Modelle der Parallelverarbeitung

11. Graphen und Netzwerke

Thomas Worsch

Institut für Theoretische Informatik
Karlsruher Institut für Technologie

Sommersemester 2018

Einleitung

- bequem Vorstellung: für jedes Datenelement ein eigener Prozessor
- von Problemgröße n abhängiger Graph G_n von Komponenten, die für Kommunikationszwecke miteinander verbunden sind: *„Topologie“* $\mathcal{G}$
- meist: gewünschte Verbindungsstruktur G_n
  $\neq$ reale Verbindungsstruktur H_n
  - unterschiedliche Struktur
  - unterschiedlich viele Knoten

Was möchte man?

Welche Eigenschaften sollte das Verbindungsnetzwerk eines Parallelrechners haben?

Überblick

Überblick

Überblick

Definition

- gerichteter *Graph* $G = (V, E)$ mit $|V| = n$ und $E \subseteq V \times V$
- $T \subseteq \mathbb{N}_+$, unendlich
- *Graphfamilie* $(G_n)_{n \in T}$ mit Graphen $G_n = (V_n, E_n)$, $|V_n| = n$.

Definition

- $n \in \mathbb{N}_+$
- $[n] = \{0, 1, \ldots, n-1\}$
  z. B. $[2] = \{0, 1\}$
- $[n]^d$ d-Tupel von Elementen aus $[n]$
- $[n]^* = \bigcup_{d \in \mathbb{N}_0} [n]^d$ alle Wörter über dem Alphabet $[n]$.
- für $0 \leq i < d$ sei $\bar{e}_i^{(d)} = 0 \cdots 010 \cdots 0 = 0^{d-1-i}10^i$
- Arithmetik
  - bei + und − in den ganzen Zahlen ausgeführt;
    falls Endergebnis nicht in $[n]$, ist Wert undefiniert.
  - bei $\oplus$ und $\ominus$ Arithmetik stets modulo n
  - mit d-Tupeln komponentenweise

Wortdistanzen

- für $\bar{a} \in [n]^d$ seien die Symbole mit $a_{d-1} \cdots a_0 = \bar{a}$ bezeichnet
- für $\bar{a}, \bar{b} \in [n]^d$ und $0 \leq i < d$ sei
  $\mathrm{dist}_i(\bar{a}, \bar{b}) := \min\{k \in \mathbb{N}_0 \mid a_i = k + b_i \vee b_i = k + a_i\}$ und
  $\mathrm{mdist}_i(\bar{a}, \bar{b}) := \min\{k \in \mathbb{N}_0 \mid a_i = k \oplus b_i \vee b_i = k \oplus a_i\}$.
- *Hamming-Distanz* $\mathrm{ham}(\bar{a}, \bar{b}) := \sum_{i=0}^{d-1} \mathrm{dist}_i(\bar{a}, \bar{b})$
- ohne Namen: $\mathrm{mham}(\bar{a}, \bar{b}) := \sum_{i=0}^{d-1} \mathrm{mdist}_i(\bar{a}, \bar{b})$

Wörter und Zahlen

Abbildung von natürlichen Zahlen in Wörter über $[2]$ und umgekehrt:

- $\mathrm{nat} : [2]^* \longrightarrow \mathbb{N}_0 : a_{d-1} \cdots a_0 \mapsto \sum_{i=0}^{d-1} a_i 2^i$
- $\mathrm{bin}_d : \mathbb{N}_0 \longrightarrow [2]^d$ mit
  - $\mathrm{bin}_d(k) = \bar{a}$ gdw. $|\bar{a}| = d \wedge \mathrm{nat}(\bar{a}) = k$
  - $\mathrm{bin}_d(k)$ undefiniert sonst

Verschmelzen von Knoten

- sei $G = (V, E)$ ein Graph und
- $P \subset 2^V$ eine Partitionierung der Knotenmenge V also
  - $u, v \in P \wedge u \neq v \Longrightarrow u \cap v = \emptyset$
  - $\bigcup_{u \in P} u = V$
- Der Graph $G' = (V', E')$, der durch Identifizierung jeweils aller Knoten einer Teilmenge $v \in P$ entsteht, ist wie folgt festgelegt:
  - $V' = P$ und
  - $E' = \{(u, v) \mid \exists \bar{a} \in u \;\; \exists \bar{b} \in v : (\bar{a}, \bar{b}) \in E\}$.

Überblick

Gitter, Tori und Hyperwürfel

Es sei $(k_1, \ldots, k_d) \in \mathbb{N}_0^d$.

- das *d-dimensionale $(k_1, \ldots, k_d)$-Gitter* ist der Graph $G = (V, E)$ mit
  - $V = [k_1] \times \cdots \times [k_d]$ und
  - $E = \{(\bar{a}, \bar{b}) \mid \mathrm{ham}(\bar{a}, \bar{b}) = 1\}$.
- der *d-dimensionale $(k_1, \ldots, k_d)$-Torus* ist der Graph $G = (V, E)$ mit
  - $V = [k_1] \times \cdots \times [k_d]$ und
  - $E = \{(\bar{a}, \bar{b}) \mid \mathrm{mham}(\bar{a}, \bar{b}) = 1\}$.
- Eindimensionale Tori heißen auch *Ringe*.
- Das d-dimensionale $(2, \ldots, 2)$-Gitter wird auch als *d-dimensionaler Hyperwürfel* bezeichnet.

Kürzeste Pfade in Hyperwürfeln

Kürzeste Pfade in Hyperwürfeln

- Kürzeste Pfade zwischen Knoten $\bar{a}$ und $\bar{b}$ eines Hyperwürfels haben Länge $\mathrm{ham}(\bar{a}, \bar{b})$.
- Zwischen zwei Knoten $\bar{a}$ und $\bar{b}$ eines Hyperwürfels gibt es $\mathrm{ham}(\bar{a}, \bar{b})!$ solche Pfade.

Cube connected cycles

Es sei $H = (V_H, E_H)$ der d-dimensionale Hyperwürfel.
d-dimensionaler CCC-Graph $C = (V_C, E_C)$:

- $V_C = V_H \times [d]$ und
- $E_C = \{((\bar{a}, i), (\bar{b}, i)) \mid (\bar{a}, \bar{b}) \in E_H \wedge i \in [d] \wedge \bar{a} \oplus \bar{b} = \bar{e}_i\}$
  $\cup \{((\bar{a}, i), (\bar{a}, j)) \mid \bar{a} \in V_H \wedge \mathrm{mdist}(i, j) = 1\}$.

- Anschaulich: Jede Ecke (Grad d) wird „abgeschnitten“ und durch einen Ring mit d Knoten (Grad 3) ersetzt.

Bäume

$T = (V_T, E_T)$ heißt *vollständiger binärer Baum* der Tiefe d, falls gilt:

- $V_T = \bigcup_{i=0}^{d} [2]^i$ und
- $E_T = \{(\bar{a}, \bar{a}b) \mid |\bar{a}| < d \wedge b \in [2]\}$.

