Contents I

1. Introduction and Networks
   - Basics
   - Motivation
   - Paths, Cycles, Grids
   - Trees
   - Hypercube

2. Paths and Cycles in...
   - ... in Trees
   - ... in Cubes
   - ... in Grids

3. Trees in...
   - ... Paths and Cycles
   - ... Cubes

4. Cube-like Networks in...

5. Embeddings
   - SE, DB, HQ
   - TR, G and HQ
   - Arbitrary Trees

6. Optical Networks
   - Basics
   - Specifications
   - Devices
   - Problem
   - Introduction
   - Wavelength-Assignment
   - Lines and Cycles
   - Trees
**Definition**

Let $G = (V, E)$ and $H = (W, F)$ be graphs. An embedding (embedding-function) from $G$ into $H$ is: $f : V \mapsto W$. We use for embeddings the following cost-functions:

- $|W|/|V|$ (Expansion)
- $\max_{w \in W} |\{v \mid f(v) = w\}|$ (Load)
- $\max \{\text{dist}_H(f(a), f(b)) \mid \{a, b\} \in E\}$ (Dilation)

**Definition**

A routing for an embedding $f : V \mapsto W$ is a function: $r : E \mapsto \{\text{Paths in } H\}$ with: $r(\{a, b\})$ is a path from $f(a)$ to $f(b)$. Note the cost-functions:

- $\max \{|r(\{a, b\})| \mid \{a, b\} \in E\}$ (Dilation)
- $\max \{|\{e \mid e \in E, e' \in r(e)\}| \mid e' \in F\}$ (Congestion)
Embeddings

Definition

Let $G = (V, E)$ and $H = (W, F)$ be graphs.
An embedding (embedding-function) from $G$ into $H$ is: $f : V \mapsto W$.
We use for embeddings the following cost-functions:

- $|W|/|V|$ (Expansion)
- $\max_{w \in W} |\{v \mid f(v) = w\}|$ (Load)
- $\max\{|\text{dist}_H(f(a), f(b))| \mid \{a, b\} \in E\}$ (Dilation)

Definition

A routing for an embedding $f : V \mapsto W$ is a function:
$r : E \mapsto \{\text{Paths in } H\}$ with: $r(\{a, b\})$ is a path from $f(a)$ to $f(b)$.
Note the cost-functions:

- $\max\{|r(\{a, b\})| \mid \{a, b\} \in E\}$ (Dilation)
- $\max\{|\{e \mid e \in E, e' \in r(e)\}| \mid e' \in F\}$ (Congestion)
**Embeddings**

**Definition**

Let $G = (V, E)$ and $H = (W, F)$ be graphs. An embedding (embedding-function) from $G$ into $H$ is: $f : V \mapsto W$. We use for embeddings the following cost-functions:

- $|W|/|V|$ (Expansion)
- $\max_{w \in W} |\{v \mid f(v) = w\}|$ (Load)
- $\max\{dist_H(f(a), f(b)) \mid \{a, b\} \in E\}$ (Dilation)

**Definition**

A routing for an embedding $f : V \mapsto W$ is a function:

$r : E \mapsto \{\text{Paths in } H\}$ with: $r(\{a, b\})$ is a path from $f(a)$ to $f(b)$. Note the cost-functions:

- $\max\{|r(\{a, b\})| \mid \{a, b\} \in E\}$ (Dilation)
- $\max\{|\{e \mid e \in E, e' \in r(e)\}| \mid e' \in F\}$ (Congestion)
Embeddings

Definition

Let $G = (V, E)$ and $H = (W, F)$ be graphs.
An embedding (embedding-function) from $G$ into $H$ is: $f : V \mapsto W$.
We use for embeddings the following cost-functions:

- $|W|/|V|$ (Expansion)
- $\max_{w \in W} |\{v \mid f(v) = w\}|$ (Load)
- $\max \{\text{dist}_H(f(a), f(b)) \mid \{a, b\} \in E\}$ (Dilation)

Definition

A routing for an embedding $f : V \mapsto W$ is a function:
$r : E \mapsto \{\text{Paths in } H\}$ with: $r(\{a, b\})$ is a path from $f(a)$ to $f(b)$.
Note the cost-functions:

- $\max \{|r(\{a, b\})| \mid \{a, b\} \in E\}$ (Dilation)
- $\max \{|\{e \mid e \in E, e' \in r(e)\}| \mid e' \in F\}$ (Congestion)
Embeddings

**Definition**

Let $G = (V, E)$ and $H = (W, F)$ be graphs.
An embedding (embedding-function) from $G$ into $H$ is: $f : V \mapsto W$.
We use for embeddings the following cost-functions:

- $|W|/|V|$ (Expansion)
- $\max_{w \in W} |\{v \mid f(v) = w\}|$ (Load)
- $\max\{dist_H(f(a), f(b)) \mid \{a, b\} \in E\}$ (Dilation)

**Definition**

A routing for an embedding $f : V \mapsto W$ is a function: $r : E \mapsto \{\text{Paths in } H\}$ with: $r(\{a, b\})$ is a path from $f(a)$ to $f(b)$.
Note the cost-functions:

- $\max\{|r(\{a, b\})| \mid \{a, b\} \in E\}$ (Dilation)
- $\max\{||\{e \mid e \in E, e' \in r(e)\}|| \mid e' \in F\}$ (Congestion)
Embeddings

**Definition**

Let $G = (V, E)$ and $H = (W, F)$ be graphs. An embedding (embedding-function) from $G$ into $H$ is: $f : V \mapsto W$.

We use for embeddings the following cost-functions:

- $|W|/|V|$ (Expansion)
- $\max_{w \in W} |\{v \mid f(v) = w\}|$ (Load)
- $\max\{\text{dist}_H(f(a), f(b)) \mid \{a, b\} \in E\}$ (Dilation)

**Definition**

A routing for an embedding $f : V \mapsto W$ is a function: $r : E \mapsto \{\text{Paths in } H\}$ with: $r(\{a, b\})$ is a path from $f(a)$ to $f(b)$.

Note the cost-functions:

- $\max\{|r(\{a, b\})| \mid \{a, b\} \in E\}$ (Dilation)
- $\max\{|\{e \mid e \in E, e' \in r(e)\}| \mid e' \in F\}$ (Congestion)
Embeddings

**Definition**

Let \( G = (V, E) \) and \( H = (W, F) \) be graphs.

An embedding (embedding-function) from \( G \) into \( H \) is: \( f : V \mapsto W \).

We use for embeddings the following cost-functions:

- \( |W|/|V| \) (Expansion)
- \( \max_{w \in W} |\{v \mid f(v) = w\}| \) (Load)
- \( \max\{\text{dist}_H(f(a), f(b)) \mid \{a, b\} \in E\} \) (Dilation)

**Definition**

A routing for an embedding \( f : V \mapsto W \) is a function:

\( r : E \mapsto \{\text{Paths in } H\} \) with: \( r(\{a, b\}) \) is a path from \( f(a) \) to \( f(b) \).

Note the cost-functions:

- \( \max\{|r(\{a, b\})| \mid \{a, b\} \in E\} \) (Dilation)
- \( \max\{|\{e \mid e \in E, e' \in r(e)\}| \mid e' \in F\} \) (Congestion)
Embeddings

Definition

Let \( G = (V, E) \) and \( H = (W, F) \) be graphs.
An embedding (embedding-function) from \( G \) into \( H \) is: \( f : V \mapsto W \).
We use for embeddings the following cost-functions:

- \( \frac{|W|}{|V|} \) (Expansion)
- \( \max_{w \in W} |\{v \mid f(v) = w\}| \) (Load)
- \( \max \{ \text{dist}_H(f(a), f(b)) \mid \{a, b\} \in E \} \) (Dilation)

Definition

A routing for an embedding \( f : V \mapsto W \) is a function:
\( r : E \mapsto \{\text{Paths in } H\} \) with: \( r(\{a, b\}) \) is a path from \( f(a) \) to \( f(b) \).
Note the cost-functions:

- \( \max \{|r(\{a, b\})| \mid \{a, b\} \in E \} \) (Dilation)
- \( \max \{|\{e \mid e \in E, e' \in r(e)\}| \mid e' \in F\} \) (Congestion)
Example

- Load:
- Dilation:
- Congestion:
Example

- Load:
- Dilation:
- Congestion:
Example

- Load: 1
- Dilation:
- Congestion:
Example

- Load: 1
- Dilation: 5
- Congestion:
Example

- Load: 1
- Dilation: 5
- Congestion: 2
Example

- Load:
- Dilation:
- Congestion:
Example

- Load: 2
- Dilation:
- Congestion:
Example

- Load: 2
- Dilation: 1
- Congestion:
Example

- Load: 2
- Dilation: 1
- Congestion: 2
Example

- Load:
- Dilation:
- Congestion:
Example

- Load: 1
- Dilation:
- Congestion:
Example

- Load: 1
- Dilation: 1
- Congestion:
Example

- Load: 1
- Dilation: 1
- Congestion: 1
Iterated Embeddings

Let $G_i = (V_i, E_i)$ be graphs for $i \in \{1, 2, 3\}$

- Let $G_1$ in $G_2$ with dilation $d$, load $l$ and congestion $c$ embeddable.
- Let $G_2$ in $G_3$ with dilation $d'$, load $l'$ and congestion $c'$ embeddable.
- Then is $G_1$ in $G_3$ embeddable with:
  - Dilation $d \cdot d'$,
  - Load $l \cdot l'$ and
  - Congestion $c \cdot c'$.

Proof obvious.
Iterated Embeddings

Let $G_i = (V_i, E_i)$ be graphs for $i \in \{1, 2, 3\}$

- Let $G_1$ in $G_2$ with dilation $d$, load $l$ and congestion $c$ embeddable.
- Let $G_2$ in $G_3$ with dilation $d'$, load $l'$ and congestion $c'$ embeddable.
- Then is $G_1$ in $G_3$ embeddable with:
  - Dilation $d \cdot d'$,
  - Load $l \cdot l'$ and
  - Congestion $c \cdot c'$.

Proof obvious.
Iterated Embeddings

Let $G_i = (V_i, E_i)$ be graphs for $i \in \{1, 2, 3\}$

- Let $G_1$ in $G_2$ with dilation $d$, load $l$ and congestion $c$ embeddable.
- Let $G_2$ in $G_3$ with dilation $d'$, load $l'$ and congestion $c'$ embeddable.
- Then is $G_1$ in $G_3$ embeddable with:
  - Dilation $d \cdot d'$,
  - Load $l \cdot l'$ and
  - Congestion $c \cdot c'$.

Proof obvious.
Iterated Embeddings

Let $G_i = (V_i, E_i)$ be graphs for $i \in \{1, 2, 3\}$

- Let $G_1$ in $G_2$ with dilation $d$, load $l$ and congestion $c$ embeddable.
- Let $G_2$ in $G_3$ with dilation $d'$, load $l'$ and congestion $c'$ embeddable.
- Then is $G_1$ in $G_3$ embeddable with:
  - Dilation $d \cdot d'$,
  - Load $l \cdot l'$ and
  - Congestion $c \cdot c'$.

Proof obvious.
Iterated Embeddings

Let $G_i = (V_i, E_i)$ be graphs for $i \in \{1, 2, 3\}$

- Let $G_1$ in $G_2$ with dilation $d$, load $l$ and congestion $c$ embeddable.
- Let $G_2$ in $G_3$ with dilation $d'$, load $l'$ and congestion $c'$ embeddable.
- Then is $G_1$ in $G_3$ embeddable with:
  - Dilation $d \cdot d'$,
  - Load $l \cdot l'$ and
  - Congestion $c \cdot c'$.

Proof obvious.
Iterated Embeddings

Let $G_i = (V_i, E_i)$ be graphs for $i \in \{1, 2, 3\}$

- Let $G_1$ in $G_2$ with dilation $d$, load $l$ and congestion $c$ embeddable.
- Let $G_2$ in $G_3$ with dilation $d'$, load $l'$ and congestion $c'$ embeddable.
- Then is $G_1$ in $G_3$ embeddable with:
  - Dilation $d \cdot d'$,
  - Load $l \cdot l'$ and
  - Congestion $c \cdot c'$.

Proof obvious.
Iterated Embeddings

Let \( G_i = (V_i, E_i) \) be graphs for \( i \in \{1, 2, 3\} \)

- Let \( G_1 \) in \( G_2 \) with dilation \( d \), load \( l \) and congestion \( c \) embeddable.
- Let \( G_2 \) in \( G_3 \) with dilation \( d' \), load \( l' \) and congestion \( c' \) embeddable.
- Then is \( G_1 \) in \( G_3 \) embeddable with:
  - Dilation \( d \cdot d' \),
  - Load \( l \cdot l' \) and
  - Congestion \( c \cdot c' \).

Proof obvious.
Iterated Embeddings

Let $G_i = (V_i, E_i)$ be graphs for $i \in \{1, 2, 3\}$

- Let $G_1$ in $G_2$ with dilation $d$, load $l$ and congestion $c$ embeddable.
- Let $G_2$ in $G_3$ with dilation $d'$, load $l'$ and congestion $c'$ embeddable.
- Then is $G_1$ in $G_3$ embeddable with:
  - Dilation $d \cdot d'$,
  - Load $l \cdot l'$ and
  - Congestion $c \cdot c'$.

Proof obvious.
**Definition (Embedding-Problem)**

Given: $G$, $H$ graphs and $d$, $c$, $l \in \mathbb{N}$. Questions: Could $G$ be embedded into $H$ with dilation $d$, load $l$ and congestion $c$.

**Theorem**

*The embedding-problem is in $NP$.*

**Proof:**

- Let $d = c = l = 1$.
- Choose $G$ to be a cycle (or path) of length $|V(H)|$.
- We will investigate in the following some special networks.
  - pathes, cycles, grids, ...
  - trees and extended trees, ...
  - hyper-cubes and related structures, ...
Definition (Embedding-Problem)

Given: \( G, H \) graphs and \( d, c, l \in \mathbb{N} \). Questions: Could \( G \) be embedded into \( H \) with dilation \( d \), load \( l \) and congestion \( c \).

Theorem

The embedding-problem is in \( NP \).

Proof:

- Let \( d = c = l = 1 \).
- Choose \( G \) to be a cycle (or path) of length \( |V(H)| \).
- We will investigate in the following some special networks.
  - pathes, cycles, grids, ...
  - trees and extended trees, ...
  - hyper-cubes and related structures, ...
Definition (Embedding-Problem)

Given: $G, H$ graphs and $d, c, l \in \mathbb{N}$. Questions: Could $G$ be embedded into $H$ with dilation $d$, load $l$ and congestion $c$.

Theorem

*The embedding-problem is in $\mathcal{NPC}$.*

Proof:

- Let $d = c = l = 1$.
- Choose $G$ to be a cycle (or path) of length $|V(H)|$.
- We will investigate in the following some special networks.
  - pathes, cycles, grids, ...
  - trees and extended trees, ...
  - hyper-cubes and related structures, ...
**Definition (Embedding-Problem)**

Given: $G, H$ graphs and $d, c, l \in \mathbb{N}$. Questions: Could $G$ be embedded into $H$ with dilation $d$, load $l$ and congestion $c$.

**Theorem**

*The embedding-problem is in $NP$.*

**Proof:**

- Let $d = c = l = 1$.
- Choose $G$ to be a cycle (or path) of length $|V(H)|$.
- We will investigate in the following some special networks. 
  - pathes, cycles, grids, ...
  - trees and extended trees, ...
  - hyper-cubes and related structures, ...

---

**Motivation**

- Paths, Cycles in ...
- Trees in ...
- Cube-like Netw...
- Embeddings
- Optical Networks

6:4 Motivation 4/10

Definition (Embedding-Problem)

Given: \( G, H \) graphs and \( d, c, l \in \mathbb{N} \). Questions: Could \( G \) be embedded into \( H \) with dilation \( d \), load \( l \) and congestion \( c \).

Theorem

*The embedding-problem is in \( \mathcal{NPC} \).*

Proof:

- Let \( d = c = l = 1 \).
- Choose \( G \) to be a cycle (or path) of length \( |V(H)| \).
- We will investigate in the following some special networks.
  - pathes, cycles, grids, ...
  - trees and extended trees, ...
  - hyper-cubes and related structures, ...
Motivation

Definition (Embedding-Problem)

Given: $G, H$ graphs and $d, c, l \in \mathbb{N}$. Questions: Could $G$ be embedded into $H$ with dilation $d$, load $l$ and congestion $c$.

Theorem

*The embedding-problem is in $\mathcal{NP}$.*

Proof:

- Let $d = c = l = 1$.
- Choose $G$ to be a cycle (or path) of length $|V(H)|$.
- We will investigate in the following some special networks.
- pathes, cycles, grids, ...
- trees and extended trees, ...
- hyper-cubes and related structures, ...
Definition (Embedding-Problem)

Given: $G, H$ graphs and $d, c, l \in \mathbb{N}$. Questions: Could $G$ be embedded into $H$ with dilation $d$, load $l$ and congestion $c$.

Theorem

*The embedding-problem is in $NP$.*

Proof:

- Let $d = c = l = 1$.
- Choose $G$ to be a cycle (or path) of length $|V(H)|$.

We will investigate in the following some special networks.

- paths, cycles, grids, ...
- trees and extended trees, ...
- hyper-cubes and related structures, ...
Definition (Embedding-Problem)

Given: $G, H$ graphs and $d, c, l \in \mathbb{N}$. Questions: Could $G$ be embedded into $H$ with dilation $d$, load $l$ and congestion $c$.

Theorem

*The embedding-problem is in $\mathcal{NPC}$.*

Proof:

- Let $d = c = l = 1$.
- Choose $G$ to be a cycle (or path) of length $|V(H)|$.
- We will investigate in the following some special networks.
  - pathes, cycles, grids, ...
  - trees and extended trees, ...
  - hyper-cubes and related structures, ...
Definition (Embedding-Problem)

Given: $G, H$ graphs and $d, c, l \in \mathbb{N}$. Questions: Could $G$ be embedded into $H$ with dilation $d$, load $l$ and congestion $c$.

Theorem

*The embedding-problem is in $\mathcal{NPC}$.*

Proof:

- Let $d = c = l = 1$.
- Choose $G$ to be a cycle (or path) of length $|V(H)|$.
- We will investigate in the following some special networks.
  - pathes, cycles, grids, ...
  - trees and extended trees, ...
  - hyper-cubes and related structures, ...
Definition (Embedding-Problem)

Given: $G, H$ graphs and $d, c, l \in \mathbb{N}$. Questions: Could $G$ be embedded into $H$ with dilation $d$, load $l$ and congestion $c$.

Theorem

*The embedding-problem is in $\mathcal{NP}$.*

Proof:

- Let $d = c = l = 1$.
- Choose $G$ to be a cycle (or path) of length $|V(H)|$.
- We will investigate in the following some special networks.
  - pathes, cycles, grids, ...
  - trees and extended trees, ...
  - hyper-cubes and related structures, ...
Properties of the Networks to be considered

- **Number of nodes.**
- Number of edges.
- Degree.
- Length of paths in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. the shortest of all longest paths.
- Connectivity, i.e. is there a bottle-neck.
  - Node-connectivity
  - Edge-connectivity
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
  - May be the graph is based on some group-structure.
  - How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.
- Length of paths in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. the shortest of all longest paths.
- Connectivity, i.e. is there a bottle-neck.
  - Node-connectivity
  - Edge-connectivity
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
  - May be the graph is based on some group-structure.
  - How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- **Degree**.
  - Length of paths in the network:
    - Diameter, i.e. the longest of all shortest paths.
    - Radius, i.e. the shortest of all longest paths.
- Connectivity, i.e. is there a bottleneck.
  - Node-connectivity
  - Edge-connectivity
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
- May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.
- **Length of paths in the network:**
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. the shortest of all longest paths.
- Connectivity, i.e. is there a bottle-neck.
  - Node-connectivity
  - Edge-connectivity
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
- May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.
- Length of pathes in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. the shortest of all longest paths.
- Connectivity, i.e. is there a bottle-neck.
  - Node-connectivity
  - Edge-connectivity
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
- May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.

Length of pathes in the network:
- Diameter, i.e. the longest of all shortest paths.
- Radius, i.e. the shortest of all longest paths.

Connectivity, i.e. is there a bottle-neck.
- Node-connectivity
- Edge-connectivity

Regularity,
- May be all nodes look ‘similar’.
- May be all edges look ‘similar’.

Easy routing
- May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.
- Length of paths in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. the shortest of all longest paths.
- Connectivity, i.e. is there a bottle-neck.
  - Node-connectivity
  - Edge-connectivity
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
- May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.

Length of paths in the network:
- Diameter, i.e. the longest of all shortest paths.
- Radius, i.e. the shortest of all longest paths.

Connectivity, i.e. is there a bottle-neck.
- Node-connectivity
- Edge-connectivity

Regularity,
- May be all nodes look ‘similar’.
- May be all edges look ‘similar’.

Easy routing
- May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.
- Length of paths in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. the shortest of all longest paths.
- Connectivity, i.e. is there a bottle-neck.
  - Node-connectivity
  - Edge-connectivity
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
- May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.
- Length of paths in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. the shortest of all longest paths.
- Connectivity, i.e. is there a bottleneck.
  - Node-connectivity
  - Edge-connectivity
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
  - May be the graph is based on some group-structure.
  - How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.
- Length of paths in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. the shortest of all longest paths.
- Connectivity, i.e. is there a bottle-neck.
  - Node-connectivity
  - Edge-connectivity
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
  - May be the graph is based on some group-structure.
  - How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.
- Length of paths in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. the shortest of all longest paths.
- Connectivity, i.e. is there a bottle-neck.
  - Node-connectivity
  - Edge-connectivity
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
- May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.
- Length of paths in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. the shortest of all longest paths.
- Connectivity, i.e. is there a bottle-neck.
  - Node-connectivity
  - Edge-connectivity
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
  - May be the graph is based on some group-structure.
  - How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.
- Length of paths in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. the shortest of all longest paths.
- Connectivity, i.e. is there a bottle-neck.
  - Node-connectivity
  - Edge-connectivity
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
  - May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
Properties of the Networks to be considered

- Number of nodes.
- Number of edges.
- Degree.
- Length of paths in the network:
  - Diameter, i.e. the longest of all shortest paths.
  - Radius, i.e. the shortest of all longest paths.
- Connectivity, i.e. is there a bottleneck.
  - Node-connectivity
  - Edge-connectivity
- Regularity,
  - May be all nodes look ‘similar’.
  - May be all edges look ‘similar’.
- Easy routing
  - May be the graph is based on some group-structure.
- How many graphs are in some family of networks?
Paths and cycles with $n$ nodes

- **Path:**
  \[
  L(n) = (V_{L(n)}, E_{L(n)})
  \]
  \[
  V_{L(n)} = \{0, 1, 2, \ldots, n - 1\}
  \]
  \[
  E_{L(n)} = \{\{i, i + 1\} \mid 0 \leq i < n - 1\}
  \]

- **Cycle:**
  \[
  C(n) = (V_{C(n)}, E_{C(n)})
  \]
  \[
  V_{C(n)} = \{0, 1, 2, \ldots, n - 1\}
  \]
  \[
  E_{C(n)} = \{\{i, (i + 1) \text{ mod } n\} \mid 0 \leq i < n\}
  \]
**Paths and cycles with \( n \) nodes**

- **Path:**
  \[
  L(n) = (V_{L(n)}, E_{L(n)})
  \]
  \[
  V_{L(n)} = \{0, 1, 2, \cdots, n - 1\}
  \]
  \[
  E_{L(n)} = \{\{i, i + 1\} \mid 0 \leq i < n - 1\}
  \]

  Number of nodes: \( n \)  
  Degrees: \( \{1, 2\} \)  
  Number of edges: \( n - 1 \)  
  Diameter: \( n - 1 \)  
  Node-con.: 1  
  Edge-con.: 1

- **Cycle:**
  \[
  C(n) = (V_{C(n)}, E_{C(n)})
  \]
  \[
  V_{C(n)} = \{0, 1, 2, \cdots, n - 1\}
  \]
  \[
  E_{C(n)} = \{\{i, (i + 1) \mod n\} \mid 0 \leq i < n\}
  \]
Paths and cycles with $n$ nodes

**Path:**
\[ L(n) = (V_{L(n)}, E_{L(n)}) \]
\[ V_{L(n)} = \{0, 1, 2, \ldots, n - 1\} \]
\[ E_{L(n)} = \{\{i, i + 1\} \mid 0 \leq i < n - 1\} \]

- Number of nodes: $n$
- Degrees: \{1, 2\}
- Number of edges: $n - 1$
- Diameter: $n - 1$
- Node-con.: 1
- Edge-con.: 1

$L(8)$:

**Cycle:**
\[ C(n) = (V_{C(n)}, E_{C(n)}) \]
\[ V_{C(n)} = \{0, 1, 2, \ldots, n - 1\} \]
\[ E_{C(n)} = \{\{i, (i + 1) \mod n\} \mid 0 \leq i < n\} \]

- Number of nodes: $n$
- Degrees: 2
- Number of edges: $n$
- Diameter: $\lfloor n / 2 \rfloor$
- Node-con.: 2
- Edge-con.: 2

$C(8)$:
Paths and cycles with \( n \) nodes

- **Path:**
  \[
  L(n) = (V_{L(n)}, E_{L(n)})
  \]
  \[
  V_{L(n)} = \{0, 1, 2, \ldots, n - 1\}
  \]
  \[
  E_{L(n)} = \{\{i, i + 1\} \mid 0 \leq i < n - 1\}
  \]
  Number of nodes: \( n \)
  Number of edges: \( n - 1 \)
  Degrees: \( \{1, 2\} \)
  Diameter: \( n - 1 \)
  Node-con.: 1
  Edge-con.: 1

- **Cycle:**
  \[
  C(n) = (V_{C(n)}, E_{C(n)})
  \]
  \[
  V_{C(n)} = \{0, 1, 2, \ldots, n - 1\}
  \]
  \[
  E_{C(n)} = \{\{i, (i + 1) \mod n\} \mid 0 \leq i < n\}
  \]
  Number of nodes: \( n \)
  Number of edges: \( n \)
  Degree: 2
  Diameter: \( \lfloor n/2 \rfloor \)
  Node-con.: 2
  Edge-con.: 2
Product of Graphs

**Definition:**

Let $G = (V, E)$ and $G' = (V', E')$ be graphs. With $G \times G'$ we denote the product of $G$ and $G'$:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) | a' = b' \land (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) | a = b \land (a', b') \in E'\}$.

Example $L(10) \times C(4)$:
Definition:

Let $G = (V, E)$ and $G' = (V', E')$ be graphs. With $G \times G'$ we denote the product of $G$ and $G'$:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{(a, a'), (b, b') | a' = b' \land (a, b) \in E\}$.
- $E_2 = \{(a, a'), (b, b') | a = b \land (a', b') \in E'\}$.

Example $L(10) \times C(4)$:
Definition:
Let $G = (V, E)$ and $G' = (V', E')$ be graphs. With $G \times G'$ we denote the product of $G$ and $G'$:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) | a' = b' \land (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) | a = b \land (a', b') \in E'\}$.

Example $L(10) \times C(4)$:
Product of Graphs

Definition:

Let \( G = (V, E) \) and \( G' = (V', E') \) be graphs. With \( G \times G' \) we denote the product of \( G \) and \( G' \):

- \( G \times G' = (V \times V', E_1 \cup E_2) \).
- \( E_1 = \{((a, a'), (b, b')) \mid a' = b' \land (a, b) \in E\} \).
- \( E_2 = \{((a, a'), (b, b')) \mid a = b \land (a', b') \in E'\} \).

Example \( L(10) \times C(4) \):
**Definition:**

Let $G = (V, E)$ and $G' = (V', E')$ be graphs. With $G \times G'$ we denote the product of $G$ and $G'$:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) \mid a' = b' \land (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) \mid a = b \land (a', b') \in E'\}$.

**Example $L(10) \times C(4)$:**

![Diagram](attachment://example.png)
Definition:

Let $G = (V, E)$ and $G' = (V', E')$ be graphs. With $G \times G'$ we denote the product of $G$ and $G'$:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) \mid a' = b' \land (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) \mid a = b \land (a', b') \in E'\}$.

Example $L(10) \times C(4)$:
Definition:

Let $G = (V, E)$ and $G' = (V', E')$ be graphs. With $G \times G'$ we denote the product of $G$ and $G'$:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) | a' = b' \land (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) | a = b \land (a', b') \in E'\}$.

Example $L(10) \times C(4)$:
**Definition:**

Let $G = (V, E)$ and $G' = (V', E')$ be graphs. With $G \times G'$ we denote the product of $G$ and $G'$:

- $G \times G' = (V \times V', E_1 \cup E_2)$.
- $E_1 = \{((a, a'), (b, b')) | a' = b' \land (a, b) \in E\}$.
- $E_2 = \{((a, a'), (b, b')) | a = b \land (a', b') \in E'\}$.

