Parallel Algorithms for Scheduling and Load Balancing

Ernst W. Mayr

Lehrstuhl für Effiziente Algorithmen

Technische Universität München

- models of parallel and distributed computation
- embedding problems
- parallel scheduling algorithms
- call scheduling
- load balancing
- conclusion and open problems

The Case for Parallel and Distributed Computation

It is possible because of

- the immense advances in VLSI,
- the much higher speed of communication links,
- much improved packaging technology.

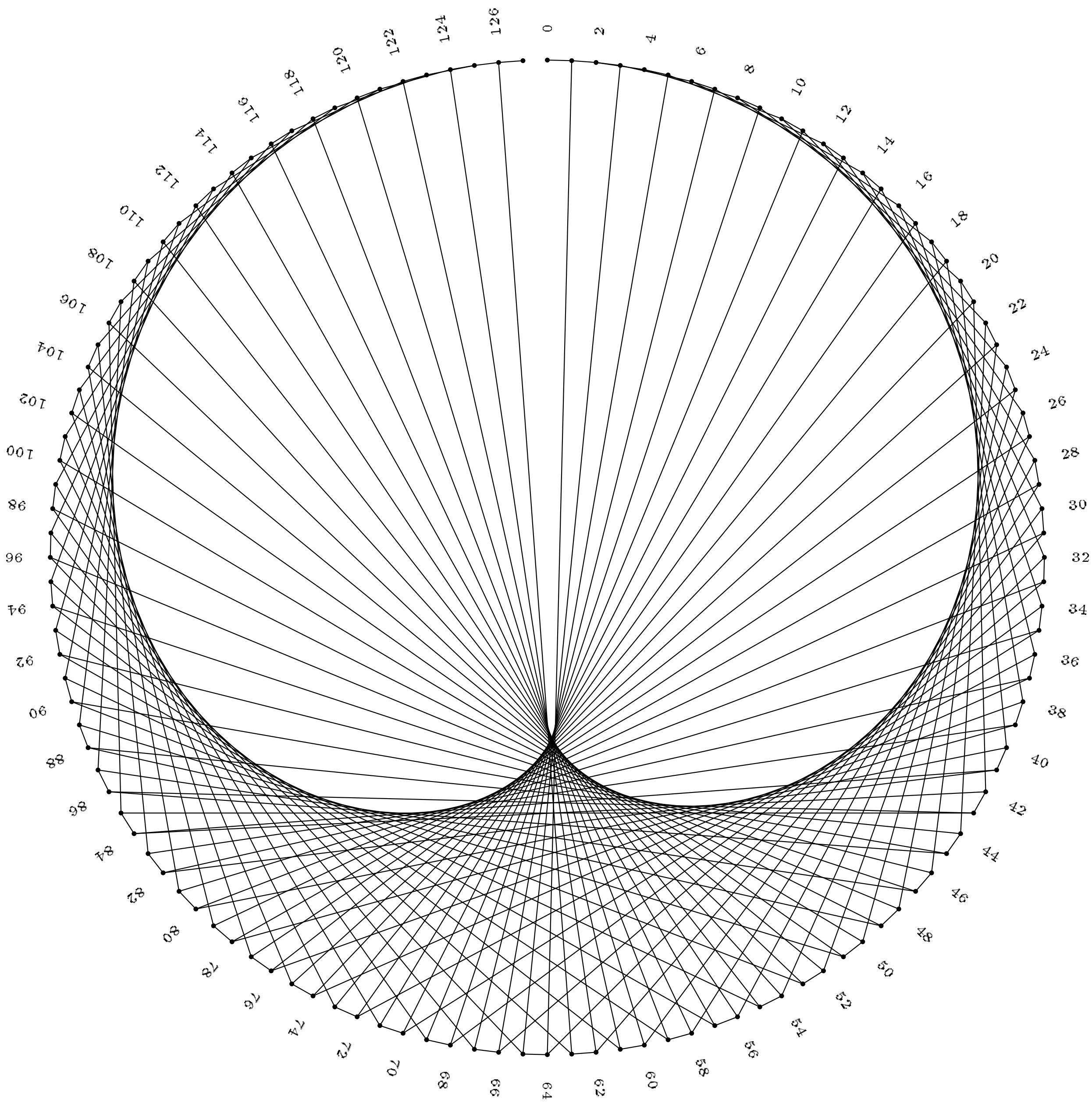
It is necessary because of

- the much increased need for computing power,
- the limits of the von Neumann architecture,
- the need for (globally) shared information resources.

Models for Parallel and Distributed Computation

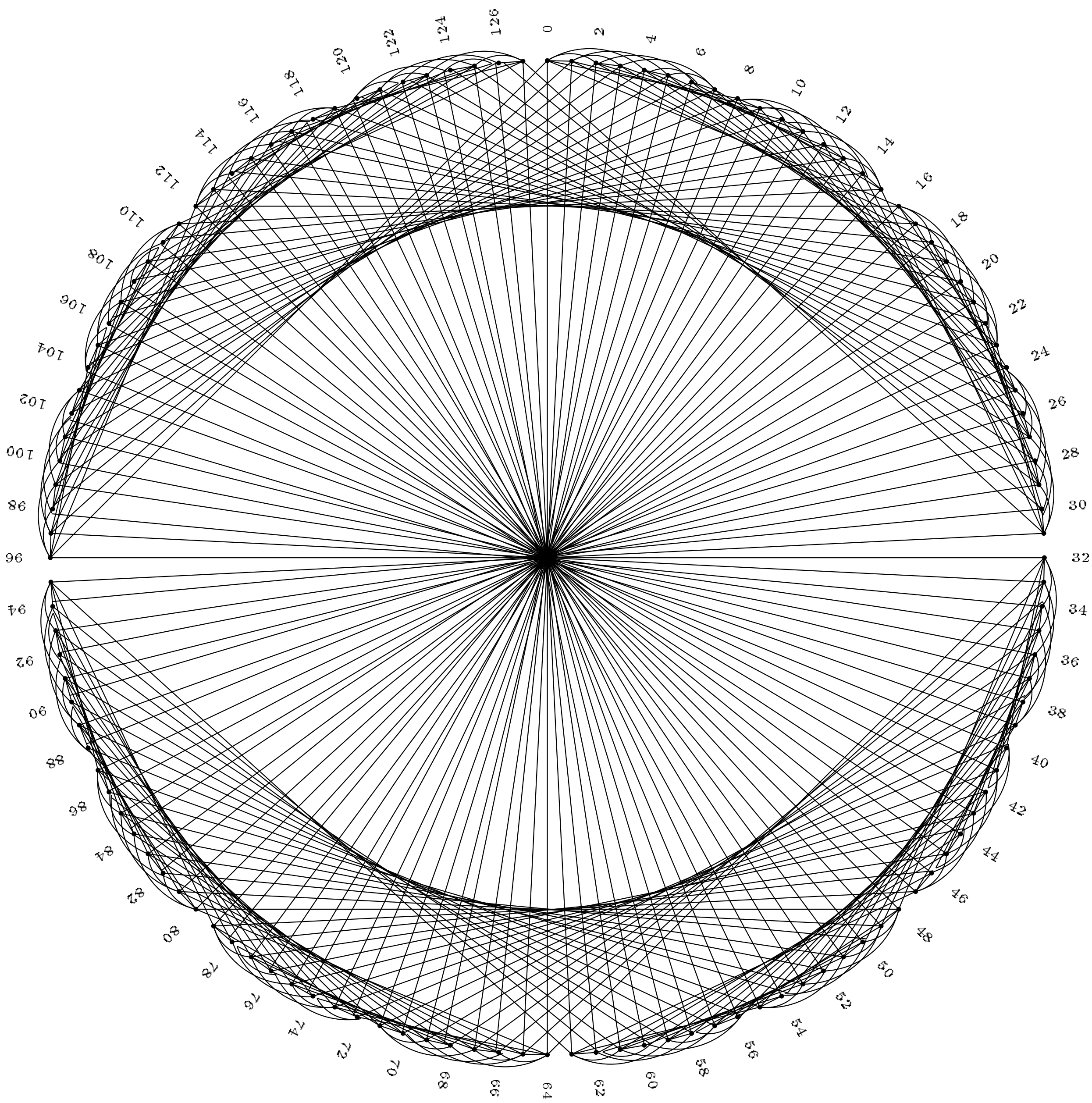
- PRAM: parallel random access
- DMM: distributed memory modules
- LogP, BSP: latency, bulk synchronous
- NOP: network of processors
- NOW: network of workstations
- WAN: wide area network
- **the** Net