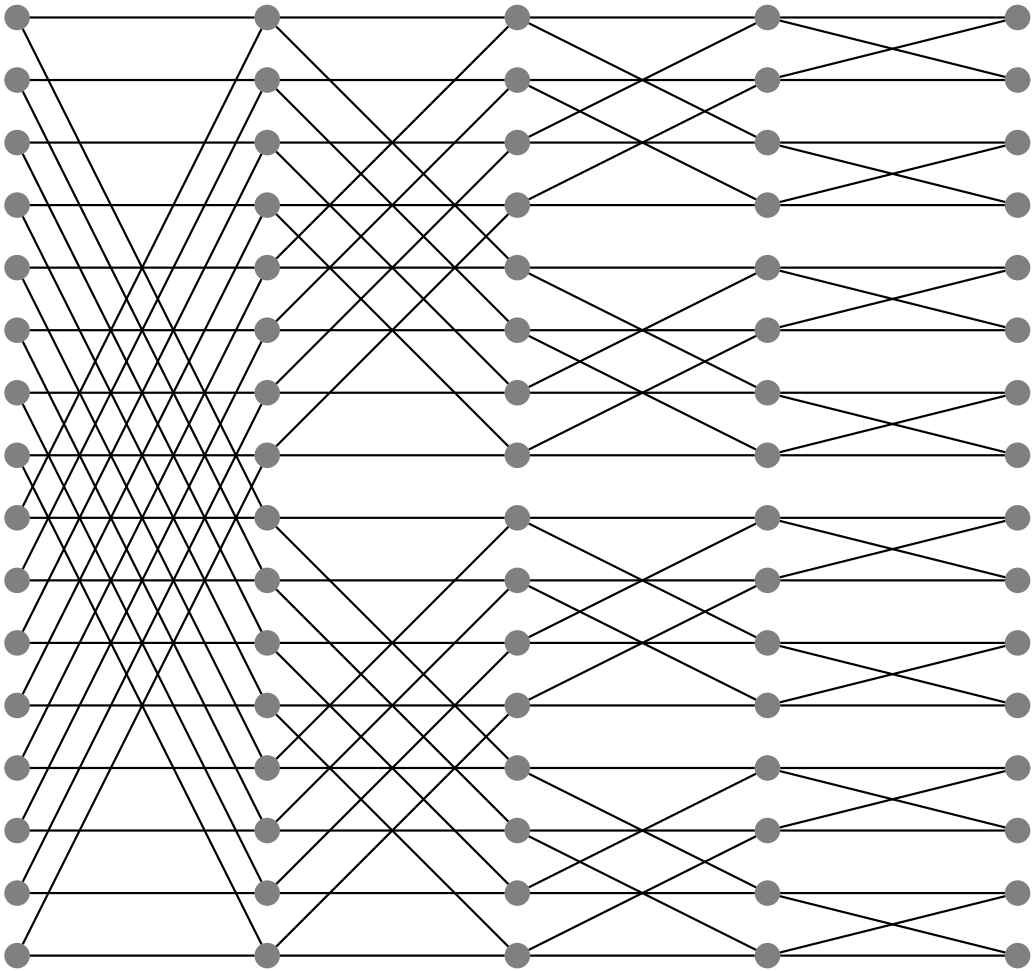
Ein Graph heißt *binärer Baum*, falls er zusammenhängender Teilgraph eines vollständigen binären Baumes ist und den Knoten ε enthält.

Meshes of trees

$M = (V_M, E_M)$ heißt *MOT(-Graph)* der Tiefe d, falls gilt:

- $V_M = [2]^d \times \bigcup_{i=0}^{d} [2]^i \cup \bigcup_{i=0}^{d} [2]^i \times [2]^d$ und
- $E_M = \{((\bar{a}, \bar{b}), (\bar{a}, \bar{b}c)) \mid \bar{a} \in [2]^d \wedge |\bar{b}| < d \wedge c \in [2]\}$
  $\cup \{((\bar{b}, \bar{a}), (\bar{b}c, \bar{a}) \mid \bar{a} \in [2]^d \wedge |\bar{b}| < d \wedge c \in [2]\}.$

Butterfly-Graphen



Butterfly-Graphen

- *d-dimensionaler Butterfly(-Graph)* $B = (V_B, E_B)$:
  - $V_B = [2]^d \times [d+1]$ und
  - $E_B = \{((\bar{a}, i), (\bar{b}, i+1)) \mid i \in [d] \wedge \bar{a} = \bar{b}\}$
    $\cup \{((\bar{a}, i), (\bar{b}, i+1)) \mid i \in [d] \wedge \bar{a} \oplus \bar{b} = \bar{e}_{d-1-i}\}$

- *zyklische Butterfly-Graphen* (wrapped butterfly):
  identifiziere für alle $\bar{a}$ die Knoten $(\bar{a}, 0)$ und $(\bar{a}, d)$

Eigenschaften von Butterfly-Graphen

- Verschmelzen jeweils aller Knoten einer Zeile liefert

Eigenschaften von Butterfly-Graphen

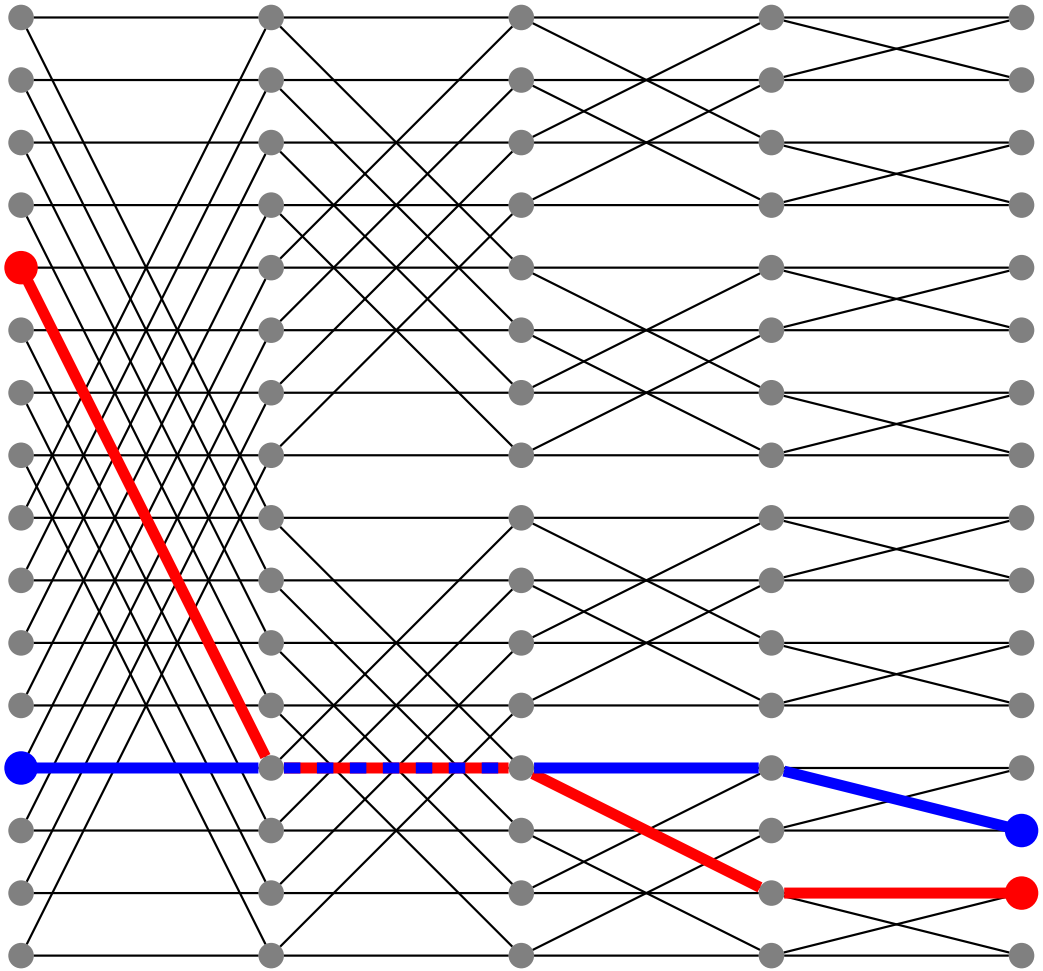
- Verschmelzen jeweils aller Knoten einer Zeile liefert einen Hyperwürfel

Eigenschaften von Butterfly-Graphen

- Verschmelzen jeweils aller Knoten einer Zeile liefert einen Hyperwürfel

- In einem d-dimensionalen Butterfly-Graphen gibt es von jedem Knoten $(\bar{a}, 0)$ zu jedem Knoten $(\bar{b}, d)$ genau einen kürzesten Pfad.
- Man findet leicht zwei Knotenpaare $((\bar{a}, 0), (\bar{b}, d))$ und $((\bar{g}, 0), (\bar{h}, d))$, deren verbindende kürzeste Pfade eine Kante des Butterfly-Graphen gemeinsam haben.