**Example** $L(10) \times C(4)$:
Grid of dimension $d$

- **Grids:** $G(n_1, n_2, \cdots, n_d) = L(n_1) \times L(n_2) \times \cdots \times L(n_d)$ with $n_i > 1$

---

**Grid:** $G(14, 4)$:

```
0,0 1,0 2,0 3,0 4,0 5,0 6,0 7,0 8,0 9,0 10,0 11,0 12,0 13,0
0,1 1,1 2,1 3,1 4,1 5,1 6,1 7,1 8,1 9,1 10,1 11,1 12,1 13,1
0,2 1,2 2,2 3,2 4,2 5,2 6,2 7,2 8,2 9,2 10,2 11,2 12,2 13,2
0,3 1,3 2,3 3,3 4,3 5,3 6,3 7,3 8,3 9,3 10,3 11,3 12,3 13,3
```
Grid of dimension $d$

- **Grids:** $G(n_1, n_2, \ldots, n_d) = L(n_1) \times L(n_2) \times \cdots \times L(N_d)$ with $n_i > 1$
  
  - Number of nodes: $\prod_{i=1}^{d} n_i$
  - Degrees: $\{d, \ldots, 2 \cdot d\}$
  
  - Number of edges: $\sum_{i=1}^{d} (n_i - 1) \prod_{j=1, j \neq i}^{d} n_j$
  - Diameter: $\sum_{i=1}^{d} (n_i - 1)$
  
  - Node-con.: $d$
  - Edge-con.: $d$

- **Grid:** $G(14, 4)$:

```
  0,3 1,3 2,3 3,3 4,3 5,3 6,3 7,3 8,3 9,3 10,3 11,3 12,3 13,3
  0,2 1,2 2,2 3,2 4,2 5,2 6,2 7,2 8,2 9,2 10,2 11,2 12,2 13,2
  0,1 1,1 2,1 3,1 4,1 5,1 6,1 7,1 8,1 9,1 10,1 11,1 12,1 13,1
  0,0 1,0 2,0 3,0 4,0 5,0 6,0 7,0 8,0 9,0 10,0 11,0 12,0 13,0
```
Torus of dimension \( d \)

- Torus: \( Tr(n_1, n_2, \cdots, n_d) = C(n_1) \times C(n_2) \times \cdots \times C(N_d) \) with \( n_i > 1 \)

- Torus: \( Tr(14, 4) \):

![Torus Diagram]

- Torus of dimension \( d \)

- Torus: \( Tr(n_1, n_2, \cdots, n_d) = C(n_1) \times C(n_2) \times \cdots \times C(N_d) \) with \( n_i > 1 \)

- Torus: \( Tr(14, 4) \):

![Torus Diagram]

- Torus of dimension \( d \)

- Torus: \( Tr(n_1, n_2, \cdots, n_d) = C(n_1) \times C(n_2) \times \cdots \times C(N_d) \) with \( n_i > 1 \)

- Torus: \( Tr(14, 4) \):

![Torus Diagram]
Torus of dimension \(d\)

- Torus: \(Tr(n_1, n_2, \ldots, n_d) = C(n_1) \times C(n_2) \times \cdots \times C(N_d)\) with \(n_i > 1\)
  - Number of nodes: \(\prod_{i=1}^d n_i\)
  - Degree: \(2 \cdot d\)
  - Number of edges: \(\prod_{i=1}^d n_i\)
  - Diameter: \(\sum_{i=1}^d \lfloor n_i/2 \rfloor\)
  - Node-con.: \(2 \cdot d\)
  - Edge-con.: \(2 \cdot d\)

- Torus: \(Tr(14, 4)\):

```
0,0 | 0,1 | 0,2 | 0,3 | 1,0 | 1,1 | 1,2 | 1,3 | 2,0 | 2,1 | 2,2 | 2,3 | 3,0 | 3,1 | 3,2 | 3,3
0,1 | 1,1 | 2,1 | 3,1 | 4,1 | 5,1 | 6,1 | 7,1 | 8,1 | 9,1 | 10,1 | 11,1 | 12,1 | 13,1
0,2 | 1,2 | 2,2 | 3,2 | 4,2 | 5,2 | 6,2 | 7,2 | 8,2 | 9,2 | 10,2 | 11,2 | 12,2 | 13,2
0,3 | 1,3 | 2,3 | 3,3 | 4,3 | 5,3 | 6,3 | 7,3 | 8,3 | 9,3 | 10,3 | 11,3 | 12,3 | 13,3
```
Complete binary tree

\[ T(d) = (V_{T(d)}, E_{T(d)}) \]
\[ V_{T(d)} = \{ w \in \{0, 1\}^* \mid |w| \leq d \} \]
\[ E_{T(d)} = \{ \{w, wa\} \mid w, wa \in V, a \in \{0, 1\} \} \]
Complete binary tree

\[ T(d) = (V_{T(d)}, E_{T(d)}) \]
\[ V_{T(d)} = \{ w \in \{0, 1\}^* \mid |w| \leq d \} \]
\[ E_{T(d)} = \{ \{w, wa\} \mid w, wa \in V, a \in \{0, 1\} \} \]
Complete binary tree

\[ T(d) = (V_{T(d)}, E_{T(d)}) \]
\[ V_{T(d)} = \{ w \in \{0, 1\}^* \mid |w| \leq d \} \]
\[ E_{T(d)} = \{ \{w, wa\} \mid w, wa \in V, a \in \{0, 1\} \} \]
Complete binary tree

\[ T(d) = (V_{T(d)}, E_{T(d)}) \]
\[ V_{T(d)} = \{ w \in \{0, 1\}^* | |w| \leq d \} \]
\[ E_{T(d)} = \{ \{w, wa\} | w, wa \in V, a \in \{0, 1\} \} \]
Complete binary tree

\[ T(d) = (V_{T(d)}, E_{T(d)}) \]
\[ V_{T(d)} = \{ w \in \{0, 1\}^* \mid |w| \leq d \} \]
\[ E_{T(d)} = \{ \{w, wa\} \mid w, wa \in V, a \in \{0, 1\} \} \]
Complete binary tree

\[ T(d) = (V_{T(d)}, E_{T(d)}) \]
\[ V_{T(d)} = \{ w \in \{0, 1\}^* \mid |w| \leq d \} \]
\[ E_{T(d)} = \{ \{w, wa\} \mid w, wa \in V, a \in \{0, 1\} \} \]
Complete binary tree

\[ T(d) = (V_T(d), E_T(d)) \]

\[ V_T(d) = \{ w \in \{0, 1\}^* | |w| \leq d \} \]

\[ E_T(d) = \{ \{w, wa\} | w, wa \in V, a \in \{0, 1\} \} \]

Number of nodes: \( 2^{d+1} - 1 \)
Degrees: \( \{1, 2, 3\} \)
Number of edges: \( 2^{d+1} - 2 \)
Diameter: \( 2 \cdot d \)
Node-con.: 1
Edge-con.: 1
Complete $k$-nary tree

$$T_k(d) = (V_{T_k(d)}, E_{T_k(d)})$$

$$V_{T_k(d)} = \{w \in \{0, 1, \ldots, k - 1\}^* \mid |w| \leq d\}$$

$$E_{T_k(d)} = \{\{w, wa\} \mid w, wa \in V_{T_k(d)}, a \in \{0, 1, \ldots, k - 1\}\}$$
Complete $k$-nary tree

\[
T_k(d) = (V_{T_k(d)}, E_{T_k(d)})
\]

\[
V_{T_k(d)} = \{w \in \{0, 1, \ldots, k - 1\}^* \mid |w| \leq d\}
\]

\[
E_{T_k(d)} = \{\{w, wa\} \mid w, wa \in V_{T_k(d)}, a \in \{0, 1, \ldots, k - 1\}\}
\]
Complete $k$-nary tree

\[
T_k(d) = (V_{T_k(d)}, E_{T_k(d)})
\]
\[
V_{T_k(d)} = \{ w \in \{0, 1, \cdots, k - 1 \}^* \mid |w| \leq d \}
\]
\[
E_{T_k(d)} = \{ \{w, wa\} \mid w, wa \in V_{T_k(d)}, a \in \{0, 1, \cdots, k - 1 \}\}
\]
Complete $k$-nary tree

$$T_k(d) = (V_{T_k(d)}, E_{T_k(d)})$$

$$V_{T_k(d)} = \{ w \in \{0, 1, \ldots, k-1\}^* \mid |w| \leq d \}$$

$$E_{T_k(d)} = \{ \{w, wa\} \mid w, wa \in V_{T_k(d)}, a \in \{0, 1, \ldots, k-1\} \}$$
Complete $k$-nary tree

$$T_k(d) = (V_{T_k(d)}, E_{T_k(d)})$$

$$V_{T_k(d)} = \{w \in \{0,1,\cdots,k-1\}^* \mid |w| \leq d\}$$

$$E_{T_k(d)} = \{\{w, wa\} \mid w, wa \in V_{T_k(d)}, a \in \{0,1,\cdots,k-1\}\}$$
Complete $k$-nary tree

$$T_k(d) = (V_{T_k(d)}, E_{T_k(d)})$$

$$V_{T_k(d)} = \{w \in \{0, 1, \ldots, k-1\}^* | |w| \leq d\}$$

$$E_{T_k(d)} = \{\{w, wa\} | w, wa \in V_{T_k(d)}, a \in \{0, 1, \ldots, k-1\}\}$$

Number of nodes: $\sum_{i=0}^{d} k^i$  
Degrees: $\{1, k, k+1\}$  
Number of edges: $\sum_{i=0}^{d} k^i - 1$  
Diameter: $2 \cdot d$  
Node-con.: 1  
Edge-con.: 1
X-Tree

\[ \begin{align*}
X_T(d) &= (V_{XT(d)}, E_{XT(d)}^1 \cup E_{XT(d)}^2) \\
V_{XT(d)} &= \{ w \in \{0, 1\}^* \mid |w| \leq d \} \\
E_{XT(d)}^1 &= \{ \{ w, wa \} \mid w, wa \in V, a \in \{0, 1\} \} \\
E_{XT(d)}^2 &= \{ \{ w, w' \} \mid w, w' \in V_{XT(d)}, |w| = |w'|, \text{int}(w) + 1 = \text{int}(w') \}
\end{align*} \]
X-Tree

\[ XT(d) = (V_{XT(d)}, E^1_{XT(d)} \cup E^2_{XT(d)}) \]

\[ V_{XT(d)} = \{ w \in \{0, 1\}^* | |w| \leq d \} \]

\[ E^1_{XT(d)} = \{ \{w, wa\} | w, wa \in V, a \in \{0, 1\} \} \]

\[ E^2_{XT(d)} = \{ \{w, w'\} | w, w' \in V_{XT(d)}, |w| = |w'|, \text{int}(w) + 1 = \text{int}(w') \} \]
X-Tree

\[ X_T(d) = (V_{X_T(d)}, E_{X_T(d)}^1 \cup E_{X_T(d)}^2) \]

\[ V_{X_T(d)} = \{ w \in \{0, 1\}^* \mid |w| \leq d \} \]

\[ E_{X_T(d)}^1 = \{\{w, wa\} \mid w, wa \in V, a \in \{0, 1\}\} \]

\[ E_{X_T(d)}^2 = \{\{w, w'\} \mid w, w' \in V_{X_T(d)}, |w| = |w'|, \text{int}(w) + 1 = \text{int}(w')\} \]
\[ XT(d) = (V_{XT(d)}, E^1_{XT(d)} \cup E^2_{XT(d)}) \]

\[ V_{XT(d)} = \{ w \in \{0, 1\}^* | |w| \leq d \} \]

\[ E^1_{XT(d)} = \{ \{ w, wa \} | w, wa \in V, a \in \{0, 1\} \} \]

\[ E^2_{XT(d)} = \{ \{ w, w' \} | w, w' \in V_{XT(d)}, |w| = |w'|, \text{int}(w) + 1 = \text{int}(w') \} \]
\[ \begin{align*}
X_T(d) &= (V_{XT(d)}; E^1_{XT(d)} \cup E^2_{XT(d)}) \\
V_{XT(d)} &= \{ w \in \{0,1\}^* \mid |w| \leq d \} \\
E^1_{XT(d)} &= \{ \{w, wa\} \mid w, wa \in V, a \in \{0,1\} \} \\
E^2_{XT(d)} &= \{ \{w, w'\} \mid w, w' \in V_{XT(d)}, |w| = |w'|, \text{int}(w) + 1 = \text{int}(w') \}
\end{align*} \]
X-Tree

\[
X_T(d) = (V_{X_T(d)}, E_{X_T(d)}^1 \cup E_{X_T(d)}^2)
\]

\[
V_{X_T(d)} = \{ w \in \{0, 1\}^* \mid |w| \leq d \}
\]

\[
E_{X_T(d)}^1 = \{ \{w, wa\} \mid w, wa \in V, a \in \{0, 1\} \}
\]

\[
E_{X_T(d)}^2 = \{ \{w, w'\} \mid w, w' \in V_{X_T(d)}, |w| = |w'|, \text{int}(w) + 1 = \text{int}(w') \}
\]
X-Tree

\[ X_T(d) = (V_{XT(d)}, E^1_{XT(d)} \cup E^2_{XT(d)}) \]
\[ V_{XT(d)} = \{ w \in \{0, 1\}^* | |w| \leq d \} \]
\[ E^1_{XT(d)} = \{ \{w, wa\} | w, wa \in V, a \in \{0, 1\} \} \]
\[ E^2_{XT(d)} = \{ \{w, w'\} | w, w' \in V_{XT(d)}, |w| = |w'|, \text{int}(w) + 1 = \text{int}(w') \} \]

Number of nodes: \(2^{d+1} - 1\)  
Degrees: \(\{2, 3, 4, 5\}\)  
Number of edges: \(2^{d+2} - 4 - d\)  
Diameter: \(2 \cdot d - 1\)  
Node-con.: 2  
Edge-con.: 2
Hypercube of dimension $d$

$$\begin{align*}
HQ(d) & = (V_{HQ(d)}, E_{HQ(d)}) \\
V_{HQ(d)} & = \{0, 1\}^d \\
E_{HQ(d)} & = \{\{w0', w1'\} \mid w0', w1' \in V_{HQ(d)}\}
\end{align*}$$
Hypercube of dimension $d$

$$HQ(d) = (V_{HQ(d)}, E_{HQ(d)})$$

$$V_{HQ(d)} = \{0, 1\}^d$$

$$E_{HQ(d)} = \{\{w0, w1\} \mid w0, w1 \in V_{HQ(d)}\}$$
Hypercube of dimension $d$

\[ HQ(d) = (V_{HQ(d)}, E_{HQ(d)}) \]
\[ V_{HQ(d)} = \{0, 1\}^d \]
\[ E_{HQ(d)} = \{\{w0, w1\}' | w0, w1, w1' \in V_{HQ(d)}\} \]
Hypercube of dimension $d$

\[ HQ(d) = (V_{HQ(d)}, E_{HQ(d)}) \]
\[ V_{HQ(d)} = \{0,1\}^d \]
\[ E_{HQ(d)} = \\{\{w0w', w1w'\} \mid w0w', w1w' \in V_{HQ(d)}\} \]
Hypercube of dimension $d$

\[ HQ(d) = (V_{HQ(d)}, E_{HQ(d)}) \]
\[ V_{HQ(d)} = \{0, 1\}^d \]
\[ E_{HQ(d)} = \{\{w0', w1'\} | w0', w1' \in V_{HQ(d)}\} \]
Hypercube of dimension $d$

\[
HQ(d) = (V_{HQ(d)}, E_{HQ(d)}) \\
V_{HQ(d)} = \{0, 1\}^d \\
E_{HQ(d)} = \\{\{w0w', w1w'\} | w0w', w1w' \in V_{HQ(d)}\}
\]
Hypercube of dimension $d$

$$HQ(d) = \left( V_{HQ(d)}, E_{HQ(d)} \right)$$

$$V_{HQ(d)} = \{0, 1\}^d$$

$$E_{HQ(d)} = \{ \{w0w', w1w'\} | w0w', w1w' \in V_{HQ(d)} \}$$

Note the Gray-Code.
Hypercube of dimension $d$

$$HQ(d) = (V_{HQ(d)}, E_{HQ(d)})$$

$$V_{HQ(d)} = \{0, 1\}^d$$

$$E_{HQ(d)} = \{\{w0w', w1w'\} | w0w', w1w' \in V_{HQ(d)}\}$$

Number of nodes: $2^d$  
Degree: $d$  
Node-con.: $d$  
Number of edges: $d \cdot 2^{d-1}$  
Diameter: $d$  
Edge-con.: $d$
Hypercube of dimension $d$ (alternative view)

\[
\begin{align*}
HQ(d) &= (V_{HQ(d)}, E_{HQ(d)}) \\
V_{HQ(d)} &= \{0, 1\}^d \\
E_{HQ(d)} &= \{\{w0w', w1w'\} \mid w0w', w1w' \in V_{HQ(d)}\}
\end{align*}
\]
Hypercube of dimension \(d\) (alternative view)

\[
HQ(d) = (V_{HQ(d)}, E_{HQ(d)})
\]

\[
V_{HQ(d)} = \{0, 1\}^d
\]

\[
E_{HQ(d)} = \{\{w0w', w1w'\} | w0w', w1w' \in V_{HQ(d)}\}
\]
Hypercube of dimension $d$ (alternative view)

$$HQ(d) = (V_{HQ(d)}, E_{HQ(d)})$$

$$V_{HQ(d)} = \{0, 1\}^d$$

$$E_{HQ(d)} = \{\{w0w', w1w'\} \mid w0w', w1w' \in V_{HQ(d)}\}$$
Hypercube of dimension $d$ (alternative view)

$$
HQ(d) = (V_{HQ(d)}, E_{HQ(d)})
$$

$$
V_{HQ(d)} = \{0, 1\}^d
$$

$$
E_{HQ(d)} = \{\{w0', w1'\} | w0', w1' \in V_{HQ(d)}\}
$$
Hypercube of dimension $d$ (alternative view)

\[
HQ(d) = (V_{HQ(d)}, E_{HQ(d)})
\]

\[
V_{HQ(d)} = \{0, 1\}^d
\]

\[
E_{HQ(d)} = \{\{w0w', w1w'\} | w0w', w1w' \in V_{HQ(d)}\}
\]
Hypercube of dimension $d$ (alternative view)

$$HQ(d) = (V_{HQ(d)}, E_{HQ(d)})$$

$$V_{HQ(d)} = \{0, 1\}^d$$

$$E_{HQ(d)} = \{\{w0w', w1w'\} \mid w0w', w1w' \in V_{HQ(d)}\}$$
Hypercube of dimension $d$ (alternative view)

$$HQ(d) = (V_{HQ(d)}, E_{HQ(d)})$$

$$V_{HQ(d)} = \{0, 1\}^d$$

$$E_{HQ(d)} = \{\{w0', w1'\} \mid w0', w1' \in V_{HQ(d)}\}$$
Hypercube of dimension $d$ (alternative view)

$$HQ(d) = (V_{HQ(d)}, E_{HQ(d)})$$

$$V_{HQ(d)} = \{0, 1\}^d$$

$$E_{HQ(d)} = \{\{w0w', w1w'\} | w0w', w1w' \in V_{HQ(d)}\}$$
Hypercube of dimension $d$ (alternative view)

$$HQ(d) = (V_{HQ(d)}, E_{HQ(d)})$$
$$V_{HQ(d)} = \{0, 1\}^d$$
$$E_{HQ(d)} = \{\{w0w', w1w'\} \mid w0w', w1w' \in V_{HQ(d)}\}$$
Hypercube of dimension $d$ (alternative view)

\[ HQ(d) = (V_{HQ(d)}, E_{HQ(d)}) \]
\[ V_{HQ(d)} = \{0, 1\}^d \]
\[ E_{HQ(d)} = \{\{w0w', w1w'\} | w0w', w1w' \in V_{HQ(d)}\} \]
Hypercube of dimension $d$ (alternative view)

$$HQ(d) = (V_{HQ(d)}, E_{HQ(d)})$$

$$V_{HQ(d)} = \{0, 1\}^d$$

$$E_{HQ(d)} = \{\{w0w', w1w'\} \mid w0w', w1w' \in V_{HQ(d)}\}$$
Cube-Connected Cycles of dimension $d$

\[
\begin{align*}
CC(d) & = (V_{\text{CCC}(d)}, E_{\text{CCC}(d)}^c \cup E_{\text{CCC}(d)}^h) \\
V_{\text{CCC}(d)} & = \{0, 1, \cdots, d - 1\} \times \{0, 1\}^d \\
E_{\text{CCC}(d)}^c & = \{((i, w), ((i + 1) \mod n, w)) \mid w \in \{0, 1\}^d, 0 \leq i < n\} \\
E_{\text{CCC}(d)}^h & = \{((i, w0w'), (i, w1w')) \mid w' \in \{0, 1\}^i, w \in \{0, 1\}^{n - i - 1}\}
\end{align*}
\]
Cube-Connected Cycles of dimension $d$

\[
CCC(d) = (V_{CCC(d)}, E^c_{CCC(d)} \cup E^h_{CCC(d)})
\]

\[
V_{CCC(d)} = \{0, 1, \cdots, d-1\} \times \{0, 1\}^d
\]

\[
E^c_{CCC(d)} = \{(i, w), ((i + 1) \mod n, w)\} \mid w \in \{0, 1\}^d, 0 \leq i < n
\]

\[
E^h_{CCC(d)} = \{(i, w0w'), (i, w1w')\} \mid w' \in \{0, 1\}^i, w \in \{0, 1\}^{n-i-1}
\]
Cube-Connected Cycles of dimension \(d\)

\[
CCC(d) = (V_{CCC(d)}, E^c_{CCC(d)} \cup E^h_{CCC(d)})
\]

\[
V_{CCC(d)} = \{0, 1, \cdots, d - 1\} \times \{0, 1\}^d
\]

\[
E^c_{CCC(d)} = \{(i, w), ((i + 1) \mod n, w)\} \mid w \in \{0, 1\}^d, 0 \leq i < n
\]

\[
E^h_{CCC(d)} = \{(i, w0w'), (i, w1w')\} \mid w' \in \{0, 1\}^i, w \in \{0, 1\}^n-i-1
\]
Cube-Connected Cycles of dimension $d$

$$
\begin{align*}
CCC(d) &= (V_{CCC(d)}, E^c_{CCC(d)} \cup E^h_{CCC(d)}) \\
V_{CCC(d)} &= \{0, 1, \cdots, d-1\} \times \{0, 1\}^d \\
E^c_{CCC(d)} &= \{(i, w), ((i+1) \mod n, w) \mid w \in \{0, 1\}^d, 0 \leq i < n\} \\
E^h_{CCC(d)} &= \{(i, w0w'), (i, w1w') \mid w' \in \{0, 1\}^i, w \in \{0, 1\}^{n-i-1}\}
\end{align*}
$$
Cube-Connected Cycles of dimension $d$

$$CCC(d) = (V_{CCC(d)}, E_{CCC(c)}^c \cup E_{CCC(d)}^h)$$

$$V_{CCC(d)} = \{0, 1, \ldots, d - 1\} \times \{0, 1\}^d$$

$$E_{CCC(c)}^c = \{((i, w), ((i + 1) \mod n, w)) \mid w \in \{0, 1\}^d, 0 \leq i < n\}$$

$$E_{CCC(d)}^h = \{((i, w0w'), (i, w1w')) \mid w' \in \{0, 1\}^i, w \in \{0, 1\}^{n-i-1}\}$$
Cube-Connected Cycles of dimension $d$

\[
CCC(d) = (V_{CCC(d)}, E^c_{CCC(d)} \cup E^h_{CCC(d)})
\]

\[
V_{CCC(d)} = \{0, 1, \ldots, d - 1\} \times \{0, 1\}^d
\]

\[
E^c_{CCC(d)} = \{((i, w), ((i + 1) \mod n, w)) \mid w \in \{0, 1\}^d, 0 \leq i < n\}
\]

\[
E^h_{CCC(d)} = \{((i, w0w'), (i, w1w')) \mid w' \in \{0, 1\}^i, w \in \{0, 1\}^{n-i-1}\}
\]

Number of nodes: $d \cdot 2^d$
Degree: 3
Number of edges: $3 \cdot d \cdot 2^{d-1}$
Diameter: $2 \cdot d - 2 + \lceil d/2 \rceil$
Node-con.: 3
Edge-con.: 3
Butterfly of dimension $d$

$$BF(d) = (V_{BF(d)}, E_{BF(d)}^c \cup E_{BF(d)}^h)$$

$$V_{BF(d)} = \{0, 1, \ldots, d - 1\} \times \{0, 1\}^d$$

$$E_{BF(d)}^c = \{(i, w), ((i + 1) \mod n, w)\} \mid w \in \{0, 1\}^d, 0 \leq i < n$$

$$E_{BF(d)}^h = \{(i, w0w'), ((i + 1) \mod n, w1w')\} \mid w \in \{0, 1\}^i, w' \in \{0, 1\}^{n-i-1}$$

$E_{CCC}(d) = \{(i, w0w'), (i, w1w')\} \mid w \in \{0, 1\}^i, w' \in \{0, 1\}^{n-i-1}$
Butterfly of dimension $d$

$$BF(d) = (V_{BF(d)}, E^c_{BF(d)} \cup E^h_{BF(d)})$$

$V_{BF(d)} = \{0, 1, \ldots, d - 1\} \times \{0, 1\}^d$

$E^c_{BF(d)} = \{((i, w), ((i + 1) \mod n, w)) \mid w \in \{0, 1\}^d, 0 \leq i < n\}$

$E^h_{BF(d)} = \{((i, w0w'), ((i + 1) \mod n, w1w')) \mid w \in \{0, 1\}^i, w' \in \{0, 1\}^{n-i-1}\}$
Butterfly of dimension $d$

$$BF(d) = (V_{BF(d)}, E^c_{BF(d)} \cup E^h_{BF(d)})$$

$$V_{BF(d)} = \{0, 1, \ldots, d - 1\} \times \{0, 1\}^d$$

$$E^c_{BF(d)} = \{(i, w), ((i + 1) \mod n, w)\} \mid w \in \{0, 1\}^d, 0 \leq i < n$$

$$E^h_{BF(d)} = \{(i, w0w'), ((i + 1) \mod n, w1w')\} \mid w \in \{0, 1\}^i, w' \in \{0, 1\}^{n-i-1}$$
Butterfly of dimension $d$

$$BF(d) = (V_{BF(d)}, E^c_{BF(d)} \cup E^h_{BF(d)})$$

$$V_{BF(d)} = \{0, 1, \ldots, d - 1\} \times \{0, 1\}^d$$

$$E^c_{BF(d)} = \{((i, w), ((i + 1) \mod n, w)) \mid w \in \{0, 1\}^d, 0 \leq i < n\}$$

$$E^h_{BF(d)} = \{((i, w0w'), ((i + 1) \mod n, w1w')) \mid w \in \{0, 1\}^i, w' \in \{0, 1\}^{n-i-1}\}$$
Butterfly of dimension $d$

\[
BF(d) = (V_{BF(d)}, E_{BF(d)}^c \cup E_{BF(d)}^h)
\]

\[
V_{BF(d)} = \{0, 1, \ldots, d - 1\} \times \{0, 1\}^d
\]

\[
E_{BF(d)}^c = \{((i, w), ((i + 1) \mod n, w)) \mid w \in \{0, 1\}^d, 0 \leq i < n\}
\]

\[
E_{BF(d)}^h = \{((i, w0w'), ((i + 1) \mod n, w1w')) \mid w \in \{0, 1\}^i, w' \in \{0, 1\}^{n-i-1}\}
\]

\[
E_{CCC(d)} = \{(i, w0w'), (i, w1w')\} \mid w \in \{0, 1\}^i, w' \in \{0, 1\}^{n-i-1}\}
\]
Butterfly of dimension $d$

$$BF(d) = (V_{BF(d)}, E^c_{BF(d)} \cup E^h_{BF(d)})$$

$$V_{BF(d)} = \{0, 1, \cdots , d - 1\} \times \{0, 1\}^d$$

$$E^c_{BF(d)} = \{((i, w), ((i + 1) \mod n, w)) \mid w \in \{0, 1\}^d, 0 \leq i < n\}$$

$$E^h_{BF(d)} = \{((i, w0w'), ((i + 1) \mod n, w1w')) \mid w \in \{0, 1\}^i, w' \in \{0, 1\}^{n-i-1}\}$$
Butterfly of dimension $d$

$BF(d) = (V_{BF(d)}, E^c_{BF(d)} \cup E^h_{BF(d)})$

$V_{BF(d)} = \{0, 1, \cdots, d-1\} \times \{0, 1\}^d$

$E^c_{BF(d)} = \{(i, w), ((i+1) \mod n, w)\} \mid w \in \{0, 1\}^d, 0 \leq i < n$

$E^h_{BF(d)} = \{(i, w0w'), ((i+1) \mod n, w1w')\} \mid w \in \{0, 1\}^i, w' \in \{0, 1\}^{n-i-1}$

$E^h_{CCC(d)} = \{(i, w0w'), (i, w1w')\} \mid w \in \{0, 1\}^i, w' \in \{0, 1\}^{n-i-1}$
Butterfly of dimension $d$

$$BF(d) = (V_{BF(d)}, E^c_{BF(d)} \cup E^h_{BF(d)})$$

$$V_{BF(d)} = \{0, 1, \ldots, d - 1\} \times \{0, 1\}^d$$

$$E^c_{BF(d)} = \{((i, w), ((i + 1) \mod n, w)) \mid w \in \{0, 1\}^d, 0 \leq i < n\}$$

$$E^h_{BF(d)} = \{((i, w0w'), ((i + 1) \mod n, w1w')) \mid w \in \{0, 1\}^i, w' \in \{0, 1\}^{n-i-1}\}$$

Number of nodes: $d \cdot 2^d$
Degree: 4

Number of edges: $d \cdot 2^d + 1$
Diameter: $d + \lfloor \frac{d}{2} \rfloor$
Node-con.: 4
Edge-con.: 4
Butterfly of dimension $d$

$$BF(d) = (V_{BF(d)}, E^c_{BF(d)} \cup E^h_{BF(d)})$$

$$V_{BF(d)} = \{0, 1, \ldots, d - 1\} \times \{0, 1\}^d$$

$$E^c_{BF(d)} = \{(i, w), ((i + 1) \mod n, w)\} \mid w \in \{0, 1\}^d, 0 \leq i < n$$

$$E^h_{BF(d)} = \{(i, w0w'), ((i + 1) \mod n, w1w')\} \mid w \in \{0, 1\}^i, w' \in \{0, 1\}^{n-i-1}$$

Number of nodes: $d \cdot 2^d$

Number of edges: $d \cdot 2^{d+1}$

Degree: 4

Diameter: $d + \lfloor d/2 \rfloor$

Node-con.: 4

Edge-con.: 4
DeBruijn network of dimension $d$

$$DB(d) = (V_{DB(d)}, E^s_{DB(d)} \cup E^{se}_{DB(d)})$$

$$V_{DB(d)} = \{0, 1\}^d$$

$$E^s_{DB(d)} = \{(aw, wa) \mid a \in \{0, 1\}, aw, wa \in V_{DB(d)}\}$$

$$E^{se}_{DB(d)} = \{(aw, wb) \mid a \in \{0, 1\}, b = 1 - a, aw, wb \in V_{DB(d)}\}$$
DeBruijn network of dimension $d$

$$DB(d) = (V_{DB(d)}, E^s_{DB(d)} \cup E^{se}_{DB(d)})$$

- $V_{DB(d)} = \{0, 1\}^d$
- $E^s_{DB(d)} = \{(aw, wa) | a \in \{0, 1\}, aw, wa \in V_{DB(d)}\}$
- $E^{se}_{DB(d)} = \{(aw, wb) | a \in \{0, 1\}, b = 1 - a, aw, wb \in V_{DB(d)}\}$
DeBruijn network of dimension $d$

DeBruijn network:

$$DB(d) = (V_{DB(d)}, E_{DB(d)}^s \cup E_{DB(d)}^{se})$$

$$V_{DB(d)} = \{0, 1\}^d$$

$$E_{DB(d)}^s = \{(aw, wa) | a \in \{0, 1\}, aw, wa \in V_{DB(d)}\}$$

$$E_{DB(d)}^{se} = \{(aw, wb) | a \in \{0, 1\}, b = 1 - a, aw, wb \in V_{DB(d)}\}$$
DeBruijn network of dimension $d$

- DeBruijn network:

$$DB(d) = (V_{DB(d)}, E^{s}_{DB(d)} \cup E^{se}_{DB(d)})$$

- $V_{DB(d)} = \{0, 1\}^d$

- $E^{s}_{DB(d)} = \{(aw, wa) \mid a \in \{0, 1\}, aw, wa \in V_{DB(d)}\}$

- $E^{se}_{DB(d)} = \{(aw, wb) \mid a \in \{0, 1\}, b = 1 - a, aw, wb \in V_{DB(d)}\}$

![DeBruijn network diagram]
DeBruijn network of dimension $d$

$$DB(d) = (V_{DB(d)}, E^s_{DB(d)} \cup E^{se}_{DB(d)})$$

$V_{DB(d)} = \{0, 1\}^d$

$E^s_{DB(d)} = \{(aw, wa) \mid a \in \{0, 1\}, aw, wa \in V_{DB(d)}\}$

$E^{se}_{DB(d)} = \{(aw, wb) \mid a \in \{0, 1\}, b = 1 - a, aw, wb \in V_{DB(d)}\}$

Number of nodes: $2^d$
Number of edges: $2^{d+1}$
Degree: 2 + 2
Diameter: $d$
DeBruijn network of dimension $d$

- Undirected DeBruijn network:

$$DB'(d) = (V_{DB(d)}, E^I_{DB(d)} \cup E^{Ise}_{DB(d)})$$

$$E^I_{DB(d)} = \{\{aw, wa\} \mid a \in \{0, 1\}, aw, wa \in V_{DB(d)}\}$$

$$E^{Ise}_{DB(d)} = \{\{aw, wb\} \mid a \in \{0, 1\}, b = 1 - a, aw, wb \in V_{DB(d)}\}$$

Number of nodes: $2^d$

Degree: \{2, 3, 4\}

Number of edges: $2^{d+1} - 3$

Diameter: $d$
Shuffle-Exchange network of dimension $d$

- **Shuffle-Exchange network:**

  $$SE(d) = (V_{SE(d)}, E^s_{SE(d)} \cup E^e_{SE(d)})$$

  - $V_{SE(d)} = \{0,1\}^d$
  - $E^s_{SE(d)} = \{(aw, wa) \mid a \in \{0,1\}, aw, wa \in V_{SE(d)}\}$
  - $E^e_{SE(d)} = \{(wa, wb) \mid a \in \{0,1\}, b = 1 - a, wa, wb \in V_{SE(d)}\}$
Shuffle-Exchange network of dimension $d$

$$SE(d) = (V_{SE(d)}, E^s_{SE(d)} \cup E^e_{SE(d)})$$

- $V_{SE(d)} = \{0, 1\}^d$
- $E^s_{SE(d)} = \{(aw, wa) \mid a \in \{0, 1\}, aw, wa \in V_{SE(d)}\}$
- $E^e_{SE(d)} = \{(wa, wb) \mid a \in \{0, 1\}, b = 1 - a, wa, wb \in V_{SE(d)}\}$
Shuffle-Exchange network of dimension $d$

$$SE(d) = \left( V_{SE(d)}, E^s_{SE(d)} \cup E^e_{SE(d)} \right)$$

- $V_{SE(d)} = \{0, 1\}^d$
- $E^s_{SE(d)} = \{(aw, wa) \mid a \in \{0, 1\}, aw, wa \in V_{SE(d)}\}$
- $E^e_{SE(d)} = \{(wa, wb) \mid a \in \{0, 1\}, b = 1 - a, wa, wb \in V_{SE(d)}\}$
Shuffle-Exchange network of dimension $d$

$SE(d) = (V_{SE(d)}, E_{SE(d)}^s \cup E_{SE(d)}^e)$

$V_{SE(d)} = \{0, 1\}^d$

$E_{SE(d)}^s = \{(aw, wa) \mid a \in \{0, 1\}, aw, wa \in V_{SE(d)}\}$

$E_{SE(d)}^e = \{(wa, wb) \mid a \in \{0, 1\}, b = 1 - a, wa, wb \in V_{SE(d)}\}$

- Number of nodes: $2^d$
- Number of edges: $2^{d+1}$
- Degree: $2 + 2$
- Diameter: $2 \cdot d - 1$
**Shuffle-Exchange network of dimension** \( d \)

- **Undirected Shuffle-Exchange network:**

\[
SE'(d) = (V_{SE(d)}, E^{fs}_{SE(d)} \cup E^{fe}_{SE(d)})
\]

\[
E^{fs}_{SE(d)} = \{ \{aw, wa\} | a \in \{0,1\}, aw, wa \in V_{SE(d)} \}
\]

\[
E^{fe}_{SE(d)} = \{ \{wa, wb\} | a \in \{0,1\}, b = 1 - a, wa, wb \in V_{SE(d)} \}
\]

- **Number of nodes:** \( 2^d \)
- **Degree:** \{1, 2, 3\}
- **Number of edges:** \( 2^{d+1}/3 \)
- **Diameter:** \( 2 \cdot d - 1 \)
Lemma:

$C(2^{d+1} - 1)$ may be embedded into $T(d)$ with load 1 and dilation 3.

