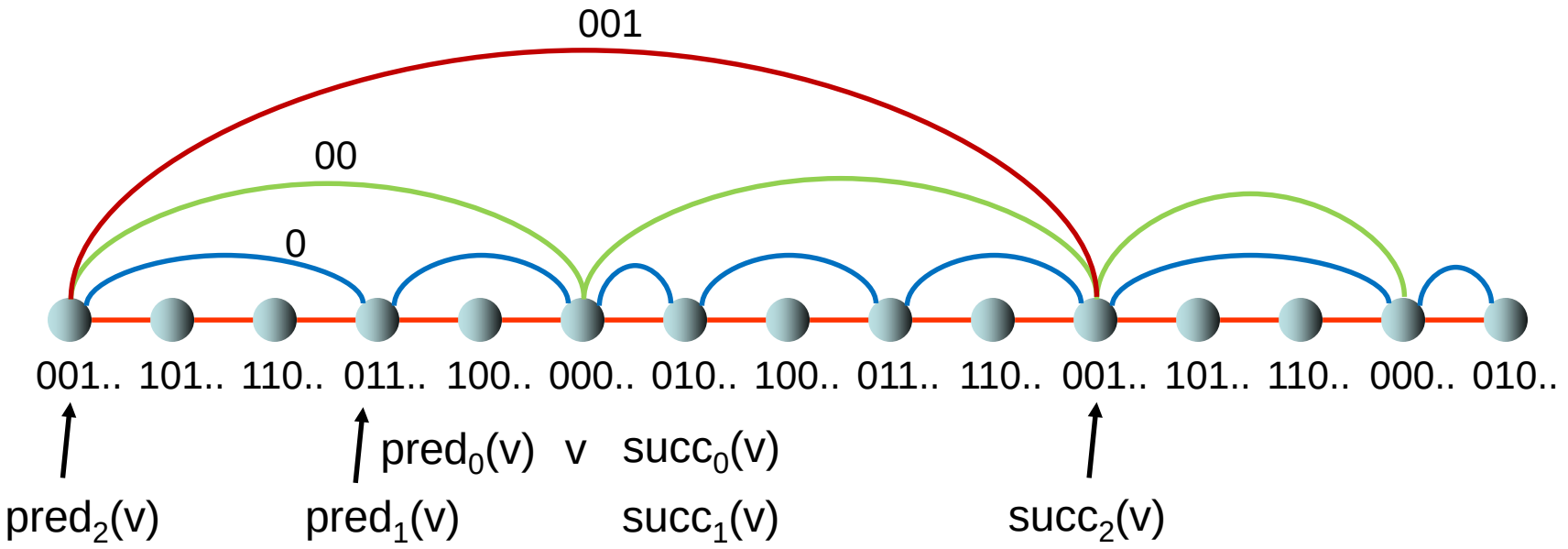
Skip Graph Regel:

Für jeden Knoten v und jedes $i \in \mathbb{N}_0$:

- v hat eine Kante zu $pred_i(v)$ und $succ_i(v)$ (sofern diese existieren)

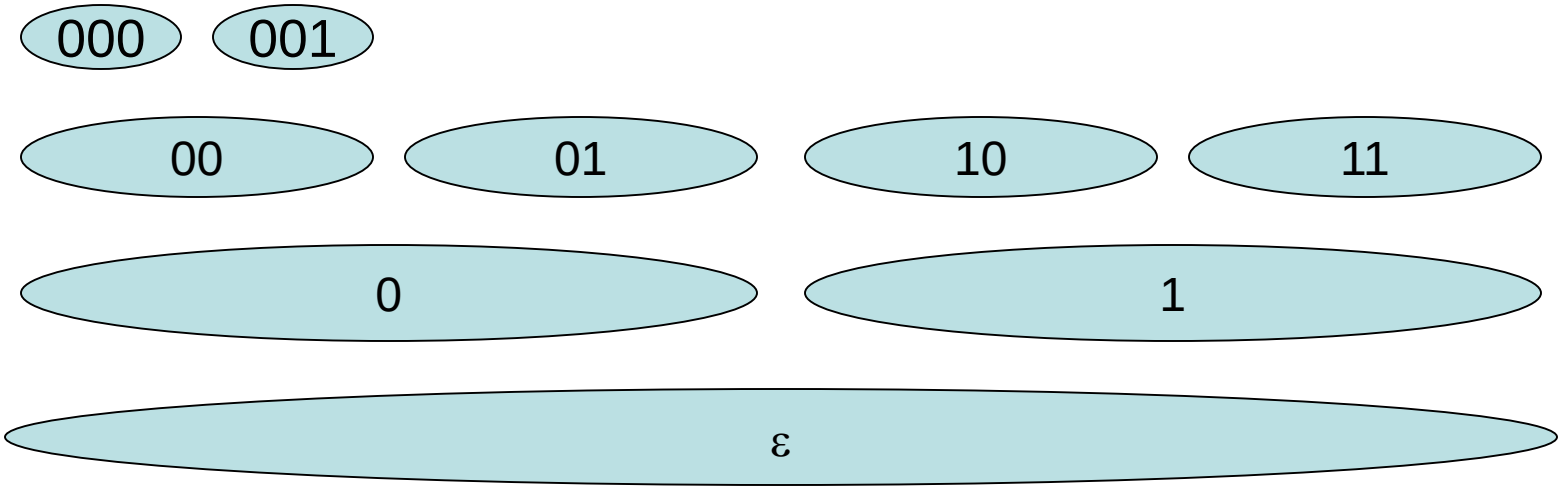
Skip Graphen

Beispiel einiger Teillisten für verschiedene Präfixlängen im Skip Graphen (Knoten seien aufsteigend gemäß ´<´ angeordnet):



Skip Graphen

Hierarchische Sicht: geordnete Listen von Knoten mit demselben Präfix.



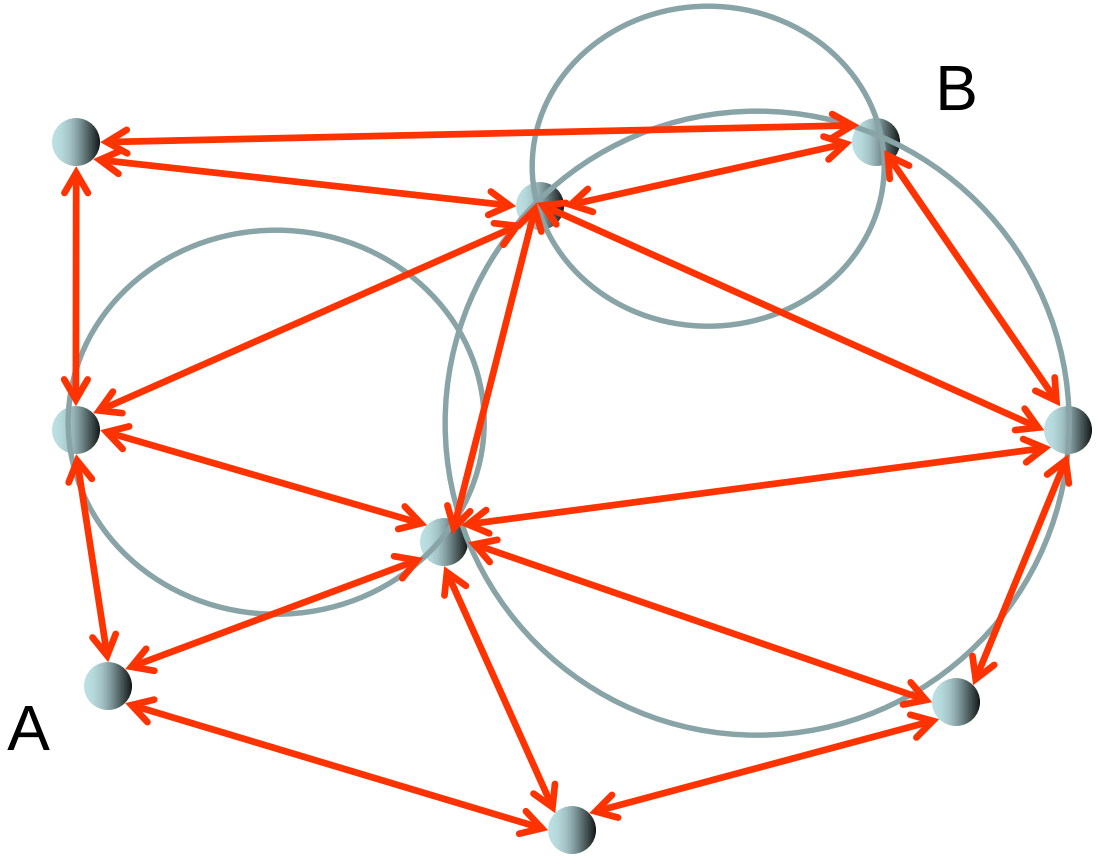
$\Theta(\log n)$ Grad, $\Theta(\log n)$ Durchmesser, $\Theta(1)$ Expansion (mit hoher Wahrscheinlichkeit)

Übersicht

- Grundlagen
- Grundlegende Graphparameter
- Klassische Graphfamilien
- Skip Graphen
- Delaunay Graphen
- Routing

Delaunay Graph

Beispiel:

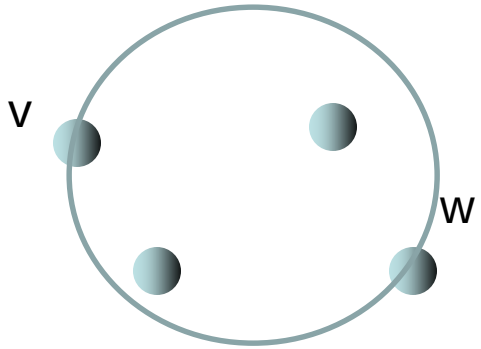


Delaunay Graph

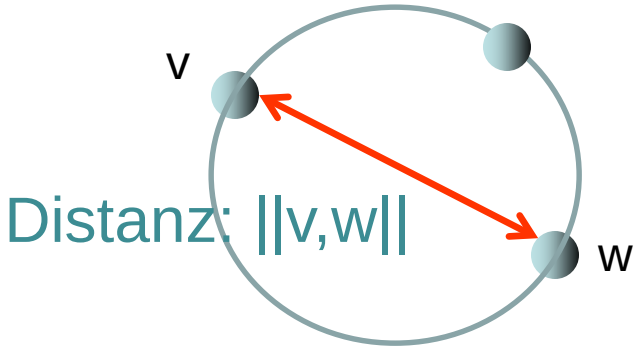
Gegeben: beliebige Punktmenge $V \in \mathbb{R}^2$

Verbindungsregel: verbinde alle Paare $v,w \in V$, für die ein Kreis K durch v und w gelegt werden kann, so dass kein Punkt in V innnerhalb von K ist (aber sie dürfen auf der Grenze von K liegen !)