Butterfly-Graphen



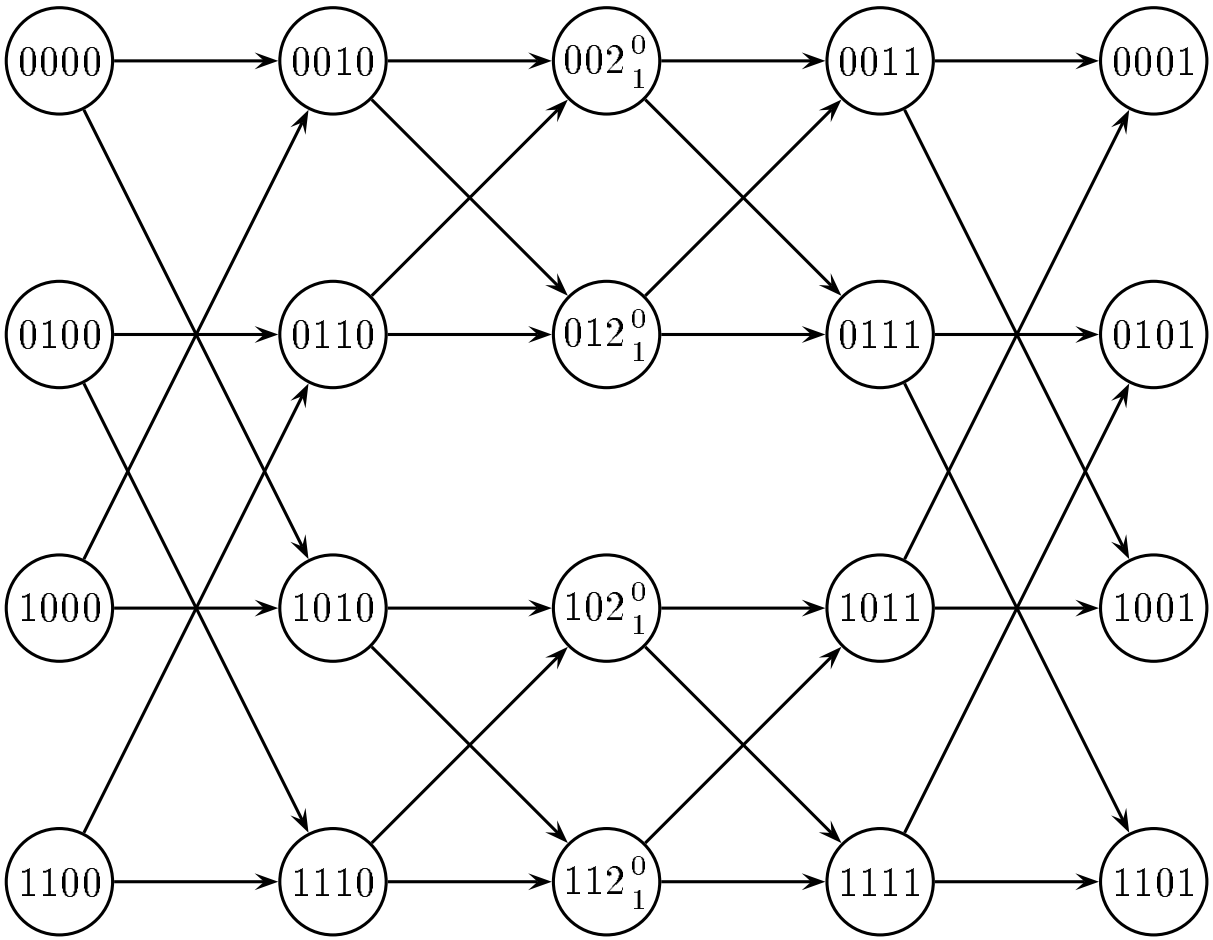
Beneš-Graphen

Es bestehe $B = (V_B, E_B)$ „im wesentlichen aus zwei gleich großen Butterfly-Graphen“:

- $V_B = [2]^d \times [d+1] \times [2]$
- $E_B = \{((\bar{a}, i, 0), (\bar{b}, i+1, 0)) \mid i \in [d] \wedge \bar{a} = \bar{b}\}$
  $\cup \{((\bar{a}, i, 0), (\bar{b}, i+1, 0)) \mid i \in [d] \wedge \bar{a} = \bar{b} \oplus \bar{e}_{d-1-i}\}$
  $\cup \{((\bar{b}, i+1, 1), (\bar{a}, i, 1)) \mid i \in [d] \wedge \bar{a} = \bar{b}\}$
  $\cup \{((\bar{b}, i+1, 1), (\bar{a}, i, 1)) \mid i \in [d] \wedge \bar{a} = \bar{b} \oplus \bar{e}_{d-1-i}\}$

d-dimensionaler Beneš-Graph: entsteht durch Identifizierung jeweils der zwei Knoten $(\bar{a}, d, 0)$ und $(\bar{a}, d, 1)$ für alle $\bar{a} \in [2]^d$

Beneš-Graphen: Beispiel



Beneš-Graphen: Eigenschaften

- Da es zwei Butterfly-Graphen sind:
  Zwischen je einem Knoten ganz links und einem ganz rechts gibt es immer einen Pfad.

Beneš-Graphen: Eigenschaften

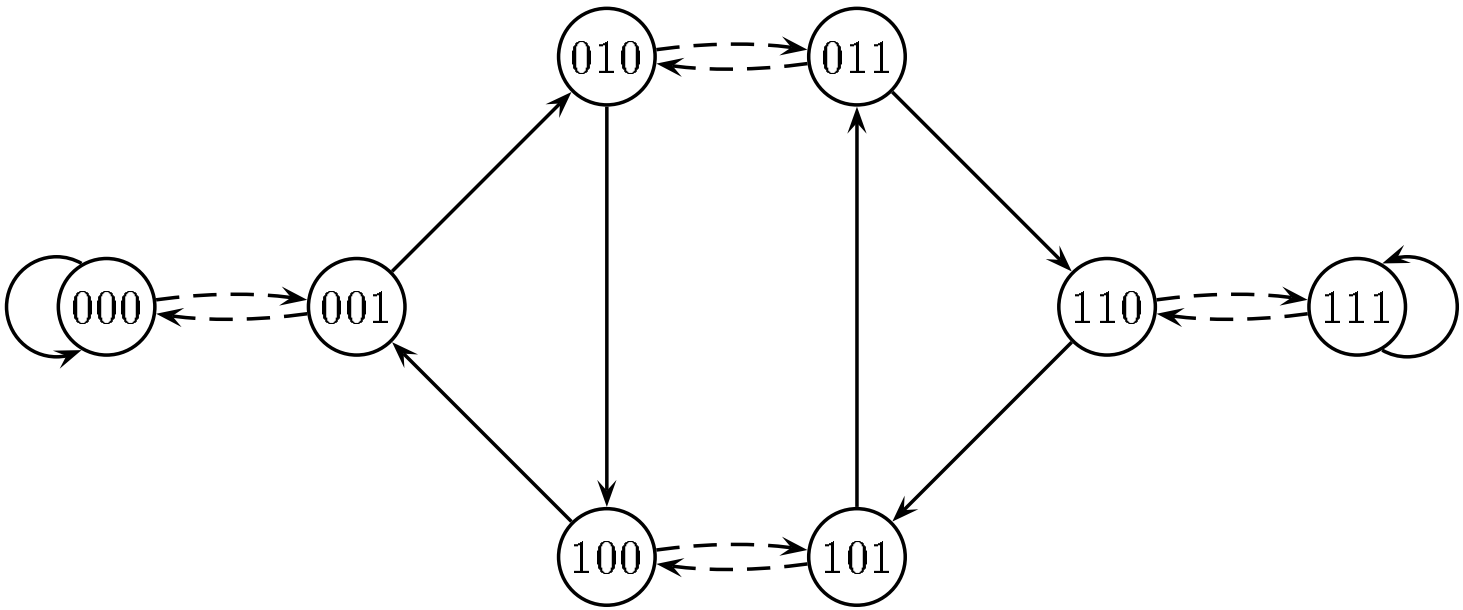
- Da es zwei Butterfly-Graphen sind:
  Zwischen je einem Knoten ganz links und einem ganz rechts gibt es immer einen Pfad.
- Wir werden sehen: Es gibt sogar immer „viele“ Pfade.

Shuffle-exchange-Graphen

$S = (V_S, E_S)$ heißt *Shuffle-exchange-Graph*, falls gilt:

- $V_S = [2]^d$ und
- $E_S = \{(b\bar{a}, \bar{a}b) \mid \bar{a} \in [2]^{d-1} \wedge b \in [2]\}$
  $\cup \{(\bar{a}b, \bar{a}b') \mid \bar{a} \in [2]^{d-1} \wedge b \in [2] \wedge b' = b \oplus 1\}$
- Kanten der ersten Menge heißen *Shuffle-Kanten*
  Kanten der zweiten Menge heißen *Exchange-Kanten*.