Proof:

[Diagram showing the embedding process]
Lemma:

$C(2^{d+1} - 1)$ may be embedded into $T(d)$ with load 1 and dilation 3.

Proof: Embed a path recursively with dilation $\leq 3$ from the root to a son of the root.
Lemma:

$C(2^d + 1 - 1)$ may be embedded into $T(d)$ with load 1 and dilation 3.

Proof: Embed a path recursively with dilation $\leq 3$ from the root to a son of the root.
Lemma:

\( C(2^{d+1} - 1) \) may be embedded into \( T(d) \) with load 1 and dilation 3.

Proof: Embed a path recursively with dilation \( \leq 3 \) from the root to a son of the root.
Lemma:

$C(2^{d+1} - 1)$ may be embedded into $T(d)$ with load 1 and dilation 3.

Proof: Embed a path recursively with dilation $\leq 3$ from the root to a son of the root.
Lemma:

\( C(2^{d+1} - 1) \) may be embedded into \( T(d) \) with load 1 and dilation 3.

Proof: Embed a path recursively with dilation \( \leq 3 \) from the root to a son of the root.
Lemma:

$C(2^{d+1} - 1)$ may be embedded into $T(d)$ with load 1 and dilation 3.

Proof: Embed a path recursively with dilation $\leq 3$ from the root to a son of the root.
Lemma:

$C(2^{d+1} - 1)$ may be embedded into $T(d)$ with load 1 and dilation 3.

Proof: Embed a path recursively with dilation $\leq 3$ from the root to a son of the root.
Lemma:

$C(2^{d+1} - 1)$ may be embedded into $T(d)$ with load 1 and dilation 3.

Proof: Embedd a path recursively with dilation $\leq 3$ from the root to a son of the root.
Lemma:

$C(2^{d+1} - 1)$ may be embedded into $T(d)$ with load 1 and dilation 3.

Proof: Embed a path recursively with dilation $\leq 3$ from the root to a son of the root.
Lemma:

$C(2^{d+1} - 1)$ may be embedded into $T(d)$ with load 1 and dilation 3.

Proof: Embed a path recursively with dilation $\leq 3$ from the root to a son of the root.
Lemma:

$C(2^{d+1} - 1)$ may be embedded into $T(d)$ with load 1 and dilation 3.

Proof: Embed a path recursively with dilation $\leq 3$ from the root to a son of the root.
Lemma:

$C(2^{d+1} - 1)$ may be embedded into $T(d)$ with load 1 and dilation 3.

Proof: Embed a path recursively with dilation $\leq 3$ from the root to a son of the root.
Lemma:

$C(3 \cdot (2^{d+1} - 1))$ may be embedded into $T(d)$ with load 3 and dilation 1.

Proof:
Lemma:

$C(3 \cdot (2^{d+1} - 1))$ may be embedded into $T(d)$ with load 3 and dilation 1.

Proof: Use the in-order traversal through the tree.
Lemma:

\[ C(3 \cdot (2^{d+1} - 1)) \] may be embedded into \( T(d) \) with load 3 and dilation 1.

Proof: Use the in-order traversal through the tree.
Lemma:

$C(3 \cdot (2^{d+1} - 1))$ may be embedded into $T(d)$ with load 3 and dilation 1.

Proof: Use the in-order traversal through the tree.
Lemma:

$C(3 \cdot (2^{d+1} - 1))$ may be embedded into $T(d)$ with load 3 and dilation 1.

Proof: Use the in-order traversal through the tree.
Lemma:

\[ C(3 \cdot (2^{d+1} - 1)) \] may be embedded into \( T(d) \) with load 3 and dilation 1.

Proof: Use the in-order traversal through the tree.
Lemma:

\( C(3 \cdot (2^{d+1} - 1)) \) may be embedded into \( T(d) \) with load 3 and dilation 1.

Proof: Use the in-order traversal through the tree.
Lemma:

$C(3 \cdot (2^{d+1} - 1))$ may be embedded into $T(d)$ with load 3 and dilation 1.

Proof: Use the in-order traversal through the tree.
Lemma:

$C(3 \cdot (2^{d+1} - 1))$ may be embedded into $T(d)$ with load 3 and dilation 1.

Proof: Use the in-order traversal through the tree.
Lemma:

\[ C(3 \cdot (2^{d+1} - 1)) \] may be embedded into \( T(d) \) with load 3 and dilation 1.

Proof: Use the in-order traversal through the tree.
Lemma:

$C(3 \cdot (2^{d+1} - 1))$ may be embedded into $T(d)$ with load 3 and dilation 1.

Proof: Use the in-order traversal through the tree.
Lemma:

$C(2 \cdot (2^{d+1} - 1))$ may be embedded into $T(d)$ with load 2 and dilation 2.

Proof:
Lemma:

\[ C(2 \cdot (2^{d+1} - 1)) \] may be embedded into \( T(d) \) with load 2 and dilation 2.

Proof: Use the in-order traversal through the tree and jump the 'in-order’ nodes.
Lemma:

\( C(2 \cdot (2^{d+1} - 1)) \) may be embedded into \( T(d) \) with load 2 and dilation 2.

Proof: Use the in-order traversal through the tree and jump the ‘in-order’ nodes.
Lemma:

\[ C(2 \cdot (2^{d+1} - 1)) \] may be embedded into \( T(d) \) with load 2 and dilation 2.

Proof: Use the in-order traversal through the tree and jump the ‘in-order” nodes.
Lemma:

$C(2 \cdot (2^{d+1} - 1))$ may be embedded into $T(d)$ with load 2 and dilation 2.

Proof: Use the in-order traversal through the tree and jump the ‘in-order” nodes.
Lemma:

\[ C(2 \cdot (2^{d+1} - 1)) \] may be embedded into \( T(d) \) with load 2 and dilation 2.

Proof: Use the in-order traversal through the tree and jump the ‘in-order” nodes.
Lemma:

$C(2 \cdot (2^d+1 - 1))$ may be embedded into $T(d)$ with load 2 and dilation 2.

Proof: Use the in-order traversal through the tree and jump the ‘in-order’ nodes.
Lemma:

$C(2 \cdot (2^{d+1} - 1))$ may be embedded into $T(d)$ with load 2 and dilation 2.

Proof: Use the in-order traversal through the tree and jump the ‘in-order’ nodes.
Lemma:

$C(2 \cdot (2^{d+1} - 1))$ may be embedded into $T(d)$ with load 2 and dilation 2.

Proof: Use the in-order traversal through the tree and jump the ‘in-order” nodes.
Lemma:

\( C(2 \cdot (2^{d+1} - 1)) \) may be embedded into \( T(d) \) with load 2 and dilation 2.

Proof: Use the in-order traversal through the tree and jump the 'in-order" nodes.
**Lemma:**

$L(2^{d+1} - 1)$ may be embedded into $XT(d)$ with load 1 and dilation 1.

**Proof:**

```
L(n) into XT(d)
```

Graphical representation of the embedding process.
Lemma:

$L(2^{d+1} - 1)$ may be embedded into $XT(d)$ with load 1 and dilation 1.

Proof: Place the path in levels through the tree.
Lemma:

\( C(2^{d+1} - 1) \) may be embedded into \( XT(d) \) with load 1 and dilation 1.

Proof: Place the path in levels through the left part and through the right part and connect both to a cycle.
Lemma:

$C(2^{d+1} - 1)$ may be embedded into $X_T(d)$ with load 1 and dilation 1.

Proof: Place the path in levels through the left part and through the right part and connect both to a cycle.
Lemma:

$C(2^{d+1} - 1)$ may be embedded into $XT(d)$ with load 1 and dilation 1.

Proof: Place the path in levels through the left part and through the right part and connect both to a cycle.
Lemma:

$C(2^{d+1} - 1)$ may be embedded into $XT(d)$ with load 1 and dilation 1.

Proof: Place the path in levels through the left part and through the right part and connect both to a cycle.
Lemma:

\( C(2^{d+1} - 1) \) may be embedded into \( XT(d) \) with load 1 and dilation 1.

Proof: Place the path in levels through the left part and through the right part and connect both to a cycle.
Lemma:

$C(2^d)$ may be embedded into $HQ(d)$ with load 1 and dilation 1.

Proof: Gray-code.
Lemma:

$C(2^d)$ may be embedded into $HQ(d)$ with load 1 and dilation 1.

Proof: Gray-code.
Lemma:

$C(2^d)$ may be embedded into $HQ(d)$ with load 1 and dilation 1.

Proof: Gray-code.
Lemma:

If $2n \leq 2^d$ holds, then $C(2n)$ could be embedded into $HQ(d)$ with load 1 and dilation 1.

Proof: recursive structure of $HQ(d)$
Lemma:
If $2n \leq 2^d$ holds, then $C(2n)$ could be embedded into $HQ(d)$ with load 1 and dilation 1.

Proof: recursive structure of $HQ(d)$
Lemma:

If $2n \leq 2^d$ holds, then $C(2n)$ could be embedded into $HQ(d)$ with load 1 and dilation 1.

Proof: recursive structure of $HQ(d)$
Lemma:
If $2n \leq 2^d$ holds, then $C(2n)$ could be embedded into $HQ(d)$ with load 1 and dilation 1.

Proof: recursive structure of $HQ(d)$
Alternative proof: $G(2, 2^{d-1})$ is a sub-graph of $HQ(d)$. 
Lemma:

$C(d \cdot 2^d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.

Proof:
Lemma:

$C(d \cdot 2^d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.

Proof: Join cycles of length $d, 2d, 4d, ...$.
Lemma:

$C(d \cdot 2^d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.

Proof: Join cycles of length $d, 2d, 4d, ...$. 
Lemma:

$C(d \cdot 2^d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.

Proof: Join cycles of length $d, 2d, 4d, \ldots$. 

---

$C(n)$ into $BF(d)$
Lemma:

\( C(d \cdot 2^d) \) may be embedded into \( BF(d) \) with load 1 and dilation 1.

Proof: Join cycles of length \( d, 2d, 4d, \ldots \)
Lemma:

\( C(d \cdot 2^d) \) may be embedded into \( BF(d) \) with load 1 and dilation 1.

Proof: Join cycles of length \( d, 2d, 4d, \ldots \).
Lemma:

$C(d \cdot 2^d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.

Proof: Join cycles of length $d$, $2d$, $4d$, ....
Lemma:

$C(d \cdot 2^d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.

Proof:
Lemma:

\[ C(d \cdot 2^d) \] may be embedded into \( BF(d) \) with load 1 and dilation 1.

Proof: Join cycles of length \( d, 2d, 4d, \ldots \) (view using the gray-code).
Lemma:

$C(d \cdot 2^d)$ may be embedded into $BF(d)$ with load $1$ and dilation $1$.

Proof: Join cycles of length $d, 2d, 4d, ...$ (view using the gray-code).
Lemma:

$C(d \cdot 2^d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.

Proof: Join cycles of length $d, 2d, 4d, \ldots$ (view using the gray-code).
Lemma:

$C(d \cdot 2^d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.

Proof: Join cycles of length $d, 2d, 4d, \ldots$ (view using the gray-code).
Lemma:

\[ C(d \cdot 2^d) \] may be embedded into \( BF(d) \) with load 1 and dilation 1.

Proof: Join cycles of length \( d, 2d, 4d, \ldots \) (view using the gray-code).
Lemma:

$C(d \cdot 2^d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.

Proof: Join cycles of length $d, 2d, 4d, \ldots$ (view using the gray-code).
Lemma:

$C(d \cdot 2^d)$ may be embedded into $CCC(d)$ with load 1 and dilation 2.

Proof:

[Diagram showing the embedding process]

$C(n)$ into $CCC(d)$
Lemma:

$C(d \cdot 2^d)$ may be embedded into $CCC(d)$ with load 1 and dilation 2.

Proof: Embed cycles in $BF(d)$ and embed $BF(d)$ in $CCC(d)$ with dilation 2.
Lemma:

$L(n \cdot n_2 \cdots n_d)$ may be embedded into $G(n_1, n_2, \ldots, n_d)$ with load 1 and dilation 1.

Proof:

```
0, 0  1, 0  2, 0  3, 0  4, 0  5, 0  6, 0  7, 0  8, 0  9, 0 10, 0 11, 0 12, 0 13, 0
0, 1  1, 1  2, 1  3, 1  4, 1  5, 1  6, 1  7, 1  8, 1  9, 1 10, 1 11, 1 12, 1 13, 1
0, 2  1, 2  2, 2  3, 2  4, 2  5, 2  6, 2  7, 2  8, 2  9, 2 10, 2 11, 2 12, 2 13, 2
0, 3  1, 3  2, 3  3, 3  4, 3  5, 3  6, 3  7, 3  8, 3  9, 3 10, 3 11, 3 12, 3 13, 3
```
Lemma:

$L(n_1 \cdot n_2 \cdots \cdot n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 1.

Proof: Place the path snake-wise through the grid.
Lemma:

$L(n)$ into $G(n_1, n_2, \cdots, n_d)$

Lemma:

$L(n_1 \cdot n_2 \cdot \cdots \cdot n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 1.

Lemma:

$C(n)$ may be embedded into $L(n)$ with load 1 and dilation 2.

- For each direction of the path, use every second node.
- Or: use the bandwidth 2 embedding of the cycle.

Lemma:

$C(n_1 \cdot n_2 \cdot \cdots \cdot n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 2.

- Embed cycle in the path with dilation 2.
- Embed the path in the grid with dilation 1.
Lemma:

$L(n_1 \cdot n_2 \cdot \cdots \cdot n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 1.

Lemma:

$C(n)$ may be embedded into $L(n)$ with load 1 and dilation 2.

- For each direction of the path, use every second node.
- Or: use the bandwidth 2 embedding of the cycle.

Lemma:

$C(n_1 \cdot n_2 \cdot \cdots \cdot n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 2.

- Embedd cycle in the path with dilation 2.
- Embedd the path in the grid with dilation 1.
Lemma:

$L(n \cdot n_2 \cdots n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 1.

Lemma:

$C(n)$ may be embedded into $L(n)$ with load 1 and dilation 2.

- For each direction of the path, use every second node.
- Or: use the bandwidth 2 embedding of the cycle.

Lemma:

$C(n_1 \cdot n_2 \cdots n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 2.

- Embed cycle in the path with dilation 2.
- Embed the path in the grid with dilation 1.
Lemma:

$L(n_1 \cdot n_2 \cdot \cdots \cdot n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 1.

Lemma:

$C(n)$ may be embedded into $L(n)$ with load 1 and dilation 2.

- For each direction of the path, use every second node.
- Or: use the bandwidth 2 embedding of the cycle.

Lemma:

$C(n_1 \cdot n_2 \cdot \cdots \cdot n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 2.

- Embedd cycle in the path with dilation 2.
- Embedd the path in the grid with dilation 1.
Lemma:
$L(n \cdot n_2 \cdot \cdots \cdot n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 1.

Lemma:
$C(n)$ may be embedded into $L(n)$ with load 1 and dilation 2.

- For each direction of the path, use every second node.
- Or: use the bandwidth 2 embedding of the cycle.

Lemma:
$C(n_1 \cdot n_2 \cdot \cdots \cdot n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 2.

- Embed cycle in the path with dilation 2.
- Embed the path in the grid with dilation 1.
Lemma:

$L(n) \ into \ G(n_1, n_2, \cdots, n_d)$

Lemma:

$L(n_1 \cdot n_2 \cdots \cdot n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 1.

Lemma:

$C(n)$ may be embedded into $L(n)$ with load 1 and dilation 2.

- For each direction of the path, use every second node.
- Or: use the bandwidth 2 embedding of the cycle.

Lemma:

$C(n_1 \cdot n_2 \cdots \cdot n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 2.

- Embed cycle in the path with dilation 2.
- Embed the path in the grid with dilation 1.
Lemma:

$L(n_1 \cdot n_2 \cdot \cdots \cdot n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 1.

Lemma:

$C(n)$ may be embedded into $L(n)$ with load 1 and dilation 2.

- For each direction of the path, use every second node.
- Or: use the bandwidth 2 embedding of the cycle.

Lemma:

$C(n_1 \cdot n_2 \cdot \cdots \cdot n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 2.

- Embedd cycle in the path with dilation 2.
- Embedd the path in the grid with dilation 1.
Lemma:
$L(n_1 \cdot n_2 \cdots n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 1.

Lemma:
$C(n)$ may be embedded into $L(n)$ with load 1 and dilation 2.

- For each direction of the path, use every second node.
- Or: use the bandwidth 2 embedding of the cycle.

Lemma:
$C(n_1 \cdot n_2 \cdots n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 2.

- Embed cycle in the path with dilation 2.
- Embedd the path in the grid with dilation 1.
Lemma:

$L(n_1 \cdot n_2 \cdot \cdots \cdot n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 1.

Lemma:

$C(n)$ may be embedded into $L(n)$ with load 1 and dilation 2.

- For each direction of the path, use every second node.
- Or: use the bandwidth 2 embedding of the cycle.

Lemma:

$C(n_1 \cdot n_2 \cdot \cdots \cdot n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 2.

- Embed cycle in the path with dilation 2.
- Embedd the path in the grid with dilation 1.
Lemma:

$L(n_1 \cdot n_2 \cdots n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 1.

Lemma:

$C(n)$ may be embedded into $L(n)$ with load 1 and dilation 2.

- For each direction of the path, use every second node.
- Or: use the bandwidth 2 embedding of the cycle.

Lemma:

$C(n_1 \cdot n_2 \cdots n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 2.

- Embed cycle in the path with dilation 2.
- Embed the path in the grid with dilation 1.
Lemma:

\( C(n_1 \cdot n_2 \cdots n_d) \) may be embedded into \( G(n_1, n_2, \cdots, n_d) \) with load 1 and dilation 1, if at least one \( n_i \) is even.

Proof:

\[
\begin{array}{cccccccccccccccc}
0,3 & 1,3 & 2,3 & 3,3 & 4,3 & 5,3 & 6,3 & 7,3 & 8,3 & 9,3 & 10,3 & 11,3 & 12,3 & 13,3 \\
0,2 & 1,2 & 2,2 & 3,2 & 4,2 & 5,2 & 6,2 & 7,2 & 8,2 & 9,2 & 10,2 & 11,2 & 12,2 & 13,2 \\
0,1 & 1,1 & 2,1 & 3,1 & 4,1 & 5,1 & 6,1 & 7,1 & 8,1 & 9,1 & 10,1 & 11,1 & 12,1 & 13,1 \\
0,0 & 1,0 & 2,0 & 3,0 & 4,0 & 5,0 & 6,0 & 7,0 & 8,0 & 9,0 & 10,0 & 11,0 & 12,0 & 13,0 \\
\end{array}
\]
Lemma:

\( C(n_1 \cdot n_2 \cdots n_d) \) may be embedded into \( G(n_1, n_2, \cdots, n_d) \) with load 1 and dilation 1, if at least one \( n_i \) is even.

Proof: Place the path snake-wise through the grid.
Lemma:

\( C(n_1 \cdot n_2 \cdots \cdot n_d) \) may be embedded into \( G(n_1, n_2, \cdots, n_d) \) with load 1 and dilation 1, if at least one \( n_i \) is even.

Lemma:

\( C(n_1 \cdot n_2 \cdots \cdot n_d) \) may not be embedded into \( G(n_1, n_2, \cdots, n_d) \) with load 1 and dilation 1, if all \( n_i \) are odd.

Proof:
Lemma:

$C(n_1 \cdot n_2 \cdots \cdot n_d)$ may be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 1, if at least one $n_i$ is even.

Lemma:

$C(n_1 \cdot n_2 \cdots \cdot n_d)$ may not be embedded into $G(n_1, n_2, \cdots, n_d)$ with load 1 and dilation 1, if all $n_i$ are odd.

Proof: Consider the 2-colouring of the grid.

\[
\begin{array}{cccccccccccccccc}
0,0 & 1,0 & 2,0 & 3,0 & 4,0 & 5,0 & 6,0 & 7,0 & 8,0 & 9,0 & 10,0 & 11,0 & 12,0 & 13,0 & 14,0 \\
0,1 & 1,1 & 2,1 & 3,1 & 4,1 & 5,1 & 6,1 & 7,1 & 8,1 & 9,1 & 10,1 & 11,1 & 12,1 & 13,1 & 14,1 \\
0,2 & 1,2 & 2,2 & 3,2 & 4,2 & 5,2 & 6,2 & 7,2 & 8,2 & 9,2 & 10,2 & 11,2 & 12,2 & 13,2 & 14,2 \\
0,3 & 1,3 & 2,3 & 3,3 & 4,3 & 5,3 & 6,3 & 7,3 & 8,3 & 9,3 & 10,3 & 11,3 & 12,3 & 13,3 & 14,3 \\
0,4 & 1,4 & 2,4 & 3,4 & 4,4 & 5,4 & 6,4 & 7,4 & 8,4 & 9,4 & 10,4 & 11,4 & 12,4 & 13,4 & 14,4 \\
\end{array}
\]
Lemma:

$T(d)$ may be embedded into $L(2^{d+1} - 1)$ with load 1 and dilation $\lceil 2^{d+1}/2d \rceil$.

Idea of Proof:

- Stretch the longest path of $T(d)$ on the path.
- Or use the bandwidth-embedding of the $T(d)$. 
Lemma:

$T(d)$ may be embedded into $L(2^{d+1} - 1)$ with load 1 and dilation $\lceil 2^{d+1}/2d \rceil$.

Idea of Proof:

- Stretch the longest path of $T(d)$ on the path.
- Or use the bandwidth-embedding of the $T(d)$.
Lemma:

$T(d)$ may be embedded into $L(2^{d+1} - 1)$ with load 1 and dilation $\lceil 2^{d+1}/2d \rceil$.

Idea of Proof:

- Stretch the longest path of $T(d)$ on the path.
- Or use the bandwidth-embedding of the $T(d)$. 

$$
\begin{array}{c}
0
\end{array}
\begin{array}{c}
0
\end{array}
\begin{array}{c}
0
\end{array}
\begin{array}{c}
0
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
0
\end{array}
\begin{array}{c}
0
\end{array}
\begin{array}{c}
0
\end{array}
\begin{array}{c}
0
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
0
\end{array}
\begin{array}{c}
0
\end{array}
\begin{array}{c}
0
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
1
\end{array}

\begin{array}{c}
0
\end{array}
\begin{array}{c}
0
\end{array}
\begin{array}{c}
0
\end{array}
\begin{array}{c}
0
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
0
\end{array}
\begin{array}{c}
0
\end{array}
\begin{array}{c}
0
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
0
\end{array}
\begin{array}{c}
0
\end{array}
\begin{array}{c}
0
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
1
\end{array}
\begin{array}{c}
1
\end{array}
Lemma:

\(T(d)\) may be embedded into \(HQ(d+1)\) with load 1 and dilation 2.

Proof:

- \(f : \{w \in \{0,1\}^* \mid |w| \leq d\} \mapsto \{w \in \{0,1\}^* \mid |w| = d + 1\}\).
- Add to \(w\) a bit-sequence of length \(x(w) = d + 1 - |w| \geq 1\).
- \(f(w) = w10^{x(w)-1}\).
- Edges: \(f((w,wa)) = f((w10^{x(w)-1}, wa10^{x(wa)-1}))\)
- Dilation is 2.
Lemma:

\( T(d) \) may be embedded into \( HQ(d + 1) \) with load 1 and dilation 2.

Proof:

- \( f : \{w \in \{0,1\}^* \mid |w| \leq d\} \mapsto \{w \in \{0,1\}^* \mid |w| = d + 1\} \).
- Add to \( w \) a bit-sequence of length \( x(w) = d + 1 - |w| \geq 1 \).
- \( f(w) = w10^{x(w)-1} \).
- Edges: \( f((w, wa)) = f((w10^{x(w)}-1, wa10^{x(wa)}-1)) \).
- Dilation is 2.
**Lemma:**

\[ T(d) \text{ may be embedded into } HQ(d + 1) \text{ with load 1 and dilation 2.} \]

**Proof:**

- \( f : \{ w \in \{0,1\}^* \mid |w| \leq d \} \mapsto \{ w \in \{0,1\}^* \mid |w| = d + 1 \}. \)
- Add to \( w \) a bit-sequence of length \( x(w) = d + 1 - |w| \geq 1. \)
- \( f(w) = w10^{x(w)-1}. \)
- Edges: \( f((w, wa)) = f((w10^{x(w)-1}, wa10^{x(wa)-1})) \)
- Dilation is 2.
Lemma:

$T(d)$ may be embedded into $HQ(d + 1)$ with load 1 and dilation 2.

Proof:

- $f : \{w \in \{0, 1\}^* \mid |w| \leq d\} \mapsto \{w \in \{0, 1\}^* \mid |w| = d + 1\}$.  
- Add to $w$ a bit-sequence of length $x(w) = d + 1 - |w| \geq 1$.  
- $f(w) = w10^{x(w)-1}$.  
- Edges: $f((w, wa)) = f((w10^{x(w)-1}, wa10^{x(wa)-1}))$.  
- Dilation is 2.
Lemma:

\( T(d) \) may be embedded into \( HQ(d + 1) \) with load 1 and dilation 2.

Proof:

- \( f : \{w \in \{0,1\}^* \mid |w| \leq d \} \mapsto \{w \in \{0,1\}^* \mid |w| = d + 1\} \).
- Add to \( w \) a bit-sequence of length \( x(w) = d + 1 - |w| \geq 1 \).
- \( f(w) = w10^{x(w)-1} \).
- Edges: \( f((w, wa)) = f((w10^{x(w)-1}, wa10^{x(wa)-1})) \).
- Dilation is 2.
Lemma:

\( T(d) \) may be embedded into \( HQ(d + 1) \) with load 1 and dilation 2.

Proof:

- \( f : \{ w \in \{0, 1\}^* \mid |w| \leq d \} \mapsto \{ w \in \{0, 1\}^* \mid |w| = d + 1 \} \).
- Add to \( w \) a bit-sequence of length \( x(w) = d + 1 - |w| \geq 1 \).
- \( f(w) = w10^{x(w)-1} \).
- Edges: \( f((w, wa)) = f((w10^{x(w)-1}, wa10^{x(wa)-1})) \)
- Dilation is 2.
**Lemma:**

$XT(d)$ may be embedded into $HQ(d + 1)$ with load 1 and dilation 2.

- $f : \{w \in \{0, 1\}^* \mid |w| \leq d\} \mapsto \{w \in \{0, 1\}^* \mid |w| = d + 1\}$.
- Add to $w$ a bit-sequence of length $x(w) = d + 1 - |w| \geq 1$.
- $f(w) = \text{GrayCode}(w)10^{x(w)-1}$.
- Edges: $f((w, wa)) = f((\text{GrayCode}(w)10^{x(w)-1}, \text{GrayCode}(wa)10^{x(wa)-1}))$.
- Dilation is 2, because $\text{GrayCode}(wa) = \text{GrayCode}(w)a_{w,b}$.
Lemma:

$XT(d)$ may be embedded into $HQ(d + 1)$ with load 1 and dilation 2.

- $f : \{w \in \{0,1\}^* \mid |w| \leq d\} \mapsto \{w \in \{0,1\}^* \mid |w| = d + 1\}$.
- Add to $w$ a bit-sequence of length $x(w) = d + 1 - |w| \geq 1$.
- $f(w) = \text{GrayCode}(w)10^{x(w)-1}$.
- Edges: $f((w, wa)) = f((\text{GrayCode}(w)10^{x(w)-1}, \text{GrayCode}(wa)10^{x(wa)-1}))$
- Dilation is 2, because $\text{GrayCode}(wa) = \text{GrayCode}(w)a_{w,b}$. 
Lemma:

$XT(d)$ may be embedded into $HQ(d + 1)$ with load 1 and dilation 2.

- $f : \{w \in \{0, 1\}^* \mid |w| \leq d\} \mapsto \{w \in \{0, 1\}^* \mid |w| = d + 1\}.$
- Add to $w$ a bit-sequence of length $x(w) = d + 1 - |w| \geq 1$.
- $f(w) = \text{GrayCode}(w)10^{x(w)-1}$.
- Edges: $f((w, wa)) = f((\text{GrayCode}(w)10^{x(w)-1}, \text{GrayCode}(wa)10^{x(wa)-1}))$
- Dilation is 2, because $\text{GrayCode}(wa) = \text{GrayCode}(w)a_{w, b}$. 

$$
E_{T(d)} = \{\{w, wa\} \mid w, wa \in V, a \in \{0, 1\}\}
$$

$$
E_{HQ(d)} = \{\{w0w', w1w'\} \mid w0w', w1w' \in V_{HQ(d)}\}
$$
Lemma:

$XT(d)$ may be embedded into $HQ(d + 1)$ with load 1 and dilation 2.

- $f : \{w \in \{0,1\}^* \mid |w| \leq d\} \mapsto \{w \in \{0,1\}^* \mid |w| = d + 1\}$.
- Add to $w$ a bit-sequence of length $x(w) = d + 1 − |w| \geq 1$.
- $f(w) = \text{GrayCode}(w)10^{x(w)−1}$.
- Edges: $f((w, wa)) = f((\text{GrayCode}(w)10^{x(w)−1}, \text{GrayCode}(wa)10^{x(wa)−1}))$.
- Dilation is 2, because $\text{GrayCode}(wa) = \text{GrayCode}(w)a_w,b$. 

$$
E_T(d) = \{\{w, wa\} \mid w, wa \in V, a \in \{0, 1\}\} \text{ and } E_{HQ(d)} = \{\{w0w', w1w'\} \mid w0w', w1w' \in V_{HQ(d)}\}
$$
Lemma:

$XT(d)$ may be embedded into $HQ(d + 1)$ with load 1 and dilation 2.

- $f : \{w \in \{0, 1\}^* \mid |w| \leq d\} \mapsto \{w \in \{0, 1\}^* \mid |w| = d + 1\}$.
- Add to $w$ a bit-sequence of length $x(w) = d + 1 − |w| \geq 1$.
- $f(w) = \text{GrayCode}(w)10^{x(w)−1}$.
- Edges: $f((w, wa)) = f((\text{GrayCode}(w)10^{x(w)−1}, \text{GrayCode}(wa)10^{x(wa)−1}))$
- Dilation is 2, because $\text{GrayCode}(wa) = \text{GrayCode}(w)a_w,b$. 
**Lemma:**

$XT(d)$ may be embedded into $HQ(d + 1)$ with load 1 and dilation 2.

- $f : \{w \in \{0,1\}^* \mid |w| \leq d\} \mapsto \{w \in \{0,1\}^* \mid |w| = d + 1\}$.
- Add to $w$ a bit-sequence of length $x(w) = d + 1 - |w| \geq 1$.
- $f(w) = \text{GrayCode}(w)10^{x(w)-1}$.
- Edges: $f((w, wa)) = f((\text{GrayCode}(w)10^{x(w)-1}, \text{GrayCode}(wa)10^{x(wa)-1}))$
- Dilation is 2, because $\text{GrayCode}(wa) = \text{GrayCode}(w)a_{w,b}$.
Lemma:

\( T(d) \) may not be embedded into \( HQ(d+1) \) for \( d > 1 \) with load 1 and dilation 1.

Proof:

\begin{align*}
T(d) & \rightarrow HQ(d+1) \\
00 & \rightarrow 000, 001, 010, 011 \\
01 & \rightarrow 010, 011, 100, 101 \\
10 & \rightarrow 101, 110, 111, 100 \\
11 & \rightarrow 111, 110, 111, 100 \\
\end{align*}
Lemma:

$T(d)$ may not be embedded into $HQ(d+1)$ for $d > 1$ with load 1 and dilation 1.