Example for NOP



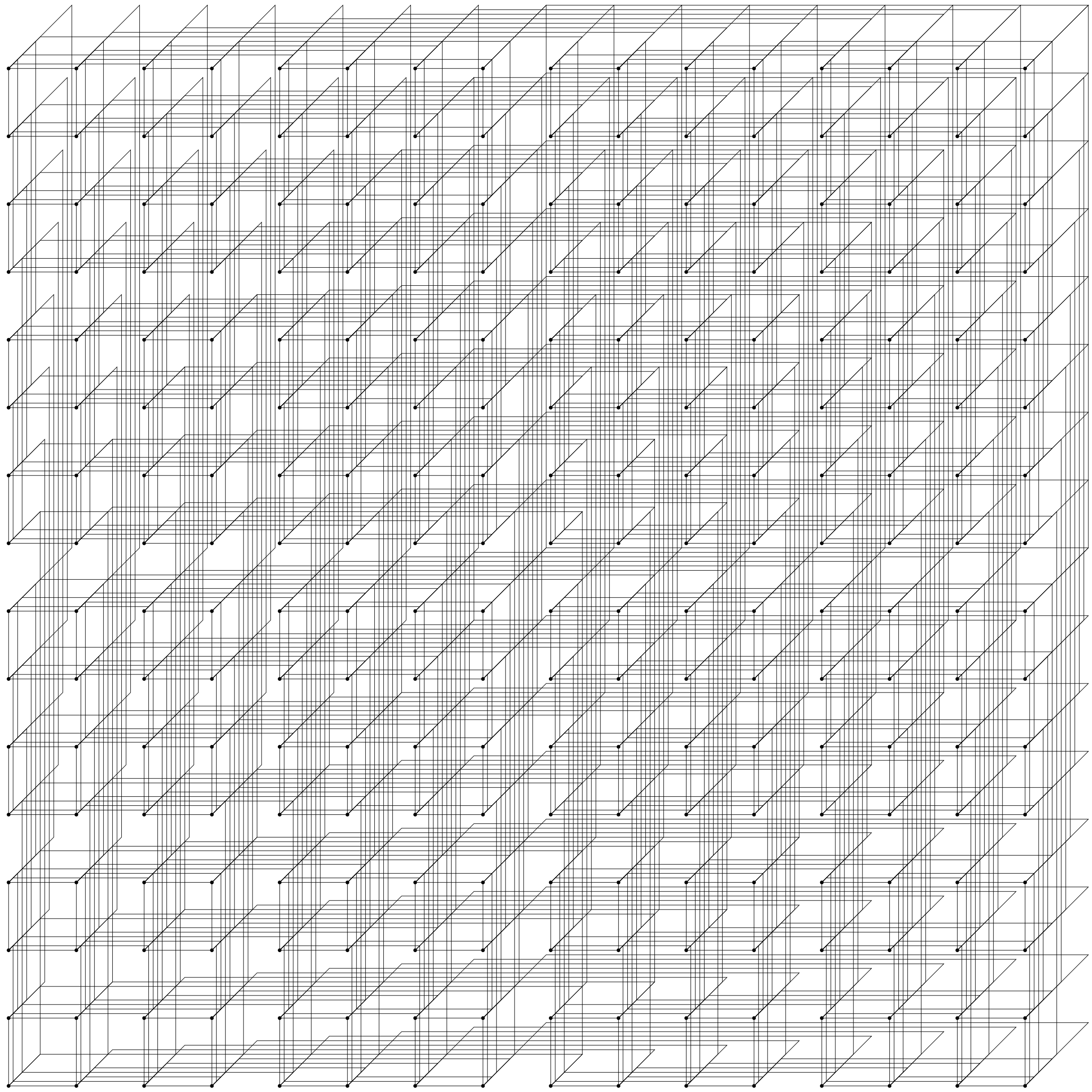
7-dimensional shuffle-exchange

Example for NOP



7-dimensional boolean hypercube

Example for NOP



8-dimensional boolean hypercube (in 2D)

Why is parallelism making such slow progress?

- no uniform model for parallel architectures
- no realistic parallel programming model
- the additional complexity/overhead of parallelism
- the need for scheduling and loadbalancing
- the need for extensive hardware support

Embeddings

(based on work with V. Heun)

An *embedding* is a pair of mappings:

$$(\varphi, \psi) : G = (V, E) \to H = (V', E')$$

where

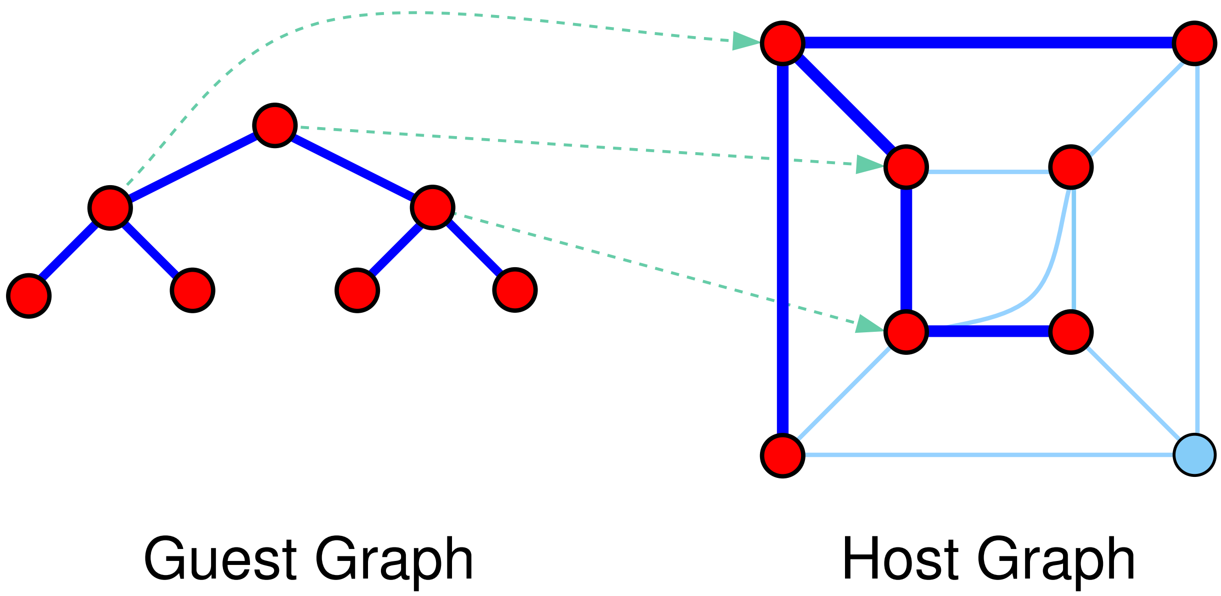
$$\varphi : V \to V'$$

$$\psi : E \to \mathcal{P}(E') \quad \text{(the set of paths in } H\text{)}$$

G is called the *guest graph* and H is called the *host graph*.