Shuffle-exchange-Graphen: Beispiel



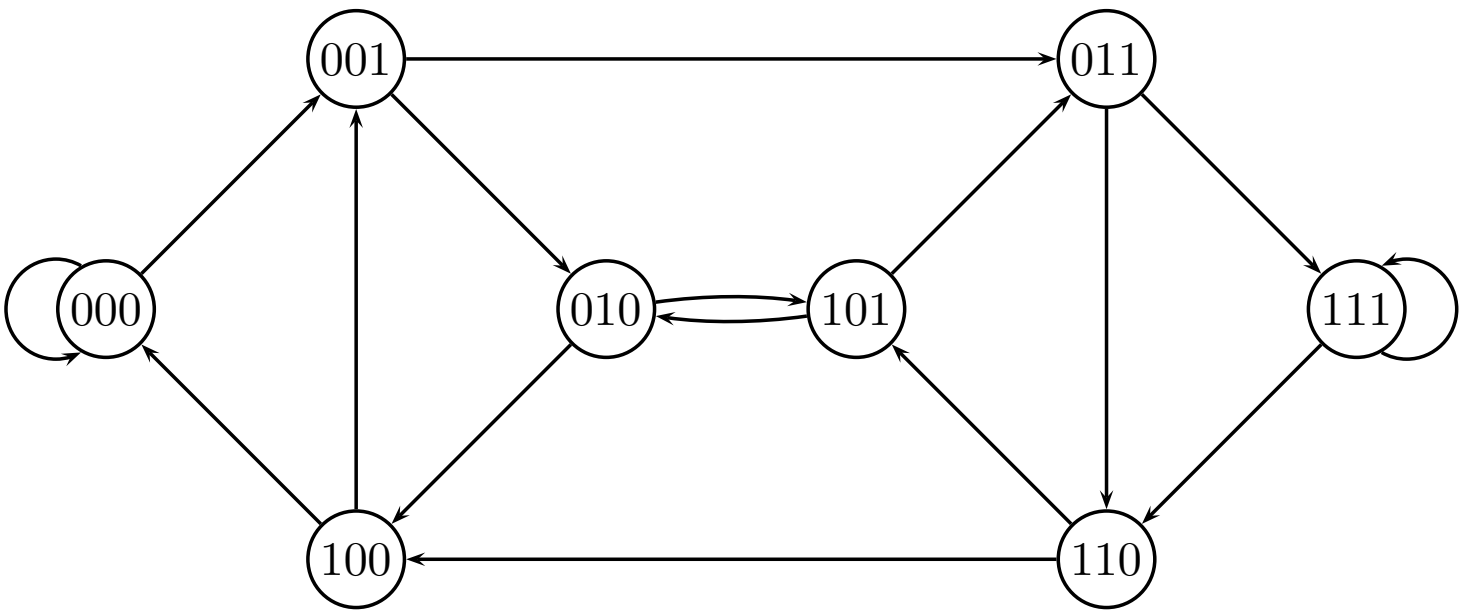
Exchange-Kanten gestrichelt
Shuffle-Kanten durchgezogen

De-Bruijn-Graphen

$B = (V_B, E_B)$ heißt *De-Bruijn-Graph*, falls gilt:

- $V_B = [2]^d$ und
- $E_B = \{(b\bar{a}, \bar{a}c) \mid \bar{a} \in [2]^{d-1} \wedge b, c \in [2]\}$.

De-Bruijn-Graphen: Beispiel



Shuffle-Exchange- und De-Bruijn-Graphen

- „ähnliche“ Definitionen
- Zusammenhang: Wenn man in einem Shuffle-exchange-Graphen Knoten identifiziert, die durch eine Exchange-Kante miteinander verbunden sind, dann erhält man einen De-Bruijn-Graphen (mit halb so vielen Knoten).

Kautz-Graphen

- K_d^k: zwei Parameter k und d
- Alphabet für Knotennummern $[d+1]$
- Knotenmenge:
  $V = \{x_0 x_1 \cdots x_{k-1} \in [d+1]^k \mid \forall\, 0 < i < k : x_{i-1} \neq x_i\}$
- Kantenmenge:
  $E = \{(x_0 x_1 \cdots x_{k-1}, x_1 \cdots x_{k-1} x_k)\} \subseteq V \times V$

Kautz-Graphen: Eigenschaften

- Grad aller Knoten: d
- Anzahl Knoten: $|V| = (d+1)d^{k-1} = d^k + d^{k-1}$
- Durchmesser: k
  - Das ist optimal: Wenn ein Graph $d^k + d^{k-1}$ Knoten mit Grad d hat, dann muss der Durchmesser $\geq k$ sein.
- Bisektionsweite: $\Theta(d^k/k)$
- Zwischen je zwei Knoten gibt es d kantendisjunkte Pfade.
- Jeder Kautzgraph besitzt einen Euler-Kreis und einen Hamilton-Kreis.

Kautz-Graphen: Anwendungsbeispiel

Rechner der Firma SiCortex (2003–2009), zum Beispiel

Kautz-Graphen: Anwendungsbeispiel

Rechner der Firma SiCortex (2003–2009), zum Beispiel

- 5832 Prozessoren, 5.8 TFlops, 18kW
- $d = 3, k = 6 \rightsquigarrow 792$ Knoten
- Knoten: 6 64-Bit Prozessoren, Netzwerk-Prozessor, …

Überblick

Von nun an betrachten wir auch die Varianten der eingeführten Graphen mit *ungerichteten* Kanten.

Definitionen

- Der *Durchmesser* eines Graphen ist das Maximum, genommen über alle Knotenpaare $(\bar{a}, \bar{b})$, der Länge der kürzesten Pfade von $\bar{a}$ nach $\bar{b}$.
- Die *Bisektionsweite* eines Graphen $G = (V, E)$ ist die minimale Anzahl von Kanten, die man aus E entfernen muss, damit G in zwei Teilgraphen zerfällt, deren Knotenzahlen sich um höchstens 1 unterscheiden und zwischen denen keine Kante verläuft.

Beispiel: vollständiger binärer Baum hat Bisektionsweite 1

Maßzahlen

Graph	Anzahl Knoten	Grad	Durch-messer	Bisektions-weite
$(k,\ldots,k)$-Torus	k^d	$2d$	$d\lceil\frac{k-1}{2}\rceil$	$2k^{d-1}$
$(k,\ldots,k)$-Gitter	k^d	$2d$	$d(k-1)$	k^{d-1}
Hyperwürfel	2^d	d	d	2^{d-1}
CCC	$d2^d$	3	$\lfloor\frac{5d}{2}\rfloor-1$	2^{d-1}
ger. shuff. ex.	2^d	4	$2d-1$	$\Theta(\frac{2^d}{d})$
ger. de Bruijn	2^d	4	d	$\Theta(\frac{2^d}{d})$
butterfly	$(d+1)2^d$	4	$2d$	2^d
wrapped butterfly	$d2^d$	4	$\lfloor\frac{3d}{2}\rfloor$	2^d
vollst. Baum	$2^{d+1}-1$	3	$2d$	1
MOT	$3\cdot 2^{2d}-2^{d+1}$	3	$4d$	2^d

Überblick

Überblick

Problemstellung

Gegeben: zwei Graphfamilien

- „guest“ $\mathcal{G} = (G_n)_{n \in T_G}$ und
- „host“ $\mathcal{H} = (H_n)_{n \in T_H}$

Gesucht: Möglichkeit,

- jeden Algorithmus für $\mathcal{G}$ auf einem Rechner mit „Topologie“ $\mathcal{H}$ auszuführen,
- indem man $f : T_G \longrightarrow T_H$ und „Simulationsverfahren“ festlegt, die angeben, wie man Algorithmus für G_n auf einem Rechner mit Verbindungsstruktur $H_{f(n)}$ simuliert.

Definition

- $G = (V_G, E_G)$ und $H = (V_H, E_H)$ seien zwei Graphen.
- Paar (f_V, f_E) von Abbildungen heißt eine *Einbettung* von G in H, wenn gilt:
  - $f_V : V_G \to V_H$
  - $f_E : E_G \to E_H^*$
    - für jedes $e \in E_G$ beschreibt $f_E(e)$ einen Pfad in H
    - für jede Kante $e = (u, v) \in E_G$ führt Pfad $f_E(e)$ von Knoten $f_V(u)$ zu Knoten $f_V(v)$
- G heißt *Gastgraph* (guest graph) und H heißt *Wirtsgraph* (host graph).

einfaches Beispiele: Teilgraphen, i. e.

- $f_V = id_{V_H}|_{V_G}$
- $f_E = id_{E_H}|_{E_G}$

Überblick

Lemma

Jedes eindimensionale Gitter G der Größe k ist Teilgraph des d-dimensionalen Hyperwürfels H mit

- $d = \lceil \log k \rceil$
- also $2^{\lceil \log k \rceil}$ Knoten.

Definition

Für $d \geq 1$ heißen die Abbildungen $g_d : [2^d] \longrightarrow [2]^d$ mit $g_1(0) = 0$ und $g_1(1) = 1$ und

$$g_{d+1}(k) = \begin{cases} 0g_d(k) & \text{falls } 0 \leq k < 2^d \\ 1g_d(2^{d+1} - 1 - k) & \text{falls } 2^d \leq k < 2^{d+1} \end{cases}$$

die *d-stelligen gespiegelten Gray-Codes*.