Proof: Consider the 2-colouring of $T(d)$ in $HQ(d+1)$.
Lemma:

\[ T(d) \] may be embedded into \( HQ(d + 1) \) with load 1 and dilation 2, such that only one edge is stretched.

Proof:

```
00 - 00 - 000
01 - 01 - 001
010 - 010 - 011
011 - 011 - 010
100 - 100 - 101
101 - 101 - 110
110 - 110 - 111
```

```
000
001
010
011
100
101
110
111
```
Lemma:

\( T(d) \) may be embedded into \( HQ(d + 1) \) with load 1 and dilation 2, such that only one edge is stretched.

Proof: Recursive embedding of the double-rooted tree as a sub-graph of the \( HQ \).
Lemma:

$T(d)$ may be embedded into $HQ(d + 1)$ with load 1 and dilation 2, such that only one edge is stretched.

Proof:
Lemma:

\( T(d) \) may be embedded into \( HQ(d + 1) \) with load 1 and dilation 2, such that only one edge is stretched.

Proof: Recursive embedding of the double-rooted tree as a sub-graph of the \( HQ \).
Lemma:

$T(d)$ may be embedded into $HQ(d + 1)$ with load 1 and dilation 2, such that only one edge is stretched.

Proof: Recursive embedding of the double-rooted tree as a sub-graph of the $HQ$. 

$$\begin{array}{c}
\text{T(d) into HQ(d + 1)}
\end{array}$$
Lemma:

$T(d)$ may be embedded into $HQ(d + 1)$ with load 1 and dilation 2, such that only one edge is stretched.

Proof: Recursive embedding of the double-rooted tree as a sub-graph of the $HQ$. 
Lemma:

$T(d)$ may be embedded into $HQ(d + 1)$ with load 1 and dilation 2, such that only one edge is stretched.

Proof: Recursive embedding of the double-rooted tree as a sub-graph of the $HQ$. 

$$T(d) \text{ into } HQ(d + 1)$$
Lemma:

\(T(d)\) may be embedded into \(HQ(d + 1)\) with load 1 and dilation 2, such that only one edge is stretched.

Proof: Recursive embedding of the double-rooted tree as a sub-graph of the \(HQ\).
Lemma:

$T(d)$ may be embedded into $HQ(d + 1)$ with load 1 and dilation 2, such that only one edge is stretched.

Proof: Recursive embedding of the double-rooted tree as a sub-graph of the $HQ$. 
Lemma:

$T(d)$ may be embedded into $HQ(d + 1)$ with load 1 and dilation 2, such that only one edge is stretched.

Proof: Recursive embedding of the double-rooted tree as a sub-graph of the $HQ$. 

\[ T(d) \text{ into } HQ(d + 1) \]
Lemma:

\(T(d)\) may be embedded into \(HQ(d + 1)\) with load 1 and dilation 2, such that only one edge is stretched.

Proof: Recursive embedding of the double-rooted tree as a sub-graph of the \(HQ\).
Lemma:

$T(d)$ may be embedded into $HQ(d + 1)$ with load 1 and dilation 2, such that only one edge is stretched.

Proof: Recursive embedding of the double-rooted tree as a sub-graph of the $HQ$. 

![Diagram of tree and hypercube embeddings](attachment:image.png)
Lemma:

*T(d)* may be embedded into *DB(d + 1)* with load 1 and dilation 1.

Proof: \( f(w) \rightarrow 0^{d-|w|-1}1w \)

- Show: Edge of the tree is placed to an edge of the DeBruijn.
- Edge of the tree: \( w \) nach \( wa \)
- Placed to: \( 0^{n-|w|-1}1w \) and \( 0^{n-|w|-2}1wa \)
- That is a shuffle or shuffle-exchange edge in the DeBruijn.
- Note: there is a second edge-disjoined tree in the DeBruijn.
Lemma:

*T(d)* may be embedded into *DB(d + 1)* with load 1 and dilation 1.

Proof: \( f(w) \rightarrow 0^{d-|w|-1}1w \)

- **Show**: Edge of the tree is placed to an edge of the DeBruijn.
- Edge of the tree: \( w \) nach \( wa \)
- Placed to: \( 0^{n-|w|-1}1w \) and \( 0^{n-|w|-2}1wa \)
- That is a shuffle or shuffle-exchange edge in the DeBruijn.
- Note: there is a second edge-disjoined tree in the DeBruijn.
Lemma:

$T(d)$ may be embedded into $DB(d + 1)$ with load 1 and dilation 1.

Proof: $f(w) \rightarrow 0^{d-|w|-1}1w$

- Show: Edge of the tree is placed to an edge of the DeBruijn.
- Edge of the tree: $w$ nach $wa$
- Placed to: $0^{n-|w|-1}1w$ and $0^{n-|w|-2}1wa$
- That is a shuffle or shuffle-exchange edge in the DeBruijn.
- Note: there is a second edge-disjoined tree in the DeBruijn.
$T(d)$ into $DB(d + 1)$

**Lemma:**

$T(d)$ may be embedded into $DB(d + 1)$ with load 1 and dilation 1.

**Proof:** $f(w) \rightarrow 0^{d-|w|-1}1w$

- Show: Edge of the tree is placed to an edge of the DeBruijn.
- Edge of the tree: $w$ nach $wa$
- Placed to: $0^{n-|w|-1}1w$ and $0^{n-|w|-2}1wa$
- That is a shuffle or shuffle-exchange edge in the DeBruijn.
- Note: there is a second edge-disjoined tree in the DeBruijn.
Lemma:

$T(d)$ may be embedded into $DB(d + 1)$ with load 1 and dilation 1.

Proof: $f(w) \rightarrow 0^{d-|w|-1} 1w$

- Show: Edge of the tree is placed to an edge of the DeBruijn.
- Edge of the tree: $w$ nach $wa$
- Placed to: $0^{n-|w|-1} 1w$ and $0^{n-|w|-2} 1wa$
- That is a shuffle or shuffle-exchange edge in the DeBruijn.
- Note: there is a second edge-disjoined tree in the DeBruijn.
Lemma:

$T(d)$ may be embedded into $DB(d+1)$ with load 1 and dilation 1.

Proof: $f(w) \rightarrow 0^{d-|w|-1}1w$

- Show: Edge of the tree is placed to an edge of the DeBruijn.
- Edge of the tree: $w$ nach $wa$
- Placed to: $0^{n-|w|-1}1w$ and $0^{n-|w|-2}1wa$
- That is a shuffle or shuffle-exchange edge in the DeBruijn.
- Note: there is a second edge-disjoined tree in the DeBruijn.
Lemma:

\[ \text{CCC}(2d) \text{ may be embedded into } HQ(2d + \lceil \log 2d \rceil) \text{ with load 1 and dilation 1.} \]

Proof:

[Diagram showing the embedding process]
Lemma:

CCC(2d) may be embedded into HQ(2d + ⌈log 2d⌉) with load 1 and dilation 1.

Proof: Embed the cycles into sub-cubes.
CCC(4) into HQ (Example)
CCC(4) into HQ (Example)
CCC(4) into HQ (Example)
CCC(4) into HQ (Example)
CCC(4) into HQ (Example)
Steps of the Proof:

- **Embedd the cycles of length** $2d$ **into the** $HQ(\lceil \log 2d \rceil)$.
- Use the recursive embedding of the cycle of length $2^{\lceil \log 2d \rceil}$.
- **Note:**
  - IF $G$ is embedded in $H$ with dilation $k$ and
  - if $G'$ is embedded $H'$ with dilation $k'$, the we may
  - embed $G \times G'$ in $H \times H'$ with dilation $\max(k, k')$.
  - Holds due to the definition of the product of graphs.

- Furthermore we have: $CCC(2d)$ is a sub-graph of $C_{2d} \times HQ(2d)$.
- Also we have: $C_{2d} \times HQ(2d)$ is sub-graph of $HQ(2d + \lceil \log 2d \rceil)$. 
Steps of the Proof:

- Embed the cycles of length $2d$ into the $HQ(\lceil \log 2d \rceil)$.
- Use the recursive embedding of the cycle of length $2^{\lceil \log 2d \rceil}$.
- Note:
  - If $G$ is embedded in $H$ with dilation $k$ and
  - if $G'$ is embedded in $H'$ with dilation $k'$, then we may
  - embed $G \times G'$ in $H \times H'$ with dilation $\max(k, k')$.
  - Holds due to the definition of the product of graphs.

- Furthermore we have: $CCC(2d)$ is a sub-graph of $C_{2d} \times HQ(2d)$.
- Also we have: $C_{2d} \times HQ(2d)$ is sub-graph of $HQ(2d + \lceil \log 2d \rceil)$.
Steps of the Proof:

- Embed the cycles of length $2d$ into the $HQ(\lceil \log 2d \rceil)$.
- Use the recursive embedding of the cycle of length $2^{\lceil \log 2d \rceil}$.
- **Note:**
  - If $G$ is embedded in $H$ with dilation $k$ and if $G'$ is embedded in $H'$ with dilation $k'$, then we may
  - embed $G \times G'$ in $H \times H'$ with dilation $\max(k, k')$.
  - Holds due to the definition of the product of graphs.

- Furthermore we have: $CCC(2d)$ is a sub-graph of $C_{2d} \times HQ(2d)$.
- Also we have: $C_{2d} \times HQ(2d)$ is sub-graph of $HQ(2d + \lceil \log 2d \rceil)$. 
Steps of the Proof:

- Embed the cycles of length $2d$ into the $HQ(\lceil \log 2d \rceil)$.
- Use the recursive embedding of the cycle of length $2^{\lceil \log 2d \rceil}$.
- Note:
  - IF $G$ is embedded in $H$ with dilation $k$ and
  - if $G'$ is embedded $H'$ with dilation $k'$, the we may
  - embed $G \times G'$ in $H \times H'$ with dilation $\max(k, k')$.
  - Holds due to the definition of the product of graphs.

Furthermore we have: $CCC(2d)$ is a sub-graph of $C_{2d} \times HQ(2d)$.
Also we have: $C_{2d} \times HQ(2d)$ is sub-graph of $HQ(2d + \lceil \log 2d \rceil)$.
Steps of the Proof:

- Embed the cycles of length $2d$ into the $HQ(\lceil \log 2d \rceil)$.
- Use the recursive embedding of the cycle of length $2 \lceil \log 2d \rceil$.

Note:

- IF $G$ is embedded in $H$ with dilation $k$ and $G'$ is embedded in $H'$ with dilation $k'$, then we may embed $G \times G'$ in $H \times H'$ with dilation $\max(k, k')$.
- Holds due to the definition of the product of graphs.

Furthermore we have: $CCC(2d)$ is a sub-graph of $C_{2d} \times HQ(2d)$.
Also we have: $C_{2d} \times HQ(2d)$ is sub-graph of $HQ(2d + \lceil \log 2d \rceil)$. 
Steps of the Proof:

- Embed the cycles of length $2d$ into the $HQ(\lceil \log 2d \rceil)$.
- Use the recursive embedding of the cycle of length $2^{\lceil \log 2d \rceil}$.
- Note:
  - IF $G$ is embedded in $H$ with dilation $k$ and
  - if $G'$ is embedded $H'$ with dilation $k'$, then we may
  - embed $G \times G'$ in $H \times H'$ with dilation $\max(k, k')$.
- Holds due to the definition of the product of graphs.

Furthermore we have: $CCC(2d)$ is a sub-graph of $C_{2d} \times HQ(2d)$.

Also we have: $C_{2d} \times HQ(2d)$ is sub-graph of $HQ(2d + \lceil \log 2d \rceil)$. 
Steps of the Proof:

- Embed the cycles of length $2d$ into the $HQ(\lceil \log 2d \rceil)$.
- Use the recursive embedding of the cycle of length $2^\lceil \log 2d \rceil$.
- Note:
  - IF $G$ is embedded in $H$ with dilation $k$ and
  - if $G'$ is embedded $H'$ with dilation $k'$, the we may
  - embed $G \times G'$ in $H \times H'$ with dilation $\max(k, k')$.
  - Holds due to the definition of the product of graphs.
- Furthermore we have: $CCC(2d)$ is a sub-graph of $C_{2d} \times HQ(2d)$.
- Also we have: $C_{2d} \times HQ(2d)$ is sub-graph of $HQ(2d + \lceil \log 2d \rceil)$. 
Steps of the Proof:

- Embedd the cycles of length $2d$ into the $HQ(\lceil \log 2d \rceil)$.
- Use the recursive embedding of the cycle of length $2^{\lceil \log 2d \rceil}$.
- **Note:**
  - IF $G$ is embedded in $H$ with dilation $k$ and
  - if $G'$ is embedded $H'$ with dilation $k'$, the we may
  - embedd $G \times G'$ in $H \times H'$ with dilation $\max(k, k')$.
  - Holds due to the definition of the product of graphs.

- Furthermore we have: $CCC(2d)$ is a sub-graph of $C_{2d} \times HQ(2d)$.
- Also we have: $C_{2d} \times HQ(2d)$ is sub-graph of $HQ(2d + \lceil \log 2d \rceil)$. 
**Steps of the Proof:**

- Embed the cycles of length $2d$ into the $HQ(\lceil \log 2d \rceil)$.
- Use the recursive embedding of the cycle of length $2^\lceil \log 2d \rceil$.
- **Note:**
  - If $G$ is embedded in $H$ with dilation $k$ and
  - if $G'$ is embedded $H'$ with dilation $k'$, then we may
  - embed $G \times G'$ in $H \times H'$ with dilation $\max(k, k')$.
  - Holds due to the definition of the product of graphs.

- Furthermore we have: $CCC(2d)$ is a sub-graph of $C_{2d} \times HQ(2d)$.
- Also we have: $C_{2d} \times HQ(2d)$ is sub-graph of $HQ(2d + \lceil \log 2d \rceil)$. 
CCC(3) into HQ (Example)
CCC(3) into HQ (Example)
CCC(3) into HQ (Example)
CCC(3) into HQ (Example)
CCC(3) into HQ (Example)
Lemma:

CCC(2d − 1) may be embedded into HQ(2d − 1 + ⌈\log 2d − 1⌉) with load 1 and dilation 2.

Proof:

- Note: ⌈\log 2d⌉ = ⌈\log 2d − 1⌉.
- We have: CCC(2d − 1) is sub-graph of C_{2d−1} × HQ(2d − 1).
- Embed C(2d − 1) with dilation 2 in C(2d).
- The we get: C_{2d−1} × HQ(2d − 1) could be embedded with dilation 2 in C_{2d} × HQ(2d − 1).
- Already known: C_{2d} × HQ(2d) is sub-graph of HQ(2d + ⌈\log 2d⌉).
- Thus we get: C_{2d} × HQ(2d − 1) is sub-graph of HQ(2d + ⌈\log 2d⌉).
Lemma:

**CCC**$(2d - 1)$ may be embedded into **HQ**$(2d - 1 + \lceil \log 2d - 1 \rceil)$ with load 1 and dilation 2.

Proof:

- **Note:** $\lceil \log 2d \rceil = \lceil \log 2d - 1 \rceil$.
- We have: **CCC**$(2d - 1)$ is sub-graph of **$C_{2d-1} \times HQ(2d - 1)$**.
- Embedd **$C(2d - 1)$** with dilation 2 in **$C(2d)$**.
- The we get: **$C_{2d-1} \times HQ(2d - 1)$** could be embedded with dilation 2 in **$C_{2d} \times HQ(2d - 1)$**.
- Already known: **$C_{2d} \times HQ(2d)$** is sub-graph of **HQ**$(2d + \lceil \log 2d \rceil)$.
- Thus we get: **$C_{2d} \times HQ(2d - 1)$** is sub-graph of **HQ**$(2d + \lceil \log 2d \rceil)$. 
Lemma:

CCC\((2d - 1)\) may be embedded into HQ\((2d - 1 + \lceil \log 2d - 1 \rceil)\) with load 1 and dilation 2.

Proof:

- Note: \(\lceil \log 2d \rceil = \lceil \log 2d - 1 \rceil\).
- We have: CCC\((2d - 1)\) is sub-graph of \(C_{2d-1} \times HQ(2d - 1)\).
- Embedd \(C(2d - 1)\) with dilation 2 in \(C(2d)\).
- The we get: \(C_{2d-1} \times HQ(2d - 1)\) could be embedded with dilation 2 in \(C_{2d} \times HQ(2d - 1)\).
- Already known: \(C_{2d} \times HQ(2d)\) is sub-graph of HQ\((2d + \lceil \log 2d \rceil)\).
- Thus we get: \(C_{2d} \times HQ(2d - 1)\) is sub-graph of HQ\((2d + \lceil \log 2d \rceil)\).
CCC into HQ

**Lemma:**

CCC$(2d - 1)$ may be embedded into $HQ(2d - 1 + \lceil \log 2d - 1 \rceil)$ with load 1 and dilation 2.

**Proof:**

- **Note:** $\lceil \log 2d \rceil = \lceil \log 2d - 1 \rceil$.
- We have: CCC$(2d - 1)$ is sub-graph of $C_{2d-1} \times HQ(2d - 1)$.
- Embedd $C(2d - 1)$ with dilation 2 in $C(2d)$.
- The we get: $C_{2d-1} \times HQ(2d - 1)$ could be embedded with dilation 2 in $C_{2d} \times HQ(2d - 1)$.
- Already known: $C_{2d} \times HQ(2d)$ is sub-graph of $HQ(2d + \lceil \log 2d \rceil)$.
- Thus we get: $C_{2d} \times HQ(2d - 1)$ is sub-graph of $HQ(2d + \lceil \log 2d \rceil)$.
Lemma:

CCC\((2d - 1)\) may be embedded into \(HQ(2d - 1 + \lceil\log 2d - 1\rceil)\) with load 1 and dilation 2.

Proof:

- Note: \(\lceil\log 2d\rceil = \lceil\log 2d - 1\rceil\).
- We have: \(CCC(2d - 1)\) is sub-graph of \(C_{2d-1} \times HQ(2d - 1)\).
- Embedd \(C(2d - 1)\) with dilation 2 in \(C(2d)\).
- The we get: \(C_{2d-1} \times HQ(2d - 1)\) could be embedded with dilation 2 in \(C_{2d} \times HQ(2d - 1)\).
- Already known: \(C_{2d} \times HQ(2d)\) is sub-graph of \(HQ(2d + \lceil\log 2d\rceil)\).
- Thus we get: \(C_{2d} \times HQ(2d - 1)\) is sub-graph of \(HQ(2d + \lceil\log 2d\rceil)\).
Lemma:

$\text{CCC}(2d - 1)$ may be embedded into $\text{HQ}(2d - 1 + \lceil \log 2d - 1 \rceil)$ with load 1 and dilation 2.

Proof:

- Note: $\lceil \log 2d \rceil = \lceil \log 2d - 1 \rceil$.
- We have: $\text{CCC}(2d - 1)$ is sub-graph of $C_{2d-1} \times \text{HQ}(2d - 1)$.
- Embed $C(2d - 1)$ with dilation 2 in $C(2d)$.
- The we get: $C_{2d-1} \times \text{HQ}(2d - 1)$ could be embedded with dilation 2 in $C_{2d} \times \text{HQ}(2d - 1)$.
- Already known: $C_{2d} \times \text{HQ}(2d)$ is sub-graph of $\text{HQ}(2d + \lceil \log 2d \rceil)$.
- Thus we get: $C_{2d} \times \text{HQ}(2d - 1)$ is sub-graph of $\text{HQ}(2d + \lceil \log 2d \rceil)$. 
CCC into HQ

Lemma:

\[ \text{CCC}(2d - 1) \text{ may be embedded into } \text{HQ}(2d - 1 + \lceil \log 2d - 1 \rceil) \text{ with load } 1 \text{ and dilation } 2. \]

Proof:

- Note: \( \lceil \log 2d \rceil = \lceil \log 2d - 1 \rceil. \)
- We have: \( \text{CCC}(2d - 1) \) is sub-graph of \( C_{2d-1} \times \text{HQ}(2d - 1). \)
- Embedd \( C(2d - 1) \) with dilation 2 in \( C(2d). \)
- The we get: \( C_{2d-1} \times \text{HQ}(2d - 1) \) could be embedded with dilation 2 in \( C_{2d} \times \text{HQ}(2d - 1). \)
- Already known: \( C_{2d} \times \text{HQ}(2d) \) is sub-graph of \( \text{HQ}(2d + \lceil \log 2d \rceil). \)
- Thus we get: \( C_{2d} \times \text{HQ}(2d - 1) \) is sub-graph of \( \text{HQ}(2d + \lceil \log 2d \rceil). \)
Lemma:

\( BF(d) \) may be embedded into \( HQ(d + \lceil \log d \rceil) \) with load 1 and dilation 2.

Proof:

- Embed \( BF(d) \) in \( CCC(d) \) with dilation 2 (trivial).
- Embed \( CCC(d) \) in \( HQ(d + \lceil \log d \rceil) \) with dilation 1.
**Lemma:**

$BF(d)$ may be embedded into $HQ(d + \lceil \log d \rceil)$ with load 1 and dilation 2.

**Proof:**

- Embed $BF(d)$ in $CCC(d)$ with dilation 2 (trivial).
- Embed $CCC(d)$ in $HQ(d + \lceil \log d \rceil)$ with dilation 1.
Lemma:

$BF(d)$ may be embedded into $HQ(d + \lceil \log d \rceil)$ with load 1 and dilation 2.

Proof:

- Embed $BF(d)$ in $CCC(d)$ with dilation 2 (trivial).
- Embed $CCC(d)$ in $HQ(d + \lceil \log d \rceil)$ with dilation 1.
Lemma:

$BF(2d)$ may be embedded into $HQ(2d + \lceil \log 2d \rceil)$ with load 1 and dilation 1.
BF(4) in HQ (Beispiel)
BF(4) in HQ (Beispiel)
BF(4) in HQ (Beispiel)
BF(4) in HQ (Beispiel)
BF(4) in HQ (Beispiel)
BF(4) in HQ (Beispiel)
BF(4) in HQ (Beispiel)
BF(4) in HQ (Beispiel)
BF(4) in HQ (Beispiel)
BF(4) in HQ (Beispiel)
BF(4) in HQ (Beispiel)
BF(4) in HQ (Beispiel)
BF(4) in HQ (Beispiel)
Steps of the Proof:

- **Embed cycle** $C_{2d}$ into $HQ(\lceil \log 2d \rceil)$ as a subgraph by some function $f_C$.
- **Embed** $BF_{2d}$ into $HQ(2d + \lceil \log 2d \rceil)$:
  \[
  (i, w) \mapsto f_{2d}(i)w
  \]

- **Assume** that $(i, w)$ is now embedded onto $cw$ for $0 \leq i < 2d$ and $w \in \{0, 1\}^{2d}$.
- **For** $i$ from 0 to $2d - 1$ do the following:
  - Let $i' = (i + 1) \mod 2d$.
  - Exchange now node of the form $(i, w)$ with $(i', w)$ for $w = w'1w''$ with $|w'| = i$.
  - Let $t = f_{2d}(i) \oplus f_{2d}(i')$.
  - Let $cw'1w''$ be a node of the hypercube.
  - The move $cw'1w''$ to $(c \oplus t)w'1w''$.
  - Note, the dilation is not enlarged for any edge.
  - The edges of the form $\{(i, w'0w''), (i', w'1w'')\}$ have now a dilation of 1.
Steps of the Proof:

- Embedd cycle $C_{2d}$ into $HQ(\lceil \log 2d \rceil)$ as a subgraph by some function $f_C$.
- Embedd $BF_{2d}$ into $HQ(2d + \lceil \log 2d \rceil)$:

$$(i, w) \mapsto f_{2d}(i)w$$

- Assume that $(i, w)$ is now embedded onto $cw$ for $0 \leq i < 2d$ and $w \in \{0, 1\}^{2d}$.
- For $i$ from 0 to $2d - 1$ do the following:
  - Let $i' = (i + 1) \mod 2d$.
  - Exchange now node of the form $(i, w)$ with $(i', w)$ for $w = w'1w''$ with $|w'| = i$.
  - Let $t = f_{2d}(i) \oplus f_{2d}(i')$.
  - Let $cw'1w''$ be a node of the hypercube.
  - The move $cw'1w''$ to $(c \oplus t)w'1w''$.
  - Note, the dilation is not enlarged for any edge.
  - The edges of the form $\{(i, w'0w''), (i', w'1w'')\}$ have now a dilation of 1.
BF into HQ

Steps of the Proof:

- Embed cycle $C_{2d}$ into $HQ(\lceil\log 2d\rceil)$ as a subgraph by some function $f_C$.
- Embed $BF_{2d}$ into $HQ(2d + \lceil\log 2d\rceil)$:
  \[
  (i, w) \mapsto f_{2d}(i)w
  \]

- Assume that $(i, w)$ is now embedded onto $cw$ for $0 \leq i < 2d$ and $w \in \{0, 1\}^{2d}$.
- For $i$ from 0 to $2d - 1$ do the following:
  - Let $i' = (i + 1) \mod 2d$.
  - Exchange now node of the form $(i, w)$ with $(i', w)$ for $w = w'1w''$ with $|w'| = i$.
  - Let $t = f_{2d}(i) \oplus f_{2d}(i')$.
  - Let $cw'1w''$ be a node of the hypercube.
  - The move $cw'1w''$ to $(c \oplus t)w'1w''$.
  - Note, the dilation is not enlarged for any edge.
  - The edges of the form $\{(i, w'0w''), (i', w'1w'')\}$ have now a dilation of 1.
Steps of the Proof:

- Embed cycle $C_{2d}$ into $HQ(\lceil \log 2d \rceil)$ as a subgraph by some function $f_C$.
- Embed $BF_{2d}$ into $HQ(2d + \lceil \log 2d \rceil)$:
  \[(i, w) \mapsto f_{2d}(i)w\]

- Assume that $(i, w)$ is now embedded onto $cw$ for $0 \leq i < 2d$ and $w \in \{0, 1\}^{2d}$.
- For $i$ from 0 to $2d - 1$ do the following:
  - Let $i' = (i + 1) \mod 2d$.
  - Exchange now node of the form $(i, w)$ with $(i', w)$ for $w = w'1w''$ with $|w'| = i$.
  - Let $t = f_{2d}(i) \oplus f_{2d}(i')$.
  - Let $cw'1w''$ be a node of the hypercube.
  - The move $cw'1w''$ to $(c \oplus t)w'1w''$.
  - Note, the dilation is not enlarged for any edge.
  - The edges of the form $\{(i, w'0w''), (i', w'1w'')\}$ have now a dilation of 1.
BF into HQ

Steps of the Proof:

- Embed cycle $C_{2d}$ into $HQ(\lceil \log 2d \rceil)$ as a subgraph by some function $f_C$.
- Embed $BF_{2d}$ into $HQ(2d + \lceil \log 2d \rceil)$:
  
  $$(i, w) \mapsto f_{2d}(i)w$$

- Assume that $(i, w)$ is now embedded onto $cw$ for $0 \leq i < 2d$ and $w \in \{0, 1\}^{2d}$.
- For $i$ from 0 to $2d - 1$ do the following:
  
  - Let $i' = (i + 1) \mod 2d$.
  - Exchange now node of the form $(i, w)$ with $(i', w)$ for $w = w'1w''$ with $|w'| = i$.
  - Let $t = f_{2d}(i) \oplus f_{2d}(i')$.
  - Let $cw'1w''$ be a node of the hypercube.
  - The move $cw'1w''$ to $(c \oplus t)w'1w''$.
  - Note, the dilation is not enlarged for any edge.
  - The edges of the form $\{(i, w'0w''), (i', w'1w'')\}$ have now a dilation of 1.
BF into HQ

Steps of the Proof:

- Embed cycle $C_{2d}$ into $HQ(\lceil \log 2d \rceil)$ as a subgraph by some function $f_C$.
- Embed $BF_{2d}$ into $HQ(2d + \lceil \log 2d \rceil)$:
  \[(i, w) \mapsto f_{2d}(i)w\]

- Assume that $(i, w)$ is now embedded onto $cw$ for $0 \leq i < 2d$ and $w \in \{0, 1\}^{2d}$.
- For $i$ from 0 to $2d - 1$ do the following:
  - Let $i' = (i + 1) \text{ mod } 2d$.
  - Exchange now node of the form $(i, w)$ with $(i', w)$ for $w = w'1w''$ with $|w'| = i$.
  - Let $t = f_{2d}(i) \oplus f_{2d}(i')$.
  - Let $cw'1w''$ be a node of the hypercube.
  - The move $cw'1w''$ to $(c \oplus t)w'1w''$.
  - Note, the dilation is not enlarged for any edge.
  - The edges of the form $\{(i, w'0w''), (i', w'1w'')\}$ have now a dilation of 1.
Steps of the Proof:

- Embed cycle $C_{2d}$ into $HQ(\lceil \log 2d \rceil)$ as a subgraph by some function $f_C$.
- Embed $BF_{2d}$ into $HQ(2d + \lceil \log 2d \rceil)$:

  $$(i, w) \mapsto f_{2d}(i)w$$

- Assume that $(i, w)$ is now embedded onto $cw$ for $0 \leq i < 2d$ and $w \in \{0, 1\}^{2d}$.
- For $i$ from 0 to $2d - 1$ do the following:
  - Let $i' = (i + 1) \mod 2d$.
  - Exchange now node of the form $(i, w)$ with $(i', w)$ for $w = w'1w''$ with $|w'| = i$.
  - Let $t = f_{2d}(i) \oplus f_{2d}(i')$.
  - Let $cw'1w''$ be a node of the hypercube.
  - The move $cw'1w''$ to $(c \oplus t)w'1w''$.
  - Note, the dilation is not enlarged for any edge.
  - The edges of the form $\{(i, w'0w''), (i', w'1w'')\}$ have now a dilation of 1.
BF into HQ

Steps of the Proof:

- Embed cycle $C_{2d}$ into $HQ(\lceil \log 2d \rceil)$ as a subgraph by some function $f_C$.
- Embed $BF_{2d}$ into $HQ(2d + \lceil \log 2d \rceil)$:

  $$(i, w) \mapsto f_{2d}(i)w$$

- Assume that $(i, w)$ is now embedded onto $cw$ for $0 \leq i < 2d$ and $w \in \{0, 1\}^{2d}$.
- For $i$ from 0 to $2d - 1$ do the following:
  - Let $i' = (i + 1) \mod 2d$.
  - Exchange now node of the form $(i, w)$ with $(i', w)$ for $w = w'1w''$ with $|w'| = i$.
  - Let $t = f_{2d}(i) \oplus f_{2d}(i')$.
  - Let $cw'1w''$ be a node of the hypercube.
  - The move $cw'1w''$ to $(c \oplus t)w'1w''$.
  - Note, the dilation is not enlarged for any edge.
  - The edges of the form $\{(i, w'0w''), (i', w'1w'')\}$ have now a dilation of 1.
BF into HQ

Steps of the Proof:

- Embed cycle $C_{2d}$ into $HQ(\lceil\log 2d\rceil)$ as a subgraph by some function $f_C$.
- Embed $BF_{2d}$ into $HQ(2d + \lceil\log 2d\rceil)$:

  $$(i, w) \mapsto f_{2d}(i)w$$

- Assume that $(i, w)$ is now embedded onto $cw$ for $0 \leq i < 2d$ and $w \in \{0, 1\}^{2d}$.
- For $i$ from 0 to $2d - 1$ do the following:
  
  - Let $i' = (i + 1) \mod 2d$.
  - Exchange now node of the form $(i, w)$ with $(i', w)$ for $w = w'1w''$ with $|w'| = i$.
  - Let $t = f_{2d}(i) \oplus f_{2d}(i')$.
  - Let $cw'1w''$ be a node of the hypercube.
  - The move $cw'1w''$ to $(c \oplus t)w'1w''$.
  - Note, the dilation is not enlarged for any edge.
  - The edges of the form $\{(i, w'0w''), (i', w'1w'')\}$ have now a dilation of 1.
BF into HQ

Steps of the Proof:

- Embed cycle $C_{2d}$ into $HQ(\lceil \log 2d \rceil)$ as a subgraph by some function $f_C$.
- Embed $BF_{2d}$ into $HQ(2d + \lceil \log 2d \rceil)$:

  $$(i, w) \mapsto f_{2d}(i)w$$

- Assume that $(i, w)$ is now embedded onto $cw$ for $0 \leq i < 2d$ and $w \in \{0, 1\}^{2d}$.
- For $i$ from 0 to $2d - 1$ do the following:
  - Let $i' = (i + 1) \text{ mod } 2d$.
  - Exchange now node of the form $(i, w)$ with $(i', w)$ for $w = w'1w''$ with $|w'| = i$.
  - Let $t = f_{2d}(i) \oplus f_{2d}(i')$.
  - Let $cw'1w''$ be a node of the hypercube.
  - The move $cw'1w''$ to $(c \oplus t)w'1w''$.
  - Note, the dilation is not enlarged for any edge.
  - The edges of the form $\{(i, w'0w''), (i', w'1w'')\}$ have now a dilation of 1.
BF into HQ

Steps of the Proof:

- Embed cycle $C_{2d}$ into $HQ(\lceil \log 2d \rceil)$ as a subgraph by some function $f_C$.
- Embed $BF_{2d}$ into $HQ(2d + \lceil \log 2d \rceil)$:
  \[(i, w) \mapsto f_{2d}(i)w\]

- Assume that $(i, w)$ is now embedded onto $cw$ for $0 \leq i < 2d$ and $w \in \{0, 1\}^{2d}$.
- For $i$ from 0 to $2d - 1$ do the following:
  - Let $i' = (i + 1) \mod 2d$.
  - Exchange now node of the form $(i, w)$ with $(i', w)$ for $w = w'1w''$ with $|w'| = i$.
  - Let $t = f_{2d}(i) \oplus f_{2d}(i')$.
  - Let $cw'1w''$ be a node of the hypercube.
  - The move $cw'1w''$ to $(c \oplus t)w'1w''$.
  - Note, the dilation is not enlarged for any edge.
  - The edges of the form \{$(i, w'0w''), (i', w'1w'')$\} have now a dilation of 1.
Lemma:

$CCC(d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.
Lemma:

$CCC(d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.
Lemma:

$CCC(d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.
Lemma:

$CCC(d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.
**Lemma:**

$CCC(d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.
**Lemma:**

$CCC(d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.
**Lemma:**

$CCC(d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.