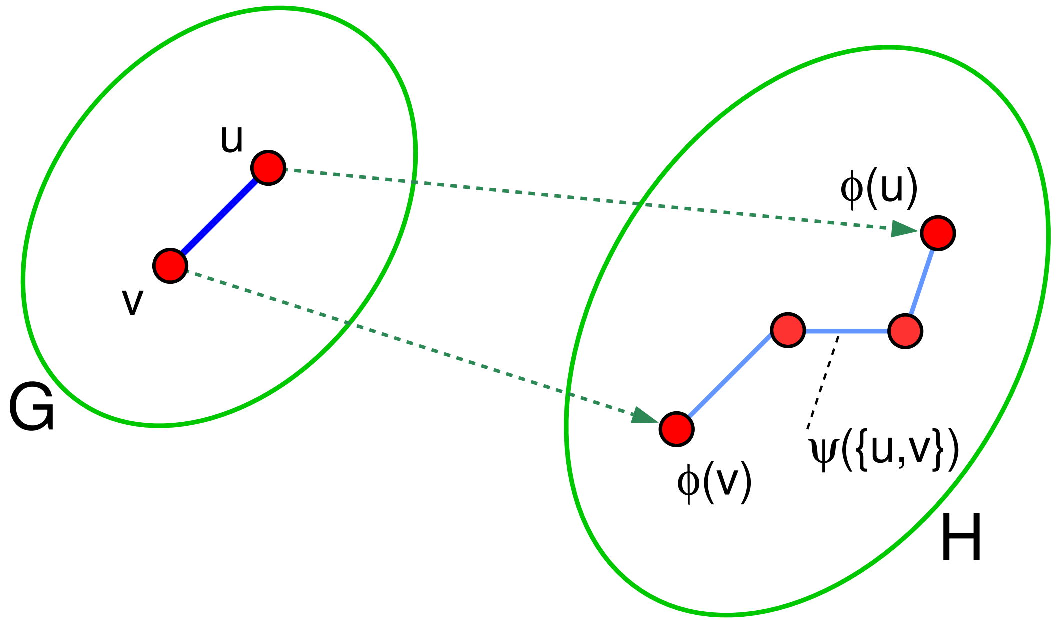
The ratio $\frac{|V'|}{|V|}$ is called the *expansion* of the embedding.

Example:

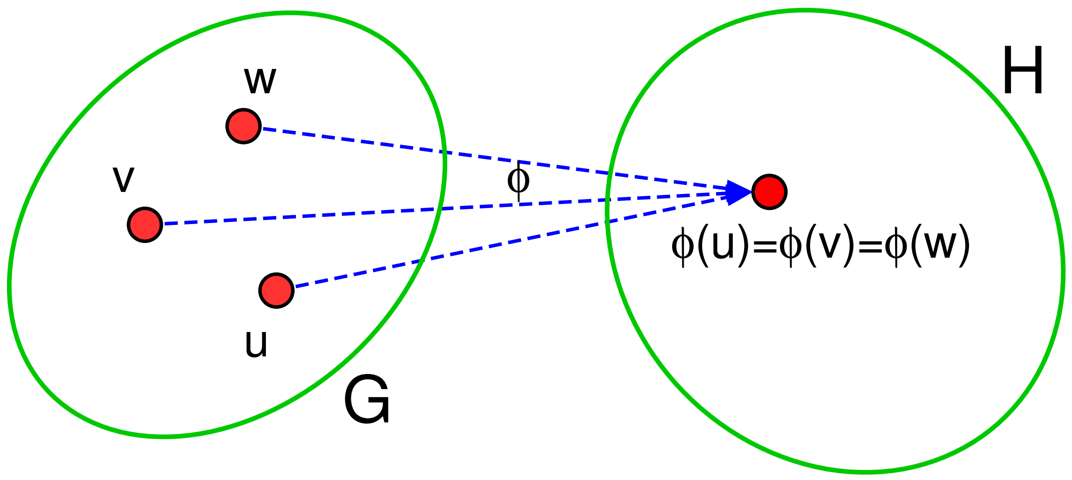


Dilation and Load of an Embedding

Dilation:

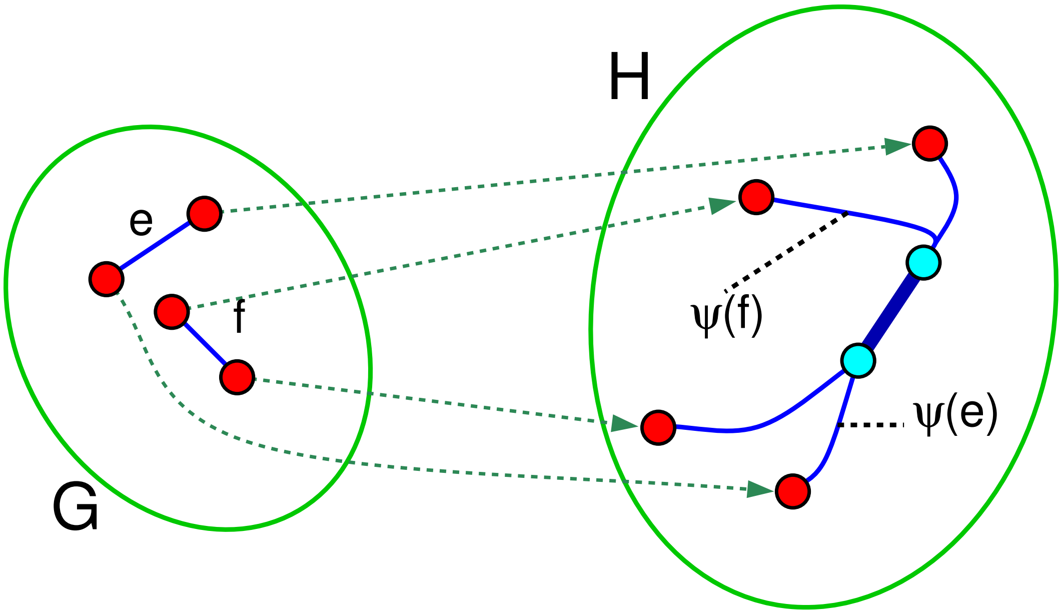


Load:

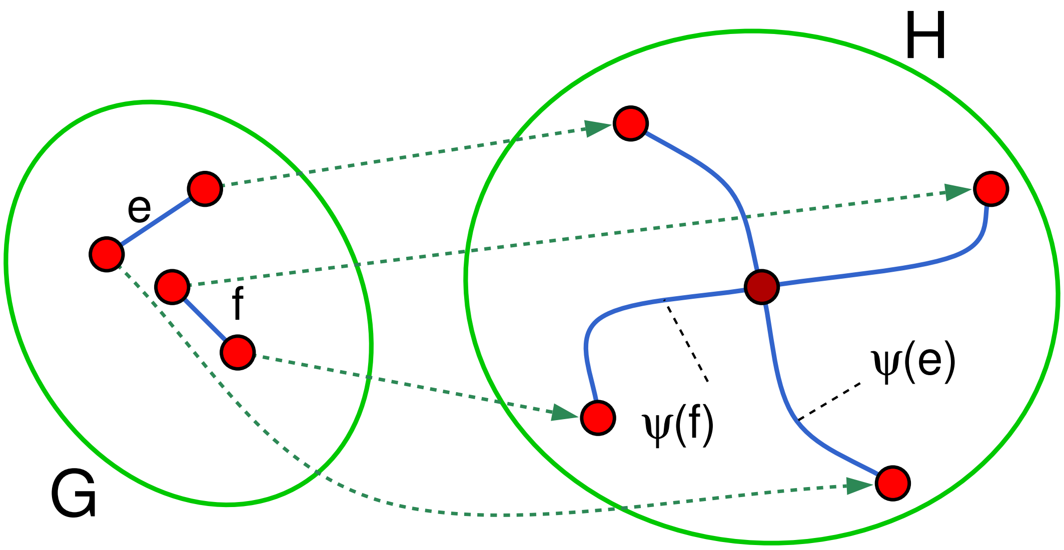


Congestion of an Embedding

Edge-Congestion:



Node-Congestion:

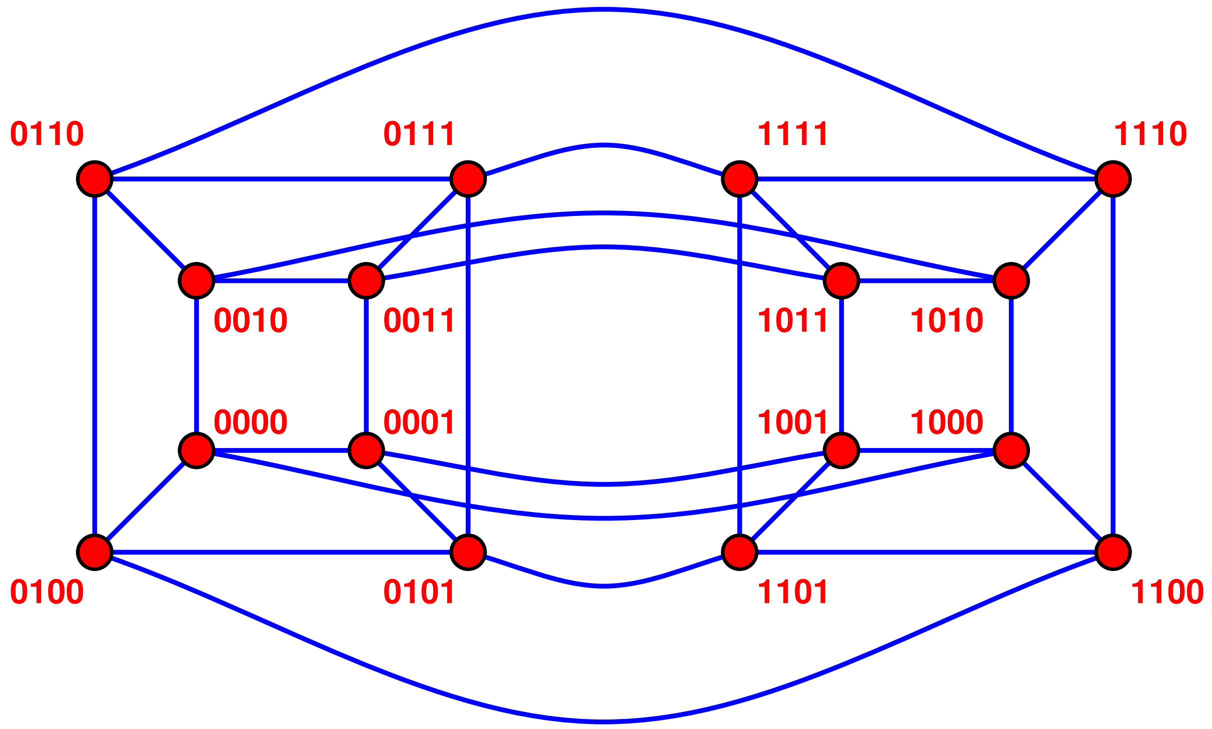


Hypercubes

A *d-dimensional hypercube* is the following graph:

$$H_d = \left(\{0,1\}^d, \{\{u,v\} : \delta(u,v) = 1\} \right).$$

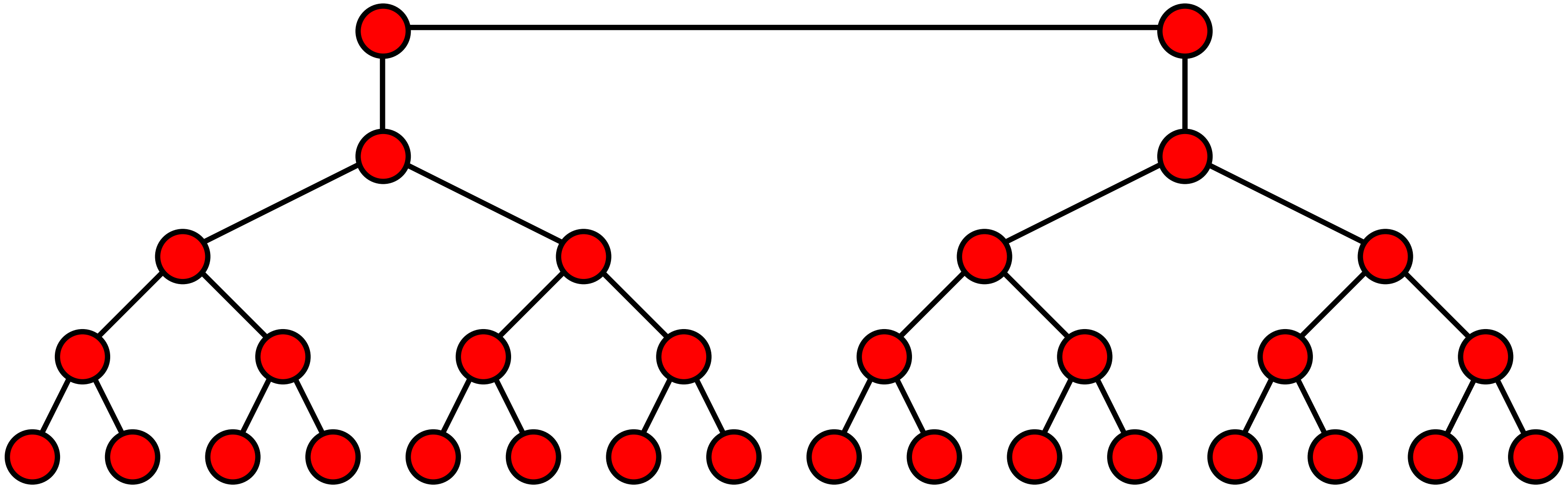
Example: a 4-dimensional hypercube



The smallest hypercube into which a given guest graph can be embedded 1-1 is called its *optimal* hypercube. The next larger hypercube is called its *next to optimal* hypercube.

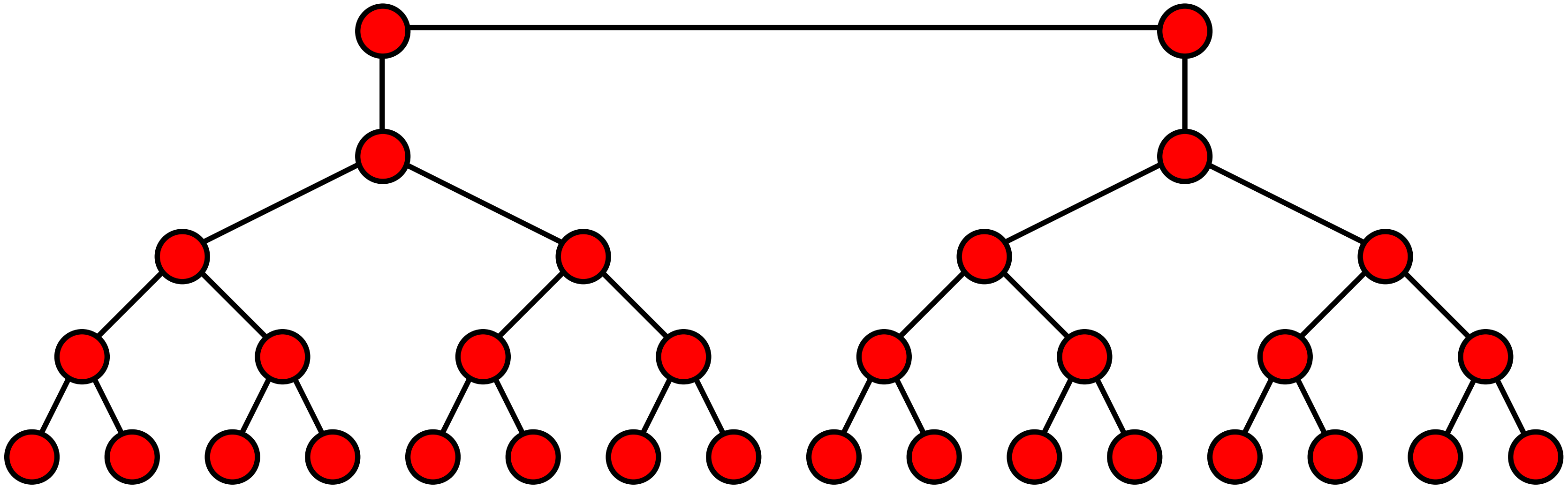
The Double-Rooted Complete Binary Tree (DRCBT)

A *double-rooted complete binary tree* of height h (or briefly a DRCBT) is a complete binary tree of height h where the root is replaced by an edge.



A DRCBT of height h consists of 2^h vertices.

The Double-Rooted Complete Binary Tree (DRCBT)



Basic Observations:

A complete binary tree of height h can be embedded with unit load and dilation 2 into a double-rooted complete binary tree of height h.

A complete binary tree of height $h-1$ is a subgraph of a double-rooted complete binary tree of height h.