Gray-Codes: Beispiele und Eigenschaften

- Gray-Codes für $d = 1, 2, 3$:

0	1	2	3	4	5	6	7
0	1	1	0	0	1	1	0
0	0	1	1	1	1	0	0
0	0	0	0	1	1	1	1

- Leicht zu sehen: Die Abbildungen g_d sind alle bijektiv, und die Codes aufeinanderfolgender Zahlen haben stets Hamming-Distanz 1.

Beweis des Lemmas

- salopp: codiere Knotennummern der linearen Kette mit Gray-Code
- technisch:
  - setze $d = \lceil \log k \rceil$
  - wähle $f_V = g_d$
    - im Gitter benachbarte Knoten werden auf im Hyperwürfel benachbarte Knoten abgebildet
  - wähle $f_E((i,j)) = (g_D(i), g_D(j))$

Satz

Jedes d-dimensionale Gitter G mit den Ausdehnungen $(k_1, \ldots, k_d)$ ist Teilgraph des D-dimensionalen Hyperwürfels H mit

- $D = \lceil \log k_1 \rceil + \cdots + \lceil \log k_d \rceil$
- also $2^{\lceil \log k_1 \rceil} \ldots 2^{\lceil \log k_d \rceil}$ Knoten

Beweis des Satzes

Induktionsanfang $d = 1$: siehe Lemma

Beweis des Satzes (2)

Induktionsschritt $d \longrightarrow d + 1$:

Beweis des Satzes (2)

Induktionsschritt $d \longrightarrow d+1$:

- Fasse $(d+1)$-dimensionales $(k_1, \ldots, k_{d+1})$-Gitter auf als k_{d+1} „Schichten“ d-dimensionaler $(k_1, \ldots, k_d)$-Gitter
- Jede Schicht nach Induktionsvoraussetzung in Hyperwürfel mit Adressen $\bar{a} \in [2]^\ell$ einbettbar
  mit $\ell = \lceil \log k_1 \rceil + \cdots + \lceil \log k_d \rceil$ ist.
  Es sei f'_V die entsprechende Knoteneinbettungsfunktion.
- Es sei nun $m = \lceil \log k_{d+1} \rceil$.
  Betrachte $\ell + m$-dimensionalen Hyperwürfel mit Adressen $\bar{a}\bar{b} \in [2]^\ell [2]^m$.
- Er besitzt 2^m paarweise disjunkte ℓ-dimensionale Unterhyperwürfel als Teilgraphen.
  $f_V((i_1, \ldots, i_d, i_{d+1})) = f'_V((i_1, \ldots, i_d)) g_m(i_{d+1})$.

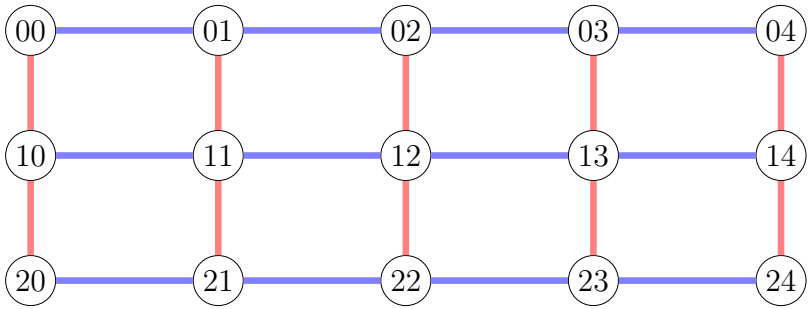
Probleme

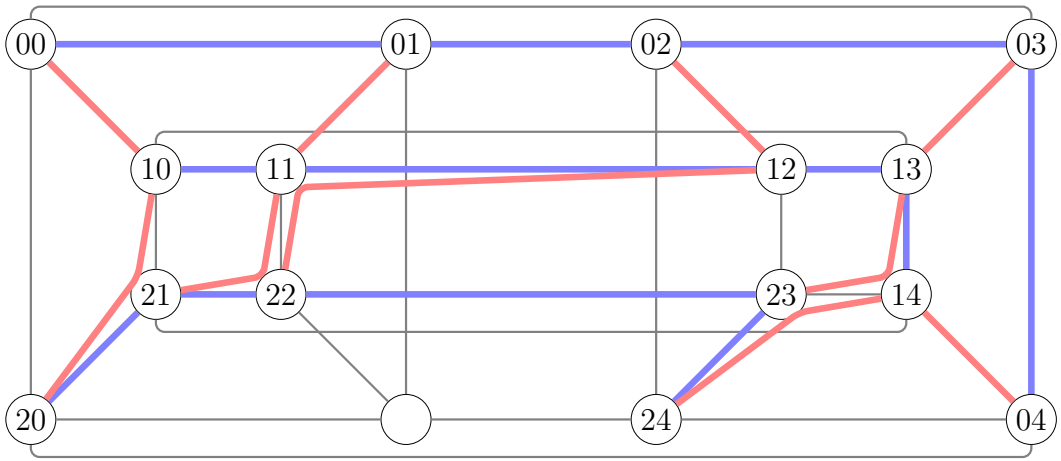
zu beachten:

- Was bedeuten „rechts“ und „links“ im Hyperwürfel?
- bei manchen Rechnern in einem Schritt nur Kommunikation in *einer* Dimension möglich
- Knotenzahl bei dieser Art der Einbettung von z. B. 3×5-Gitter nicht ganz so klein …

Aber

Einbettung von 3×5-Gitter in 2^4-Hyperwürfel:





Überblick

Definition

Es sei (f_V, f_E) eine Einbettung eines Graphen $G = (V_G, E_G)$ in einen Graphen $H = (V_H, E_H)$.

- *Expansion* (engl. expansion) der Einbettung: $\frac{|V_H|}{|V_G|}$
- *Dilatation* (engl. dilation) der Einbettung: maximale Länge $\max\{|f_E(e)| \mid e \in E_G\}$ eines Bildpfades
- *Kantenlast* (engl. congestion) der Einbettung: maximale Zahl $\max\left\{\left|\{e_G \mid e_H \text{ kommt in } f_E(e_G) \text{ vor}\}\right| \,\middle|\, e_H \in E_H\right\}$ von Bildpfaden, die über eine Kante $e_H \in E_H$ laufen
- *(Knoten-)Last* (engl. load) der Einbettung: maximale Zahl $\max\{|f^{-1}(v_H)| \mid v_H \in V_H\}$ von Gastknoten, die auf einen Wirtsknoten abgebildet werden

Beispiele

- Man kann das $(3, 5)$-Gitter mit Expansion $\frac{16}{15}$, Dilatation 2, Kantenlast 2 und Last 1 in den 4-dimensionalen Hyperwürfel einbetten.
- Man kann jedes d-dimensionale Gitter mit Dilatation, Kantenlast und Last 1 und einer Expansion kleiner gleich 2^d in Hyperwürfel einbetten kann.
- Behauptung: Jedes zweidimensionale Gitter mit n Knoten kann man in einen Hyperwürfel mit $2^{\lceil \log n \rceil}$ Knoten mit Dilatation 2 einbetten
  - Für höherdimensionale Gitter wächst bei den bekannten Konstruktionen die Dilatation.

Überblick

Lemma

Ein vollständiger binärer Baum der Tiefe $d \geq 2$

- ist nicht Teilgraph des Hyperwürfels mit 2^{d+1} Knoten,

Lemma

Ein vollständiger binärer Baum der Tiefe $d \geq 2$

- ist nicht Teilgraph des Hyperwürfels mit 2^{d+1} Knoten,
- kann also nicht injektiv in seinen optimalen Hyperwürfel mit Dilatation 1 eingebettet werden.

Beweis

Beweis

- Beide Graphen sind bipartit.