- Let $P(w) := \#_1(w) \mod 2$.
- $f(i, w) = (i, w)$ if $P(w) = 0$.
- $f(i, w) = ((i + 1) \mod d, w)$ if $P(w) = 1$.
- Consider the edges on the cycles: $\{(i, w), ((i + 1) \mod d, w)\}$:
  - $w_i$ has the $i^{th}$ bit of $w$ flipped.
  - $f(i, w) = (i, w)$ if $P(w) = 0$.
  - $f((i + 1) \mod d, w) = ((i + 1) \mod d, w)$ if $P(w) = 0$.
  - $f(i, w) = ((i + 1) \mod d, w)$ if $P(w) = 1$.
  - $f((i + 1) \mod d, w) = ((i + 2) \mod d, w)$ if $P(w) = 1$. 
Lemma:

**CCC**($d$) may be embedded into **BF**($d$) with load 1 and dilation 1.

- Let $P(w) := \#_1(w) \mod 2$.
- $f(i, w) = (i, w)$ if $P(w) = 0$.
- $f(i, w) = ((i + 1) \mod d, w)$ if $P(w) = 1$.

Consider the edges on the cycles: $\{(i, w), ((i + 1) \mod d, w)\}$:

- $w_i$ has the $i^{th}$ bit of $w$ flipped.
- $f(i, w) = (i, w)$ if $P(w) = 0$.
- $f(((i + 1) \mod d, w) = ((i + 1) \mod d, w)$ if $P(w) = 0$.
- $f(i, w) = ((i + 1) \mod d, w)$ if $P(w) = 1$.
- $f(((i + 1) \mod d, w) = ((i + 2) \mod d, w)$ if $P(w) = 1$. 
Lemma:

\( \text{CCC}(d) \) may be embedded into \( \text{BF}(d) \) with load 1 and dilation 1.

- Let \( P(w) := \#_1(w) \mod 2 \).
- \( f(i, w) = (i, w) \) if \( P(w) = 0 \).
- \( f(i, w) = ((i + 1) \mod d, w) \) if \( P(w) = 1 \).
- Consider the edges on the cycles: \( \{(i, w), ((i + 1) \mod d, w)\} \):
  - \( w_i \) has the \( i^{\text{th}} \) bit of \( w \) flipped.
  - \( f(i, w) = (i, w) \) if \( P(w) = 0 \).
  - \( f(((i + 1) \mod d, w) = ((i + 1) \mod d, w) \) if \( P(w) = 0 \).
  - \( f(i, w) = ((i + 1) \mod d, w) \) if \( P(w) = 1 \).
  - \( f(((i + 1) \mod d, w) = ((i + 2) \mod d, w) \) if \( P(w) = 1 \).
Lemma:

$CCC(d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.

- Let $P(w) := \#_1(w) \mod 2$.
- $f(i, w) = (i, w)$ if $P(w) = 0$.
- $f(i, w) = ((i + 1) \mod d, w)$ if $P(w) = 1$.
- Consider the edges on the cycles: $\{(i, w), ((i + 1) \mod d, w)\}$:
  - $w_i$ has the $i^{th}$ bit of $w$ flipped.
  - $f(i, w) = (i, w)$ if $P(w) = 0$.
  - $f(((i + 1) \mod d, w) = ((i + 1) \mod d, w)$ if $P(w) = 0$.
  - $f(i, w) = ((i + 1) \mod d, w)$ if $P(w) = 1$.
  - $f(((i + 1) \mod d, w) = ((i + 2) \mod d, w)$ if $P(w) = 1$. 
Lemma:

$CCC(d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.

- Let $P(w) := \#_1(w) \mod 2$.
- $f(i, w) = (i, w)$ if $P(w) = 0$.
- $f(i, w) = ((i + 1) \mod d, w)$ if $P(w) = 1$.
- Consider the edges on the cycles: $\{(i, w), ((i + 1) \mod d, w)\}$:
  - $w_i$ has the $i^{th}$ bit of $w$ flipped.
  - $f(i, w) = (i, w)$ if $P(w) = 0$.
  - $f((i + 1) \mod d, w) = ((i + 1) \mod d, w)$ if $P(w) = 0$.
  - $f(i, w) = ((i + 1) \mod d, w)$ if $P(w) = 1$.
  - $f((i + 1) \mod d, w) = ((i + 2) \mod d, w)$ if $P(w) = 1$. 
Lemma:

$CCC(d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.

- Let $P(w) := \#_1(w) \mod 2$.
- $f(i, w) = (i, w)$ if $P(w) = 0$.
- $f(i, w) = ((i + 1) \mod d, w)$ if $P(w) = 1$.
- Consider the edges on the cycles: $\{(i, w), ((i + 1) \mod d, w)\}$:
  - $w_i$ has the $i^{th}$ bit of $w$ flipped.
  - $f(i, w) = (i, w)$ if $P(w) = 0$.
  - $f(((i + 1) \mod d, w) = ((i + 1) \mod d, w)$ if $P(w) = 0$.
  - $f(i, w) = ((i + 1) \mod d, w)$ if $P(w) = 1$.
  - $f(((i + 1) \mod d, w) = ((i + 2) \mod d, w)$ if $P(w) = 1$. 
Lemma:

\[ \text{CCC}(d) \] may be embedded into \[ \text{BF}(d) \] with load 1 and dilation 1.

- Let \( P(w) := \#_1(w) \mod 2. \)
- \( f(i, w) = (i, w) \) if \( P(w) = 0. \)
- \( f(i, w) = ((i + 1) \mod d, w) \) if \( P(w) = 1. \)
- Consider the edges on the cycles: \( \{(i, w), ((i + 1) \mod d, w)\} \):
  - \( w \) has the \( i \)th bit of \( w \) flipped.
  - \( f(i, w) = (i, w) \) if \( P(w) = 0. \)
  - \( f(((i + 1) \mod d, w) = ((i + 1) \mod d, w) \) if \( P(w) = 0. \)
  - \( f(i, w) = ((i + 1) \mod d, w) \) if \( P(w) = 1. \)
  - \( f(((i + 1) \mod d, w) = ((i + 2) \mod d, w) \) if \( P(w) = 1. \).
Lemma:

$CCC(d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.

- Let $P(w) := \#_1(w) \mod 2$.
- $f(i, w) = (i, w)$ if $P(w) = 0$.
- $f(i, w) = ((i + 1) \mod d, w)$ if $P(w) = 1$.
- Consider the edges on the cycles: $\{(i, w), ((i + 1) \mod d, w)\}$:
  - $w_i$ has the $i^{th}$ bit of $w$ flipped.
  - $f(i, w) = (i, w)$ if $P(w) = 0$.
  - $f(((i + 1) \mod d, w) = ((i + 1) \mod d, w)$ if $P(w) = 0$.
  - $f(i, w) = ((i + 1) \mod d, w)$ if $P(w) = 1$.
  - $f(((i + 1) \mod d, w) = ((i + 2) \mod d, w)$ if $P(w) = 1$. 
**Lemma:**

$CCC(d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.

- Let $P(w) := \#_1(w) \mod 2$.
- $f(i, w) = (i, w)$ if $P(w) = 0$.
- $f(i, w) = ((i + 1) \mod d, w)$ if $P(w) = 1$.
- Consider the edges on the cycles: $\{(i, w), ((i + 1) \mod d, w)\}$:
  - $w_i$ has the $i^{th}$ bit of $w$ flipped.
  - $f(i, w) = (i, w)$ if $P(w) = 0$.
  - $f((i + 1) \mod d, w) = ((i + 1) \mod d, w)$ if $P(w) = 0$.
  - $f(i, w) = ((i + 1) \mod d, w)$ if $P(w) = 1$.
  - $f((i + 1) \mod d, w) = ((i + 2) \mod d, w)$ if $P(w) = 1$. 

Lemma:

$CCC(d)$ may be embedded into $BF(d)$ with load 1 and dilation 1.

- Let $P(w) := \#_1(w) \mod 2$.
- $f(i, w) = (i, w)$ if $P(w) = 0$.
- $f(i, w) = ((i + 1) \mod d, w)$ if $P(w) = 1$.
- Consider the edges on the cycles: $\{(i, w), ((i + 1) \mod d, w)\}$:
  - $w_i$ has the $i^{th}$ bit of $w$ flipped.
  - $f(i, w) = (i, w)$ if $P(w) = 0$.
  - $f((i + 1) \mod d, w) = ((i + 1) \mod d, w)$ if $P(w) = 0$.
  - $f(i, w) = ((i + 1) \mod d, w)$ if $P(w) = 1$.
  - $f((i + 1) \mod d, w) = ((i + 2) \mod d, w)$ if $P(w) = 1$. 
CCC into BF

- \( f(i, w) = (i, w) \) if \( P(w) = 0 \).
- \( f(i, w) = ((i + 1) \mod d, w) \) if \( P(w) = 1 \).
- Consider the cube-edges: \( \{(i, w), (i, w_i)\} \):
  - \( f(i, w) = (i, w) \) if \( P(w) = 0 \).
  - \( f(i, w_i) = ((i + 1) \mod d, w_i) \) if \( P(w) = 0 \).
  - \( f(i, w) = ((i + 1) \mod d, w) \) if \( P(w) = 1 \).
  - \( f(i, w_i) = (i, w_i) \) if \( P(w) = 1 \).
CCC into BF

- $f(i, w) = (i, w)$ if $P(w) = 0$.
- $f(i, w) = ((i + 1) \mod d, w)$ if $P(w) = 1$.

Consider the cube-edges: $\{(i, w), (i, w_i)\}$:
- $f(i, w) = (i, w)$ if $P(w) = 0$.
- $f(i, w_i) = ((i + 1) \mod d, w_i)$ if $P(w) = 0$.
- $f(i, w) = ((i + 1) \mod d, w)$ if $P(w) = 1$.
- $f(i, w_i) = (i, w_i)$ if $P(w) = 1$. 
CCC into BF

- \( f(i, w) = (i, w) \) if \( P(w) = 0 \).
- \( f(i, w) = ((i + 1) \mod d, w) \) if \( P(w) = 1 \).

Consider the cube-edges: \( \{(i, w), (i, w_i)\} \):

- \( f(i, w) = (i, w) \) if \( P(w) = 0 \).
- \( f(i, w_i) = ((i + 1) \mod d, w_i) \) if \( P(w) = 0 \).
- \( f(i, w) = ((i + 1) \mod d, w) \) if \( P(w) = 1 \).
- \( f(i, w_i) = (i, w_i) \) if \( P(w) = 1 \).
CCC into BF

- \( f(i, w) = (i, w) \) if \( P(w) = 0 \).
- \( f(i, w) = ((i + 1) \mod d, w) \) if \( P(w) = 1 \).

Consider the cube-edges: \( \{(i, w), (i, w_i)\} \):
  - \( f(i, w) = (i, w) \) if \( P(w) = 0 \).
  - \( f(i, w_i) = ((i + 1) \mod d, w_i) \) if \( P(w) = 0 \).
  - \( f(i, w) = ((i + 1) \mod d, w) \) if \( P(w) = 1 \).
  - \( f(i, w_i) = (i, w_i) \) if \( P(w) = 1 \).
CCC into BF

- \( f(i, w) = (i, w) \) if \( P(w) = 0 \).
- \( f(i, w) = ((i + 1) \mod d, w) \) if \( P(w) = 1 \).

Consider the cube-edges: \( \{(i, w), (i, w_i)\} \):

- \( f(i, w) = (i, w) \) if \( P(w) = 0 \).
- \( f(i, w_i) = ((i + 1) \mod d, w_i) \) if \( P(w) = 0 \).
- \( f(i, w) = ((i + 1) \mod d, w) \) if \( P(w) = 1 \).
- \( f(i, w_i) = (i, w_i) \) if \( P(w) = 1 \).
CCC into BF

- \( f(i, w) = (i, w) \) if \( P(w) = 0 \).
- \( f(i, w) = ((i + 1) \mod d, w) \) if \( P(w) = 1 \).
- Consider the cube-edges: \( \{ (i, w), (i, w_i) \} \):
  - \( f(i, w) = (i, w) \) if \( P(w) = 0 \).
  - \( f(i, w_i) = ((i + 1) \mod d, w_i) \) if \( P(w) = 0 \).
  - \( f(i, w) = ((i + 1) \mod d, w) \) if \( P(w) = 1 \).
  - \( f(i, w_i) = (i, w_i) \) if \( P(w) = 1 \).
CCC into BF

- $f(i, w) = (i, w)$ if $P(w) = 0$. 
- $f(i, w) = ((i + 1) \mod d, w)$ if $P(w) = 1$. 

Consider the cube-edges: $\{(i, w), (i, w_i)\}$:

- $f(i, w) = (i, w)$ if $P(w) = 0$. 
- $f(i, w_i) = ((i + 1) \mod d, w_i)$ if $P(w) = 0$. 
- $f(i, w) = ((i + 1) \mod d, w)$ if $P(w) = 0$. 
- $f(i, w_i) = (i, w_i)$ if $P(w) = 1$. 

Lemma:

$SE(d)$ may be embedded into $DB(d)$ with load 1 and dilation 1.

Proof: Exercise
**Lemma:**

$DB(d)$ may be embedded into $HQ(d)$ with load 1 and dilation $\lceil d/4 \rceil$.

**Proof:**

- Consider edge in DB: $aw \leftrightarrow wb$.
- Split the node-strings into blocks: $awa'w' \leftrightarrow wbw'b'$ with $b = a'$.
- This makes small virtual DeBruijn within the original DeBruijn.
- Each virtual part is embedded in a hyper-cubes.
- The dilation sums up during this process.
- The proof is done by embedding the $DB(8)$ into the $HQ(8)$ with dilation 2.
Lemma:

$DB(d)$ may be embedded into $HQ(d)$ with load 1 and dilation $\lceil d/4 \rceil$.

Proof:

- Consider edge in $DB$: $aw \leftrightarrow wb$.
- Split the node-strings into blocks: $awa'w' \leftrightarrow wbw'b'$ with $b = a'$.
- This makes small virtual DeBruijn within the original DeBruijn.
- Each virtual part is embedded in a hyper-cubes.
- The dilation sums up during this process.
- The proof is done by embedding the $DB(8)$ into the $HQ(8)$ with dilation 2.
Lemma:

$DB(d)$ may be embedded into $HQ(d)$ with load 1 and dilation $\lceil d/4 \rceil$.

Proof:

- Consider edge in DB: $aw \leftrightarrow wb$.
- Split the node-strings into blocks: $awa'w' \leftrightarrow wbw'b'$ with $b = a'$.
- This makes small virtual DeBruijn within the original DeBruijn.
- Each virtual part is embedded in a hyper-cubes.
- The dilation sums up during this process.
- The proof is done by embedding the $DB(8)$ into the $HQ(8)$ with dilation 2.
Lemma:

$DB(d)$ may be embedded into $HQ(d)$ with load 1 and dilation $\lceil d/4 \rceil$.

Proof:

- Consider edge in DB: $aw \leftrightarrow wb$.
- Split the node-strings into blocks: $awa'w' \leftrightarrow wbw'b'$ with $b = a'$.
- This makes small virtual DeBruijn within the original DeBruijn.
- Each virtual part is embedded in a hyper-cubes.
- The dilation sums up during this process.
- The proof is done by embedding the $DB(8)$ into the $HQ(8)$ with dilation 2.
Lemma:

DB\(d\) may be embedded into HQ\(d\) with load 1 and dilation \(\lceil d/4 \rceil\).

Proof:

- Consider edge in DB: \(aw \leftrightarrow wb\).
- Split the node-strings into blocks: \(awa'w' \leftrightarrow wbw'b'\) with \(b = a'\).
- This makes small virtual DeBruijn within the original DeBruijn.
- Each virtual part is embedded in a hyper-cubes.
- The dilation sums up during this process.
- The proof is done by embedding the DB\(8\) into the HQ\(8\) with dilation 2.
Lemma:

$DB(d)$ may be embedded into $HQ(d)$ with load 1 and dilation $\lceil d/4 \rceil$.

Proof:

- Consider edge in DB: $aw \leftrightarrow wb$.
- Split the node-strings into blocks: $awa'w' \leftrightarrow wbw'b'$ with $b = a'$.
- This makes small virtual DeBruijn within the original DeBruijn.
- Each virtual part is embedded in a hyper-cubes.
- The dilation sums up during this process.
- The proof is done by embedding the $DB(8)$ into the $HQ(8)$ with dilation 2.
Lemma:

$DB(d)$ may be embedded into $HQ(d)$ with load 1 and dilation $\lceil d/4 \rceil$.

Proof:
- Consider edge in DB: $aw \leftrightarrow wb$.
- Split the node-strings into blocks: $awa'w' \leftrightarrow wbw'b'$ with $b = a'$.
- This makes small virtual DeBruijn within the original DeBruijn.
- Each virtual part is embedded in a hyper-cubes.
- The dilation sums up during this process.
- The proof is done by embedding the $DB(8)$ into the $HQ(8)$ with dilation 2.
Lemma:

$G(n_1, n_2, \ldots, n_t)$ may be embedded into $HQ(d)$ with load 1 and dilation 1, iff $d \geq \sum_{i=1}^{t} \lceil \log n_i \rceil$.

Proof:

- Check the dimension-changes of the edges of the grid:
- In each square are precisely 2 dimensions.
- Thus each path of the form $L(n_i)$ has to be embedded into a sub-cube.

Lemma:

$TR(n_1, n_2, \ldots, n_t)$ may be embedded into $HQ(d)$ with load 1 and dilation 1, iff $d \geq \sum_{i=1}^{t} \lceil \log n_i \rceil$ and all $n_i$ are even.
Torus and Hypercube

Lemma:

\[ G(n_1, n_2, \cdots, n_t) \] may be embedded into \( HQ(d) \) with load 1 and dilation 1, iff \( d \geq \sum_{i=1}^{t} \lceil \log n_i \rceil \).

Proof:

- Check the dimension-changes of the edges of the grid:
- In each square are precisely 2 dimensions.
- Thus each path of the form \( L(n_i) \) has to be embedded into a sub-cube.

Lemma:

\[ TR(n_1, n_2, \cdots, n_t) \] may be embedded into \( HQ(d) \) with load 1 and dilation 1, iff \( d \geq \sum_{i=1}^{t} \lceil \log n_i \rceil \) and all \( n_i \) are even.
Torus and Hypercube

Lemma:

\[ G(n_1, n_2, \ldots, n_t) \text{ may be embedded into } HQ(d) \text{ with load 1 and dilation 1, iff } d \geq \sum_{i=1}^{t} \lceil \log n_i \rceil. \]

Proof:

- Check the dimension-changes of the edges of the grid:
- In each square are precisely 2 dimensions.
- Thus each path of the form \( L(n_i) \) has to be embedded into a sub-cube.

Lemma:

\[ TR(n_1, n_2, \ldots, n_t) \text{ may be embedded into } HQ(d) \text{ with load 1 and dilation 1, iff } d \geq \sum_{i=1}^{t} \lceil \log n_i \rceil \text{ and all } n_i \text{ are even.} \]
Lemma:

\[ G(n_1, n_2, \cdots, n_t) \text{ may be embedded into } HQ(d) \text{ with load 1 and dilation 1, iff } d \geq \sum_{i=1}^{t} \lceil \log n_i \rceil. \]

Proof:

- Check the dimension-changes of the edges of the grid:
- In each square are precisely 2 dimensions.
- Thus each path of the form \( L(n_i) \) has to be embedded into a sub-cube.

Lemma:

\[ TR(n_1, n_2, \cdots, n_t) \text{ may be embedded into } HQ(d) \text{ with load 1 and dilation 1, iff } d \geq \sum_{i=1}^{t} \lceil \log n_i \rceil \text{ and all } n_i \text{ are even.} \]
Lemma:

\[ G(n_1, n_2, \cdots, n_t) \] may be embedded into \( HQ(d) \) with load 1 and dilation 1, iff \( d \geq \sum_{i=1}^{t} \lceil \log n_i \rceil \).

Proof:

- Check the dimension-changes of the edges of the grid:
- In each square are precisely 2 dimensions.
- Thus each path of the form \( L(n_i) \) has to be embedded into a sub-cube.

Lemma:

\[ TR(n_1, n_2, \cdots, n_t) \] may be embedded into \( HQ(d) \) with load 1 and dilation 1, iff \( d \geq \sum_{i=1}^{t} \lceil \log n_i \rceil \) and all \( n_i \) are even.
Torus and Hypercube

Lemma:

\( G(n_1, n_2, \ldots, n_t) \) may be embedded into \( HQ(d) \) with load 1 and dilation 1, iff \( d \geq \sum_{i=1}^{t} \lceil \log n_i \rceil \).

Proof:

- Check the dimension-changes of the edges of the grid:
- In each square are precisely 2 dimensions.
- Thus each path of the form \( L(n_i) \) has to be embedded into a sub-cube.

Lemma:

\( TR(n_1, n_2, \ldots, n_t) \) may be embedded into \( HQ(d) \) with load 1 and dilation 1, iff \( d \geq \sum_{i=1}^{t} \lceil \log n_i \rceil \) and all \( n_i \) are even.
Theorem:

A binary tree may be embedded with dilation 3 and expansion 8 into the Hypercube.

Theorem:

A binary tree may be embedded with dilation 7 and expansion 1 into the Hypercube.
Theorem:
A binary tree may be embedded with dilation 3 and expansion 8 into the Hypercube.

Theorem:
A binary tree may be embedded with dilation 7 and expansion 1 into the Hypercube.
Theorem:
A binary tree may be embedded with dilation 3 and expansion 8 into the Hypercube.

Theorem:
A binary tree may be embedded with dilation 7 and expansion 1 into the Hypercube.
Caterpillars

Definition:
A binary tree is called caterpillar, iff all nodes with degree 3 are on a simple path. The hair-length denotes the distance of the nodes to the path.

Definition:
A graph $G$ is called balanced, iff there exists a 2-colouring of $G$, which has as many red nodes as black nodes.
Caterpillars

Definition:

A binary tree is called caterpillar, iff all nodes with degree 3 are on a simple path. The hair-length denotes the distance of the nodes to the path.

Definition:

A graph $G$ is called balanced, iff there exists a 2-colouring of $G$, which has as many red nodes as black nodes.
Theorem:

Balanced caterpillars with hair-length 1 are sub-graphs of the hypercube.

Idea of proof: Cut the caterpillar in two balanced pieces.

Theorem:

Caterpillars with $4 \cdot n$ nodes may be embedded with congestion 1 and load 1 into $G(2, 2, n)$.

Proof: Embedd step by step 4 nodes of the caterpillar into the grid.
Caterpillars

**Theorem:**
Balanced caterpillars with hair-length 1 are sub-graphs of the hypercube.

Idea of proof: Cut the caterpillar in two balanced pieces.

**Theorem:**
Caterpillars with $4 \cdot n$ nodes may be embedded with congestion 1 and load 1 into $G(2, 2, n)$.

Proof: Embedd step by step 4 nodes of the caterpillar into the grid.
Caterpillars

**Theorem:**
Balanced caterpillars with hair-length 1 are sub-graphs of the hypercube.

Idea of proof: Cut the caterpillar in two balanced pieces.

**Theorem:**
Caterpillars with $4 \cdot n$ nodes may be embedded with congestion 1 and load 1 into $G(2, 2, n)$.

Proof: Embedd step by step 4 nodes of the caterpillar into the grid.
Definition:

Given: $G, H$ graphs and $d, c, l \in \mathbb{N}$. Questions: Could $G$ be embedded into $H$ with dilation $d$, load $l$ and congestion $c$. 
Embedding-Problem

Theorem:

The embedding-problem is NP-complete into the following cases:

- $G$ is a cycle, $d = c = l = 1$ and $H$ has the same number of nodes as $G$.
- $G, H$ arbitrary, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G, H$ arbitrary, $c$ a constant, $l = 1$, $d$ arbitrary.
- $G, H$ arbitrary, $d, c, l$ constants.
- $G$ a balanciert tree, $H$ a hyper-cube, $d = l = 1$.
- $G$ arbitrary, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G$ a tree, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G$ a caterpillar, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
Embedding-Problem

Theorem:

The embedding-problem is NP-complete into the following cases:

- $G$ is a cycle, $d = c = l = 1$ and $H$ has the same number of nodes as $G$.
- $G$, $H$ arbitrary, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G$, $H$ arbitrary, $c$ a constant, $l = 1$, $d$ arbitrary.
- $G$, $H$ arbitrary, $d$, $c$, $l$ constants.
- $G$ a balanced tree, $H$ a hyper-cube, $d = l = 1$.
- $G$ arbitrary, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G$ a tree, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G$ a caterpillar, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
The embedding-problem is NP-complete into the following cases:

- $G$ is a cycle, $d = c = l = 1$ and $H$ has the same number of nodes as $G$.
- $G, H$ arbitrary, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G, H$ arbitrary, $c$ a constant, $l = 1$, $d$ arbitrary.
- $G, H$ arbitrary, $d, c, l$ constants.
- $G$ a balanced tree, $H$ a hyper-cube, $d = l = 1$.
- $G$ arbitrary, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G$ a tree, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G$ a caterpillar, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
Embedding-Problem

Theorem:

The embedding-problem is NP-complete into the following cases:

- $G$ is a cycle, $d = c = l = 1$ and $H$ has the same number of nodes as $G$.
- $G, H$ arbitrary, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G, H$ arbitrary, $c$ a constant, $l = 1$, $d$ arbitrary.
- $G, H$ arbitrary, $d, c, l$ constants.
- $G$ a balanced tree, $H$ a hyper-cube, $d = l = 1$.
- $G$ arbitrary, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G$ a tree, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G$ a caterpillar, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
Embedding-Problem

Theorem:

The embedding-problem is NP-complete into the following cases:

- $G$ is a cycle, $d = c = l = 1$ and $H$ has the same number of nodes as $G$.
- $G, H$ arbitrary, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G, H$ arbitrary, $c$ a constant, $l = 1$, $d$ arbitrary.
- $G, H$ arbitrary, $d, c, l$ constants.
- $G$ a balanced tree, $H$ a hypercube, $d = l = 1$.
- $G$ arbitrary, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G$ a tree, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G$ a caterpillar, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
Embedding-Problem

Theorem:

The embedding-problem is NP-complete into the following cases:

- $G$ is a cycle, $d = c = l = 1$ and $H$ has the same number of nodes as $G$.
- $G, H$ arbitrary, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G, H$ arbitrary, $c$ a constant, $l = 1$, $d$ arbitrary.
- $G, H$ arbitrary, $d, c, l$ constants.
- $G$ a balanced tree, $H$ a hyper-cube, $d = l = 1$.
- $G$ arbitrary, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G$ a tree, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G$ a caterpillar, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
Embedding-Problem

Theorem:

The embedding-problem is NP-complete into the following cases:

- $G$ is a cycle, $d = c = l = 1$ and $H$ has the same number of nodes as $G$.
- $G, H$ arbitrary, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G, H$ arbitrary, $c$ a constant, $l = 1$, $d$ arbitrary.
- $G, H$ arbitrary, $d, c, l$ constants.
- $G$ a balanced tree, $H$ a hyper-cube, $d = l = 1$.
- $G$ arbitrary, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G$ a tree, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G$ a caterpillar, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
Embedding-Problem

Theorem:

The embedding-problem is NP-complete into the following cases:

- $G$ is a cycle, $d = c = l = 1$ and $H$ has the same number of nodes as $G$.
- $G, H$ arbitrary, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G, H$ arbitrary, $c$ a constant, $l = 1$, $d$ arbitrary.
- $G, H$ arbitrary, $d, c, l$ constants.
- $G$ a balanced tree, $H$ a hyper-cube, $d = l = 1$.
- $G$ arbitrary, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G$ a tree, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G$ a caterpillar, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
Theorem:

The embedding-problem is NP-complete into the following cases:

- $G$ is a cycle, $d = c = l = 1$ and $H$ has the same number of nodes as $G$.
- $G, H$ arbitrary, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G, H$ arbitrary, $c$ a constant, $l = 1$, $d$ arbitrary.
- $G, H$ arbitrary, $d, c, l$ constants.
- $G$ a balanciert tree, $H$ a hyper-cube, $d = l = 1$.
- $G$ arbitrary, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G$ a tree, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
- $G$ a caterpillar, $H$ a path, $d$ a constant, $l = 1$, $c$ arbitrary.
Optical Fibers
- Optical Sender
- Optical Receiver
- Optical Amplifiers
- Wavelengths: 1450–1650 nm (Nanometer)
- C-Band: 1530–1565 nm (currently used)
- L-Band: 1565–1625 nm (used soon)
- Width of a channel: about 10 GHz.
- Distance between channels: about 100 GHz.
- About 80 channels in the C-Band.
- With a channel-distance of 25 GHz about 200 channels in the C-Band
- Critical Angle: $\sin^{-1} \frac{\mu_2}{\mu_1}$
- Technic known as “wavelength division multiplexing” (WDM)
- Nodes of an optical network: Transmitters and Routers.
- Optical paths (“lightpath”) via routers.
The Technic

- **Optical Fibers**
- **Optical Sender**
- Optical Receiver
- Optical Amplifiers
- Wavelengths: 1450–1650 nm (Nanometer)
- C-Band: 1530–1565 nm (currently used)
- L-Band: 1565–1625 nm (used soon)
- Width of a channel: about 10 GHz.
- Distance between channels: about 100 GHz.
- About 80 channels in the C-Band.
- With a channel-distance of 25 GHz about 200 channels in the C-Band
- Critical Angle: $\sin^{-1} \frac{\mu_2}{\mu_1}$
- Technic known as “wavelength division multiplexing” (WDM)
- Nodes of an optical network: Transmitters and Routers.
- Optical paths (“lightpath”) via routers.
The Technic

- Optical Fibers
- Optical Sender
- **Optical Receiver**
  - Optical Amplifiers
  - Wavelengths: 1450–1650 nm (Nanometer)
  - C-Band: 1530–1565 nm (currently used)
  - L-Band: 1565–1625 nm (used soon)
  - Width of a channel: about 10 GHz.
  - Distance between channels: about 100 GHz.
  - About 80 channels in the C-Band.
  - With a channel-distance of 25 GHz about 200 channels in the C-Band
  - Critical Angle: $\sin^{-1}\frac{\mu_2}{\mu_1}$
  - Technic known as “wavelength division multiplexing” (WDM)
  - Nodes of an optical network: Transmitters and Routers.
  - Optical paths (“lightpath”) via routers.
The Technic

- Optical Fibers
- Optical Sender
- Optical Receiver
- **Optical Amplifiers**
  - Wavelengths: 1450–1650 nm (Nanometer)
  - C-Band: 1530–1565 nm (currently used)
  - L-Band: 1565–1625 nm (used soon)
  - Width of a channel: about 10 GHz.
  - Distance between channels: about 100 GHz.
  - About 80 channels in the C-Band.
  - With a channel-distance of 25 GHz about 200 channels in the C-Band
  - Critical Angle: $\sin^{-1}\frac{\mu_2}{\mu_1}$
  - Technic known as “wavelength division multiplexing” (WDM)
  - Nodes of an optical network: Transmitters and Routers.
  - Optical paths ("lightpath") via routers.
The Technic