Previous Work

Static embeddings of complete binary trees into hypercubes:

Dilation 2, Load 1, Expansion 1 *or* [Havel-Liebl/73], [Wu/85], [Bhatt-Ipsen/85]

Dilation 1, Load 1, Expansion 2 [Saad-Schultz/85], [Bier-Loe/89], [Leiss-Reddy/91]

Dilation 2, Load 2, Edge-Congestion 1 [Ravindran-Gibbons/93]

Dilation 1, Load 1, Expansion 2 [Chan-Lee/89], [Chan-Chin-Poon/91]

(tolerates $2d - \lceil \log(d) \rceil - 3$ *node or edge faults)*

Dilation 1, Load 1, Expansion 2 [Wang-Cypher-Mayr/90]

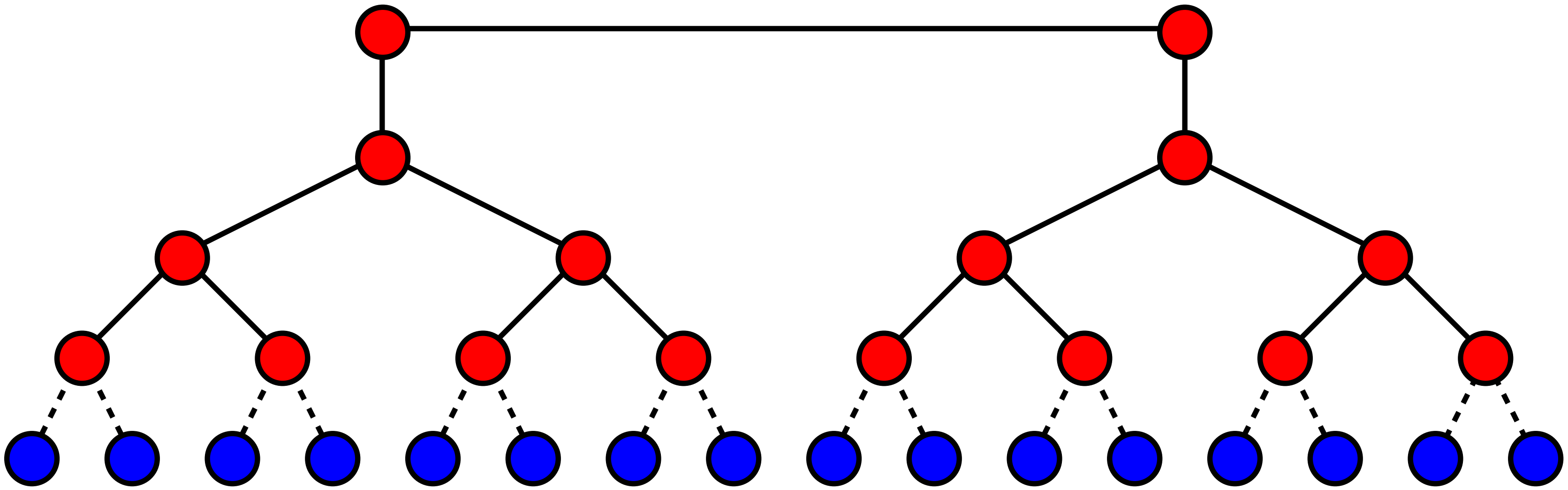
(tolerates $\approx \frac{d^2}{16 \log(d)}$ *node or edge faults)*

Dynamic embeddings of complete binary trees into hypercubes:

Dilation 2, Load 1, Expansion 1 [Feder-Mayr/87]

Spawning a New Level

In what follows, we assume that at each step the DRCBT grows by a complete level (of blue leaves).



Fundamental Property of the Embedding

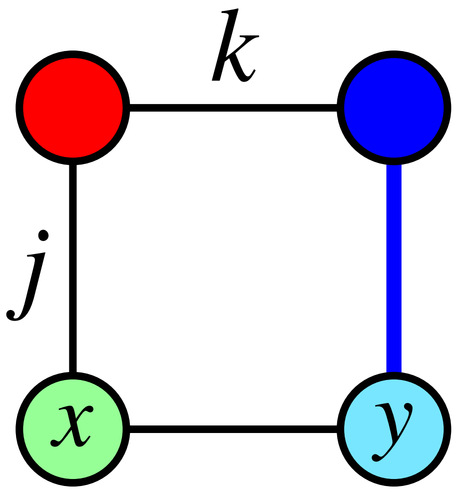
In what follows, we construct an embedding with the following property.

There exists a partition of the vertices of the DRCBT of height h into 2^{h-2} sets of four vertices such that the following three conditions hold:

1.) The four vertices are mapped to a 2-dimensional subcube

2.) each set contains two leaves
(the green and the lightblue vertex)
and two internal vertices *(the red and the blue vertex)*

3.) one internal vertex is the parent of a leaf in the set
and they are adjacent in the hypercube
(the blue vertex is the parent of the lightblue leaf),
and the two leaves are adjacent in the hypercube

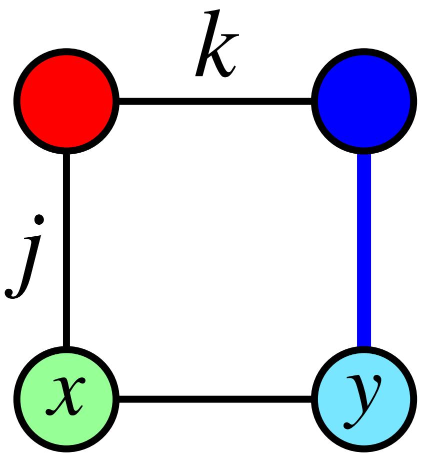


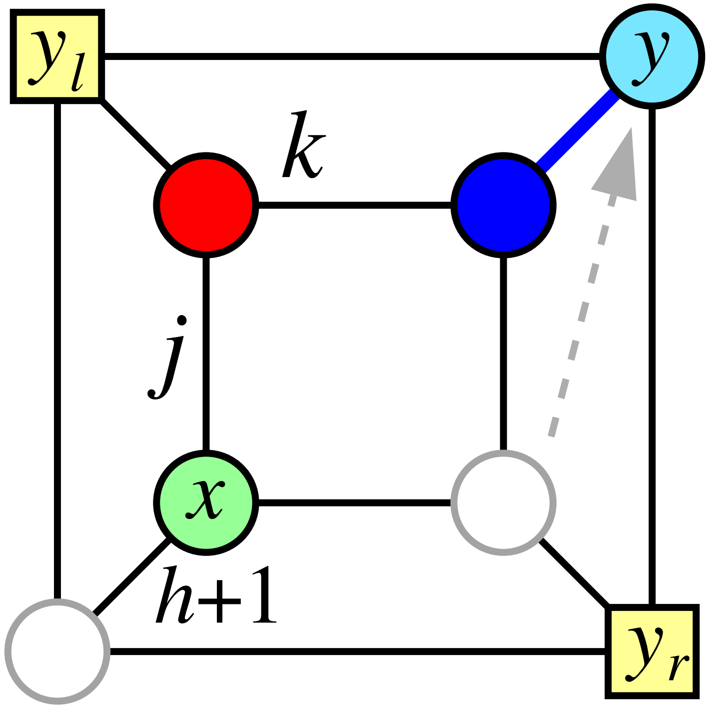
Obviously, this is true for a DRCBT of height 2.

A Pair of Leaves Spawning Children — Step 1

The leaves x and y spawn their children x_l, x_r and y_l, y_r respectively.

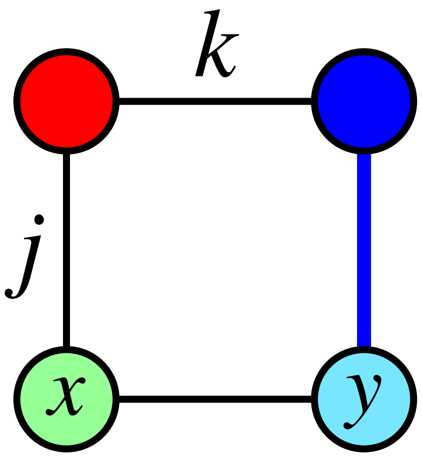
First let y spawn its children.