Beispiel:



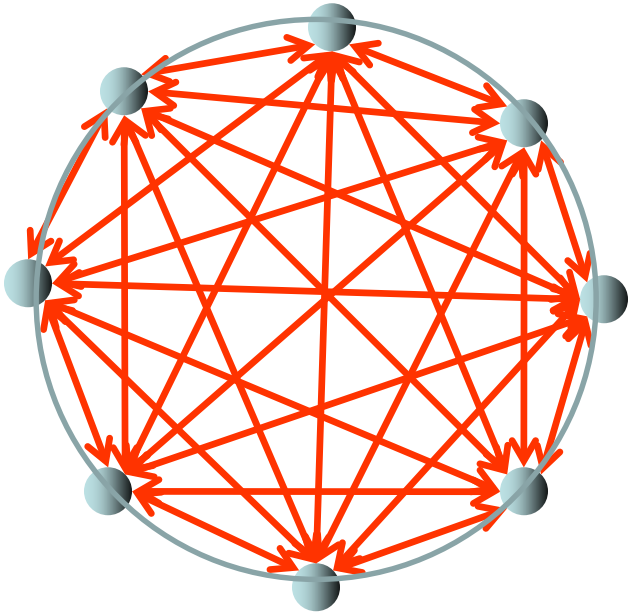
Kante {v,w} nicht in E



Kante {v,w} in E

Delaunay Graph

Spezialfälle:

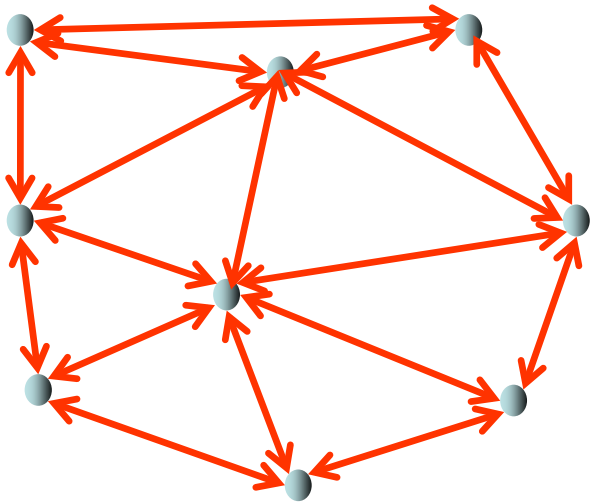


Delaunay Graph

V nichtdegeneriert:

- keine zwei Knoten auf einem Punkt
- keine drei Knoten auf einer Linie
- keine vier Knoten auf einem Kreis

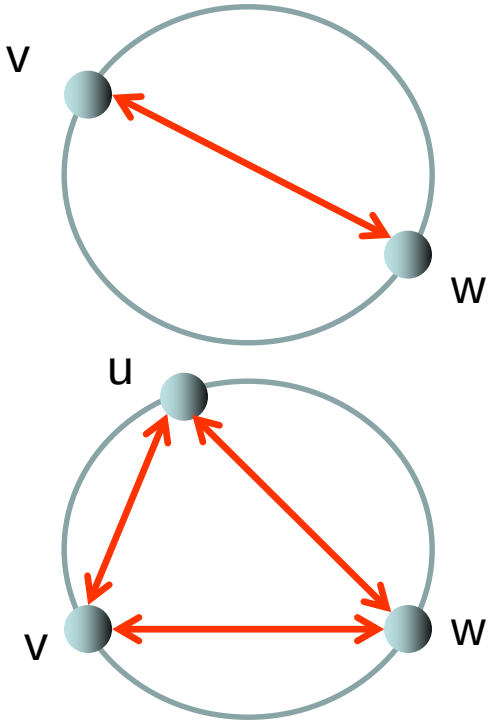
Beobachtung: Für nichtdegenerierte Punktmengen V ist der Delaunay Graph eine planare Triangulierung von V.



Delaunay Graph

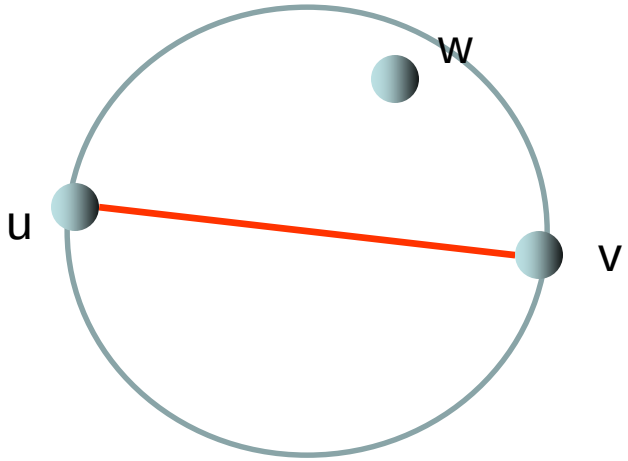
Naive Berechnung des Delaunay Graphen für eine Punktmenge $V \in \mathbb{R}^2$:

1. Für jedes Knotenpaar {u,v} in V: falls der eindeutige Kreis durch u und v mit Durchmesser ||u,v|| keinen Knoten in V enthält, dann füge die Kante {u,v} zu E hinzu.
2. Für jedes Knotentripel {u,v,w} in V, das nicht auf einer Linie liegt: falls der eindeutige Kreis durch u, v und w keinen Knoten in V enthält, dann füge die Kanten {u,v}, {v,w} und {w,u} zu E hinzu.



Delaunay Graph

Test für 1.:

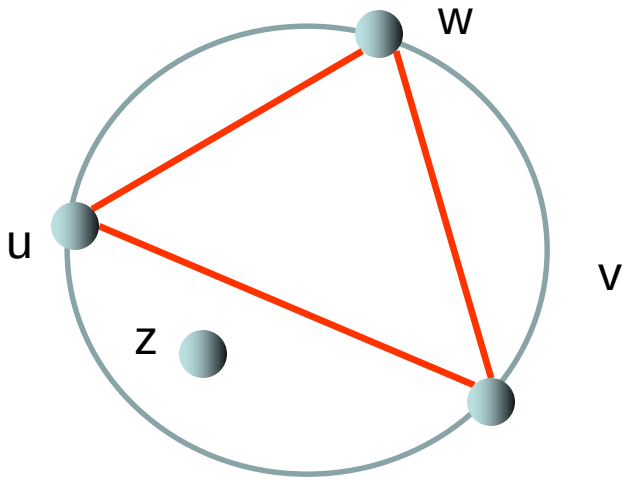


Knoten w ist innerhalb genau dann wenn

$$||u,w||^2+||w,v||^2 < ||u,v||^2.$$

Delaunay Graph

Test für 2.:



Knoten z ist innerhalb genau dann wenn

$$\det \begin{pmatrix} x_u & y_u & x_u^2+y_u^2 & 1 \\ x_v & y_v & x_v^2+y_v^2 & 1 \\ x_w & y_w & x_w^2+y_w^2 & 1 \\ x_z & y_z & x_z^2+y_z^2 & 1 \end{pmatrix} > 0$$

Delaunay Graph

Effiziente sequentielle Berechnung des Delaunay Graphen für eine Punktmenge $V \in \mathbb{R}^2$: divide-and-conquer

- Sortiere die Punkte in V in aufsteigender Ordnung gemäß ihrer x-Koordinate (und falls die x-Koordinaten gleich sind, gemäß ihrer y-Koordinate)
- führe createDT(V) aus

Algorithmus createDT(V):

- if $|V| \leq 3$ then return trivialDT(V)
- L:= left half of nodes in V
- R:= right half of nodes in V
- DT1:=createDT(L); DT2:=createDT(R)
- return stitchDT(DT1,DT2)

Details: „Geoff Leach. Improving worst-case optimal Delaunay triangulation algorithms, 4th Canadian Conf. on Computational Geometry, 1992“.

Delaunay Graph

- $||u,v||$: Euclidische Distanz zwischen u und v

Definition 2.12: Ein Graph $G=(V,E)$ über einer Punktmenge V heißt Euclidischer Spanner falls es eine Konstante c gibt, so dass es für alle v,w einen Pfad $p=(v=u_1,u_2,...,u_k=w)$ von v nach w in G gibt mit

$$||p|| = \Sigma_i ||u_i,u_{i+1}|| \leq c \cdot ||v,w||$$

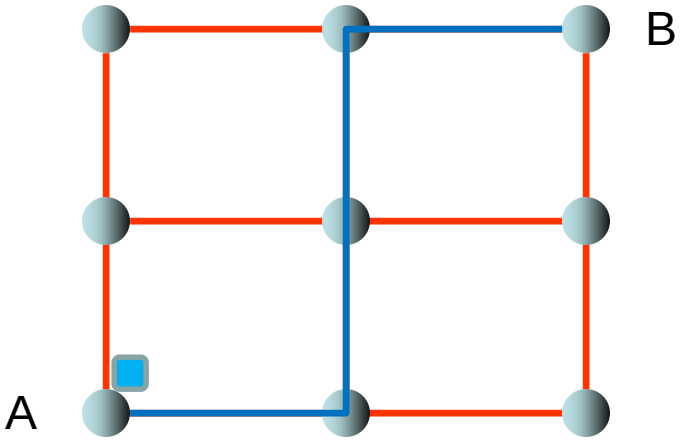
Satz 2.13: Der Delaunay Graph von V ist ein Euclidischer Spanner mit $c \leq 2,42$.