- Optical Fibers
- Optical Sender
- Optical Receiver
- Optical Amplifiers

- **Wavelengths: 1450–1650 nm (Nanometer)**
  - C-Band: 1530–1565 nm (currently used)
  - L-Band: 1565–1625 nm (used soon)
- Width of a channel: about 10 GHz.
- Distance between channels: about 100 GHz.
- About 80 channels in the C-Band.
- With a channel-distance of 25 GHz about 200 channels in the C-Band
- Critical Angle: $\sin^{-1}\frac{\mu_2}{\mu_1}$
- Technic known as “wavelength division multiplexing” (WDM)
- Nodes of an optical network: Transmitters and Routers.
- Optical paths ("lightpath") via routers.
The Technic

- Optical Fibers
- Optical Sender
- Optical Receiver
- Optical Amplifiers
- Wavelengths: 1450–1650 nm (Nanometer)
  - C-Band: 1530–1565 nm (currently used)
  - L-Band: 1565–1625 nm (used soon)
- Width of a channel: about 10 GHz.
- Distance between channels: about 100 GHz.
- About 80 channels in the C-Band.
- With a channel-distance of 25 GHz about 200 channels in the C-Band
- Critical Angle: $\sin^{-1}\mu_2/\mu_1$
- Technic known as “wavelength division multiplexing” (WDM)
- Nodes of an optical network: Transmitters and Routers.
- Optical paths (“lightpath”) via routers.
- Optical Fibers
- Optical Sender
- Optical Receiver
- Optical Amplifiers
- Wavelengths: 1450–1650 nm (Nanometer)
- C-Band: 1530–1565 nm (currently used)
- L-Band: 1565–1625 nm (used soon)
- Width of a channel: about 10 GHz.
- Distance between channels: about 100 GHz.
- About 80 channels in the C-Band.
- With a channel-distance of 25 GHz about 200 channels in the C-Band
- Critical Angle: $\sin^{-1} \frac{\mu_2}{\mu_1}$
- Technic known as “wavelength division multiplexing” (WDM)
- Nodes of an optical network: Transmitters and Routers.
- Optical paths (“lightpath”) via routers.
The Technic

- Optical Fibers
- Optical Sender
- Optical Receiver
- Optical Amplifiers
- Wavelengths: 1450–1650 nm (Nanometer)
- C-Band: 1530–1565 nm (currently used)
- L-Band: 1565–1625 nm (used soon)
- **Width of a channel:** about 10 GHz.
- Distance between channels: about 100 GHz.
- About 80 channels in the C-Band.
- With a channel-distance of 25 GHz about 200 channels in the C-Band
- Critical Angle: $\sin^{-1} \frac{\mu_2}{\mu_1}$
- Technic known as “wavelength division multiplexing” (WDM)
- Nodes of an optical network: Transmitters and Routers.
- Optical paths (“lightpath”) via routers.
The Technic

- Optical Fibers
- Optical Sender
- Optical Receiver
- Optical Amplifiers
- Wavelengths: 1450–1650 nm (Nanometer)
- C-Band: 1530–1565 nm (currently used)
- L-Band: 1565–1625 nm (used soon)
- Width of a channel: about 10 GHz.
- Distance between channels: about 100 GHz.
- About 80 channels in the C-Band.
- With a channel-distance of 25 GHz about 200 channels in the C-Band
- Critical Angle: $sin^{-1}\frac{\mu_2}{\mu_1}$
- Technic known as “wavelength division multiplexing” (WDM)
- Nodes of an optical network: Transmitters and Routers.
- Optical paths ("lightpath") via routers.
The Technic

- Optical Fibers
- Optical Sender
- Optical Receiver
- Optical Amplifiers
- Wavelengths: 1450–1650 nm (Nanometer)
- C-Band: 1530–1565 nm (currently used)
- L-Band: 1565–1625 nm (used soon)
- Width of a channel: about 10 GHz.
- Distance between channels: about 100 GHz.
- About 80 channels in the C-Band.
- With a channel-distance of 25 GHz about 200 channels in the C-Band
- Critical Angle: $\sin^{-1}\frac{\mu_2}{\mu_1}$
- Technic known as “wavelength division multiplexing” (WDM)
- Nodes of an optical network: Transmitters and Routers.
- Optical paths (“lightpath”) via routers.
The Technic

- Optical Fibers
- Optical Sender
- Optical Receiver
- Optical Amplifiers
- Wavelengths: 1450–1650 nm (Nanometer)
- C-Band: 1530–1565 nm (currently used)
- L-Band: 1565–1625 nm (used soon)
- Width of a channel: about 10 GHz.
- Distance between channels: about 100 GHz.
- About 80 channels in the C-Band.
- **With a channel-distance of 25 GHz about 200 channels in the C-Band**
- Critical Angle: \( \sin^{-1} \frac{\mu_2}{\mu_1} \)
- Technic known as “wavelength division multiplexing” (WDM)
- Nodes of an optical network: Transmitters and Routers.
- Optical paths (“lightpath”) via routers.
The Technic

- Optical Fibers
- Optical Sender
- Optical Receiver
- Optical Amplifiers
- Wavelengths: 1450–1650 nm (Nanometer)
  - C-Band: 1530–1565 nm (currently used)
  - L-Band: 1565–1625 nm (used soon)
- Width of a channel: about 10 GHz.
- Distance between channels: about 100 GHz.
- About 80 channels in the C-Band.
- With a channel-distance of 25 GHz about 200 channels in the C-Band
- Critical Angle: $\sin^{-1} \frac{\mu_2}{\mu_1}$
- Technic known as “wavelength division multiplexing” (WDM)
- Nodes of an optical network: Transmitters and Routers.
- Optical paths ("lightpath") via routers.
The Technic

- Optical Fibers
- Optical Sender
- Optical Receiver
- Optical Amplifiers
- Wavelengths: 1450–1650 nm (Nanometer)
- C-Band: 1530–1565 nm (currently used)
- L-Band: 1565–1625 nm (used soon)
- Width of a channel: about 10 GHz.
- Distance between channels: about 100 GHz.
- About 80 channels in the C-Band.
- With a channel-distance of 25 GHz about 200 channels in the C-Band
- Critical Angle: $\sin^{-1} \frac{\mu_2}{\mu_1}$
- Technic known as “wavelength division multiplexing” (WDM)
- Nodes of an optical network: Transmitters and Routers.
- Optical paths (“lightpath”) via routers.
Optical Fibers
Optical Sender
Optical Receiver
Optical Amplifiers

Wavelengths: 1450–1650 nm (Nanometer)
C-Band: 1530–1565 nm (currently used)
L-Band: 1565–1625 nm (used soon)

Width of a channel: about 10 GHz.
Distance between channels: about 100 GHz.

About 80 channels in the C-Band.
With a channel-distance of 25 GHz about 200 channels in the C-Band

Critical Angle: \( \sin^{-1} \frac{\mu_2}{\mu_1} \)

Technic known as “wavelength division multiplexing” (WDM)

Nodes of an optical network: Transmitters and Routers.

Optical paths ("lightpath") via routers.
The Technic

- Optical Fibers
- Optical Sender
- Optical Receiver
- Optical Amplifiers
- Wavelengths: 1450–1650 nm (Nanometer)
- C-Band: 1530–1565 nm (currently used)
- L-Band: 1565–1625 nm (used soon)
- Width of a channel: about 10 GHz.
- Distance between channels: about 100 GHz.
- About 80 channels in the C-Band.
- With a channel-distance of 25 GHz about 200 channels in the C-Band
- Critical Angle: $\sin^{-1}\frac{\mu_2}{\mu_1}$
- Technic known as “wavelength division multiplexing” (WDM)
- Nodes of an optical network: Transmitters and Routers.
- Optical paths (“lightpath”) via routers.
Advantages and Disadvantages

- High transfer-rate:
  - Currently: 107 Gigabit per second.
  - Theoretical $50 \cdot 10^{12}$ bits per second.
- Low signal-loss: 0.2 db/km.
- Signal is not changed a lot (less jitter).
- Not so many optical Amplifiers are used.
- Less energy, space and less cost for the material.
- More channels per fiber.
- Less disturbance by other signals.
- Fast signal distribution.
- Low cost.

- Optical Devices are expensive (or not developed so far)
- Detour via electronic devices.
Advantages and Disadvantages

- **High transfer-rate:**
  - Currently: 107 Gigabit per second.
  - Theoretical $50 \cdot 10^{12}$ bits per second.

- **Low signal-loss:** 0.2 db/km.
- Signal is not changed a lot (less jitter).
- Not so many optical Amplifiers are used.
- Less energy, space and less cost for the material.
- More channels per fiber.
- Less disturbance by other signals.
- Fast signal distribution.
- Low cost.

- Optical Devices are expensive (or not developed so far)
- Detour via electronic devices.
Advantages and Disadvantages

- **High transfer-rate:**
  - Currently: 107 Gigabit per second.
  - Theoretical $50 \cdot 10^{12}$ bits per second.
- Low signal-loss: 0.2 db/km.
- Signal is not changed a lot (less jitter).
- Not so many optical Amplifiers are used.
- Less energy, space and less cost for the material.
- More channels per fiber.
- Less disturbance by other signals.
- Fast signal distribution.
- Low cost.

- Optical Devices are expensive (or not developed so far)
- Detour via electronic devices.
Advantages and Disadvantages

- **High transfer-rate:**
  - Currently: 107 Gigabit per second.
  - Theoretical $50 \cdot 10^{12}$ bits per second.

- Low signal-loss: 0.2 db/km.

- Signal is not changed a lot (less jitter).

- Not so many optical Amplifiers are used.

- Less energy, space and less cost for the material.

- More channels per fiber.

- Less disturbance by other signals.

- Fast signal distribution.

- Low cost.

- Optical Devices are expensive (or not developed so far)

- Detour via electronic devices.
Advantages and Disadvantages

- **High transfer-rate:**
  - Currently: 107 Gigabit per second.
  - Theoretical $50 \cdot 10^{12}$ bits per second.

- Low signal-loss: 0.2 db/km.
- Signal is not changed a lot (less jitter).
- Not so many optical Amplifiers are used.
- Less energy, space and less cost for the material.
- More channels per fiber.
- Less disturbance by other signals.
- Fast signal distribution.
- Low cost.

- Optical Devices are expensive (or not developed so far)
- Detour via electronic devices.
Advantages and Disadvantages

- **High transfer-rate:**
  - Currently: 107 Gigabit per second.
  - Theoretical $50 \cdot 10^{12}$ bits per second.

- **Low signal-loss:** 0.2 db/km.
- Signal is not changed a lot (less jitter).
- Not so many optical Amplifiers are used.
- Less energy, space and less cost for the material.
- More channels per fiber.
- Less disturbance by other signals.
- Fast signal distribution.
- Low cost.

- Optical Devices are expensive (or not developed so far)
- Detour via electronic devices.
Advantages and Disadvantages

- High transfer-rate:
  - Currently: 107 Gigabit per second.
  - Theoretical $50 \cdot 10^{12}$ bits per second.

- Low signal-loss: 0.2 db/km.

- Signal is not changed a lot (less jitter).

- Not so many optical Amplifiers are used.

- Less energy, space and less cost for the material.

- More channels per fiber.

- Less disturbance by other signals.

- Fast signal distribution.

- Low cost.

- Optical Devices are expensive (or not developed so far)

- Detour via electronic devices.
Advantages and Disadvantages

- **High transfer-rate:**
  - Currently: 107 Gigabit per second.
  - Theoretical $50 \cdot 10^{12}$ bits per second.

- **Low signal-loss:** 0.2 db/km.

- **Signal is not changed a lot (less jitter).**

- **Not so many optical Amplifiers are used.**
  - Less energy, space and less cost for the material.
  - More channels per fiber.
  - Less disturbance by other signals.
  - Fast signal distribution.
  - Low cost.

- **Optical Devices are expensive (or not developed so far)**

- **Detour via electronic devices.**
Advantages and Disadvantages

- **High transfer-rate:**
  - Currently: 107 Gigabit per second.
  - Theoretical $50 \cdot 10^{12}$ bits per second.

- **Low signal-loss:** 0.2 db/km.
- **Signal is not changed a lot (less jitter).**
- **Not so many optical Amplifiers are used.**
- **Less energy, space and less cost for the material.**
  - More channels per fiber.
  - Less disturbance by other signals.
  - Fast signal distribution.
  - Low cost.

- **Optical Devices are expensive (or not developed so far)**
- **Detour via electronic devices.**
Advantages and Disadvantages

- **High transfer-rate:**
  - Currently: 107 Gigabit per second.
  - Theoretical $50 \cdot 10^{12}$ bits per second.
- **Low signal-loss:** 0.2 db/km.
- Signal is not changed a lot (less jitter).
- Not so many optical Amplifiers are used.
- Less energy, space and less cost for the material.
- **More channels per fiber.**
  - Less disturbance by other signals.
  - Fast signal distribution.
  - Low cost.
- Optical Devices are expensive (or not developed so far)
- Detour via electronic devices.
Advantages and Disadvantages

- High transfer-rate:
  - Currently: 107 Gigabit per second.
  - Theoretical $50 \cdot 10^{12}$ bits per second.

- Low signal-loss: 0.2 db/km.
- Signal is not changed a lot (less jitter).
- Not so many optical Amplifiers are used.
- Less energy, space and less cost for the material.
- More channels per fiber.
- Less disturbance by other signals.
- Fast signal distribution.
- Low cost.

- Optical Devices are expensive (or not developed so far)
- Detour via electronic devices.
Advantages and Disadvantages

- **High transfer-rate:**
  - Currently: 107 Gigabit per second.
  - Theoretical $50 \cdot 10^{12}$ bits per second.

- **Low signal-loss:** 0.2 db/km.

- **Signal is not changed a lot** (less jitter).

- **Not so many optical Amplifiers are used.**

- **Less energy, space and less cost for the material.**

- **More channels per fiber.**

- **Less disturbance by other signals.**

- **Fast signal distribution.**

- **Low cost.**

- **Optical Devices are expensive** (or not developed so far)

- **Detour via electronic devices.**
Advantages and Disadvantages

- High transfer-rate:
  - Currently: 107 Gigabit per second.
  - Theoretical $50 \cdot 10^{12}$ bits per second.

- Low signal-loss: 0.2 db/km.
- Signal is not changed a lot (less jitter).
- Not so many optical Amplifiers are used.
- Less energy, space and less cost for the material.
- More channels per fiber.
- Less disturbance by other signals.
- Fast signal distribution.
- Low cost.

- Optical Devices are expensive (or not developed so far)
- Detour via electronic devices.
Advantages and Disadvantages

- High transfer-rate:
  - Currently: 107 Gigabit per second.
  - Theoretical $50 \cdot 10^{12}$ bits per second.

- Low signal-loss: 0.2 db/km.

- Signal is not changed a lot (less jitter).

- Not so many optical Amplifiers are used.

- Less energy, space and less cost for the material.

- More channels per fiber.

- Less disturbance by other signals.

- Fast signal distribution.

- Low cost.

- Optical Devices are expensive (or not developed so far)

- Detour via electronic devices.
Advantages and Disadvantages

- High transfer-rate:
  - Currently: 107 Gigabit per second.
  - Theoretical $50 \cdot 10^{12}$ bits per second.

- Low signal-loss: 0.2 db/km.
- Signal is not changed a lot (less jitter).
- Not so many optical Amplifiers are used.
- Less energy, space and less cost for the material.
- More channels per fiber.
- Less disturbance by other signals.
- Fast signal distribution.
- Low cost.

- Optical Devices are expensive (or not developed so far)
- Detour via electronic devices.
Advantages and Disadvantages

- **High transfer-rate:**
  - Currently: 107 Gigabit per second.
  - Theoretical $50 \cdot 10^{12}$ bits per second.

- **Low signal-loss:** 0.2 db/km.

- **Signal is not changed a lot (less jitter).**

- **Not so many optical Amplifiers are used.**

- **Less energy, space and less cost for the material.**

- **More channels per fiber.**

- **Less disturbance by other signals.**

- **Fast signal distribution.**

- **Low cost.**

- **Optical Devices are expensive (or not developed so far)**

- **Detour via electronic devices.**
Advantages and Disadvantages

- High transfer-rate:
  - Currently: 107 Gigabit per second.
  - Theoretical $50 \cdot 10^{12}$ bits per second.

- Low signal-loss: 0.2 db/km.

- Signal is not changed a lot (less jitter).

- Not so many optical Amplifiers are used.

- Less energy, space and less cost for the material.

- More channels per fiber.

- Less disturbance by other signals.

- Fast signal distribution.

- Low cost.

- Optical Devices are expensive (or not developed so far)

- Detour via electronic devices.
Types of WDM and Problems

- **Types of WDM**
  - Wavelength-routed Networks: the receiver determines the wavelength statically.
  - Broadcasting Networks: Send with wavelength $\lambda$ to all. Only the receivers use $\lambda$ as input wavelength.
  - Static and dynamic optical paths.
  - Single-HOP ("all-optical Network") and Multi-HOP.

- **Important Problems on WDM**
  - Building the optical paths.
  - Building a logical connection-structure.
  - Determine communication by for this logical structure.
  - Handle errors.
Types of WDM and Problems

- Types of WDM
  - Wavelength-routed Networks: the receiver determines the wavelength statically.
  - Broadcasting Networks: Send with wavelength $\lambda$ to all. Only the receivers use $\lambda$ as input wavelength.
  - Static and dynamic optical paths.
  - Single-HOP ("all-optical Network") and Multi-HOP.

- Important Problems on WDM
  - Building the optical paths.
  - Building a logical connection-structure.
  - Determine communication by for this logical structure.
  - Handle errors.
Types of WDM and Problems

- **Types of WDM**
  - Wavelength-routed Networks: the receiver determines the wavelength statically.
  - Broadcasting Networks: Send with wavelength $\lambda$ to all. Only the receivers use $\lambda$ as input wavelength.
  - Static and dynamic optical paths.
  - Single-HOP ("all-optical Network") and Multi-HOP.

- **Important Problems on WDM**
  - Building the optical paths.
  - Building a logical connection-structure.
  - Determine communication by for this logical structure.
  - Handle errors.
Types of WDM and Problems

- Types of WDM
  - Wavelength-routed Networks: the receiver determines the wavelength statically.
  - Broadcasting Networks: Send with wavelength $\lambda$ to all. Only the receivers use $\lambda$ as input wavelength
  - Static and dynamic optical paths.
  - Single-HOP ("all-optical Network") and Multi-HOP.

- Important Problems on WDM
  - Building the optical paths.
  - Building a logical connection-structure.
  - Determine communication by for this logical structure.
  - Handle errors.
Types of WDM and Problems

- **Types of WDM**
  - Wavelength-routed Networks: the receiver determines the wavelength staticly.
  - Broadcasting Networks: Send with wavelength $\lambda$ to all. Only the receivers use $\lambda$ as input wavelength
  - Static and dynamic optical paths.
  - Single-HOP ("all-optical Network") and Multi-HOP.

- **Important Problems on WDM**
  - Building the optical paths.
  - Building a logical connection-structure.
  - Determine communication by for this logical structure.
  - Handle errors.
Types of WDM and Problems

- **Types of WDM**
  - Wavelength-routed Networks: the receiver determines the wavelength statically.
  - Broadcasting Networks: Send with wavelength $\lambda$ to all. Only the receivers use $\lambda$ as input wavelength.
  - Static and dynamic optical paths.
  - Single-HOP ("all-optical Network") and Multi-HOP.

- **Important Problems on WDM**
  - Building the optical paths.
  - Building a logical connection-structure.
  - Determine communication by for this logical structure.
  - Handle errors.
Types of WDM and Problems

- **Types of WDM**
  - Wavelength-routed Networks: the receiver determines the wavelength statically.
  - Broadcasting Networks: Send with wavelength $\lambda$ to all. Only the receivers use $\lambda$ as input wavelength.
  - Static and dynamic optical paths.
  - Single-HOP (“all-optical Network”) and Multi-HOP.

- **Important Problems on WDM**
  - Building the optical paths.
  - Building a logical connection-structure.
  - Determine communication by for this logical structure.
  - Handle errors.
Types of WDM and Problems

- **Types of WDM**
  - Wavelength-routed Networks: the receiver determines the wavelength staticly.
  - Broadcasting Networks: Send with wavelength $\lambda$ to all. Only the receivers use $\lambda$ as input wavelength.
  - Static and dynamic optical paths.
  - Single-HOP (“all-optical Network”) and Multi-HOP.

- **Important Problems on WDM**
  - Building the optical paths.
  - Building a logical connection-structure.
  - Determine communication by for this logical structure.
  - Handle errors.
Types of WDM and Problems

- **Types of WDM**
  - Wavelength-routed Networks: the receiver determines the wavelength statically.
  - Broadcasting Networks: Send with wavelength $\lambda$ to all. Only the receivers use $\lambda$ as input wavelength.
  - Static and dynamic optical paths.
  - Single-HOP (“all-optical Network”) and Multi-HOP.

- **Important Problems on WDM**
  - Building the optical paths.
  - Building a logical connection-structure.
  - **Determine communication by for this logical structure.**
  - Handle errors.
Types of WDM and Problems

- **Types of WDM**
  - Wavelength-routed Networks: the receiver determines the wavelength statically.
  - Broadcasting Networks: Send with wavelength $\lambda$ to all. Only the receivers use $\lambda$ as input wavelength.
  - Static and dynamic optical paths.
  - Single-HOP (“all-optical Network”) and Multi-HOP.

- **Important Problems on WDM**
  - Building the optical paths.
  - Building a logical connection-structure.
  - Determine communication by for this logical structure.
  - Handle errors.
Optical Coupler

- Optical coupler has value $\alpha$.
- If input $I_i$ receives a signal of strength $P_i$, then outputs $O_0 \alpha \cdot P_0$ and $O_1 (1 - \alpha) \cdot P_1$.
- Exists independent of the wavelength and dependent of the wavelength.

Two possible configurations:
- crossing and
- not crossing.
Optical Coupler

- Optical coupler has value $\alpha$.
- If input $I_i$ receives a signal of strength $P_i$,
  then outputs $O_0 \alpha \cdot P_0$ and
  $O_1 (1 - \alpha) \cdot P_1$.
- Exists independent of the wavelength and
  dependent of the wavelength.
- Two possible configurations:
  - crossing and
  - not crossing.
Optical coupler has value $\alpha$.

If input $I_i$ receives a signal of strength $P_i$,

then outputs $O_0 \alpha \cdot P_0$ and

$O_1 (1 - \alpha) \cdot P_1$.

 Exists independent of the wavelength and
dependent of the wavelength.

Two possible configurations:

- crossing and
- not crossing.
Optical coupler has value $\alpha$.

- If input $I_i$ receives a signal of strength $P_i$,
- then outputs $O_0 \alpha \cdot P_0$ and
- $O_1 (1 - \alpha) \cdot P_1$.

Exists independent of the wavelength and dependent of the wavelength.

Two possible configurations:
- crossing and
- not crossing.
Optical Coupler

- Optical coupler has value $\alpha$.
- If input $I_i$ receives a signal of strength $P_i$,
  - then outputs $O_0 \alpha \cdot P_0$ and
  - $O_1 (1 - \alpha) \cdot P_1$.
- Exists independent of the wavelength and dependent of the wavelength.

- Two possible configurations:
  - crossing and
  - not crossing.
Optical coupler has value $\alpha$.

If input $I_i$ receives a signal of strength $P_i$,

then outputs $O_0 \alpha \cdot P_0$ and

$O_1 (1 - \alpha) \cdot P_1$.

Exists independent of the wavelength and

dependent of the wavelength.

Two possible configurations:

crossing and

not crossing.
Optical Coupler

- Optical coupler has value $\alpha$.
- If input $I_i$ receives a signal of strength $P_i$, then outputs $O_0 \alpha \cdot P_0$ and $O_1 (1 - \alpha) \cdot P_1$.
- Exists independent of the wavelength and dependent of the wavelength.

Two possible configurations:
- crossing and
- not crossing.
Optical Coupler

- Optical coupler has value $\alpha$.
- If input $I_i$ receives a signal of strength $P_i$,
  then outputs $O_0 \alpha \cdot P_0$ and
  $O_1 (1 - \alpha) \cdot P_1$.
- Exists independent of the wavelength and
  dependent of the wavelength.

Two possible configurations:
- crossing and
- not crossing.
Optical Coupler

- Optical coupler has value $\alpha$.
- If input $I_i$ receives a signal of strength $P_i$,
- then outputs $O_0 \alpha \cdot P_0$ and $O_1 (1 - \alpha) \cdot P_1$.
- Exists independent of the wavelength and dependent of the wavelength.

Two possible configurations:
- crossing and
- not crossing.
Optical coupler has value \( \alpha \).
If input \( I_i \) receives a signal of strength \( P_i \),
then outputs \( O_0 \alpha \cdot P_0 \) and
\( O_1 (1 - \alpha) \cdot P_1 \).
Exists independent of the wavelength and
dependent of the wavelength.

- Two possible configurations:
  - crossing and
  - not crossing.
Optical Coupler

- Optical coupler has value $\alpha$.
- If input $I_i$ receives a signal of strength $P_i$,
- then outputs $O_0 \alpha \cdot P_0$ and
- $O_1 (1 - \alpha) \cdot P_1$.
- Exists independent of the wavelength and
- dependent of the wavelength.

- Two possible configurations:
  - crossing and
  - not crossing.
Optical Coupler

- Optical coupler has value $\alpha$.
- If input $I_i$ receives a signal of strength $P_i$, then outputs $O_0 \cdot \alpha \cdot P_0$ and $O_1 \cdot (1 - \alpha) \cdot P_1$.
- Exists independent of the wavelength and dependent of the wavelength.

Two possible configurations:
- crossing and
- not crossing.
Optical Coupler

- Optical coupler has value $\alpha$.
- If input $I_i$ receives a signal of strength $P_i$,
  then outputs $O_0 \alpha \cdot P_0$ and
  $O_1 (1 - \alpha) \cdot P_1$.
- Exists independent of the wavelength and
dependent of the wavelength.

- Two possible configurations:
  - crossing and
  - not crossing.
Optical Coupler

- Optical coupler has value $\alpha$.
- If input $I_i$ receives a signal of strength $P_i$, then outputs $O_0 \alpha \cdot P_0$ and $O_1 (1 - \alpha) \cdot P_1$.
- Exists independent of the wavelength and dependent of the wavelength.

- Two possible configurations:
  - crossing and
  - not crossing.
Optical coupler has value $\alpha$.

- If input $I_i$ receives a signal of strength $P_i$,
- then outputs $O_0 \alpha \cdot P_0$ and
- $O_1 (1 - \alpha) \cdot P_1$.

- Exists independent of the wavelength and dependent of the wavelength.

- Two possible configurations:
  - crossing and
  - not crossing.
“Crossbar” and Beneš

Theorem

A crossbar is “wide-sense nonblocking”, i.e. any permutation and any extension to a sub-permutation is possible.
Theorem
A crossbar is “wide-sense nonblocking”, i.e. any permutation and any extension to a sub-permutation is possible.
A crossbar is “wide-sense nonblocking”, i.e. any permutation and any extension to a sub-permutation is possible.
A crossbar is “wide-sense nonblocking”, i.e. any permutation and any extension to a sub-permutation is possible.
"Crossbar" and Beneš

Theorem

A crossbar is "wide-sense nonblocking", i.e. any permutation and any extension to a sub-permutation is possible.
**Theorem**

A crossbar is “wide-sense nonblocking”, i.e. any permutation and any extension to a sub-permutation is possible.
**Theorem**

A crossbar is “wide-sense nonblocking”, i.e. any permutation and any extension to a sub-permutation is possible.

**Theorem**

The Beneš Network is “nonblocking”, i.e. any permutation is possible.
The Beneš Network is nonblocking

- Each path $i$ has to traverse one of the sub-networks.
- Common inputs $2 \cdot i$ and $2 \cdot i - 1$ may not use the same sub-network.
- Common inputs $\pi(2 \cdot i)$ and $\pi(2 \cdot i - 1)$ may not use the same sub-network.
- The resulting conflict graph is bipartite (Sum of two Matchings).

Thus the paths may be placed on the two sub-networks.
- The statement holds by a simple induction.
The Beneš Network is nonblocking

- Each path \( i \) has to traverse one of the sub-networks.
- Common inputs \( 2 \cdot i \) and \( 2 \cdot i - 1 \) may not use the same sub-network.
- Common inputs \( \pi(2 \cdot i) \) and \( \pi(2 \cdot i - 1) \) may not use the same sub-network.
- The resulting conflict graph is bipartite (Sum of two Matchings).

- Thus the paths may be placed on the two sub-networks.
- The statement holds by a simple induction.
The Beneš Network is nonblocking

- Each path $i$ has to traverse one of the sub-networks.
- Common inputs $2 \cdot i$ and $2 \cdot i - 1$ may not use the same sub-network.
- Common inputs $\pi(2 \cdot i)$ and $\pi(2 \cdot i - 1)$ may not use the same sub-network.
- The resulting conflict graph is bipartite (Sum of two Matchings).

- Thus the paths may be placed on the two sub-networks.
- The statement holds by a simple induction.
The Beneš Network is nonblocking

- Each path $i$ has to traverse one of the sub-networks.
- Common inputs $2 \cdot i$ and $2 \cdot i - 1$ may not use the same sub-network.
- Common inputs $\pi(2 \cdot i)$ and $\pi(2 \cdot i - 1)$ may not use the same sub-network.
- The resulting conflict graph is bipartite (Sum of two Matchings).

Thus the pathes may be placed on the two sub-networks.

The statement holds by a simple induction.
The Beneš Network is nonblocking

- Each path $i$ has to traverse one of the sub-networks.
- Common inputs $2 \cdot i$ and $2 \cdot i - 1$ may not use the same sub-network.
- Common inputs $\pi(2 \cdot i)$ and $\pi(2 \cdot i - 1)$ may not use the same sub-network.
- The resulting conflict graph is bipartite (Sum of two Matchings).

Thus the paths may be placed on the two sub-networks.

The statement holds by a simple induction.
The Beneš Network is nonblocking

- Each path $i$ has to traverse one of the sub-networks.
- Common inputs $2 \cdot i$ and $2 \cdot i - 1$ may not use the same sub-network.
- Common inputs $\pi(2 \cdot i)$ and $\pi(2 \cdot i - 1)$ may not use the same sub-network.
- The resulting conflict graph is bipartite (Sum of two Matchings).

Thus the pathes may be placed on the two sub-networks.

The statement holds by a simple induction.
The Beneš Network is nonblocking.

- Each path $i$ has to traverse one of the sub-networks.
- Common inputs $2 \cdot i$ and $2 \cdot i - 1$ may not use the same sub-network.
- Common inputs $\pi(2 \cdot i)$ and $\pi(2 \cdot i - 1)$ may not use the same sub-network.
- The resulting conflict graph is bipartite (Sum of two Matchings).

Thus the pathes may be placed on the two sub-networks.
The statement holds by a simple induction.
The Beneš Network is nonblocking

- Each path $i$ has to traverse one of the sub-networks.
- Common inputs $2 \cdot i$ and $2 \cdot i - 1$ may not use the same sub-network.
- Common inputs $\pi(2 \cdot i)$ and $\pi(2 \cdot i - 1)$ may not use the same sub-network.
- The resulting conflict graph is bipartite (Sum of two Matchings).

- Thus the paths may be placed on the two sub-networks.
- The statement holds by a simple induction.
The Beneš Network is nonblocking.

- Each path $i$ has to traverse one of the sub-networks.
- Common inputs $2 \cdot i$ and $2 \cdot i - 1$ may not use the same sub-network.
- Common inputs $\pi(2 \cdot i)$ and $\pi(2 \cdot i - 1)$ may not use the same sub-network.
- The resulting conflict graph is bipartite (Sum of two Matchings).

- Thus the paths may be placed on the two sub-networks.
- The statement holds by a simple induction.
The Beneš Network is nonblocking

- Each path $i$ has to traverse one of the sub-networks.
- Common inputs $2 \cdot i$ and $2 \cdot i - 1$ may not use the same sub-network.
- Common inputs $\pi(2 \cdot i)$ and $\pi(2 \cdot i - 1)$ may not use the same sub-network.
- The resulting conflict graph is bipartite (Sum of two Matchings).

- Thus the pathes may be placed on the two sub-networks.
- The statement holds by a simple induction.
The Beneš Network is nonblocking

- Each path $i$ has to traverse one of the sub-networks.
- Common inputs $2 \cdot i$ and $2 \cdot i - 1$ may not use the same sub-network.
- Common inputs $\pi(2 \cdot i)$ and $\pi(2 \cdot i - 1)$ may not use the same sub-network.
- The resulting conflict graph is bipartite (Sum of two Matchings).