Beweis

- Beide Graphen sind bipartit.
- betrachte naheliegende Zweifärbungen

Beweis

- Beide Graphen sind bipartit.
- betrachte naheliegende Zweifärbungen
  - in Hyperwürfeln beide Farben gleich häufig
  - in Bäumen überwiegt die Farbe der Blätter
- also …

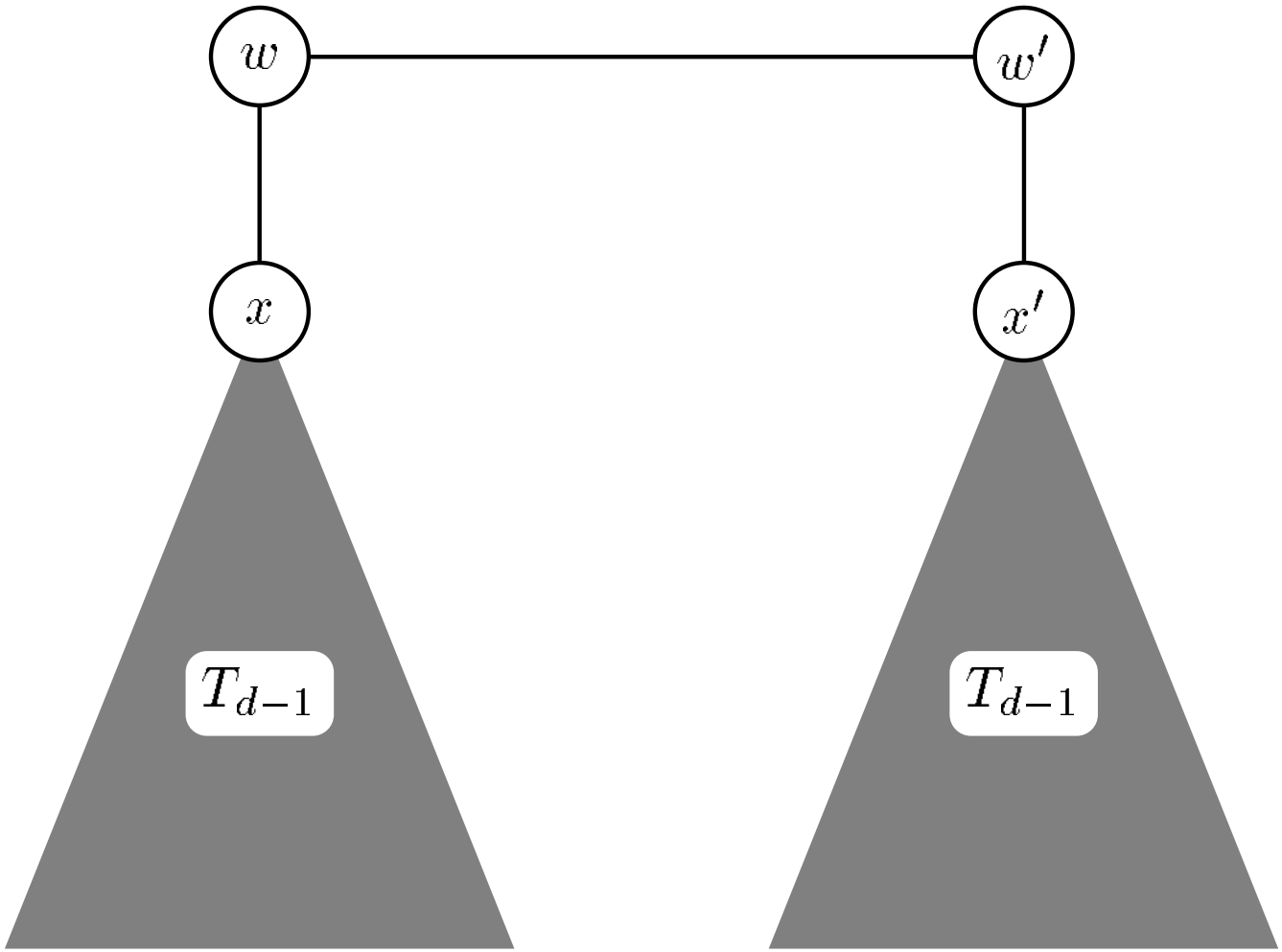
Satz

- Ein vollständiger binärer Baum der Tiefe $d \geq 2$ kann injektiv so in seinen optimalen Hyperwürfel mit Dilatation 2 eingebettet werden, dass nur eine einzige Kante auf einen Pfad der Länge 2 abgebildet wird und alle anderen Kanten auf Pfade der Länge 1.
- Zwei vollständige binäre Bäume der Tiefe $d - 1$ können injektiv mit Dilatation 1 in disjunkte Teile des (sinngemäß wieder optimalen) Hyperwürfels H_{d+1} eingebettet werden.

Definition 2-Wurzel-Baum

- Sei $T_{d-1} = (V_T, E_T)$ ein vollständiger binärer Baum der Tiefe $d-1$.
- Ein *vollständiger binärer 2-Wurzel-Baum* (engl. double-rooted complete binary tree) der Tiefe d ist der Graph $D_d = (V_D, E_D)$, für den gilt:
  - $V_D = \{w, w'\} \cup \{x\}V_T \cup \{x'\}V_T$, wobei w, w', x und x' vier neue Symbole seien.
  - $E_D = \{(w, w'), (w, x), (w', x')\} \cup \{(x\bar{a}, x\bar{b}) \mid (\bar{a}, \bar{b}) \in E_T\} \cup \{(x'\bar{a}, x'\bar{b}) \mid (\bar{a}, \bar{b}) \in E_T\}$
  - Die Knoten w und w' sind „die beiden Wurzeln“ von D_d.

2-Wurzel-Baum: Beispiel

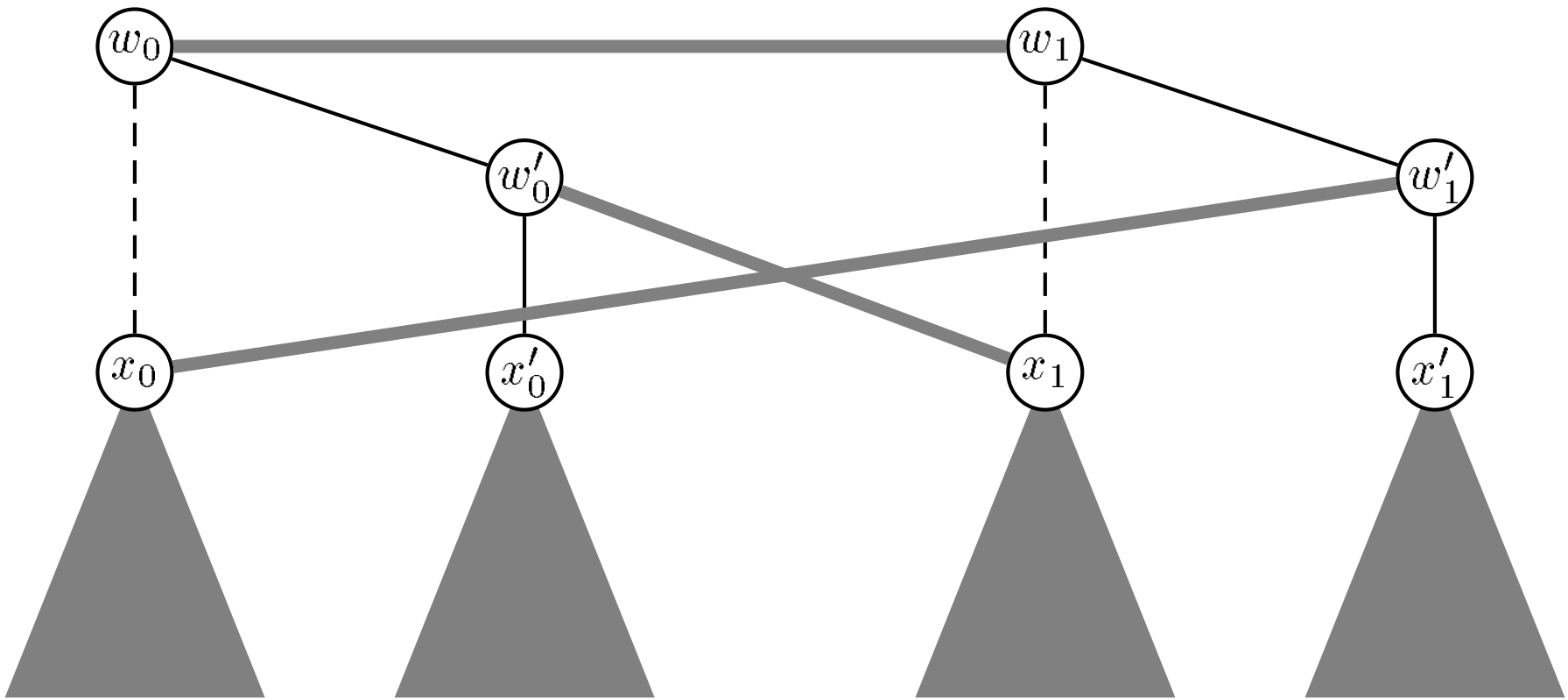


Lemma

- Es seien $D_i = (V_i, E_i)$, $i \in [2]$, zwei vollständige binäre 2-Wurzel-Bäume der Tiefe $d - 1$. Dann ist $D_0 \otimes D_1 := (V, E)$ mit
  - $V = V_0 \cup V_1$ und
  - $E = E_0 \cup E_1 \cup \{(w_0, w_1), (w_0', x_1), (w_1', x_0)\} \setminus \{(w_0, x_0), (w_1, x_1)\}$

  ein vollständiger binärer 2-Wurzel-Baum der Tiefe d mit $w = w_0$, $w' = w_1$, $x = w_0'$ und $x' = w_1'$.