Beweis: Keil, J. M.; Gutwin, C. A., "Classes of graphs which approximate the complete Euclidean graph", *Discrete and Computational Geometry* **7** (1): 13–28, 1992.

Übersicht

- Grundlagen
- Grundlegende Graphparameter
- Klassische Graphfamilien
- Skip Graphen
- Delaunay Graphen
- Routing

Routing



- Routing: finde Weg von A nach B

Routing

Ziel: Gegeben ein Graph $G=(V,E)$ und eine Menge $R=\{(s_1,t_1),\ldots,(s_k,t_k)\}\subseteq V\times V$ von Quell-Ziel-Paaren, finde einen Weg in G für jedes dieser Quell-Ziel-Paare, so dass

- die Weglänge so kurz wie möglich ist (um Nachrichten möglichst schnell auszuliefern) und
- möglichst wenige Wege über denselben Knoten wollen (um die Belastung der Knoten möglichst gering zu halten)

Wie finden wir solche Wege, und wie messen wir die Qualität solcher Wege?

Routing

Gängige Maße für die Qualität von Wegen:

Definition 2.14: Gegeben eine Menge an Wegen $P=\{p_1, p_2, p_3,\ldots\}$ mit Gewichten $w:P\rightarrow\mathbb{R}_+$ in einem Graphen $G=(V,E)$, dann ist

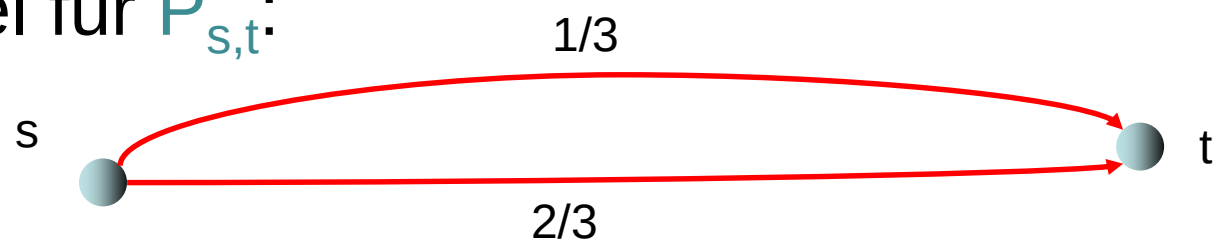
- Congestion von P (max. Knotenbelastung): $C(P) = \max_{v\in V} \Sigma_{p\in P_v} w(p)$, wobei $P_v\subseteq P$ die Menge aller Wege in P über v ist
- Dilation von P (max. Weglänge): $D(P) = \max_{p\in P} |p|$

Routing

Definition 2.15: Routingproblem: Gegeben ein Graph $G=(V,E)$ und eine Menge $R=\{(s_1,t_1),\ldots,(s_k,t_k)\}\subseteq V\times V$ von Quell-Ziel-Paaren, finde ein

- Wegesystem $P = \cup_{s,t} P_{s,t}$ mit nichtleerer Wegemenge $P_{s,t}$ für alle Quell-Ziel-Paare $(s,t)\in R$ und eine
- Gewichtsfunktion: $w:P\rightarrow[0,1]$, so dass für alle $(s,t)\in R$: $\Sigma_{p\in P_{s,t}} w(p) = 1$.

Beispiel für $P_{s,t}$:



Oblivious Routing

Einfachste Strategie zur Lösung von Routingproblemen: oblivious Routing (verwende vorberechnete Wege, um ein beliebiges Routingproblem zu lösen)

Definition 2.16: Sei $G=(V,E)$ ein Netzwerk. Ein oblivious Routing-schema (P,w) ist spezifiziert durch:

- Wegesystem: $P = \cup_{s,t} P_{s,t}$ mit nichtleerer Wegemenge $P_{s,t}$ für alle Quell-Ziel-Paare $(s,t) \in V^2$
- Gewichtsfunktion: $w: P \rightarrow \mathbb{R}_+$, so dass für alle $(s,t) \in V^2$: $\sum_{p \in P_{s,t}} w(p) = 1$.

Lösung eines Routingproblems mittels oblivious Routingschema (P,w):

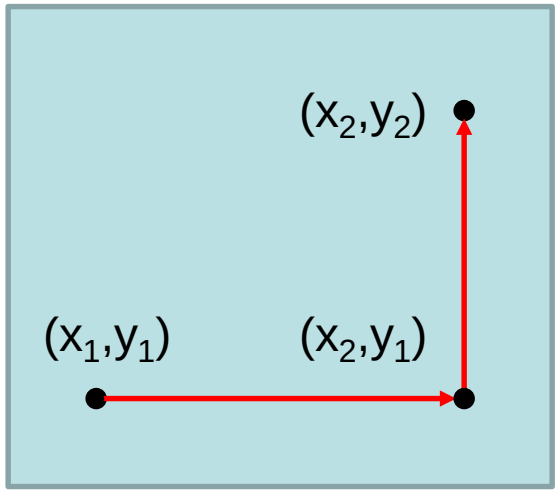
Gegeben eine Menge $R=\{(s_1,t_1),\ldots,(s_k,t_k)\}$, wähle für jedes $(s,t) \in R$ die Wegemenge $P_{s,t}$ aus P mit den vorgegebenen Gewichten in w.

Wie gut ist das?

Oblivious Routing im Gitter

Oblivious Routingschema (P,w) für $k \times k$-Gitter:

- Wegesystem P: Für jedes Paar $(x_1,y_1),(x_2,y_2) \in [k]^2$, route erst von (x_1,y_1) nach (x_2,y_1), dann von (x_2,y_1) nach (x_2,y_2).
- D.h. eindeutiger Weg p ($w(p)=1$) pro Quell-Ziel-Paar.



x-y-Routing Strategie

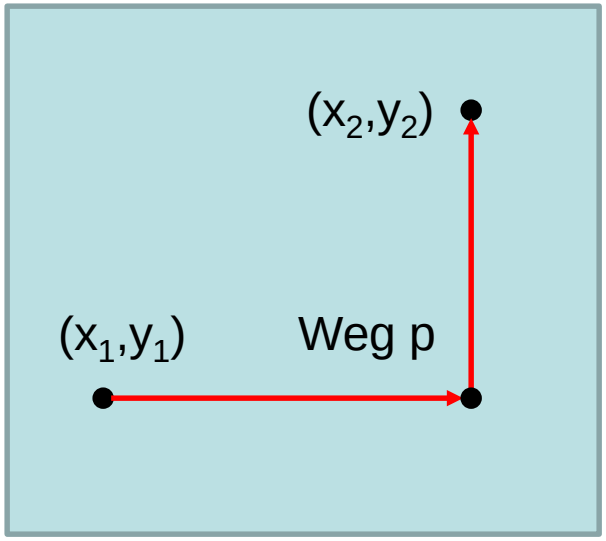
Oblivious Routing im Gitter

„Benchmark“ für Routingstrategie: kann die Strategie beliebige Permutationen $\pi:V\rightarrow V$ mit geringer Dilation und Congestion routen? (D.h. das Routingproblem R_π zu π ist definiert als $R_\pi=\{ (v,\pi(v)) \mid v\in V \}$).

Satz 2.17: Die x-y-Routingstrategie kann jede Permutation im $k\times k$-Gitter mit Congestion höchstens $4d$ und Dilation höchstens d routen, wobei d die maximale Distanz eines Quell-Ziel-Paares ist.

Oblivious Routing im Gitter

Beweis:

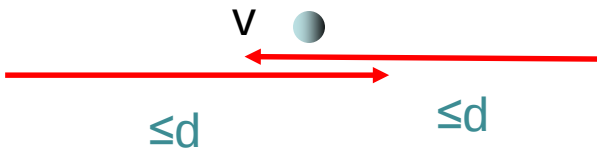


Dilation:

- p hat Länge $d((x_1,y_1),(x_2,y_2))$
- also ist max. Weglänge d

Congestion:

- #Wege über v in x-Richtung:



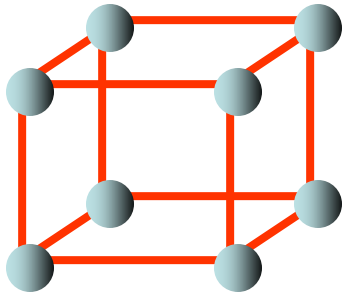
- maximal 2d Quellen haben Wege, die in x-Richtung über v laufen, also ist Congestion in x-Richtung $\leq 2d$
- dasselbe gilt auch für Ziele in y-Richtung

Oblivious Routing im Hypercube

Bitanpassungsstrategie:
Weg von $(x_1,\ldots,x_d)$ nach $(y_1,\ldots,y_d)$ führt über $(y_1,x_2,\ldots,x_d)$, $(y_1,y_2,x_3,\ldots,x_d)$, $(y_1,y_2,y_3,x_4,\ldots,x_d)$, …, $(y_1,y_2,y_3,\ldots,y_{d-1},x_d)$, $(y_1,\ldots,y_d)$

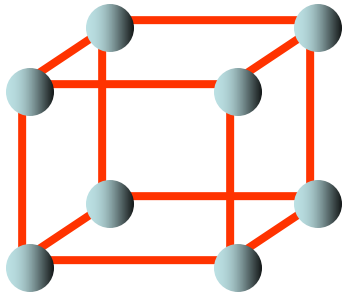
- Dilation: optimal, da Weglänge gleich Distanz
- Congestion: es gibt Permutationen, die sehr hohe Congestion haben!