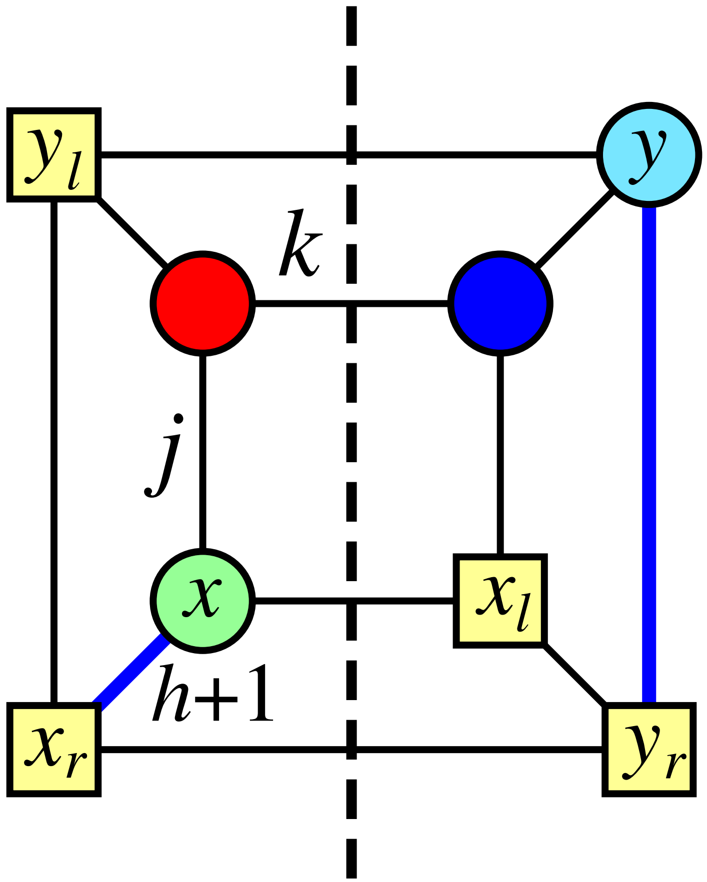




A Pair of Leaves Spawning Children — Step 2

The leaves x and y spawn their children x_l, x_r and y_l, y_r respectively.





Note that the left and right subcubes have the required property.

Computational View

Let $\mathrm{loc}_h(v)$ be the label of the vertex in an h-dimensional hypercube to which v is mapped.
Let $\sigma(x) = \sigma(y) = (j, k)$ be the pair of hypercube dimensions according to condition 1.)
We denote by x^i the label which differ in the i-th position from label x.

$$\mathrm{loc}_{h+1}(v) = \mathrm{loc}_h(v) \cdot 0 \qquad \text{for all internal vertices } v$$

$$\begin{aligned}
\mathrm{loc}_{h+1}(x) &= \mathrm{loc}_h(x) \cdot 0 & & \\
\mathrm{loc}_{h+1}(x_l) &= (\mathrm{loc}_h(x))^k \cdot 0 & \sigma(x_l) &= (j, h+1) \\
\mathrm{loc}_{h+1}(x_r) &= \mathrm{loc}_h(x) \cdot 1 & \sigma(x_r) &= (h+1, j)
\end{aligned}$$

$$\begin{aligned}
\mathrm{loc}_{h+1}(y) &= (\mathrm{loc}_h(y))^j \cdot 1 & & \\
\mathrm{loc}_{h+1}(y_l) &= \left((\mathrm{loc}_h(y))^j\right)^k \cdot 1 & \sigma(y_l) &= (h+1, j) \\
\mathrm{loc}_{h+1}(y_r) &= \mathrm{loc}_h(y) \cdot 1 & \sigma(y_r) &= (j, h+1)
\end{aligned}$$

Results (Part I)

Theorem: The double-rooted complete binary tree of height h can be dynamically embedded into its optimal hypercube with unit dilation, unit load, and unit congestion such that for each new level of leaves only one half of the old leaves have to be remapped. The computation time for each new level is constant.

Corollary: The complete binary tree of height h can be dynamically embedded into its optimal hypercube with dilation 2, unit load, and unit congestion such that for each new level of leaves only one half of the old leaves have to be remapped. Moreover, all tree edges but one have unit dilation. The computation time for each new level is constant.

Embedding the DRCBT Without Migration

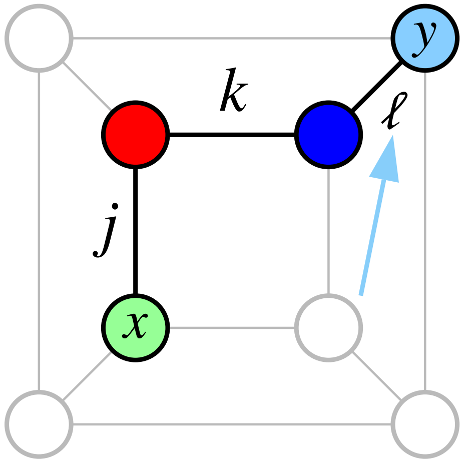
In what follows, we construct an embedding with the following property.

There exists a partition of the vertices of the DRCBT into 2^{h-2} sets of four vertices such that the following conditions hold:

1.) The four vertices are mapped to a 3-dimensional subcube

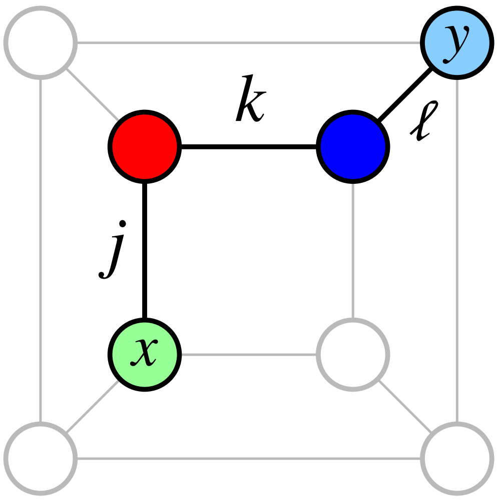
2.) each set contains two leaves
(the green and the lightblue vertex)
and two internal vertices *(the red and the blue vertex)*

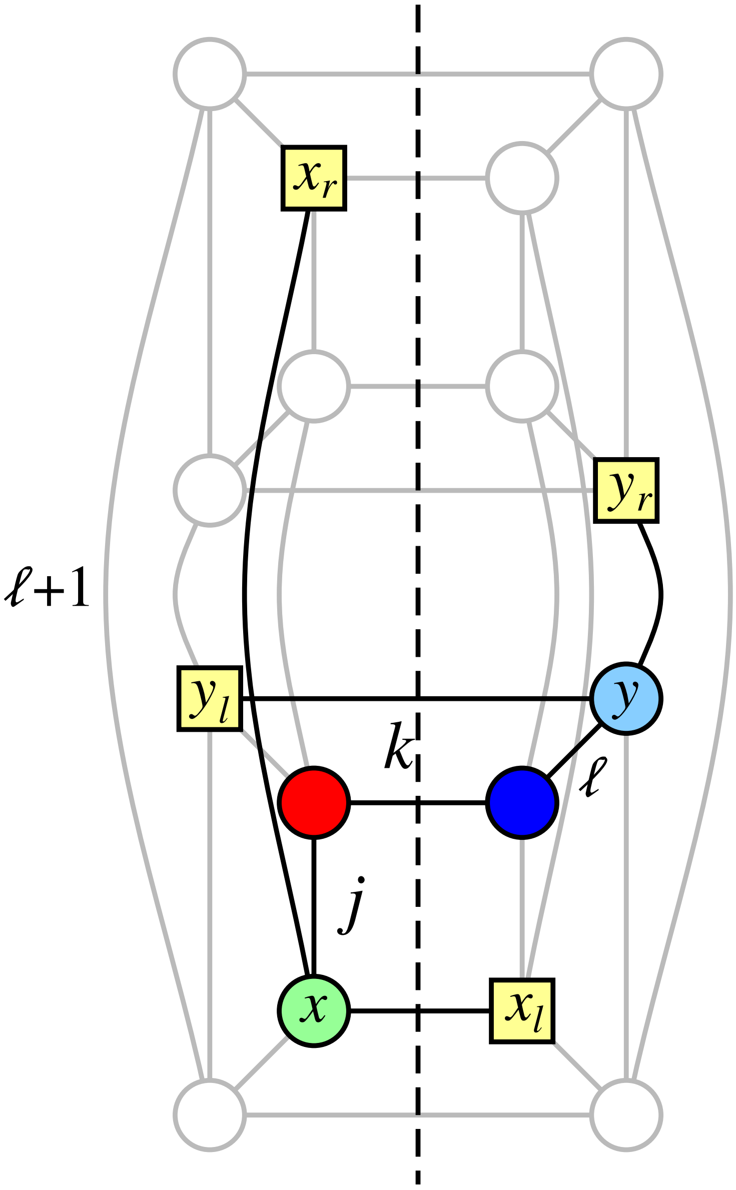
3.) one internal vertex is the parent of a leaf in the set
and they are adjacent in the hypercube
(the blue vertex is the parent of the lightblue leaf)



4.) In the hypercube, the four vertices are connected
by a path of length 3 which traverses three different dimensions (j, k, and ℓ).