Thus the pathes may be placed on the two sub-networks.

The statement holds by a simple induction.
The Beneš Network is nonblocking

- Each path $i$ has to traverse one of the sub-networks.
- Common inputs $2 \cdot i$ and $2 \cdot i - 1$ may not use the same sub-network.
- Common inputs $\pi(2 \cdot i)$ and $\pi(2 \cdot i - 1)$ may not use the same sub-network.
- The resulting conflict graph is bipartite (Sum of two Matchings).

Thus the paths may be placed on the two sub-networks.

The statement holds by a simple induction.
The Beneš Network is nonblocking.

- Each path $i$ has to traverse one of the sub-networks.
- Common inputs $2 \cdot i$ and $2 \cdot i - 1$ may not use the same sub-network.
- Common inputs $\pi(2 \cdot i)$ and $\pi(2 \cdot i - 1)$ may not use the same sub-network.
- The resulting conflict graph is bipartite (Sum of two Matchings).

- Thus the paths may be placed on the two sub-networks.
- The statement holds by a simple induction.
The Beneš Network is nonblocking

- Each path $i$ has to traverse one of the sub-networks.
- Common inputs $2 \cdot i$ and $2 \cdot i - 1$ may not use the same sub-network.
- Common inputs $\pi(2 \cdot i)$ and $\pi(2 \cdot i - 1)$ may not use the same sub-network.
- The resulting conflict graph is bipartite (Sum of two Matchings).

- Thus the pathes may be placed on the two sub-networks.
- The statement holds by a simple induction.
The Beneš Network is nonblocking

- Each path $i$ has to traverse one of the sub-networks.
- Common inputs $2 \cdot i$ and $2 \cdot i - 1$ may not use the same sub-network.
- Common inputs $\pi(2 \cdot i)$ and $\pi(2 \cdot i - 1)$ may not use the same sub-network.
- The resulting conflict graph is bipartite (Sum of two Matchings).

Thus the paths may be placed on the two sub-networks.

The statement holds by a simple induction.
The Beneš Network is nonblocking.

- Each path $i$ has to traverse one of the sub-networks.
- Common inputs $2 \cdot i$ and $2 \cdot i - 1$ may not use the same sub-network.
- Common inputs $\pi(2 \cdot i)$ and $\pi(2 \cdot i - 1)$ may not use the same sub-network.
- The resulting conflict graph is bipartite (Sum of two Matchings).

Thus the paths may be placed on the two sub-networks.

The statement holds by a simple induction.
The Beneš Network is nonblocking

- Each path $i$ has to traverse one of the sub-networks.
- Common inputs $2 \cdot i$ and $2 \cdot i - 1$ may not use the same sub-network.
- Common inputs $\pi(2 \cdot i)$ and $\pi(2 \cdot i - 1)$ may not use the same sub-network.
- The resulting conflict graph is bipartite (Sum of two Matchings).

Thus the paths may be placed on the two sub-networks.

The statement holds by a simple induction.
Introduction

Input

- Network: \( G = (V, E) \)
- Requests: \( I = \{(s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q\} \)
- Routes: \( \rho_1^i, \rho_2^i, \rho_3^i, \ldots \) paths from \( s_i \) to \( d_i \).

Routing

For the above input is a routing \( R \):

- \( R = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_q\} \) and
- \( \rho_i \) connects \( s_i \) with \( d_i \).
Introduction

**Input**

- **Network:** \( G = (V, E) \)
- **Requests:** \( I = \{(s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q\} \)
- **Routes:** \( \rho_1^i, \rho_2^i, \rho_3^i, \ldots \) paths from \( s_i \) to \( d_i \).

**Routing**

For the above input is a routing \( \mathcal{R} \):

- \( \mathcal{R} = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_q\} \) and
- \( \rho_i \) connects \( s_i \) with \( d_i \).
Introduction

**Input**

- Network: $G = (V, E)$
- Requests: $I = \{(s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q\}$
- Routes: $\rho^1_i, \rho^2_i, \rho^3_i, \ldots$ paths from $s_i$ to $d_i$.

**Routing**

For the above input is a routing $R$:

- $R = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_q\}$ and
- $\rho_i$ connects $s_i$ with $d_i$. 
Introduction

Input

- Network: \( G = (V, E) \)
- Requests: \( I = \{ (s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q \} \)
- Routes: \( \rho_1^i, \rho_2^i, \rho_3^i, \ldots \) paths from \( s_i \) to \( d_i \).

Routing

For the above input is a routing \( R \):

- \( R = \{ \rho_1, \rho_2, \rho_3, \ldots, \rho_q \} \) and
- \( \rho_i \) connects \( s_i \) with \( d_i \).
Input

- Network: \( G = (V, E) \)
- Requests: \( I = \{(s_i, d_i) | s_i, d_i \in V \land 1 \leq i \leq q\} \)
- Routes: \( \rho_1^i, \rho_2^i, \rho_3^i, \ldots \) paths from \( s_i \) to \( d_i \).

Routing

For the above input is a routing \( \mathcal{R} \):

- \( \mathcal{R} = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_q\} \) and
- \( \rho_i \) connects \( s_i \) with \( d_i \).
Input

- Network: $G = (V, E)$
- Requests: $I = \{(s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q\}$
- Routes: $\rho_1^i, \rho_2^i, \rho_3^i, \ldots$ paths from $s_i$ to $d_i$.

Routing

For the above input is a routing $R$:

- $R = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_q\}$ and
- $\rho_i$ connects $s_i$ with $d_i$. 
Introduction

Input

- Network: $G = (V, E)$
- Requests: $I = \{(s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q\}$
- Routes: $\rho_1^i, \rho_2^i, \rho_3^i, \ldots$ paths from $s_i$ to $d_i$.

Routing

For the above input is a routing $\mathcal{R}$:

- $\mathcal{R} = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_q\}$ and
- $\rho_i$ connects $s_i$ with $d_i$. 
Input

- Network: \( G = (V, E) \)
- Requests: \( I = \{ (s_i, d_i) | s_i, d_i \in V \land 1 \leq i \leq q \} \)
- Routes: \( \rho_1^i, \rho_2^i, \rho_3^i, \ldots \) paths from \( s_i \) to \( d_i \).

Routing

For the above input is a routing \( R \):

- \( R = \{ \rho_1, \rho_2, \rho_3, \ldots, \rho_q \} \) and
- \( \rho_i \) connects \( s_i \) with \( d_i \).
Wavelength-Assignment

Input

- Network: $G = (V, E)$
- Requests: $I = \{(s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q\}$
- Routing: $\mathcal{R} = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_q\}$

Wavelength-Assignment

is the colouring of the conflict-graph $G_R^I$:

- $G_R^I = (\mathcal{R}, F) \cong (I, F)$ mit: $F = \{\{\rho_i, \rho_j\} \mid \rho_i \cap \rho_j \cap E \neq \emptyset\}$
- Each request is assigned a wavelength.
- If two request share an edge (in the same direction), then differ the wavelengths.
- $w(G_R^I)$ is the number of necessary wavelengths.
Wavelength-Assignment

**Input**
- Network: \( G = (V, E) \)
- Requests: \( I = \{ (s_i, d_i) | s_i, d_i \in V \land 1 \leq i \leq q \} \)
- Routing: \( R = \{ \rho_1, \rho_2, \rho_3, \ldots, \rho_q \} \)

**Wavelength-Assignment**

is the colouring of the conflict-graph \( G^I_R \):
- \( G^I_R = (R, F) \uplus (I, F) \) mit: \( F = \{ \rho_i, \rho_j \} | \rho_i \cap \rho_j \cap E \neq \emptyset \)
- Each request is assigned a wavelength.
- If two request share an edge (in the same direction), then differ the wavelengths.
- \( w(G^I_R) \) is the number of necessary wavelengths.
Wavelength-Assignment

Input

- Network: \( G = (V, E) \)
- Requests: \( I = \{ (s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q \} \)
- Routing: \( R = \{ \rho_1, \rho_2, \rho_3, \ldots, \rho_q \} \)

Wavelength-Assignment

is the colouring of the conflict-graph \( G_R^I \):

- \( G_R^I = (R, F) \cap (I, F) \) mit: \( F = \{ \rho_i, \rho_j \mid \rho_i \cap \rho_j \cap E \neq \emptyset \} \)
- Each request is assigned a wavelength.
- If two request share an edge (in the same direction), then differ the wavelengths.
- \( w(G_R^I) \) is the number of necessary wavelengths.
Wavelength-Assignment

Input

- Network: \( G = (V, E) \)
- Requests: \( I = \{(s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q\} \)
- Routing: \( R = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_q\} \)

Wavelength-Assignment

is the colouring of the conflict-graph \( G^l_R \):

- \( G^l_R = (R, F) \cap (I, F) \) mit: \( F = \{\{\rho_i, \rho_j\} \mid \rho_i \cap \rho_j \cap E \neq \emptyset\} \)
- Each request is assigned a wavelength.
- If two requests share an edge (in the same direction), then differ the wavelengths.
- \( w(G^l_R) \) is the number of necessary wavelengths.
Wavelength-Assignment

Input
- Network: $G = (V, E)$
- Requests: $I = \{(s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q\}$
- Routing: $\mathcal{R} = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_q\}$

Wavelength-Assignment

is the colouring of the conflict-graph $G^I_{\mathcal{R}}$:
- $G^I_{\mathcal{R}} = (\mathcal{R}, F) \Delta (I, F)$ mit: $F = \{\{\rho_i, \rho_j\} \mid \rho_i \cap \rho_j \cap E \neq \emptyset\}$
- Each request is assigned a wavelength.
- If two request share an edge (in the same direction), then differ the wavelengths.
- $w(G^I_{\mathcal{R}})$ is the number of necessary wavelengths.
Wavelength-Assignment

Input

- Network: $G = (V, E)$
- Requests: $I = \{(s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q\}$
- Routing: $R = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_q\}$

Wavelength-Assignment

is the colouring of the conflict-graph $G^I_R$:

- $G^I_R = (R, F) \triangle (I, F)$ mit: $F = \{\{\rho_i, \rho_j\} \mid \rho_i \cap \rho_j \cap E \neq \emptyset\}$
- Each request is assigned a wavelength.
- If two request share an edge (in the same direction), then differ the wavelengths.
- $w(G^I_R)$ is the number of necessary wavelengths.
Wavelength-Assignment

**Input**

- Network: $G = (V, E)$
- Requests: $I = \{(s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q\}$
- Routing: $\mathcal{R} = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_q\}$

**Wavelength-Assignment**

is the colouring of the conflict-graph $G_R^I$:

- $G_R^I = (\mathcal{R}, F) \triangleq (I, F)$ mit: $F = \{\rho_i, \rho_j \mid \rho_i \cap \rho_j \cap E \neq \emptyset\}$
- Each request is assigned a wavelength.
- If two request share an edge (in the same direction), then differ the wavelengths.
- $w(G_R^I)$ is the number of necessary wavelengths.
Wavelength-Assignment

Input

- Network: $G = (V, E)$
- Requests: $I = \{(s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q\}$
- Routing: $\mathcal{R} = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_q\}$

Wavelength-Assignment

is the colouring of the conflict-graph $G_\mathcal{R}^I$:

- $G_\mathcal{R}^I = (\mathcal{R}, F) \bowtie (I, F)$ mit: $F = \{\{\rho_i, \rho_j\} \mid \rho_i \cap \rho_j \cap E \neq \emptyset\}$
- Each request is assigned a wavelength.
- If two request share an edge (in the same direction), then differ the wavelengths.
- $w(G_\mathcal{R}^I)$ is the number of necessary wavelengths.
Input

- Network: \( G = (V, E) \)
- Requests: \( I = \{(s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q\} \)
- Routing: \( \mathcal{R} = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_q\} \)

Wavelength-Assignment

is the colouring of the conflict-graph \( G^I_{\mathcal{R}} \):

- \( G^I_{\mathcal{R}} = (\mathcal{R}, F) \triangleq (I, F) \) mit: \( F = \{\{\rho_i, \rho_j\} \mid \rho_i \cap \rho_j \cap E \neq \emptyset\} \)
- Each request is assigned a wavelength.
- If two request share an edge (in the same direction), then differ the wavelengths.
- \( w(G^I_{\mathcal{R}}) \) is the number of necessary wavelengths.
Definition

Given:
- Network: $G = (V, E)$
- Requests: $I = \{(s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q\}$
- Routing: $R = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_q\}$

Then we define:
- The congestion of an edge $e$ the number of routing-paths which use $e$.
  - $c_e(G_I^R) = |\{r \in R \mid e \in r\}|$.
- $c(G_I^R) = \max_{e \in E} c_e(G_I^R)$.

Lemma

We have: $c(G_I^R) \leq w(G_I^R)$. 
Congestion

**Definition**

Given:
- Network: \( G = (V, E) \)
- Requests: \( I = \{ (s_i, d_i) | s_i, d_i \in V \land 1 \leq i \leq q \} \)
- Routing: \( R = \{ \rho_1, \rho_2, \rho_3, \ldots, \rho_q \} \)

Then we define:
- The congestion of an edge \( e \) the number of routing-paths which use \( e \).
- \( c_e(G^l_R) = |\{ r \in R | e \in r \}|. \)
- \( c(G^l_R) = \max_{e \in E} c_e(G^l_R). \)

**Lemma**

We have: \( c(G^l_R) \leq w(G^l_R). \)
Congestion

Definition

Given:
- Network: \( G = (V, E) \)
- Requests: \( I = \{(s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q\} \)
- Routing: \( R = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_q\} \)

Then we define:
- The congestion of an edge \( e \) the number of routing-paths which use \( e \).
- \( c_e(G^I_R) = |\{r \in R \mid e \in r\}| \).
- \( c(G^I_R) = \max_{e \in E} c_e(G^I_R) \).

Lemma

We have: \( c(G^I_R) \leq w(G^I_R) \).
Congestion

Definition

Given:
- Network: $G = (V, E)$
- Requests: $I = \{(s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q\}$
- Routing: $\mathcal{R} = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_q\}$

Then we define:
- The congestion of an edge $e$ the number of routing-paths which use $e$.
- $c_e(G_{IR}) = |\{r \in \mathcal{R} \mid e \in r\}|$.
- $c(G_{IR}) = \max_{e \in E} c_e(G_{IR})$.

Lemma

We have: $c(G_{IR}) \leq w(G_{IR})$. 
Congestion

Definition

Given:
- Network: \( G = (V, E) \)
- Requests: \( I = \{(s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q\} \)
- Routing: \( R = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_q\} \)

Then we define:
- The congestion of an edge \( e \) the number of routing-paths which use \( e \).
- \( c_e(G^l_R) = |\{r \in R \mid e \in r\}| \).
- \( c(G^l_R) = \max_{e \in E} c_e(G^l_R) \).

Lemma

We have: \( c(G^l_R) \leq w(G^l_R) \).
Congestion

**Definition**

Given:
- Network: $G = (V, E)$
- Requests: $I = \{(s_i, d_i) | s_i, d_i \in V \land 1 \leq i \leq q\}$
- Routing: $\mathcal{R} = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_q\}$

Then we define:
- The congestion of an edge $e$ the number of routing-paths which use $e$.
  - $c_e(G^l_\mathcal{R}) = |\{r \in \mathcal{R} | e \in r\}|$
  - $c(G^l_\mathcal{R}) = \max_{e \in E} c_e(G^l_\mathcal{R})$.

**Lemma**

We have: $c(G^l_\mathcal{R}) \leq w(G^l_\mathcal{R})$. 
Definition

Given:

- Network: \( G = (V, E) \)
- Requests: \( I = \{(s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q\} \)
- Routing: \( R = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_q\} \)

Then we define:

- The congestion of an edge \( e \) the number of routing-paths which use \( e \).
- \( c_e(G_R^l) = |\{r \in R \mid e \in r\}|. \)
- \( c(G_R^l) = \max_{e \in E} c_e(G_R^l). \)

Lemma

We have: \( c(G_R^l) \leq w(G_R^l). \)
Congestion

Definition

Given:
- Network: \( G = (V,E) \)
- Requests: \( I = \{(s_i,d_i) \mid s_i,d_i \in V \land 1 \leq i \leq q\} \)
- Routing: \( R = \{\rho_1,\rho_2,\rho_3,\ldots,\rho_q\} \)

Then we define:
- The congestion of an edge \( e \) the number of routing-paths which use \( e \).
- \( c_e(G^l_R) = \{|\{r \in R \mid e \in r\}|. \)
- \( c(G^l_R) = \max_{e \in E} c_e(G^l_R) \).

Lemma

We have: \( c(G^l_R) \leq w(G^l_R) \).
### Definition

Given:
- Network: $G = (V, E)$
- Requests: $I = \{(s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q\}$
- Routing: $R = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_q\}$

Then we define:
- The congestion of an edge $e$ the number of routing-paths which use $e$.
- $c_e(G^l_R) = |\{r \in R \mid e \in r\}|$
- $c(G^l_R) = \max_{e \in E} c_e(G^l_R)$

### Lemma

We have: $c(G^l_R) \leq w(G^l_R)$.
**Definition**

Given:
- Network: $G = (V, E)$
- Requests: $I = \{ (s_i, d_i) \mid s_i, d_i \in V \land 1 \leq i \leq q \}$
- Routing: $R = \{ \rho_1, \rho_2, \rho_3, \ldots, \rho_q \}$

Then we define:
- The congestion of an edge $e$ the number of routing-paths which use $e$.
- $c_e(G^I_R) = \{ r \in R \mid e \in r \}$.
- $c(G^I_R) = \max_{e \in E} c_e(G^I_R)$.

**Lemma**

*We have: $c(G^I_R) \leq w(G^I_R)$.***
**Greedy**

**Theorem**

Let $L$ be the maximal length of a routing-path in $G^l_R$.

- Then we have: $w(G^l_R) \leq (c(G^l_R) - 1) \cdot L + 1$
- Is also the bound for the simple greedy algorithm.

**Proof:** The node degree in the conflict-graph is at most: $(c(G^l_R) - 1) \cdot L$. 
Greedy

Theorem

Let $L$ be the maximal length of a routing-path in $G^l_R$.

- Then we have: $w(G^l_R) \leq (c(G^l_R) - 1) \cdot L + 1$
- Is also the bound for the simple greedy algorithm.

Proof: The node degree in the conflict-graph is at most: $(c(G^l_R) - 1) \cdot L$. 
**Greedy**

**Theorem**

Let $L$ be the maximal length of a routing-path in $G^l_R$.

- Then we have: $w(G^l_R) \leq (c(G^l_R) - 1) \cdot L + 1$
- Is also the bound for the simple greedy algorithm.

**Proof:** The node degree in the conflict-graph is at most: $(c(G^l_R) - 1) \cdot L$. 
**Greedy**

**Theorem**

Let $L$ be the maximal length of a routing-path in $G^l_R$.

- Then we have: $w(G^l_R) \leq (c(G^l_R) - 1) \cdot L + 1$
- Is also the bound for the simple greedy algorithm.

**Proof:** The node degree in the conflict-graph is at most: $(c(G^l_R) - 1) \cdot L$. 
Theorem

Let \( L \) be the maximal length of a routing-path in \( G^l_{IR} \).

- Then we have: \( w(G^l_{IR}) \leq (c(G^l_{IR}) - 1) \cdot L + 1 \)
- Is also the bound for the simple greedy algorithm.

Proof: The node degree in the conflict-graph is at most: \( (c(G^l_{IR}) - 1) \cdot L \).
Greedy improved

- Let $G^I_R$ be given.
- Let $\mathcal{R}_1$ be the paths of length $\geq \sqrt{|E|}$.
- Let $\mathcal{R}_2$ be the paths of length $< \sqrt{|E|}$.
- Colour each path in $\mathcal{R}_1$ with its own colour.
- Colour $\mathcal{R}_2$ with greed.

**Theorem**

We have: $w(G^I_R) \leq 2 \cdot \sqrt{|E|} \cdot c(G^I_R)$.

**Proof:**

- $|\mathcal{R}_1| \leq \sqrt{|E|} \cdot c(G^I_R)$, because
- otherwise we would have an edge $e$ with $c_e(G^I_R) > c(G^I_R)$.
- And $w(G^I_{R_2}) \leq \sqrt{|E|} \cdot c(G^I_R)$ is easy.
Greedy improved

- Let $G^I_R$ be given.
- Let $R_1$ be the paths of length $\geq \sqrt{|E|}$.
- Let $R_2$ be the paths of length $< \sqrt{|E|}$.
- Colour each path in $R_1$ with its own colour.
- Colour $R_2$ with greed.

**Theorem**

We have: $w(G^I_R) \leq 2 \cdot \sqrt{|E|} \cdot c(G^I_R)$.

**Proof:**

- $|R_1| \leq \sqrt{|E|} \cdot c(G^I_R)$, because
- otherwise we would have an edge $e$ with $c_e(G^I_R) > c(G^I_R)$.
- And $w(G^I_{R_2}) \leq \sqrt{|E|} \cdot c(G^I_R)$ is easy.
Greedy improved

- Let $G^I_R$ be given.
- Let $\mathcal{R}_1$ be the paths of length $\geq \sqrt{|E|}$.
- Let $\mathcal{R}_2$ be the paths of length $< \sqrt{|E|}$.
- Colour each path in $\mathcal{R}_1$ with its own colour.
- Colour $\mathcal{R}_2$ with greed.

**Theorem**

We have: $w(G^I_R) \leq 2 \cdot \sqrt{|E|} \cdot c(G^I_R)$.

**Proof:**

- $|\mathcal{R}_1| \leq \sqrt{|E|} \cdot c(G^I_R)$, because
- otherwise we would have an edge $e$ with $c_e(G^I_R) > c(G^I_R)$.
- And $w(G^I_{R_2}) \leq \sqrt{|E|} \cdot c(G^I_R)$ is easy.
Greedy improved

- Let $G^I_R$ be given.
- Let $R_1$ be the paths of length $\geq \sqrt{|E|}$.
- Let $R_2$ be the paths of length $< \sqrt{|E|}$.
- Colour each path in $R_1$ with its own colour.
- Colour $R_2$ with greed.

**Theorem**

We have: $w(G^I_R) \leq 2 \cdot \sqrt{|E|} \cdot c(G^I_R)$.

**Proof:**

- $|R_1| \leq \sqrt{|E|} \cdot c(G^I_R)$, because
- otherwise we would have an edge $e$ with $c_e(G^I_R) > c(G^I_R)$.
- And $w(G^I_{R_2}) \leq \sqrt{|E|} \cdot c(G^I_R)$ is easy.
Greedy improved

- Let $G^I_R$ be given.
- Let $R_1$ be the paths of length $\geq \sqrt{|E|}$.
- Let $R_2$ be the paths of length $< \sqrt{|E|}$.
- Colour each path in $R_1$ with its own colour.
- Colour $R_2$ with greed.

**Theorem**

We have: $w(G^I_R) \leq 2 \cdot \sqrt{|E|} \cdot c(G^I_R)$.

**Proof:**

- $|R_1| \leq \sqrt{|E|} \cdot c(G^I_R)$, because
- otherwise we would have an edge $e$ with $c_e(G^I_R) > c(G^I_R)$.
- And $w(G^I_{R_2}) \leq \sqrt{|E|} \cdot c(G^I_R)$ is easy.
Greedy improved

- Let $G^I_R$ be given.
- Let $\mathcal{R}_1$ be the paths of length $\geq \sqrt{|E|}$.
- Let $\mathcal{R}_2$ be the paths of length $< \sqrt{|E|}$.
- Colour each path in $\mathcal{R}_1$ with its own colour.
- Colour $\mathcal{R}_2$ with greed.

Theorem

\[ w(G^I_R) \leq 2 \cdot \sqrt{|E|} \cdot c(G^I_R). \]

Proof:

- $|\mathcal{R}_1| \leq \sqrt{|E|} \cdot c(G^I_R)$, because
- otherwise we would have an edge $e$ with $c_e(G^I_R) > c(G^I_R)$.
- And $w(G^I_{R_2}) \leq \sqrt{|E|} \cdot c(G^I_R)$ is easy.
Greedy improved

- Let $G_{\mathcal{R}}$ be given.
- Let $\mathcal{R}_1$ be the paths of length $\geq \sqrt{|E|}$.
- Let $\mathcal{R}_2$ be the paths of length $< \sqrt{|E|}$.
- Colour each path in $\mathcal{R}_1$ with its own colour.
- Colour $\mathcal{R}_2$ with greed.

**Theorem**

We have: $w(G_{\mathcal{R}}) \leq 2 \cdot \sqrt{|E|} \cdot c(G_{\mathcal{R}})$.

**Proof:**

- $|\mathcal{R}_1| \leq \sqrt{|E|} \cdot c(G_{\mathcal{R}})$, because
- otherwise we would have an edge $e$ with $c_e(G_{\mathcal{R}}) > c(G_{\mathcal{R}})$.
- And $w(G_{\mathcal{R}_2}) \leq \sqrt{|E|} \cdot c(G_{\mathcal{R}})$ is easy.
Greedy improved

- Let $G^I_R$ be given.
- Let $R_1$ be the paths of length $\geq \sqrt{|E|}$.
- Let $R_2$ be the paths of length $< \sqrt{|E|}$.
- Colour each path in $R_1$ with its own colour.
- Colour $R_2$ with greed.

**Theorem**

*We have: $w(G^I_R) \leq 2 \cdot \sqrt{|E|} \cdot c(G^I_R)$.***

**Proof:**

- $|R_1| \leq \sqrt{|E|} \cdot c(G^I_R)$, because
- otherwise we would have an edge $e$ with $c_e(G^I_R) > c(G^I_R)$.
- And $w(G^I_{R_2}) \leq \sqrt{|E|} \cdot c(G^I_R)$ is easy.
Greedy improved

- Let $G^l_R$ be given.
- Let $R_1$ be the paths of length $\geq \sqrt{|E|}$.
- Let $R_2$ be the paths of length $< \sqrt{|E|}$.
- Colour each path in $R_1$ with its own colour.
- Colour $R_2$ with greed.

**Theorem**

We have: $w(G^l_R) \leq 2 \cdot \sqrt{|E|} \cdot c(G^l_R)$.

**Proof:**

- $|R_1| \leq \sqrt{|E|} \cdot c(G^l_R)$, because
- otherwise we would have an edge $e$ with $c_e(G^l_R) > c(G^l_R)$.
- And $w(G^l_{R_2}) \leq \sqrt{|E|} \cdot c(G^l_R)$ is easy.
Greedy improved

- Let $G_R^I$ be given.
- Let $R_1$ be the paths of length $\geq \sqrt{|E|}$.
- Let $R_2$ be the paths of length $< \sqrt{|E|}$.
- Colour each path in $R_1$ with its own colour.
- Colour $R_2$ with greed.

**Theorem**

We have: $w(G_R^I) \leq 2 \cdot \sqrt{|E|} \cdot c(G_R^I)$.

**Proof:**

- $|R_1| \leq \sqrt{|E|} \cdot c(G_R^I)$, because
- otherwise we would have an edge $e$ with $c_e(G_R^I) > c(G_R^I)$.
- And $w(G_{R_2}^I) \leq \sqrt{|E|} \cdot c(G_R^I)$ is easy.
Greedy improved

- Let $G_R^1$ be given.
- Let $\mathcal{R}_1$ be the paths of length $\geq \sqrt{|E|}$.
- Let $\mathcal{R}_2$ be the paths of length $< \sqrt{|E|}$.
- Colour each path in $\mathcal{R}_1$ with its own colour.
- Colour $\mathcal{R}_2$ with greed.

**Theorem**

We have: $w(G_R^1) \leq 2 \cdot \sqrt{|E|} \cdot c(G_R^1)$.

**Proof:**

- $|\mathcal{R}_1| \leq \sqrt{|E|} \cdot c(G_R^1)$, because
- otherwise we would have an edge $e$ with $c_e(G_R^1) > c(G_R^1)$.
- And $w(G_R^2) \leq \sqrt{|E|} \cdot c(G_R^1)$ is easy.
Greedy improved

- Let $G_R^I$ be given.
- Let $\mathcal{R}_1$ be the paths of length $\geq \sqrt{|E|}$.
- Let $\mathcal{R}_2$ be the paths of length $< \sqrt{|E|}$.
- Colour each path in $\mathcal{R}_1$ with its own colour.
- Colour $\mathcal{R}_2$ with greed.

**Theorem**

We have: $w(G_R^I) \leq 2 \cdot \sqrt{|E|} \cdot c(G_R^I)$.

**Proof:**

- $|\mathcal{R}_1| \leq \sqrt{|E|} \cdot c(G_R^I)$, because
- otherwise we would have an edge $e$ with $c_e(G_R^I) > c(G_R^I)$.
- And $w(G_{R_2}^I) \leq \sqrt{|E|} \cdot c(G_R^I)$ is easy.
If $G$ is a line, then we can compute $w(G^l_R)$ in polynomial time.

Proof:

- Let $I_l$ be the requests going to the left.
- Let $I_r$ be the requests going to the right.
- $I_l$ and $I_r$ are independent.
- $w(G^l_R)$ corresponds to the colouring of an interval-graph.
- $w(G^l_R)$ corresponds to the colouring of an interval-graph.
Theorem

If $G$ is a line, then we can compute $w(G^l_R)$ in polynomial time.

Proof:

- Let $I_l$ be the requests going to the left.
- Let $I_r$ be the requests going to the right.
- $I_l$ and $I_r$ are independent.
- $w(G^l_R)$ corresponds to the colouring of an interval-graph.
- $w(G^l_R)$ corresponds to the colouring of an interval-graph.
Theorem

If \( G \) is a line, then we can compute \( w(G_{rl}^l) \) in polynomial time.

Proof:

- Let \( I_l \) be the requests going to the left.
- Let \( I_r \) be the requests going to the right.
- \( I_l \) and \( I_r \) are independent.
- \( w(G_{rl}^l) \) corresponds to the colouring of an interval-graph.
- \( w(G_{rl}^r) \) corresponds to the colouring of an interval-graph.
**Theorem**

If $G$ is a line, then we can compute $w(G^l_r)$ in polynomial time.

**Proof:**

- Let $I_l$ be the requests going to the left.
- Let $I_r$ be the requests going to the right.
- $I_l$ and $I_r$ are independent.
- $w(G^l_r)$ corresponds to the colouring of an interval-graph.
- $w(G^r_l)$ corresponds to the colouring of an interval-graph.
If $G$ is a line, then we can compute $w(G^l_R)$ in polynomial time.

Proof:

- Let $l_l$ be the requests going to the left.
- Let $l_r$ be the requests going to the right.
- $l_l$ and $l_r$ are independent.
- $w(G^l_R)$ corresponds to the colouring of an interval-graph.
- $w(G^r_R)$ corresponds to the colouring of an interval-graph.
**Theorem**

If $G$ is a line, then we can compute $w(G^l_R)$ in polynomial time.

**Proof:**
- Let $I_l$ be the requests going to the left.
- Let $I_r$ be the requests going to the right.
- $I_l$ and $I_r$ are independent.
- $w(G^l_R)$ corresponds to the colouring of an interval-graph.
- $w(G^r_R)$ corresponds to the colouring of an interval-graph.
Theorem

If $G$ is a line, then we can compute $w(G^l_R)$ in polynomial time.

Proof:

- Let $I_l$ be the requests going to the left.
- Let $I_r$ be the requests going to the right.
- $I_l$ and $I_r$ are independent.
- $w(G^l_R)$ corresponds to the colouring of an interval-graph.
- $w(G^l_R)$ corresponds to the colouring of an interval-graph.
**Theorem**

*If \( G \) is a cycle, then we can approximate \( w(G^I_R) \) in polynomial time with a factor of 2.*

**Proof:**
- Let \( e \) be an edge in \( G \).
- Let \( I_1 \) be the requests which use \( e \) in the routing.
- Let \( I_2 \) be the requests which do not use \( e \) in the routing.
- \( w(G^I_{I_1}R) \) corresponds to the colouring of an interval-graph.
- \( w(G^I_{I_2}R) \) corresponds to the colouring of an interval-graph.

**Theorem**

*If \( G \) is a cycle, then the computation of \( w(G^I_R) \) is NP-complete.*

**Proof:**
- \( w(G^I_R) \) corresponds to the colouring of an arc-graph.
Theorem

If $G$ is a cycle, then we can approximate $w(G^I_\mathcal{R})$ in polynomial time with a factor of 2.

Proof:

- Let $e$ be an edge in $G$.
- Let $I_1$ be the requests which use $e$ in the routing.
- Let $I_2$ be the requests which do not use $e$ in the routing.
- $w(G^I_\mathcal{R})$ corresponds to the colouring of an interval-graph.
- $w(G^{I_2}_\mathcal{R})$ corresponds to the colouring of an interval-graph.