Beispiel: sei $\pi(x_1,\ldots,x_d) = (x_d,\ldots,x_1)$ für alle $(x_1,\ldots,x_d)$



Oblivious Routing im Hypercube

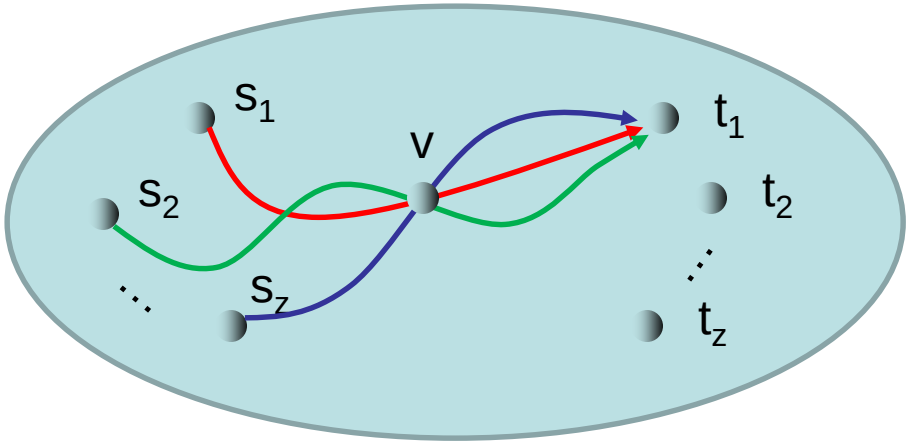
Beispiel: sei $\pi(x_1,\ldots,x_d) = (x_d,\ldots,x_1)$ für alle $(x_1,\ldots,x_d)$



- Betrachte Knotenmenge $M=\{ (x_1,\ldots,x_{d/2},0,\ldots,0) \mid x_i \in \{0,1\}$ für alle $i \in \{1,\ldots,d/2\} \}$
- Nach $d/2$ Routingschritten gemäß π sind alle Pakete mit Quelle in M im Knoten $(0,\ldots,0)$
- Da $|M|=2^{d/2} = \sqrt{2^d} = \sqrt{n}$ ist, kann Congestion wesentlich größer als Dilation (maximal $\log n$) sein. Bestmöglich wäre Congestion $O(\log n)$!

Borodin-Hopcroft Schranke

Satz 2.18: Für jeden ungerichteten Graphen G der Größe n mit Grad δ und jedes oblivious Routing-schema mit nur einem Pfad pro Quell-Ziel-Paar gibt es eine Permutation π, für die ein Knoten von mindestens $\sqrt{n/\delta}$ Pfaden durchlaufen wird.



Es gibt v und $t_1,\ldots,t_z$,so dass jedes t_i mind. z Quellen hat,deren Wege durch v führen, $z = \sqrt{n/\delta}$.

Valiants Trick

Frage: gibt es oblivious Routingschemen, die für alle Permutationen eine niedrige Congestion garantieren?

Antwort: ja, aber wir brauchen mehrere Pfade pro Quell-Ziel-Paar.

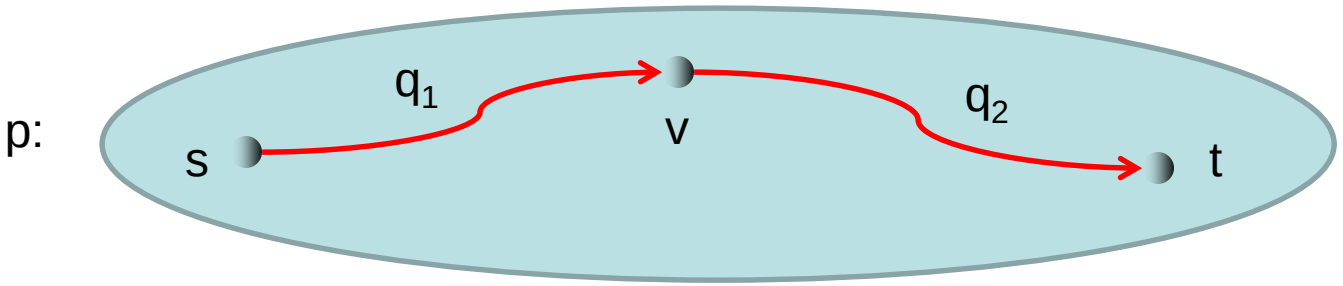
Valiants Trick:

- Konstruiere zunächst ein oblivious Routingschema (P^*,w^*) mit $P^* = \cup_{s,t} P^*_{s,t}$ mit minimaler Congestion C_{OPT} (kann in polynomieller Zeit berechnet werden).
- Konstruiere aus (P^*,w^*) mittels Zwischenzielen ein oblivious Routingschema (P,w) mit $P = \cup_{s,t} P_{s,t}$, so dass die Congestion in etwa gleich bleibt zu P^*.

Valiants Trick

Konstruktion von (P,w) für beliebigen Graphen G=(V,E):

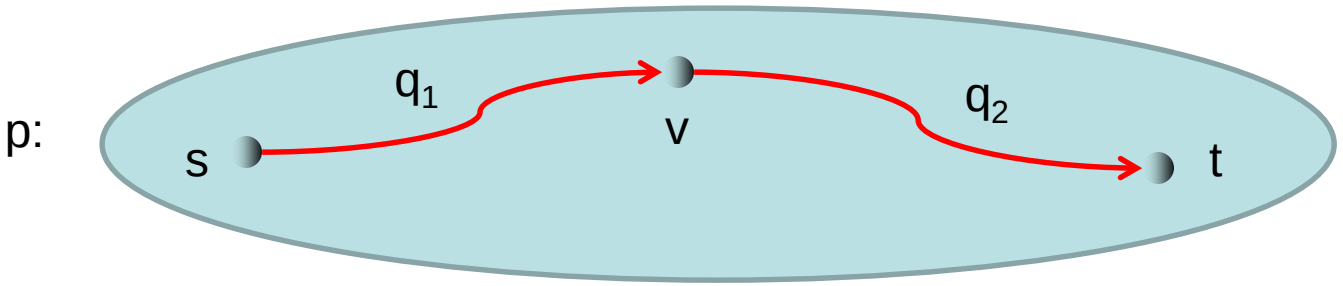
- (P*,w*): Routingschema mit min. Congestion C_{OPT}
- $P_{s,t} = \{ p=q_1 \circ q_2 \mid q_1 \in P^*_{s,v}$ und $q_2 \in P^*_{v,t}$ für ein $v \in V \}$
  ($\circ$: Konkatenation)
- Für alle $p=q_1 \circ q_2$ in $P_{s,t}$ setzen wir $w(p)=w^*(q_1) \cdot w^*(q_2)/n$
- Damit (P,w) ist ein gültiges oblivious Routingschema ist, müssen wir zeigen: $\sum_{p \in P_{s,t}} w(p) = 1$.



Valiants Trick

- $P_{s,t} = \{ p=q_1 \circ q_2 \mid q_1 \in P^*_{s,v}$ und $q_2 \in P^*_{v,t}$ für ein $v \in V \}$
- Für alle $p=q_1 \circ q_2$ in $P_{s,t}$ setzen wir $w(p)=w^*(q_1) \cdot w^*(q_2)/n$
- Zu zeigen: $\sum_{p \in P_{s,t}} w(p) = 1$.

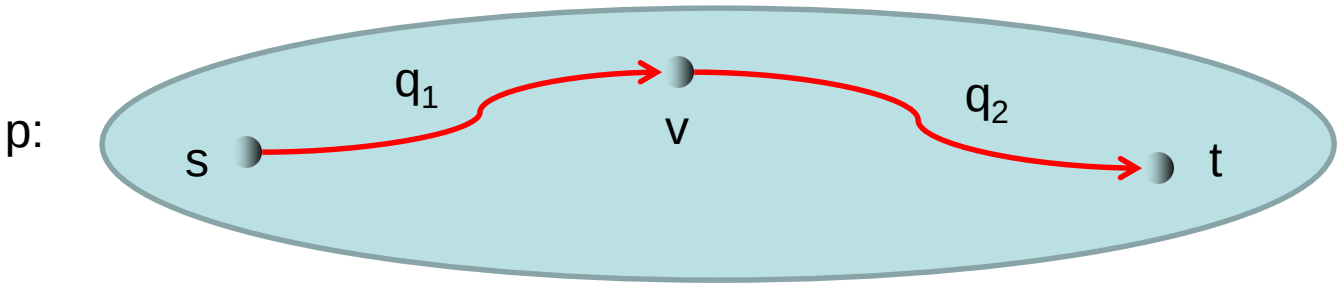
$$\begin{aligned}
\sum_{p \in P_{s,t}} w(p) &= \sum_{v \in V} \sum_{q_1 \in P^*_{s,v}} \sum_{q_2 \in P^*_{v,t}} w^*(q_1) \cdot w^*(q_2)/n \\
&= \sum_{v \in V} (1/n) \sum_{q_1 \in P^*_{s,v}} w^*(q_1) \sum_{q_2 \in P^*_{v,t}} w^*(q_2) \\
&= \sum_{v \in V} (1/n) \sum_{q_1 \in P^*_{s,v}} w^*(q_1) \\
&= \sum_{v \in V} (1/n) = 1
\end{aligned}$$



Valiants Trick

Dilation und Congestion von (P,w):