The DRCBT Spawns a New Level into a Next to Optimal Hypercube





Note that the left and right subcubes have the required property.

More Results

Theorem: The double-rooted complete binary tree of height h can be embedded dynamically into its (next to) optimal hypercube with unit dilation, unit load, and unit congestion without remapping any vertex. The computation time for each new level is constant.

Corollary: The complete binary tree of height h can be embedded dynamically into its (next to) optimal hypercube with dilation 2, unit load, and unit congestion without remapping any vertex. Moreover, all tree edges but one have unit dilation. The computation time for each new level is constant.

Embedding Graphs with Small Treewidth

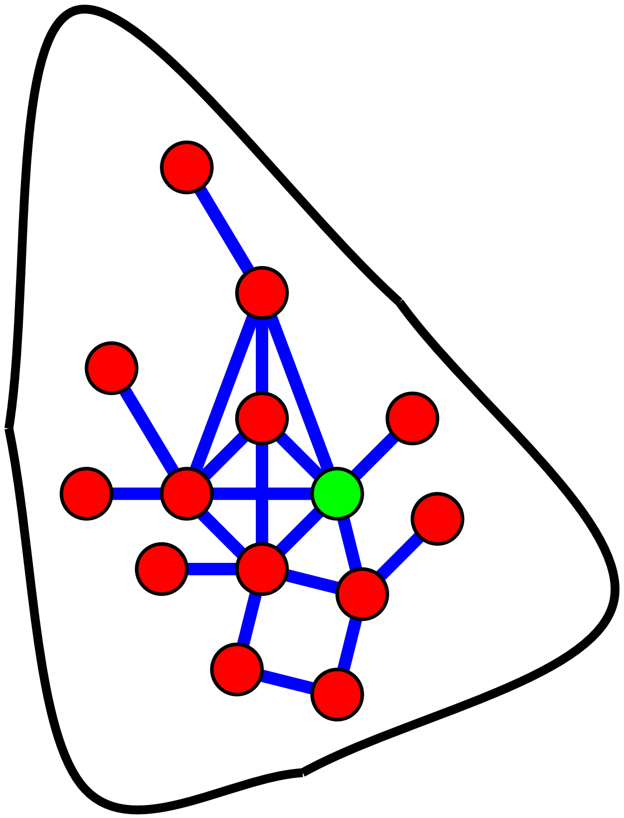
A *tree-decomposition* $\mathcal{D}_G$ of a graph $G = (V, E)$ is a pair (T, X) consisting of a tree $T = (S, F)$ and a set $X = \{X_s \subseteq V : s \in S\}$ such that the following conditions are satisfied:

1. $\bigcup_{s \in S} X_s = V$,
2. $\forall (v, w) \in E : \exists s \in S : v, w \in X_s$,
3. $\forall v \in V : T[\{s : v \in X_s\}]$ is connected.

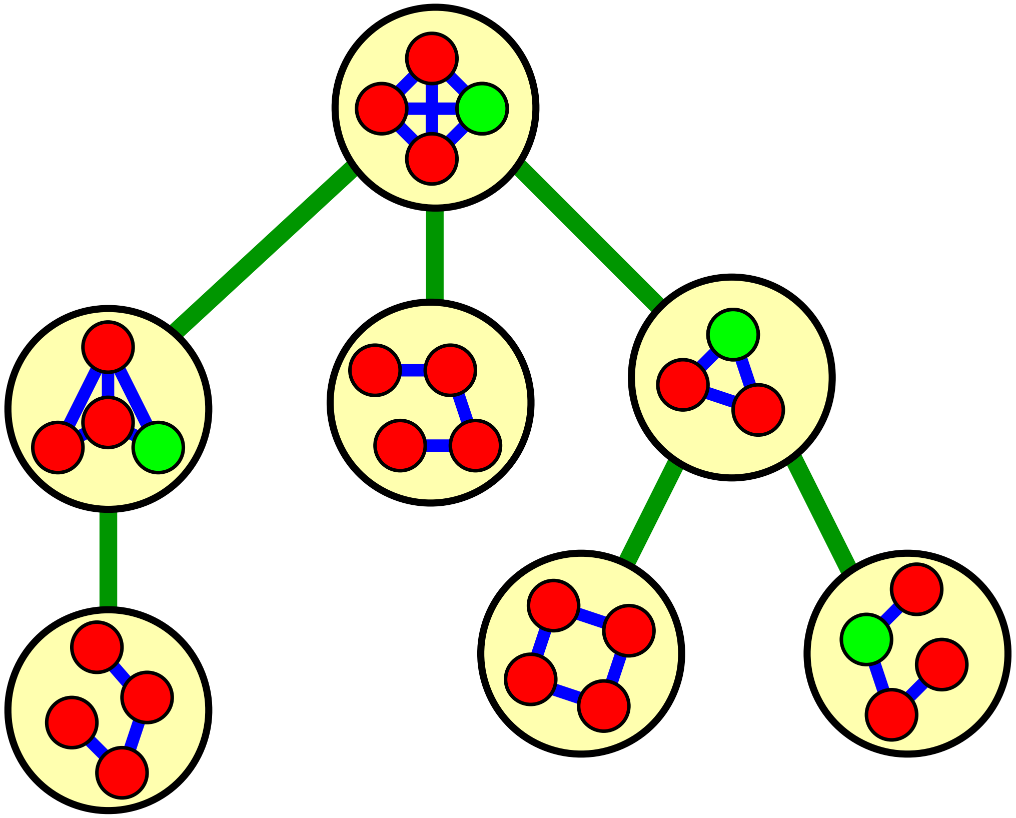
The *width* of a tree-decomposition $\mathcal{D}_G$ is $\max\{|X_s| : s \in S\} - 1$.

The *treewidth* of a graph G is the minimum width over all tree-decompositions of G.