Theorem

If $G$ is a cycle, then the computation of $w(G^I_\mathcal{R})$ is NP-complete.

Proof:

- $w(G^I_\mathcal{R})$ corresponds to the colouring of an arc-graph.
**Theorem**

*If $G$ is a cycle, then we can approximate $w(G^l_R)$ in polynomial time with a factor of 2.*

**Proof:**

- Let $e$ be an edge in $G$.
- Let $I_1$ be the requests which use $e$ in the routing.
- Let $I_2$ be the requests which do not use $e$ in the routing.
- $w(G^l_R)$ corresponds to the colouring of an interval-graph.
- $w(G^l_R)$ corresponds to the colouring of an interval-graph.

**Theorem**

*If $G$ is a cycle, then the computation of $w(G^l_R)$ is NP-complete.*

**Proof:**

- $w(G^l_R)$ corresponds to the colouring of an arc-graph.
**Theorem**

*If\ G\ is\ a\ cycle,\ then\ we\ can\ approximate\ \(w(G^I_R)\)\ in\ polynomial\ time\ with\ a\ factor\ of\ 2.*

**Proof:**

- Let \(e\) be an edge in \(G\).
- Let \(I_1\) be the requests which use \(e\) in the routing.
- Let \(I_2\) be the requests which do not use \(e\) in the routing.
- \(w(G^I_1^R)\) corresponds to the colouring of an interval-graph.
- \(w(G^I_2^R)\) corresponds to the colouring of an interval-graph.

**Theorem**

*If\ G\ is\ a\ cycle,\ then\ the\ computation\ of\ \(w(G^I_R)\)\ is\ NP-complete.*

**Proof:**

- \(w(G^I_R)\) corresponds to the colouring of an arc-graph.
Cycle

Theorem

If $G$ is a cycle, then we can approximate $w(G^I_R)$ in polynomial time with a factor of 2.

Proof:

- Let $e$ be an edge in $G$.
- Let $I_1$ be the requests which use $e$ in the routing.
- Let $I_2$ be the requests which do not use $e$ in the routing.
- $w(G^I_{1R})$ corresponds to the colouring of an interval-graph.
- $w(G^I_{2R})$ corresponds to the colouring of an interval-graph.

Theorem

If $G$ is a cycle, then the computation of $w(G^I_R)$ is NP-complete.

Proof:

- $w(G^I_{R})$ corresponds to the colouring of an arc-graph.
**Theorem**

*If $G$ is a cycle, then we can approximate $w(G^I_R)$ in polynomial time with a factor of 2.*

**Proof:**

- Let $e$ be an edge in $G$.
- Let $I_1$ be the requests which use $e$ in the routing.
- Let $I_2$ be the requests which do not use $e$ in the routing.
- $w(G^I_1 R)$ corresponds to the colouring of an interval-graph.
- $w(G^I_2 R)$ corresponds to the colouring of an interval-graph.

**Theorem**

*If $G$ is a cycle, then the computation of $w(G^I_R)$ is NP-complete.*

**Proof:**

- $w(G^I_R)$ corresponds to the colouring of an arc-graph.
**Theorem**

*If $G$ is a cycle, then we can approximate $w(G^I_R)$ in polynomial time with a factor of 2.*

**Proof:**

- Let $e$ be an edge in $G$.
- Let $I_1$ be the requests which use $e$ in the routing.
- Let $I_2$ be the requests which do not use $e$ in the routing.
- $w(G^I_R)$ corresponds to the colouring of an interval-graph.
- $w(G^{I_1}_R)$ corresponds to the colouring of an interval-graph.

**Theorem**

*If $G$ is a cycle, then the computation of $w(G^I_R)$ is NP-complete.*

**Proof:**

- $w(G^I_R)$ corresponds to the colouring of an arc-graph.
Theorem

If $G$ is a cycle, then we can approximate $w(G^I_R)$ in polynomial time with a factor of 2.

Proof:

- Let $e$ be an edge in $G$.
- Let $I_1$ be the requests which use $e$ in the routing.
- Let $I_2$ be the requests which do not use $e$ in the routing.
- $w(G^I_1)$ corresponds to the colouring of an interval-graph.
- $w(G^I_2)$ corresponds to the colouring of an interval-graph.

Theorem

If $G$ is a cycle, then the computation of $w(G^I_R)$ is NP-complete.

Proof:

- $w(G^I_R)$ corresponds to the colouring of an arc-graph.
**Theorem**

*If $G$ is a cycle, then we can approximate $w(G^l_\mathcal{R})$ in polynomial time with a factor of 2.*

**Proof:**

- Let $e$ be an edge in $G$.
- Let $I_1$ be the requests which use $e$ in the routing.
- Let $I_2$ be the requests which do not use $e$ in the routing.
- $w(G^l_{\mathcal{I}_1 \mathcal{R}})$ corresponds to the colouring of an interval-graph.
- $w(G^l_{\mathcal{I}_2 \mathcal{R}})$ corresponds to the colouring of an interval-graph.

**Theorem**

*If $G$ is a cycle, then the computation of $w(G^l_{\mathcal{R}})$ is NP-complete.*

**Proof:**

- $w(G^l_{\mathcal{R}})$ corresponds to the colouring of an arc-graph.
Theorem

If $G$ is a cycle, then we can approximate $w(G^I_R)$ in polynomial time with a factor of 2.

Proof:
- Let $e$ be an edge in $G$.
- Let $I_1$ be the requests which use $e$ in the routing.
- Let $I_2$ be the requests which do not use $e$ in the routing.
- $w(G^I_R)$ corresponds to the colouring of an interval-graph.
- $w(G^I_2)$ corresponds to the colouring of an interval-graph.

Theorem

If $G$ is a cycle, then the computation of $w(G^I_R)$ is NP-complete.

Proof:
- $w(G^I_R)$ corresponds to the colouring of an arc-graph.
Theorem

If $G$ is a cycle, then we can approximate $w(G_{IR}^I)$ in polynomial time with a factor of 2.

Proof:

- Let $e$ be an edge in $G$.
- Let $I_1$ be the requests which use $e$ in the routing.
- Let $I_2$ be the requests which do not use $e$ in the routing.
- $w(G_{IR}^{I_1})$ corresponds to the colouring of an interval-graph.
- $w(G_{IR}^{I_2})$ corresponds to the colouring of an interval-graph.

Theorem

If $G$ is a cycle, then the computation of $w(G_{IR}^I)$ is NP-complete.

Proof:

- $w(G_{IR}^I)$ corresponds to the colouring of an arc-graph.
Theorem

If \( G \) is a star, then we can compute \( w(G^I_T) \) in polynomial time.

Proof:

- Let \( G = (\{0, 1, \ldots, n\}, E) \) be the star with central node 0.
- Let \( H = (\{s_1, s_2, \ldots s_n\}, \{d_1, d_2, \ldots d_n\}, F) \) be a bipartite graph,
- with: \( F = \{(s_i, d_j) \mid (i, j) \in I\} \)
- Computing of \( w(G^I_T) \) corresponds to the edge-colouring of \( H \).
- Request of the form \( 0, i \) and \( i, 0 \) may be coloured later by greed.
Theorem

If $G$ is a star, then we can compute $w(G_{IR}^I)$ in polynomial time.

Proof:

- Let $G = (\{0, 1, \ldots, n\}, E)$ be the star with central node 0.
- Let $H = (\{s_1, s_2, \ldots s_n\}, \{d_1, d_2, \ldots d_n\}, F)$ be a bipartite graph,
- with: $F = \{(s_i, d_j) \mid (i, j) \in I\}$
- Computing of $w(G_{IR}^I)$ corresponds to the edge-colouring of $H$.
- Request of the form $0, i$ and $i, 0$ may be coloured later by greed.
Theorem

If $G$ is a star, then we can compute $w(G^I_{\mathcal{R}})$ in polynomial time.

Proof:

- Let $G = (\{0, 1, \ldots, n\}, E)$ be the star with central node 0.
- Let $H = (\{s_1, s_2, \ldots s_n\}, \{d_1, d_2, \ldots d_n\}, F)$ be a bipartite graph,
- with: $F = \{(s_i, d_j) | (i, j) \in I\}$
- Computing of $w(G^I_{\mathcal{R}})$ corresponds to the edge-colouring of $H$.
- Request of the form $0, i$ and $i, 0$ may be coloured later by greed.
Theorem

If $G$ is a star, then we can compute $w(G^I_R)$ in polynomial time.

Proof:

- Let $G = (\{0, 1, \ldots, n\}, E)$ be the star with central node 0.
- Let $H = (\{s_1, s_2, \ldots s_n\}, \{d_1, d_2, \ldots d_n\}, F)$ be a bipartite graph, with: $F = \{(s_i, d_j) \mid (i, j) \in I\}$
- Computing of $w(G^I_R)$ corresponds to the edge-colouring of $H$.
- Request of the form $0, i$ and $i, 0$ may be coloured later by greed.
Theorem

If $G$ is a star, then we can compute $w(G^l_{IR})$ in polynomial time.

Proof:

- Let $G = (\{0, 1, \ldots, n\}, E)$ be the star with central node 0.
- Let $H = (\{s_1, s_2, \ldots, s_n\}, \{d_1, d_2, \ldots, d_n\}, F)$ be a bipartite graph,
- with: $F = \{(s_i, d_j) \mid (i, j) \in I\}$
- Computing of $w(G^l_{IR})$ corresponds to the edge-colouring of $H$.
- Request of the form $0, i$ and $i, 0$ may be coloured later by greed.
Theorem

If $G$ is a star, then we can compute $w(G^l_T)$ in polynomial time.

Proof:

- Let $G = (\{0, 1, \ldots, n\}, E)$ be the star with central node 0.
- Let $H = (\{s_1, s_2, \ldots s_n\}, \{d_1, d_2, \ldots d_n\}, F)$ be a bipartite graph,
- with: $F = \{(s_i, d_j) \mid (i, j) \in I\}$
- Computing of $w(G^l_T)$ corresponds to the edge-colouring of $H$.
- Request of the form $0, i$ and $i, 0$ may be coloured later by greed.
**Star**

**Theorem**

*If $G$ is a star, then we can compute $w(G^I_R)$ in polynomial time.*

**Proof:**

- Let $G = (\{0, 1, \ldots, n\}, E)$ be the star with central node 0.
- Let $H = (\{s_1, s_2, \ldots s_n\}, \{d_1, d_2, \ldots d_n\}, F)$ be a bipartite graph,
- with: $F = \{(s_i, d_j) \mid (i, j) \in I\}$
- Computing of $w(G^I_R)$ corresponds to the edge-colouring of $H$.
- Request of the form $0, i$ and $i, 0$ may be coloured later by greed.
Theorem

If $G$ is a spider-graph, then we can compute $w(G^1_R)$ in polynomial time.

Proof:

- Colour first the center star.
- Extend the colouring on each leg of the spider-graph by using the algorithm for paths.
Theorem

If $G$ is a spider-graph, then we can compute $w(G_{IR})$ in polynomial time.

Proof:

- Colour first the center star.
- Extend the colouring on each leg of the spider-graph by using the algorithm for paths.
If \( G \) is a spider-graph, then we can compute \( w(G^l_R) \) in polynomial time.

Proof:

- Colour first the center star.
- Extend the colouring on each leg of the spider-graph by using the algorithm for paths.
Theorem

If $G$ is a spider-graph, then we can compute $w(G^1_R)$ in polynomial time.

Proof:

- Colour first the center star.
- Extend the colouring on each leg of the spider-graph by using the algorithm for paths.
Theorem

If \( G \) is a tree, then the computation of \( w(G^l_R) \) is NP-complete.

Proof:

- \( w(G^l_R) \) corresponds to the colouring of an EPT-Graph.
Baum

Theorem

*If G is a tree, then the computation of w(Gₐᵣ) is NP-complete.*

Proof:

- w(Gₐᵣ) corresponds to the colouring of an EPT-Graph.
Theorem

If $G$ is a tree, then the computation of $w(G_{IR})$ is NP-complete.

Proof:

- $w(G_{IR})$ corresponds to the colouring of an EPT-Graph.
If the requests are of type broadcast, then the wavelength-assignment becomes easy.

- We have: $I = \{(v, w) \mid w \in V\}$ for a start node $v$.
- There are $|V| - 1$ nodes to be informed from $v$.
- There have to be $|V| - 1$ paths starting in $v$.
- Let $d(w)$ be the out-degree of node $w \in V$.
- Let $d_{\min}(G) = \min_{w \in V} d(w)$.
- At least $(|V| - 1)/d(v)$ requests use the same edge of $v$.
- Thus we have: $w(G^I) \geq \lceil (|V| - 1)/d_{\min}(G) \rceil$. 
Broadcast

- If the requests are of type broadcast, then the wavelength-assignment becomes easy.
- We have: $I = \{(v, w) \mid w \in V\}$ for a start node $v$.
- There are $|V| - 1$ nodes to be informed from $v$.
- There have to be $|V| - 1$ paths starting in $v$.
- Let $d(w)$ be the out-degree of node $w \in V$.
- Let $d_{\text{min}}(G) = \min_{v \in V} d(v)$.
- At least $(|V| - 1)/d(v)$ requests use the same edge of $v$.
- Thus we have: $w(G^I_R) \geq \lceil (|V| - 1)/d_{\text{min}}(G) \rceil$. 
If the requests are of type broadcast, then the wavelength-assignment becomes easy.

We have: \( I = \{ (v, w) \mid w \in V \} \) for a start node \( v \).

There are \( |V| - 1 \) nodes to be informed from \( v \).

There have to be \( |V| - 1 \) paths starting in \( v \).

Let \( d(w) \) be the out-degree of node \( w \in V \).

Let \( d_{\min}(G) = \min_{w \in V} d(w) \).

At least \( (|V| - 1)/d(v) \) requests use the same edge of \( v \).

Thus we have: \( w(G^I_R) \geq \lceil (|V| - 1)/d_{\min}(G) \rceil \).
If the requests are of type broadcast, then the wavelength-assignment becomes easy.

We have: \( I = \{ (v, w) \mid w \in V \} \) for a start node \( v \).

There are \( |V| - 1 \) nodes to be informed from \( v \).

There have to be \( |V| - 1 \) paths starting in \( v \).

Let \( d(w) \) be the out-degree of node \( w \in V \).

Let \( d_{\text{min}}(G) = \min_{w \in V} d(w) \).

At least \( (|V| - 1)/d(v) \) requests use the same edge of \( v \).

Thus we have: \( w(G^I_R) \geq \lceil (|V| - 1)/d_{\text{min}}(G) \rceil \).
If the requests are of type broadcast, then the wavelength-assignment becomes easy.

- We have: \( \mathcal{I} = \{ (v, w) \mid w \in V \} \) for a start node \( v \).
- There are \( |V| - 1 \) nodes to be informed from \( v \).
- There have to be \( |V| - 1 \) paths starting in \( v \).
- Let \( d(w) \) be the out-degree of node \( w \in V \).
- Let \( d_{\text{min}}(G) = \min_{w \in V} d(w) \).
- At least \( (|V| - 1)/d(v) \) requests use the same edge of \( v \).
- Thus we have: \( w(G_{\mathcal{R}}) \geq \lceil (|V| - 1)/d_{\text{min}}(G) \rceil \).
If the requests are of type broadcast, then the wavelength-assignment becomes easy.

We have: \( l = \{ (v, w) \mid w \in V \} \) for a start node \( v \).

There are \(|V| - 1\) nodes to be informed from \( v \).

There have to be \(|V| - 1\) paths starting in \( v \).

Let \( d(w) \) be the out-degree of node \( w \in V \).

Let \( d_{\text{min}}(G) = \min_{w \in V} d(w) \).

At least \((|V| - 1)/d(v)\) requests use the same edge of \( v \).

Thus we have: \( w(G_I^l) \geq \lceil (|V| - 1)/d_{\text{min}}(G) \rceil \).
If the requests are of type broadcast, then the wavelength-assignment becomes easy.

- We have: \( I = \{(v, w) \mid w \in V\} \) for a start node \( v \).
- There are \(|V| - 1\) nodes to be informed from \( v \).
- There have to be \(|V| - 1\) paths starting in \( v \).
- Let \( d(w) \) be the out-degree of node \( w \in V \).
- Let \( d_{\text{min}}(G) = \min_{w \in V} d(w) \).
- At least \((|V| - 1)/d(v)\) requests use the same edge of \( v \).
- Thus we have: \( w(G^I_R) \geq \lceil (|V| - 1)/d_{\text{min}}(G) \rceil \).
If the requests are of type broadcast, then the wavelength-assignment becomes easy.

- We have: \( I = \{(v, w) \mid w \in V\} \) for a start node \( v \).
- There are \( |V| - 1 \) nodes to be informed from \( v \).
- There have to be \( |V| - 1 \) paths starting in \( v \).
- Let \( d(w) \) be the out-degree of node \( w \in V \).
- Let \( d_{\text{min}}(G) = \min_{w \in V} d(w) \).
- At least \( (|V| - 1)/d(v) \) requests use the same edge of \( v \).
- Thus we have: \( w(G_{\mathcal{R}}) \geq \lceil (|V| - 1)/d_{\text{min}}(G) \rceil \).
**Theorem**

*For an $k$ edge connected graph we have: $w(G_{IR}^l) \leq \lceil (|V| - 1)/k \rceil$.*

**Proof:**

- Let $v$ be the start-node.
- Split $V \setminus \{v\}$ into $s = \lceil (|V| - 1)/k \rceil$ subsets, with:
  - $V_1, V_2, \ldots, V_s$ have a size of at most $k$.
  - For each $i$ exist $k$ edge-disjoined paths from $v$ to $V_i$.
- Each $V_i$ will be informed by using colour $i$.
- In total are $s = \lceil (|V| - 1)/k \rceil$ colours used.
Theorem

For an $k$ edge connected graph we have: $w(G^I_R) \leq \lceil(|V| - 1)/k \rceil$.

Proof:

- Let $v$ be the start-node.
- Split $V \setminus \{v\}$ into $s = \lceil(|V| - 1)/k \rceil$ subsets, with:
  - $V_1, V_2, \ldots, V_s$ have a size of at most $k$.
  - For each $i$ exist $k$ edge-disjoined paths from $v$ to $V_i$.
- Each $V_i$ will be informed by using colour $i$.
- In total are $s = \lceil(|V| - 1)/k \rceil$ colours used.
Theorem

For an $k$ edge connected graph we have: $w(G^r) \leq \lceil (|V| - 1)/k \rceil$.

Proof:

- Let $v$ be the start-node.
- Split $V \setminus \{v\}$ into $s = \lceil (|V| - 1)/k \rceil$ subsets, with:
  - $V_1, V_2, \ldots, V_s$ have a size of at most $k$.
  - For each $i$ exist $k$ edge-disjoined paths from $v$ to $V_i$.
  - Each $V_i$ will be informed by using colour $i$.
- In total are $s = \lceil (|V| - 1)/k \rceil$ colours used.
**Theorem**

For an \( k \) edge connected graph we have: \( w(G^I_R) \leq \lceil (|V| - 1)/k \rceil \).

**Proof:**

- Let \( v \) be the start-node.
- Split \( V \setminus \{v\} \) into \( s = \lceil (|V| - 1)/k \rceil \) subsets, with:
  - \( V_1, V_2, \ldots, V_s \) have a size of at most \( k \).
  - For each \( i \) exist \( k \) edge-disjoined paths from \( v \) to \( V_i \).
  - Each \( V_i \) will be informed by using colour \( i \).
- In total are \( s = \lceil (|V| - 1)/k \rceil \) colours used.
Broadcast

Theorem

For an $k$ edge connected graph we have: $w(G^I_R) \leq \lceil (|V| - 1)/k \rceil$.

Proof:

- Let $v$ be the start-node.
- Split $V \setminus \{v\}$ into $s = \lceil (|V| - 1)/k \rceil$ subsets, with:
  - $V_1, V_2, \ldots, V_s$ have a size of at most $k$.
  - For each $i$ exist $k$ edge-disjoined paths from $v$ to $V_i$.
  - Each $V_i$ will be informed by using colour $i$.
- In total are $s = \lceil (|V| - 1)/k \rceil$ colours used.
Broadcast

Theorem

For an $k$ edge connected graph we have: $w(G^l_R) \leq \lceil(|V| - 1)/k \rceil$.

Proof:

- Let $v$ be the start-node.
- Split $V \setminus \{v\}$ into $s = \lceil(|V| - 1)/k \rceil$ subsets, with:
  - $V_1, V_2, \ldots, V_s$ have a size of at most $k$.
  - For each $i$ exist $k$ edge-disjoined paths from $v$ to $V_i$.
- Each $V_i$ will be informed by using colour $i$.
- In total are $s = \lceil(|V| - 1)/k \rceil$ colours used.
Broadcast

Theorem

For an \( k \) edge connected graph we have: \( w(G^I_R) \leq \lceil (|V| - 1)/k \rceil \).

Proof:

- Let \( v \) be the start-node.
- Split \( V \setminus \{v\} \) into \( s = \lceil (|V| - 1)/k \rceil \) subsets, with:
  - \( V_1, V_2, \ldots, V_s \) have a size of at most \( k \).
  - For each \( i \) exist \( k \) edge-disjoined paths from \( v \) to \( V_i \).
  - Each \( V_i \) will be informed by using colour \( i \).
- In total are \( s = \lceil (|V| - 1)/k \rceil \) colours used.
Broadcast

**Theorem**

For an $k$ edge connected graph we have: $w(G_{IR}) \leq \lceil (|V| - 1)/k \rceil$.

**Proof:**

- Let $v$ be the start-node.
- Split $V \setminus \{v\}$ into $s = \lceil (|V| - 1)/k \rceil$ subsets, with:
  - $V_1, V_2, \ldots, V_s$ have a size of at most $k$.
  - For each $i$ exist $k$ edge-disjoined paths from $v$ to $V_i$.
  - Each $V_i$ will be informed by using colour $i$.
- In total are $s = \lceil (|V| - 1)/k \rceil$ colours used.
For an $k$ edge connected graph we have: $w(G^l_R) = \lceil (|V| - 1)/k \rceil$.

Proof:

- Known: $w(G^l_R) \geq \lceil (|V| - 1)/d_{\text{min}}(G) \rceil$.
- Known: $w(G^l_R) \leq \lceil (|V| - 1)/k \rceil$.
- Known: $k \leq d_{\text{min}} G$.
- Thus we have: $w(G^l_R) = \lceil (|V| - 1)/k \rceil$. 

Broadcast
Theorem

For an $k$ edge connected graph we have: $w(G^l_R) = \lceil(|V| - 1)/k \rceil$.

Proof:
- Known: $w(G^l_R) \geq \lceil(|V| - 1)/d_{min}(G) \rceil$.
- Known: $w(G^l_R) \leq \lceil(|V| - 1)/k \rceil$.
- Known: $k \leq d_{min}G$.
- Thus we have: $w(G^l_R) = \lceil(|V| - 1)/k \rceil$. 
Theorem

For an $k$ edge connected graph we have: $w(G^I_R) = \lceil (|V| - 1)/k \rceil$.

Proof:

- Known: $w(G^I_R) \geq \lceil (|V| - 1)/d_{\min}(G) \rceil$.
- Known: $w(G^I_R) \leq \lceil (|V| - 1)/k \rceil$.
- Known: $k \leq d_{\min}G$.
- Thus we have: $w(G^I_R) = \lceil (|V| - 1)/k \rceil$. 

Broadcast
**Theorem**

For an $k$ edge connected graph we have: $w(G_{IR}^l) = \lceil (|V| - 1)/k \rceil$.

**Proof:**

- Known: $w(G_{IR}^l) \geq \lceil (|V| - 1)/d_{min}(G) \rceil$.
- Known: $w(G_{IR}^l) \leq \lceil (|V| - 1)/k \rceil$.
- Known: $k \leq d_{min} G$.
- Thus we have: $w(G_{IR}^l) = \lceil (|V| - 1)/k \rceil$. 
Theorem

For an $k$ edge connected graph we have: $w(G_I^R) = \lceil (|V| - 1)/k \rceil$.

Proof:

- **Known:** $w(G_I^R) \geq \lceil (|V| - 1)/d_{\text{min}}(G) \rceil$.
- **Known:** $w(G_I^R) \leq \lceil (|V| - 1)/k \rceil$.
- **Known:** $k \leq d_{\text{min}} G$.
- Thus we have: $w(G_I^R) = \lceil (|V| - 1)/k \rceil$. 

Theorem

For an $k$ edge connected graph we have: $w(G^l_R) = \lceil (|V| - 1)/k \rceil$.

Proof:

- Known: $w(G^l_R) \geq \lceil (|V| - 1)/d_{min}(G) \rceil$.
- Known: $w(G^l_R) \leq \lceil (|V| - 1)/k \rceil$.
- Known: $k \leq d_{min}G$.
- Thus we have: $w(G^l_R) = \lceil (|V| - 1)/k \rceil$. 
Theorem

For the following graphs it is NP-complete to compute $w(G_{r_{\min}}^I)$:

- cycles,
- trees,
- binary trees and
- grids.
More Results

Theorem

For the following graphs it is NP-complete to compute $w(G^I_{r_{\text{min}}})$:

- cycles,
- trees,
- binary trees and
- grids.
More Results

Theorem

For the following graphs it is NP-complete to compute $w(G^I_{\text{min}})$:

- cycles,
- trees,
- binary trees and
- grids.
More Results

Theorem

For the following graphs it is NP-complete to compute $w(G_{R_{min}}^I)$:

- cycles,
- trees,
- binary trees and
- grids.
More Results

Theorem

For the following graphs it is NP-complete to compute $w(G^I_{R_{min}})$:

- cycles,
- trees,
- binary trees and
- grids.
Theorem

Let $G^I_{\min}$ given with $L = \max_{(x,y) \in I} \text{dist}(x,y)$. Then we have: $w(G^I_{\min}) = O(L \cdot c(G^I_{\min}))$. 

Theorem

For each $L$ and $c$ there exists $G^I_{\min}$ with: $L = \max_{(x,y) \in I} \text{dist}(x,y)$, $c = c(G^I_{\min})$ $w(G^I_{\min}) = \Omega(L \cdot c)$. 

Theorem

Let $G^I_{\min}$ given with $I$ is “one-to-many” communication. Then we have: $w(G^I_{\min}) = c(G^I_{\min})$. 

More Results
More Results

**Theorem**

Let $G_{I_{\text{min}}}^l$ given with $L = \max_{(x,y) \in I} \dist(x,y)$. Then we have: $\w(G_{I_{\text{R}}}^l) = O(L \cdot c(G_{I_{\text{R}}}^l))$.

**Theorem**

For each $L$ and $c$ there exists $G_{I_{\text{min}}}^l$ with: $L = \max_{(x,y) \in I} \dist(x,y)$, $c = c(G_{I_{\text{min}}}^l)$ $\w(G_{I_{\text{R}}}^l) = \Omega(L \cdot c)$.

**Theorem**

Let $G_{I_{\text{min}}}^l$ given with $I$ is “one-to-many” communication. Then we have: $\w(G_{I_{\text{R}}}^l) = c(G_{I_{\text{R}}}^l)$. 
More Results

**Theorem**

Let $G_{R_{\text{min}}}^I$ given with $L = \max_{(x,y) \in I} \text{dist}(x, y)$. Then we have: $w(G_{R_{\text{min}}}^I) = O(L \cdot c(G_{R_{\text{min}}}^I))$.

**Theorem**

For each $L$ and $c$ there exists $G_{R_{\text{min}}}^I$ with: $L = \max_{(x,y) \in I} \text{dist}(x, y)$, $c = c(G_{R_{\text{min}}}^I)$ $w(G_{R_{\text{min}}}^I) = \Omega(L \cdot c)$.

**Theorem**

Let $G_{R_{\text{min}}}^I$ given with $I$ is “one-to-many” communication. Then we have: $w(G_{R_{\text{min}}}^I) = c(G_{R_{\text{min}}}^I)$.
Theorem

Let $G_{R_{\min}}^I$ given with $L = \max_{(x,y) \in I} \text{dist}(x,y)$. Then we have: $w(G_{R}^I) = O(L \cdot c(G_{R}^I))$.

Theorem

For each $L$ and $c$ there exists $G_{R_{\min}}^I$ with: $L = \max_{(x,y) \in I} \text{dist}(x,y)$, $c = c(G_{R_{\min}}^I)$ $w(G_{R}^I) = \Omega(L \cdot c)$.

Theorem

Let $G_{R_{\min}}^I$ given with $I$ is “one-to-many” communication. Then we have: $w(G_{R}^I) = c(G_{R}^I)$. 
More Results

**Theorem**

Let $G^I_{\text{R min}}$ given with $L = \max_{(x,y) \in I} \text{dist}(x,y)$. Then we have: $w(G^I_{\text{R}}) = O(L \cdot c(G^I_{\text{R}}))$.

**Theorem**

For each $L$ and $c$ there exists $G^I_{\text{R min}}$ with: $L = \max_{(x,y) \in I} \text{dist}(x,y)$, $c = c(G^I_{\text{R min}})$ $w(G^I_{\text{R}}) = \Omega(L \cdot c)$.

**Theorem**

Let $G^I_{\text{R min}}$ given with $l$ is “one-to-many” communication. Then we have: $w(G^I_{\text{R}}) = c(G^I_{\text{R}})$. 
Literature

Dissemination of Information in Optical Networks
From Technology to Algorithms
Questions

- Which problems are interesting for optical networks?
- For which is the Beneš Network used, what are it's properties?
- What is the relation between wavelength-assignment and colouring a graph?
- How is the wavelength-assignment solved on the following graphs?
  - paths and cycles.
  - stars and spider-graphs.
- On which graphs is the wavelength-assignment hard?
- May the wavelength-assignment be solved if the connection structure is of type broadcast?
Questions

- Which problems are interesting for optical networks?
- For which is the Beneš Network used, what are its properties?
- What is the relation between wavelength-assignment and colouring a graph?
- How is the wavelength-assignment solved on the following graphs?
  - paths and cycles.
  - stars and spider-graphs.
- On which graphs is the wavelength-assignment hard?
- May the wavelength-assignment be solved if the connection structure is of type broadcast?
Questions

- Which problems are interesting for optical networks?
- For which is the Beneš Network used, what are its properties?
- What is the relation between wavelength-assignment and colouring a graph?
- How is the wavelength-assignment solved on the following graphs?
  - paths and cycles.
  - stars and spider-graphs.
- On which graphs is the wavelength-assignment hard?
- May the wavelength-assignment be solved if the connection structure is of type broadcast?
Questions

- Which problems are interesting for optical networks?
- For which is the Beneš Network used, what are its properties?
- What is the relation between wavelength-assignment and colouring a graph?
- How is the wavelength-assignment solved on the following graphs?
  - paths and cycles.
  - stars and spider-graphs.
- On which graphs is the wavelength-assignment hard?
- May the wavelength-assignment be solved if the connection structure is of type broadcast?
Questions

- Which problems are interesting for optical networks?
- For which is the Beneš Network used, what are it's properties?
- What is the relation between wavelength-assignment and colouring a graph?
- How is the wavelength-assignment solved on the following graphs?
  - paths and cycles.
  - stars and spider-graphs.
- On which graphs is the wavelength-assignment hard?
- May the wavelength-assignment be solved if the connection structure is of type broadcast?
Questions

- Which problems are interesting for optical networks?
- For which is the Beneš Network used, what are it's properties?
- What is the relation between wavelength-assignment and colouring a graph?
- How is the wavelength-assignment solved on the following graphs?
  - paths and cycles.
  - stars and spider-graphs.
- On which graphs is the wavelength-assignment hard?
- May the wavelength-assignment be solved if the connection structure is of type broadcast?
Questions

- Which problems are interesting for optical networks?
- For which is the Beneš Network used, what are its properties?
- What is the relation between wavelength-assignment and colouring a graph?
- How is the wavelength-assignment solved on the following graphs?
  - paths and cycles.
  - stars and spider-graphs.
- On which graphs is the wavelength-assignment hard?
- May the wavelength-assignment be solved if the connection structure is of type broadcast?
Questions

- Which problems are interesting for optical networks?
- For which is the Beneš Network used, what are its properties?
- What is the relation between wavelength-assignment and colouring a graph?
- How is the wavelength-assignment solved on the following graphs?
  - paths and cycles.
  - stars and spider-graphs.
- On which graphs is the wavelength-assignment hard?
- May the wavelength-assignment be solved if the connection structure is of type broadcast?
Legend

- : Not of relevance
- : implicitly used basics
- : idea of proof or algorithm
- : structure of proof or algorithm
- : Full knowledge