- (P^*,w^*): Routingschema mit min. Congestion C_{OPT}
- $P_{s,t} = \{ p=q_1 \circ q_2 \mid q_1 \in P^*_{s,v}$ und $q_2 \in P^*_{v,t}$ für ein $v \in V \}$
- Für alle $p=q_1 \circ q_2$ in $P_{s,t}$ setzen wir $w(p)=w^*(q_1) \cdot w^*(q_2)/n$
- Dilation von P: offensichtlich $\leq 2 \cdot$(Dilation von P^*)
- Congestion von P: wir zeigen, dass $C(P) \leq 2C_{OPT}$



Valiants Trick

$C(P) \leq 2C_{OPT}$:
Sei $C_u(P^*)=\sum_{p\in P^*:\, u\in p} w(p) \leq C_{OPT}$ die Congestion von u für P^*.
Für $C_u(P)$ gilt:

$$
\begin{aligned}
C_u(P) &= \sum_{p\in P:\, u\in p} w(p) \\
&= \sum_{s,t\in V} \sum_{p\in P_{s,t}:\, u\in p} w(p) \\
&= \sum_{s,t\in V} \sum_{p=q_1\circ q_2\in P_{s,t}:\, u\in q_1 \text{ oder } u\in q_2} w(p) \\
&\leq \sum_{s,t\in V} \left(\sum_{p=q_1\circ q_2\in P_{s,t}:\, u\in q_1} w(p) + \sum_{p=q_1\circ q_2\in P_{s,t}:\, u\in q_2} w(p)\right)
\end{aligned}
$$

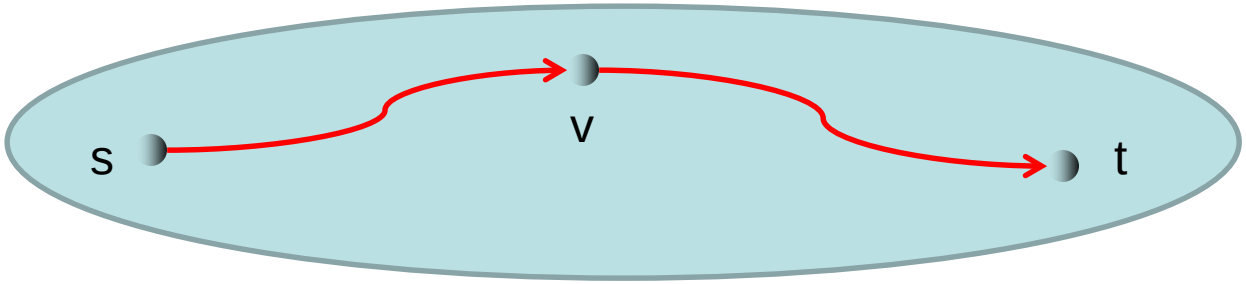
$$
\begin{aligned}
\sum_{s,t\in V} \sum_{p=q_1\circ q_2\in P_{s,t}:\, u\in q_1} w(p) &= \sum_{s,t\in V} \sum_{v\in V} \sum_{q_1\in P^*_{s,v}:\, u\in q_1} \sum_{q_2\in P^*_{v,t}} w^*(q_1)\cdot w^*(q_2)/n \\
&= \sum_{t\in V} (1/n) \sum_{s,v\in V} \sum_{q_1\in P^*_{s,v}:\, u\in q_1} w^*(q_1) \\
&= \sum_{t\in V} (1/n) \sum_{q\in P^*:\, u\in q} w^*(q) \\
&= \sum_{t\in V} (1/n) \cdot C_u(P^*) \\
&= C_u(P^*)
\end{aligned}
$$

Analog: $\sum_{s,t\in V} \sum_{p=q_1\circ q_2\in P_{s,t}:\, u\in q_2} w(p) = C_u(P^*)$. Also ist $C_u(P) \leq 2C_{OPT}$.

Valiants Trick

Satz 2.19: Mit dem aus Valiants Trick resultierenden oblivious Routingschema (P,w) kann in jedem Graphen G=(V,E) jede Permutation mit Congestion höchstens $2C_{OPT}/n$ geroutet werden.

Beweis: Übung.

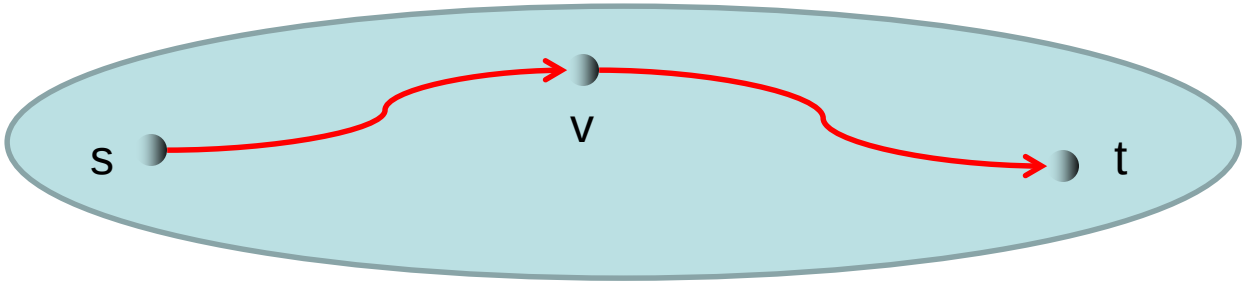


Valiants Trick

Satz 2.20: In jedem Graphen G=(V,E) benötigt eine Permutation im Durchschnitt mindestens C_{OPT}/n Congestion, um geroutet zu werden.

Beweis: Übung.

→ Mit (P,w): Congestion max. doppelt so groß wie Durchschnitt

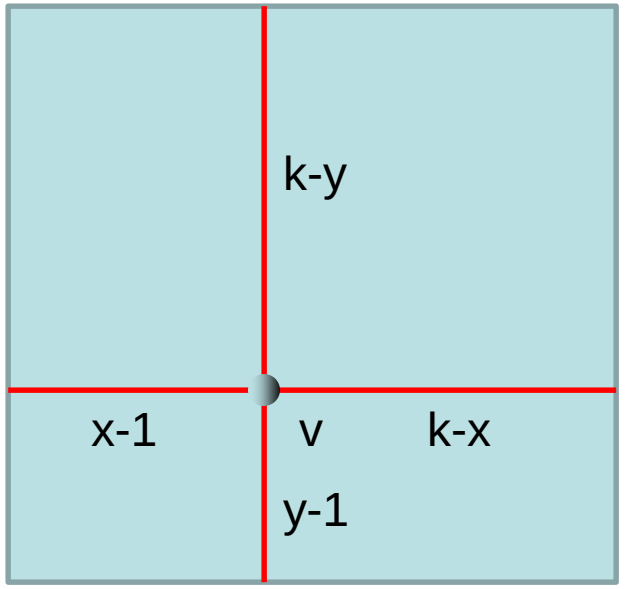


Valiants Trick

Obere Schranke für C_{OPT} für k×k-Gitter:

Für das x-y-Routing gilt für jeden Knoten v an Position $(x,y) \in \{1,\ldots,k\}^2$:

- Es gibt maximal
  $(x-1)(k-x+1)k + k^2 + (k-x)x \cdot k$
  Quell-Ziel-Paare, deren Pfade v in x-Richtung durchlaufen
- Entsprechend gibt es maximal
  $(y-1)(k-y+1)k + k^2 + (k-y)y \cdot k$
  Quell-Ziel-Paare, deren Pfade v in y-Richtung durchlaufen
- Also ist $C_{OPT}=O(k^3)$ und damit $C_{OPT}/n = O(k)$.

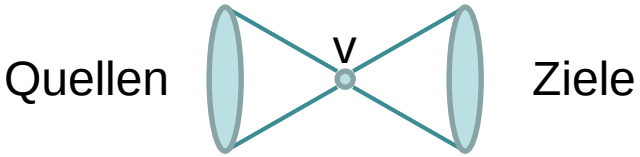


Valiants Trick

Obere Schranke für C_{OPT} für d-dim. Hypercube:

Für die Bitanpassungsstrategie gilt für jeden Knoten v:

- Für jedes $i \in \{0,\ldots,d\}$ hat v 2^i Quellen, die v in i Schritten erreichen können, und die von v aus noch 2^{d-i} Ziele erreichen können.



- Es können also maximal

$$\Sigma_{i=0}^{d} 2^i \cdot 2^{d-i} = (d+1)2^d$$

  Pfade über v laufen.
- Also ist $C_{OPT} \leq (d+1)2^d$ und damit $C_{OPT}/n = O(d)$.

Valiants Trick

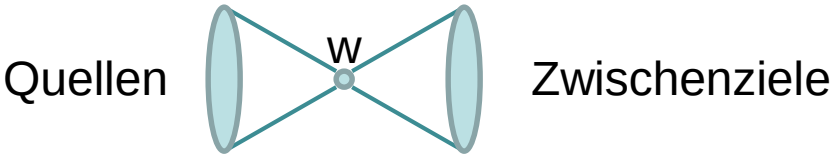
Valiants Trick für d-dimensionalen Hypercube:

- Bitanpassungsstrategie ergibt sogar optimales oblivious Routingschema (P^*,w^*).
- D.h. $P^*_{s,t}$ besteht lediglich aus einem Weg für jedes Knotenpaar (s,t) im Hypercube, d.h. $P^*_{s,t} = \{p_{s,t}\}$ für den Bitanpassungsweg $p_{s,t}$ und $w^*(p_{s,t})=1$.
- Also ist $P_{s,t} = \{ p_{s,v,t} = p_{s,v} \circ p_{v,t} \mid v \in V \}$ mit $w(p_{s,v,t})= 1/n$.

Valiants Trick

Ist das oblivious Routingschema (P,w) wirklich gut?