Beweis



Lemma

- Wenn
  - (f_V, f_E) die injektive Einbettung eines Graphen G in einen d-dimensionalen Hyperwürfel mit Dilatation 1 ist,
  - $\pi' : [d] \to [d]$ eine Permutation und
  - $\pi : [2]^d \to [2]^d : b_{d-1} \cdots b_1 b_0 \mapsto b_{\pi'(d-1)} \cdots b_{\pi'(1)} b_{\pi'(0)}$ die dadurch induzierte Abbildung
- dann auch (f'_V, f'_E) eine injektive Einbettung von G in einen d-dimensionalen Hyperwürfel mit Dilatation 1, wobei
  - $f'_V(v) = \pi(f_V(v))$ und
  - $f'_E((v_1, v_2)) = (\pi(f_V(v_1)), \pi(f_V(v_2)))$
- Beweis: leichte Übung

Lemma

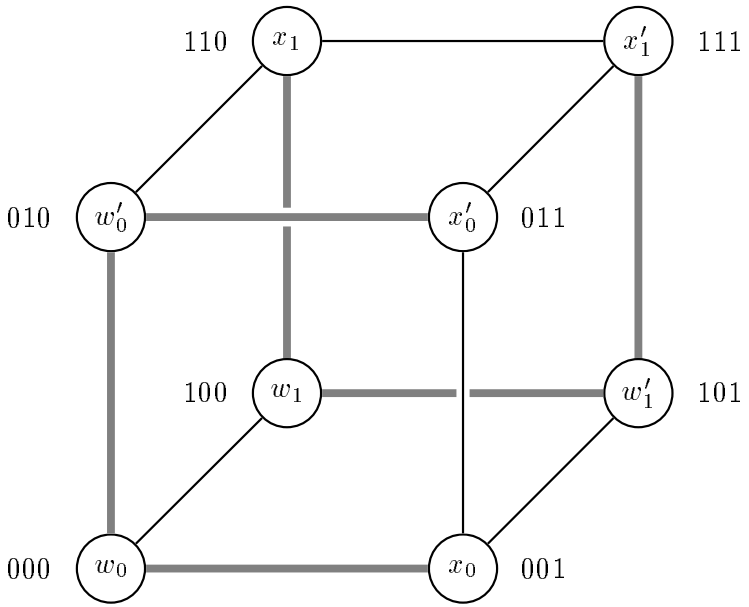
Für alle $d \geq 2$ kann man zwei vollständige binäre 2-Wurzel-Bäume $D_i = (V_i, E_i)$, $i \in [2]$, der Tiefe $d - 1$ injektiv mit Dilatation 1 in den $(d + 1)$-dimensionalen Hyperwürfel $H = (V_H, E_H)$ so einbetten, dass gilt:

- $\forall i \in [2]\ \forall v \in V_i : f_V(v) \in i[2]^d$ und
- die Knotenabbildungen f_i für D_i legen auch eine injektive Einbettung des vollständigen binären 2-Wurzel-Baumes $D_0 \otimes D_1$ der Tiefe d mit Dilatation 1 in H fest.

Beweis

Induktionsanfang $d = 2$:

- betrachte



- Offensichtlich:

$$f_0(w_0) = 0000^{d-2} \quad f_0(w_0') = 0100^{d-2} \quad f_0(x_0) = 0010^{d-2}$$
$$f_1(w_1) = 1000^{d-2} \quad f_1(w_1') = 1010^{d-2} \quad f_1(x_1) = 1100^{d-2}.$$

Beweis (2)

Induktionsanfang $d = 2$ (Fortsetzung);

- $\text{ham}(f_0(w_0), f_1(w_1)) = 1$, $\text{ham}(f_0(w_0'), f_1(x_1)) = 1$ und $\text{ham}(f_1(w_1'), f_0(x_0)) = 1$, so dass
- auch $D = D_0 \otimes D_1$ Teilgraph von H ist.
- für Knotenabbildungsfunktion f gilt:

$$f(w) = f_0(w_0) = 0000^{d-2} \qquad f(w') = f_1(w_1) = 1000^{d-2}$$
$$f(x) = f_0(w_0') = 0100^{d-2}$$

- Im Induktionsschritt werden wir die vorher genannten Eigenschaften von f_0 und f_1 erhalten.
- Damit ist das Lemma dann bewiesen.

Beweis (3)

Induktionsschritt $d \longrightarrow d+1$:

- Seien D_0 und D_1 zwei vollständige binäre 2-Wurzel-Bäume der Tiefe d.
- Es sei $\pi : [2]^{d+2} \longrightarrow [2]^{d+2}$ diejenige Abbildung, die das zweite und das dritte Bit von links vertauscht.
- Definiere injektive Einbettungen f_0 bzw. f_1 von D_0 bzw. D_1 in den $(d+2)$-dimensionalen Hyperwürfel wie folgt:
  - $f_0(v) = 0f(v)$ und
  - $f_1(v) = \pi(1f(v))$
- leichte Rechnung:

$$f_0(w_0) = 0000^{d-1} \quad f_0(w_0') = 0100^{d-1} \quad f_0(x_0) = 0010^{d-1}$$
$$f_1(w_1) = 1000^{d-1} \quad f_1(w_1') = 1010^{d-1} \quad f_1(x_1) = 1100^{d-1}.$$

Überblick

Aufgabenstellung

- Ein Prozessor oder manche oder alle Prozessoren haben *Pakete*, die von ihrem Quellprozessor zu Zielprozessor(en) transportiert werden müssen.
- Probleme:
  - Alles soll möglichst schnell am Ziel sein.
  - Pfadlängen?
  - Staus?

Überblick

Aufgabestellung

Probleminstanz:

- Menge M von Nachrichten
- Funktionen $src : M \longrightarrow A$ und $dest : M \longrightarrow A$,
- die für Nachricht m die Adressen von Startknoten $src(m)$ und Zielknoten $dest(m)$ spezifizieren.

Gesucht:

- die Pfade im Netzwerk
- die zeitliche Anordnung

Spezialfälle

- *Permutationsrouting*:
  jedes PE Quelle und Ziel von genau einer Nachricht
- *h-Relation*:
  jedes PE Quelle und Ziel von genau $h \in \mathbb{N}_+$ Nachrichten

Leitungsvermittlung

- Für jede Nachricht wird zu einem Zeitpunkt ein ganzer Pfad (die „Leitung“) von Quelle zu Ziel exklusiv reserviert und
- dann die Nachricht in so vielen Schritten wie der Pfad lang ist übertragen.
- In dieser Zeit kann keine der am Pfad beteiligten Kanten auch noch für einen andere Nachrichtentransport benutzt werden.

Das werden wir nicht betrachten.

Paketvermittlung

Das werden wir betrachten.

Übertragung einer Nachricht wird „in Einzelschritte zerlegt“:

- für Übertragung einer Nachricht über eine Kante Kante exklusiv dafür reserviert
- auf nächstem Knoten muss die Nachricht u. U. auf Weitertransport warten, da nächste Kante belegt
- wegen der Wartezeiten in allen Knoten *Warteschlangen* hinreichender Größe nötig
- (widersprüchliche?) Anforderungen an Routing-Algorithmus:
  - möglichst kurze maximale Warteschlangen (Hardware!)
  - Minimierung der Gesamt-Routingzeit

Lösung für ein Routing-Problem $(M, src, dest)$

für jede Nachricht m ein „Fahrplan“ $(t_1, e_1), \ldots, (t_{l(m)}, e_{l(m)})$.

- $(e_1, \ldots, e_{l(m)})$ ein Pfad von $src(m)$ nach $dest(m)$,
- $t_i < t_j$ für $i < j$, und
- für keine zwei verschiedene Nachrichten m und m' existieren Indizes i und i', so daß $(t_i, e_i) = (t'_{i'}, e'_{i'})$ ist.

- *Off-line routing*:
  Algorithmus weiß alles über alle Nachrichten.
  Eingabe sind die vollständigen Funktionen $src : M \to A$ und $dest : M \to A$.
- *On-line routing*:
  Algorithmus für jedes PE: für jede ankommende Nachricht kann nur aufgrund des bisher gesammelten lokalen Wissens wie
  - PE-Adresse
  - Zieladresse der Nachricht
  - bisher geroutete Nachrichten,
  - …

  entschieden werden, wann sie über welche Kante weitergeschickt werden soll.

Überblick

Lemma

Für jeden Graphen mit Durchmesser x gibt es eine Permutation π, so dass das Routing von π mindestens x Schritte dauert.