- Betrachte eine beliebige Permutation π (d.h. wir haben Quell-Ziel-Paare $(v, \pi(v))$ für alle $v \in V$).
- Zunächst streuen sich die Wege von s über alle n Zwischenziele v
- Dilation: maximal d bis v, also insgesamt maximal $2d$.
- Congestion: betrachte einen festen Knoten w und für ein festes $i \in \{0,\ldots,d\}$ alle Quell-Zwischenziel-Paare, deren Wege w in i Schritten erreichen.
- Mit der Bitanpassungsstrategie gibt es 2^i mögliche Quellen für w.
- Weiterhin können 2^{d-i} Zwischenziele von w aus erreicht werden.



- D.h. die Anzahl der Quell-Ziel-Paare, dessen Pfade w durchlaufen, ist max.
$$\Sigma_{i=0}^{d} 2^i \cdot 2^{d-i} = (d+1)\cdot 2^d = (d+1)\cdot n.$$
- Da jeder dieser Pfade ein Gewicht von $1/n$ hat, ist die Congestion maximal
$$(d+1)\cdot n \cdot 1/n = d+1.$$
- Da diese Congestion zweimal entsteht (von der Quelle zum Zwischenziel, und vom Zwischenziel zum Ziel), ist die Gesamtcongestion $\leq 2(d+1)$.

Routing zu zufälligen Zielen

- Bisher haben wir nur Permutationsroutingprobleme betrachtet.
- Welche Congestion können wir für den Fall erreichen, dass jeder Knoten eine Nachricht zu einem zufälligen Knoten schicken will?
- Dafür verallgemeinern wir die Definition eines Routingproblems.

Definition 2.21 (Allgemeines Routingproblem): Gegeben ein Graph $G=(V,E)$ und eine Menge $R=\{((s_1,t_1),c_1),\ldots,((s_k,t_k),c_k)\} \subseteq V\times V\times \mathbb{R}_+$ von Quell-Ziel-Paaren mit Gewichten c_i, finde ein

- Wegesystem $P = \cup_{s,t} P_{s,t}$ mit nichtleerer Wegemenge $P_{s,t}$ für alle $((s,t),c)\in R$ und eine
- Gewichtsfunktion: $w:P\rightarrow\mathbb{R}_+$, so dass für alle $((s,t),c)\in R$: $\sum_{p\in P_{s,t}} w(p) = c$ (vorher: 1!).

Routing zu zufälligen Zielen

Routingproblem zu zufälligen Zielen in einem Graphen $G=(V,E)$ mit n Knoten:

$$R_{rand}=\{ ((s,t),1/n) \mid s,t \in V \}$$

↑

Wahrscheinlichkeit, dass s t auswählt

Satz 2.22: Unter Verwendung des optimalen oblivious Routingschemas (P^*,w^*) in G kann R_{rand} mit Congestion C_{OPT}/n geroutet werden. (Hier benötigen wir also nicht Valiants Trick.)

D.h. die erwartete Congestion (Gewichte sind Wahrscheinlichkeiten!) an jedem Knoten in G ist $\leq C_{OPT}/n$.

Routing zu zufälligen Zielen

Satz 2.22: Unter Verwendung des optimalen oblivious Routingschemas (P^*,w^*) in G kann R_{rand} mit Congestion C_{OPT}/n geroutet werden.

Beweis:

- $C_u(R_{rand})$: Congestion bei u für R_{rand}.
- Da für alle Quell-Ziel-Paare ein Gewicht von $1/n$ geroutet wird, gilt:

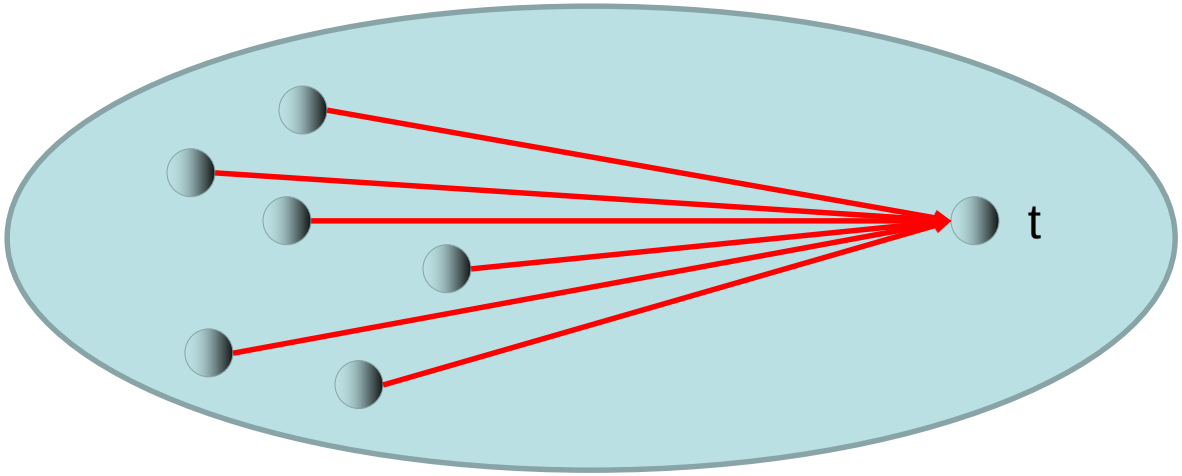
$$C_u(R_{rand}) = \sum_{p \in P^*:\, u \in p} w^*(p)/n$$
$$= (1/n) \cdot C_u(P^*) \leq C_{OPT}/n$$

Routing mit Combining

- Angenommen, wir haben ein Routingproblem $R=\{(s_1,t_1),\ldots,(s_k,t_k)\}\subseteq V\times V$, in dem zwar jeder Knoten höchstens einmal als Quelle auftaucht, aber ein Knoten beliebig oft als Ziel auftauchen kann. (Das kann z.B. passieren, wenn ein Knoten eine sehr populäre Information enthält.)
- Können wir die Congestion auch in diesem Fall klein halten, und wenn ja, unter welchen Annahmen?

Routing mit Combining

Falls Wege zum selben Ziel nicht „kombiniert“ werden können, ist eine hohe Congestion unvermeidbar!



Was heißt, dass Wege kombiniert werden können?

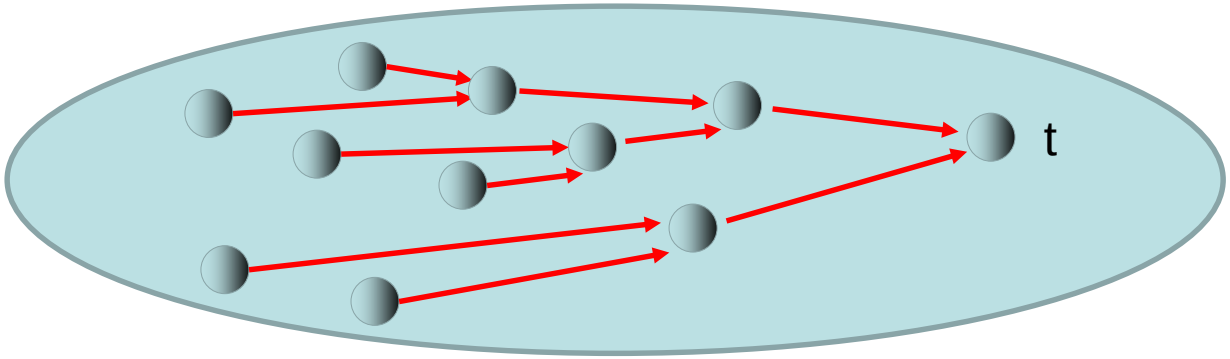
Routing mit Combining

Congestion für u bezüglich Ziel t für ein (einfaches) Routing-problem R, das oblivious Routingschema (P,w) benutzt:

- Bisher: $\sum_{s:\,(s,t)\in R} \sum_{p\in P_{s,t}:\,u\in p} w(p)$
- Wege zum selben Ziel kombinierbar:
  $\min\{1, \sum_{s:\,(s,t)\in R} \sum_{p\in P_{s,t}:\,u\in p} w(p)\}$

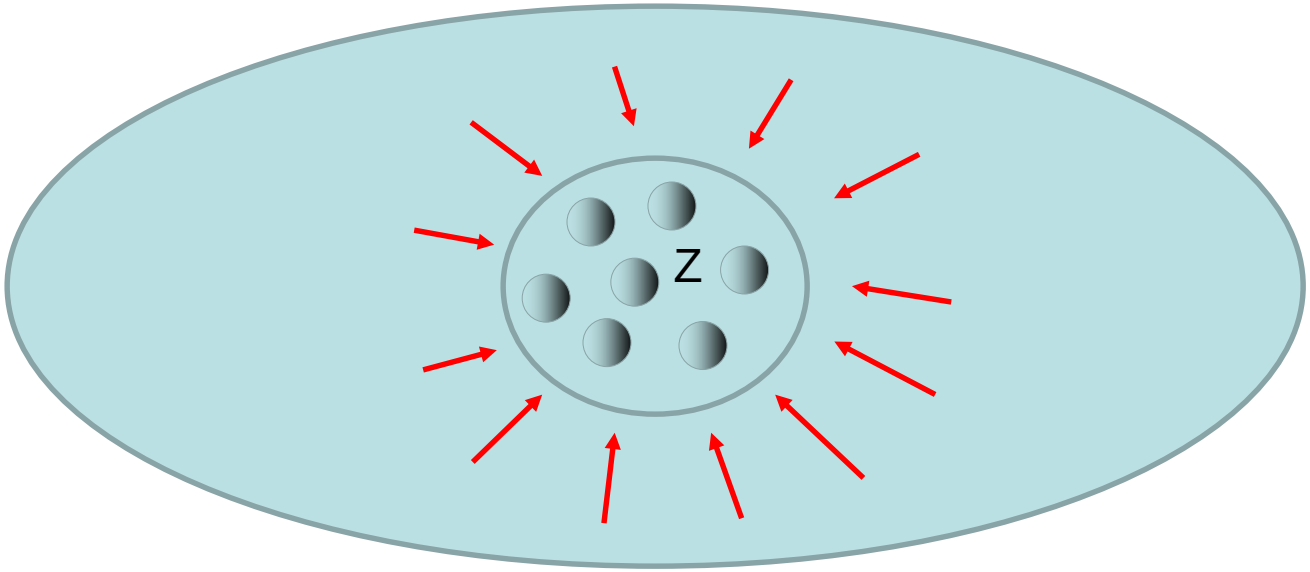
Motivation: es reicht, für ein Ziel t maximal eine Anfrage über Knoten u zu verschicken.

Beispiel für Routing mit Combining (ein Weg in $P_{s,t}$ pro (s,t)):



Routing mit Combining

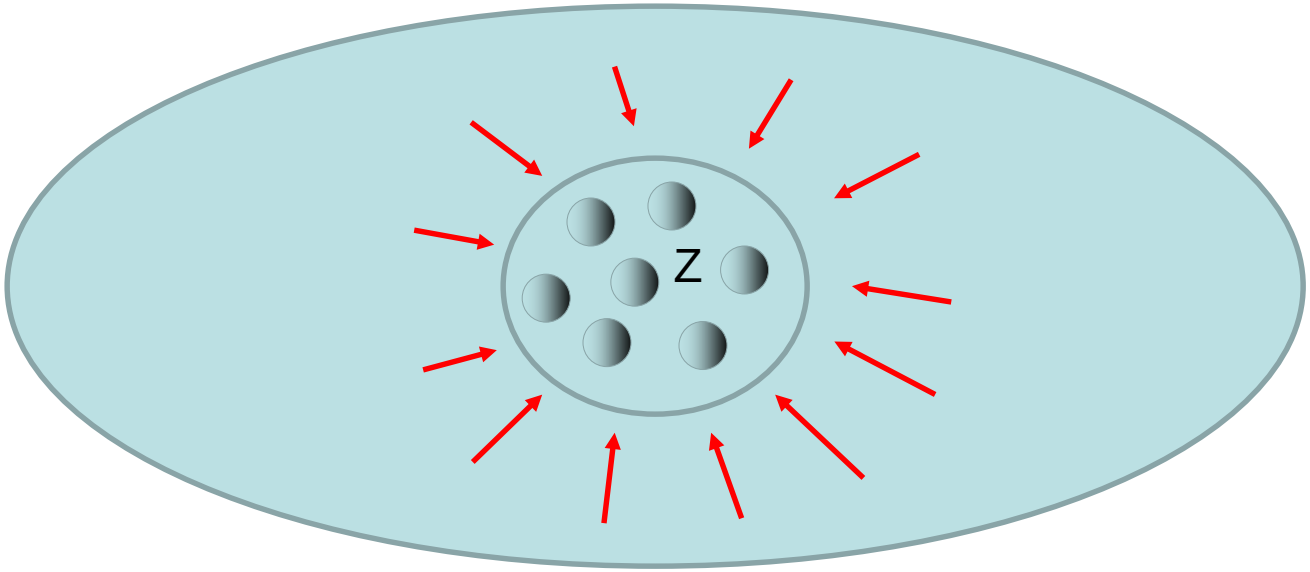
Die Kombinierbarkeit alleine reicht noch nicht für eine kleine Congestion!



Jedes Ziel in Z hat viele Quellen!

Routing mit Combining

Die Kombinierbarkeit alleine reicht noch nicht für eine kleine Congestion!



→ Anfragen müssen im Vorfeld kombiniert werden!

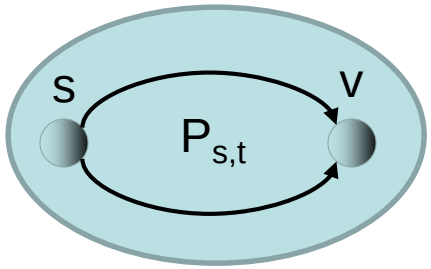
Routing mit Combining

Routing mit Combining Strategie:

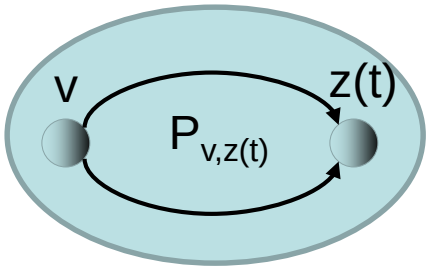
Sei R ein einfaches Routingproblem und (P,w) ein (optimales) oblivious Routingschema. Für jedes $(s,t) \in R$:

1. Wähle $P_{s,v}$ aus für ein zufälliges $v \in V$
2. Sei $z(t)$ ein zufälliges Zwischenziel für Ziel t (welches für alle s' mit $(s',t) \in R$ das gleiche ist). Die Zwischenziele sorgen für eine gute Streuung von Zielen t' mit vielen Paaren $(s',t') \in R$!
   Wähle ausgehend von v $P_{v,z(t)}$ aus (aufgrund der Kombinierung mit anderen $(s',t) \in R$ ist danach nur ein $((z(t),t),1)$ übrig, das zu routen ist!)
3. Wähle schließlich $P_{z(t),t}$ aus, um nach t zu gelangen

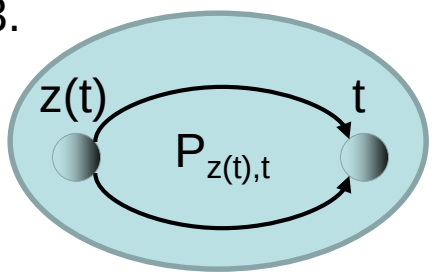
1.



2.



3.

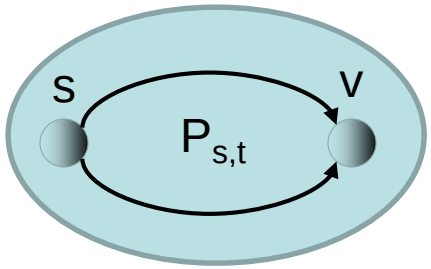


Routing mit Combining

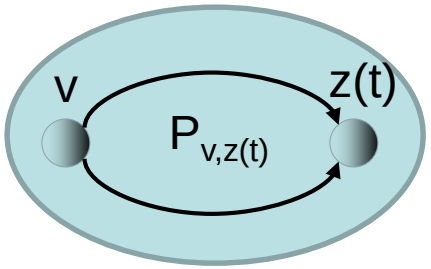
Routingprobleme für ein $(s,t) \in R$:

1. $\{ ((s,v(s,t)),1/n) \mid v(s,t) \in V \}$
2. $\{ ((v(s,t),z(t)),1/n^2) \mid z(t) \in V \}$
3. $\{ ((z(t),t),1/n) \mid z(t) \in V \}$ (nur Gewicht 1 für alle s' mit $(s',t) \in R$)

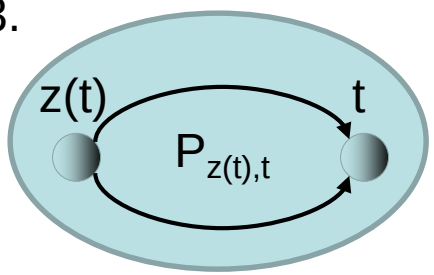
1.



2.



3.



Routing mit Combining

Routingprobleme für ein $(s,t) \in R$:

1. $\{ ((s,v(s,t)),1/n) \mid v(s,t) \in V \}$
2. $\{ ((v(s,t),z(t)),1/n^2) \mid z(t) \in V \}$
3. $\{ ((z(t),t),1/n) \mid z(t) \in V \}$ (nur Gewicht 1 für alle $s´$ mit $(s´,t) \in R$)

Routingprobleme 1. und 3. entsprechen (im worst case) dem Routingproblem R_{rand}! Mit oblivious Routing-schema (P^*,w^*) ist Congestion also C_{OPT}/n.

Es bleibt also, Routingproblem 2. zu betrachten.

Routing mit Combining

2. Routingproblem:

$R' = \{ ((v(s,t),z(t)),1/n^2) \mid (s,t) \in R,\ v(s,t) \in V,\ z(t) \in V \}$

Congestion bei Knoten u:

$$\sum_{(s,t)\in R} \sum_{v\in V} \sum_{z\in V} \sum_{p\in P^*_{v,z}:\, u\in p} w^*(p)/n^2$$

$$= \sum_{(s,t)\in R} C_u(P^*)/n^2$$

$$\leq C_u(P^*)/n \leq C_{OPT}/n \quad \text{(da } |R| \leq n\text{)}$$

Zusammen über alle Routingprobleme ist die maximale Congestion also $\leq 3C_{OPT}/n$.