Lemma

Für jeden Graphen mit n Knoten und Bisektionsweite x gibt es eine Permutation π, so dass das Routing von π mindestens $n/2x$ Schritte dauert.

Überblick

Routing in eindimensionalen Gittern

Wie?

Routing in eindimensionalen Gittern

Wie?

So:

if hier startende Nachricht muss zu anderem Knoten
then schicke Nachricht als nächstes über die Kante,
entlang der der Weg zum Ziel am kürzesten ist
fi
if Nachricht trifft über eine Kante ein
und Zieladresse ≠ aktuelle Knotennummer
then schicke Nachricht als nächstes über die andere Kante
fi

Analyse

- Alle Nachrichten starten gleichzeitig und
- bewegen sich in jedem Schritt weiter, solange sie noch nicht am Ziel sind.
- Also können sich Nachrichten, die sich in die gleiche Richtung bewegen, nie einholen.
- Es können in einem Knoten höchstens zwei Nachrichten zusammentreffen, die in entgegengesetzte Richtungen unterwegs sind.
- Es reichen Warteschlangen konstanter Größe.
- Alle Nachrichten gelangen schnellstmöglich an ihr Ziel.

Online-Routing für 2-dimensionale Gitter

Wie?

Greedy Online-Routing für 2-dimensionale Gitter

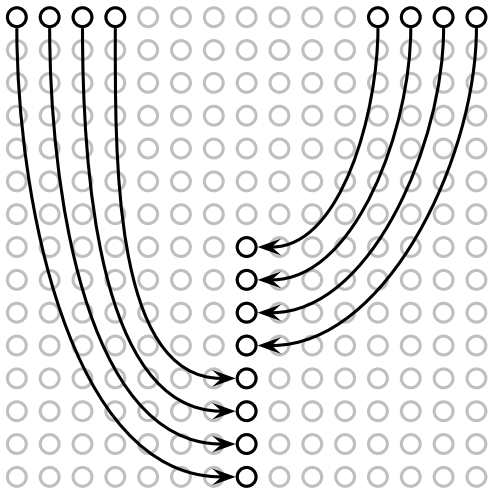
Wie?

So:

- Transportiere jedes Paket zuerst in die passende Spalte
- und anschließend in die richtige Zeile.
- Bevorzuge stets Pakete, die noch weite Wege vor sich haben.

Problem beim Greedy Online-Routing für zweidimensionale Gitter

Problem beim Greedy Online-Routing für zweidimensionale Gitter



- Problemfall:
  alle Pakete einer Zeile wollen in die gleiche Spalte
- Warteschlangen der Größe $\sqrt{n}$ erzwungen.
- Immerhin: Nach $2\sqrt{n}$ Schritten sind alle am Ziel:
  zeitoptimal (Durchmesser)

Schnelles Routing für zweidimensionale Gitter mit konstant großen Warteschlangen

Ideen?

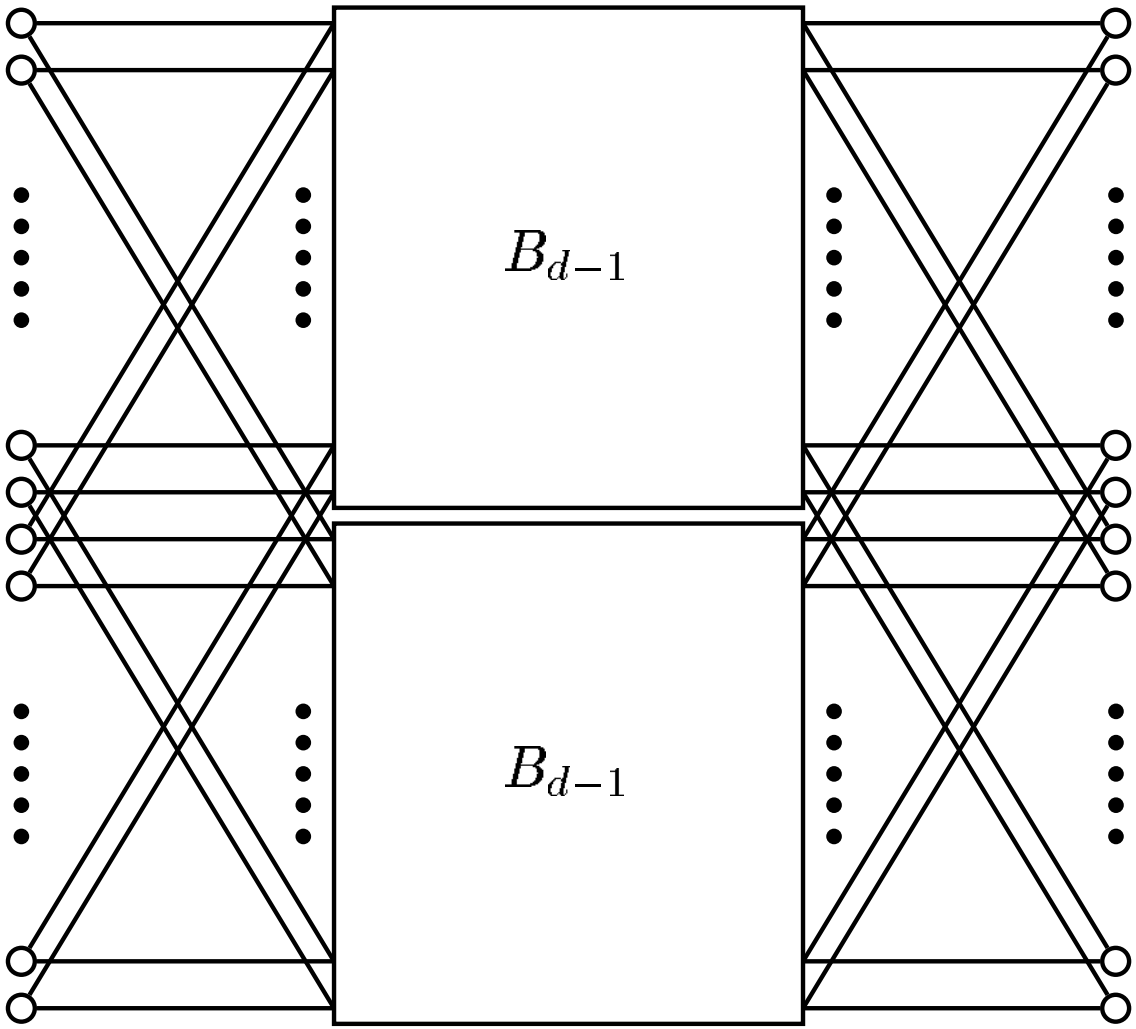
Schnelles Routing für zweidimensionale Gitter mit konstant großen Warteschlangen

Ideen?

- (Randomisierung)
- Permutationsrouting entspricht Sortieren:
  Es gibt Sortieralgorithmen für zweidimensionale Gitter, die
  - in Zeit $\Theta(\sqrt{n})$ arbeiten und
  - nur lokalen *Austausch von Paketen* zwischen benachbarten Knoten benötigen.

Überblick

Rekursiver Aufbau von Beneš-Netzwerken



Problemstellung

- Bezeichne die Knoten am linken Rand des Graphen als Eingänge
- und die am rechten Rand des Graphen als Ausgänge.
- Problem: Route eine beliebige 2-Relation von den Eingängen zu den Ausgängen möglichst „geschickt“.

Satz

In Beneš-Graphen können 2-Relationen von den Eingängen zu den Ausgängen auf paarweise kantendisjunkten Pfaden geroutet werden.

Beachte:

- Butterfly-Graphen haben diese Eigenschaft nicht.
- Für das Routing werden *alle* Kanten benötigt.

Beweisidee

- Induktion
- an der Tafel

Überblick

Universelle Netzwerke

Z. B. Butterfly-Netzwerke sind $\log n$-universell, d. h.

- *jedes* Netzwerk mit konstantem Knotengrad und „ungefähr ebensovielen“ Knoten wie ein Butterfly-Netzwerk
- kann mit einem („nur“) um den Faktor $\log n$ größeren Zeitbedarf simuliert werden.

Emulationen

- Einbettungen:
  - motiviert durch den Wunsch, ein Netzwerk auf einem anderen zu „simulieren“
  - Idee: Jeder Gast-Knoten wird von genau einem Wirts-Knoten simuliert
  - das ist zwar naheliegend, *aber nicht immer das Beste*
- Emulationen:
  - In manchen Fällen kommt man zu *nachweisbar* asymptotisch schnelleren Verfahren,
  - wenn man erlaubt, dass jeder Gast-Knoten von mehreren Wirts-Knoten simuliert werden darf.
    - Konsistenz … ? …