Routing mit Combining

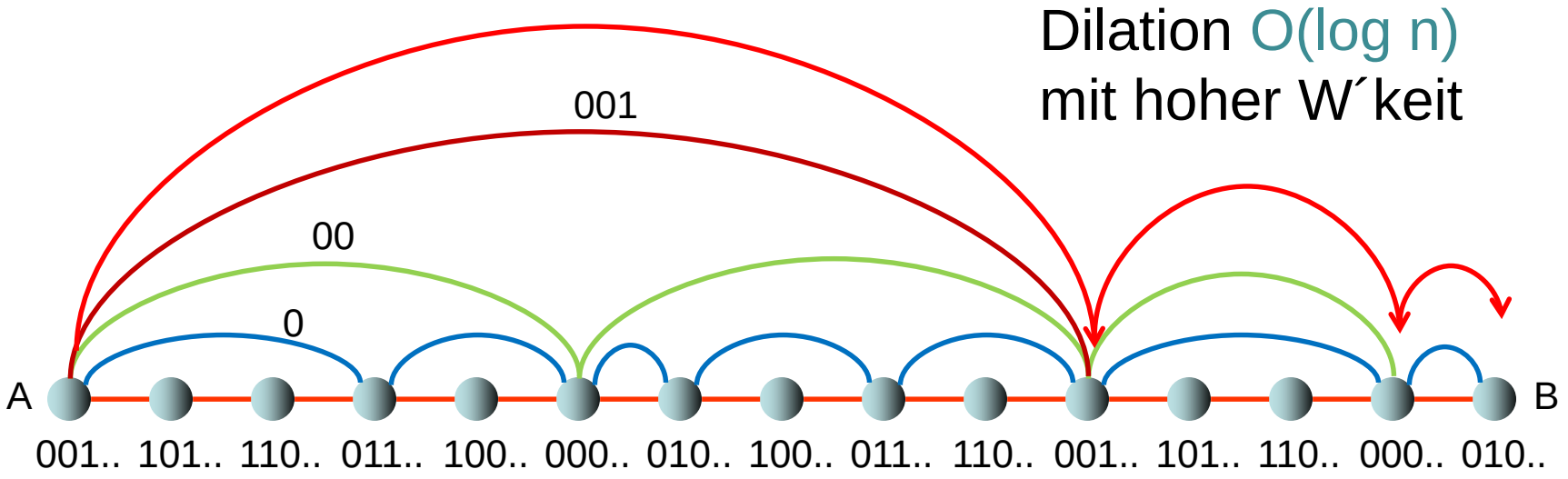
Satz 2.23: Ein beliebiges einfaches Routingproblem R, in dem jeder Knoten nur einmal als Quelle vorkommt, kann mit der Routing mit Combining Strategie mittels (P*,w*) mit Congestion höchstens $3C_{OPT}/n$ geroutet werden.

Bemerkung:

- Wir nehmen hier an, dass sich Informationen beliebig über Wege aufspalten können. Interpretieren wir die Gewichte als Wahrscheinlichkeiten, dann handelt es sich bei $3C_{OPT}/n$ um die maximale erwartete Congestion in einem Knoten. Die Abweichung von $3C_{OPT}/n$ ist gering mit hoher Wahrschein-lichkeit, da jedes Ziel nur einen Wert von maximal 1 beitragen kann.

Oblivious Routing im Skip Graph

Geeignete Routingstrategie (A kennt ID von B): wähle möglichst lange Kante in der Richtung des Ziels B (Greedy Routing)

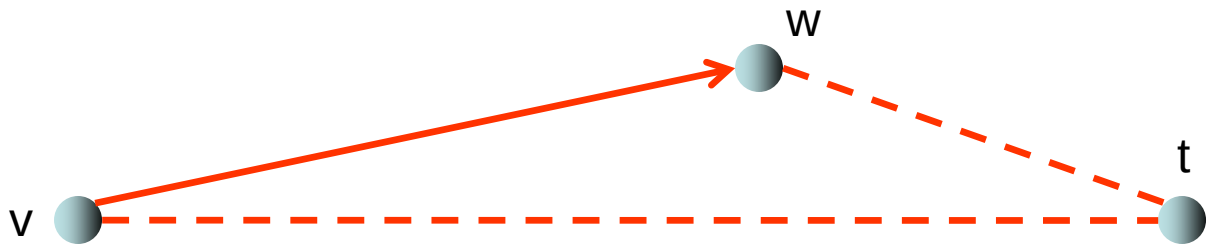


Delaunay Graph

Problem: wie finden wir einen Pfad von s nach t in beliebigen Delaunay Graphen?

Greedy Strategien:

- Distanz Routing: Knoten v leitet Paket P an den Knoten $w \in N(v)$ mit minimaler Euklidischer Distanz $||w,t||$ weiter

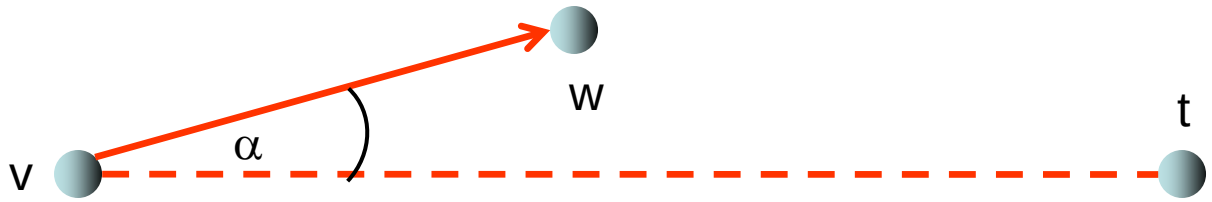


Delaunay Graph

Problem: wie finden wir einen Pfad von s nach t in beliebigen Delaunay Graphen?

Greedy Strategien:

- Kompass Routing: Knoten v leitet Paket P an den Knoten $w \in N(v)$ mit minimalem Winkel $\alpha = \measuredangle tvw$ weiter

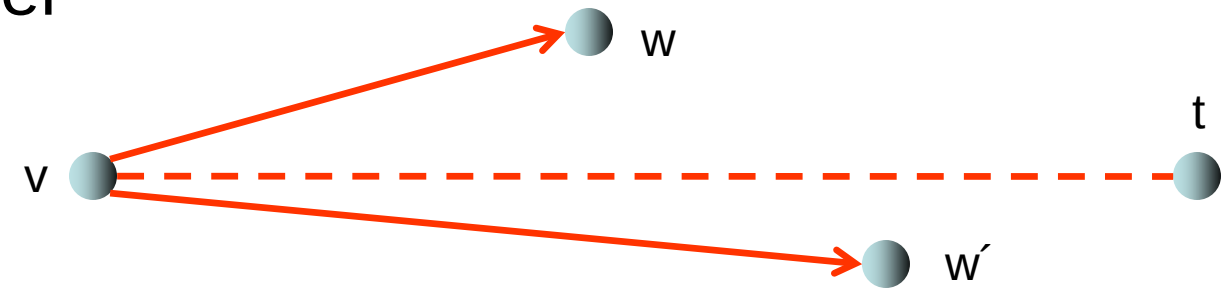


Delaunay Graph

Problem: wie finden wir einen Pfad von s nach t in beliebigen Delaunay Graphen?

Greedy Strategien:

- Distanz-Kompass Routing: Knoten v bestimmt zunächst die Knoten $w,w' \in N(v)$ mit kleinstem Winkel im Uhrzeigersinn und gegen den Uhrzeigersinn zu t leitet das Paket P an den Knoten $w'' \in \{w,w'\}$ mit minimaler Distanz $||w'',t||$ weiter



Delaunay Graph

Definition 2.24: Eine Routingstrategie ist c-kompetitiv für einen Graphen G falls sie für jedes Paar v,w einen Pfad $p=(v=u_1,u_2,...,u_k=w)$ von v nach w in G auswählt mit

$$||p|| = \Sigma_i ||u_i,u_{i+1}|| \leq c \cdot d_G(v,w)$$

wobei $d_G(v,w)$ die Eukl. Länge des kürzesten Weges von v nach w in G ist.

Satz 2.25:

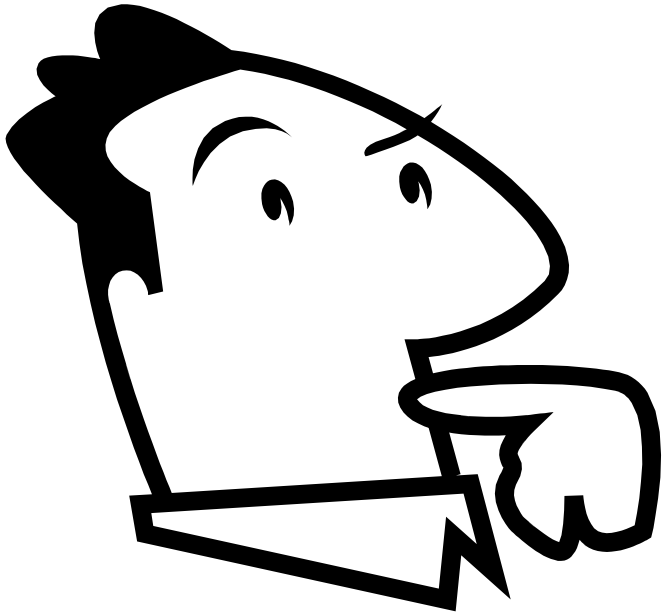
- Distanz Routing und Kompass Routing finden einen Pfad zum Ziel für beliebige Delaunay Graphen aber nicht für beliebige planare Triangulierungen.
- Distanz-Kompass Routing findet einen Pfad zum Ziel für beliebige planare Triangulierungen.
- Keine der drei Strategien ist allerdings in Delaunay Graphen c-kompetitiv für eine Konstante c.

Beweis:

Siehe „P. Bose, A. Brodnik, S. Carlsson, E. Demaine, R. Fleischer, A. Lopez-Ortiz, P. Morin, and I. Munro. Online routing in convex subdivisions. *Intl. Journal of Computational Geometry, vol. 12, pp. 283-295, 2002.”*

Referenzen

- James Aspnes, Udi Wieder: The expansion and mixing time of skip graphs with applications. *Distributed Computing* 21(6): 385-393, 2009.
- N. Bonichon, P. Bose, J.-L. De Carufel, L. Perkovic, and A. van Renssen. Upper and lower bounds for online routing on Delaunay triangulations. Discrete Computational Geometry 58: 482-504, 2017.
- O. Gabber and Z. Galil. Explicit constructions of linear-sized superconcentrators. *Journal of Computer and System Sciences, 22:407–420, 1981.*
- F. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays · Trees · Hypercubes. Morgan* Kaufmann Publishers, 1992.
- A. Lubotzky, R. Phillips, and R. Sarnak. Ramanujan graphs. *Combinatorica, 8(3):261–277, 1988.*
- C. Scheideler. *Universal Routing Strategies for Interconnection Networks.* Lecture Notes in Computer Science 1390. Springer, 1998.



Fragen?