A Tree-Decomposition of a Graph

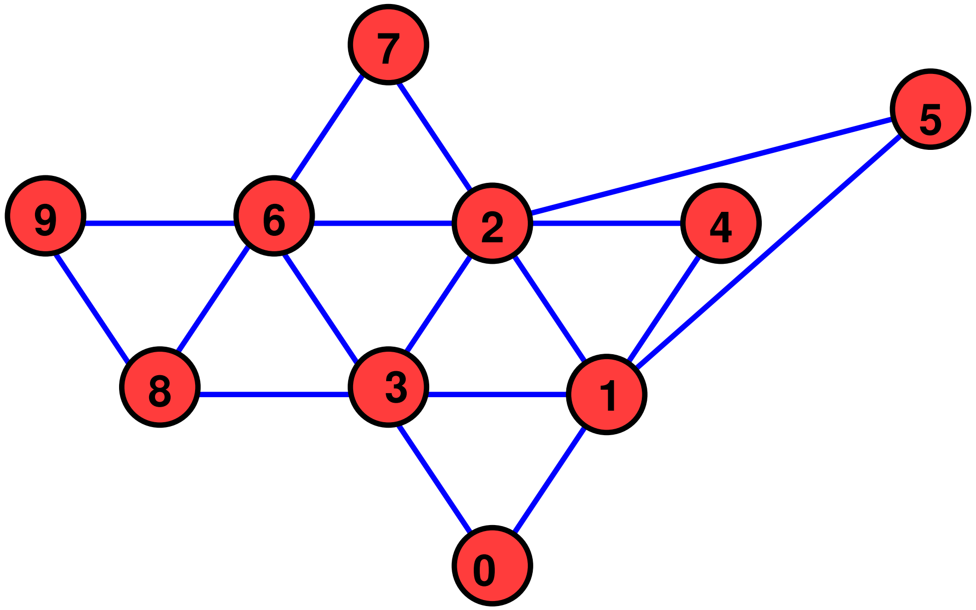


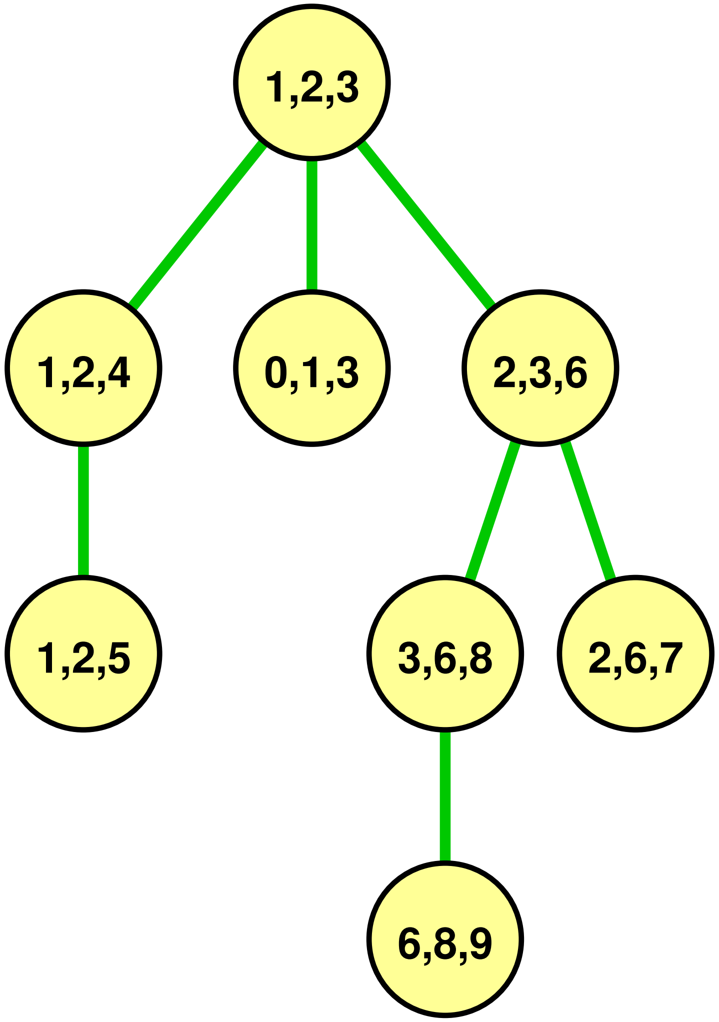
Graph G



Tree-decomposition for G

Example: A Graph and Its Decomposition Tree

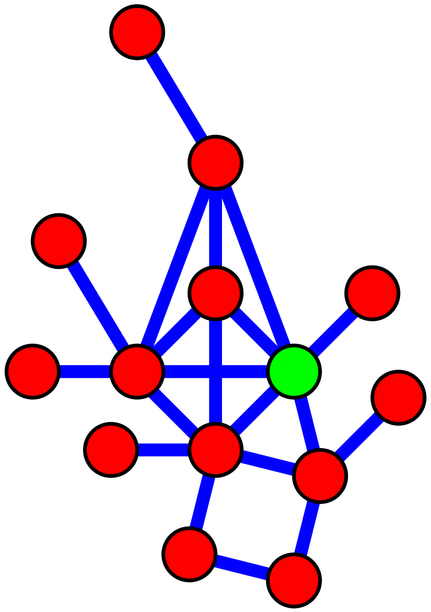


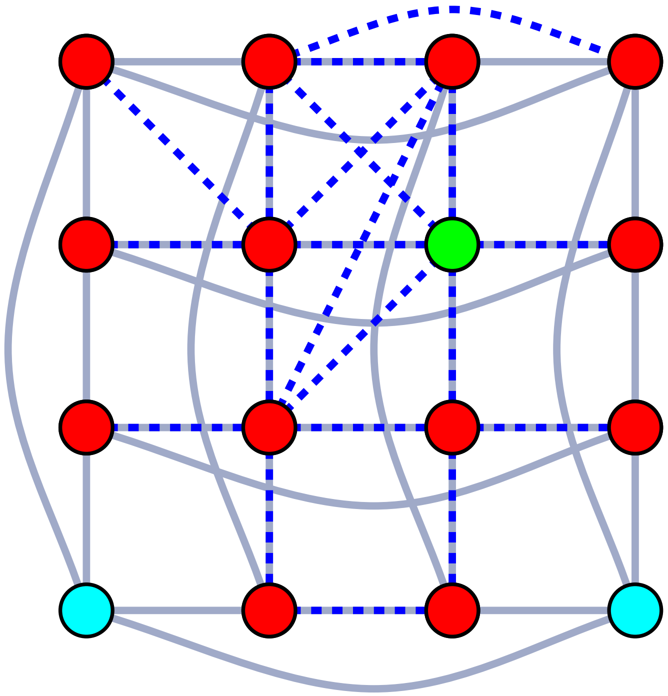


Main Theorem

Theorem: Let G be a graph with treewidth t, maximal degree d, and size n.

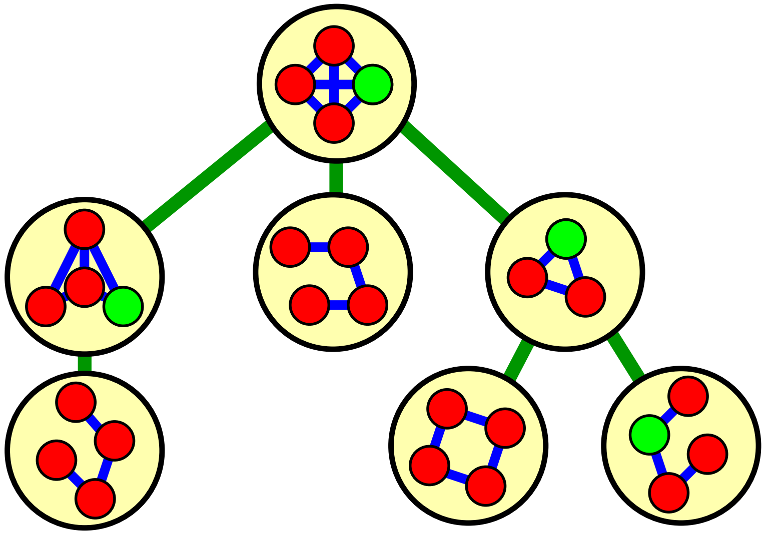
There exists a one-to-one embedding of G into its optimal hypercube with dilation $3\lceil \log((d+1)(t+1)) \rceil + 8$ and node-congestion $O((d+t)^4)$.





Main Theorem (cont.)

Provided that the graph is given by a tree-decomposition of size $O(n)$, this embedding can be computed on the optimal hypercube itself in time



$$O\left(\max\left\{T_{\mathrm{LR}}(n), (d+t^2)\log^2(n)\right\}\right)$$

$$= \begin{cases} O(\log^2(n)\,\mathrm{logloglog}(n)\log^*(n)) & \text{if } d+t^2 = O(\mathrm{logloglog}(n)\log^*(n)) \\ O((d+t^2)\log^2(n)\}) & \text{if } d+t^2 = \Omega(\mathrm{logloglog}(n)\log^*(n)) \end{cases}$$

Related Work

Embedding regular structures into (optimal) hypercubes:

2-dimensional Grids: Dilation 2 [Chan/89]

3-dimensional Grids: Dilation 5 [Chan-Chin-Chu-Mak/93]

d-dimensional Grids: Dilation $O(d)$ [Chan/91]

(Double-Rooted) Complete Binary Trees: Dilation 2 (1) [Havel-Liebl/73,. . .]

Complete k-ary trees: Dilation $\Theta(\log(k))$, expansion $\omega(1)$ [Shen-Hu-Liang/95]

X-Trees: Dilation 2 [Stout/86]

Pyramids: Dilation 2 [Stout/86]

Embedding of arbitrary binary trees into optimal hypercubes:

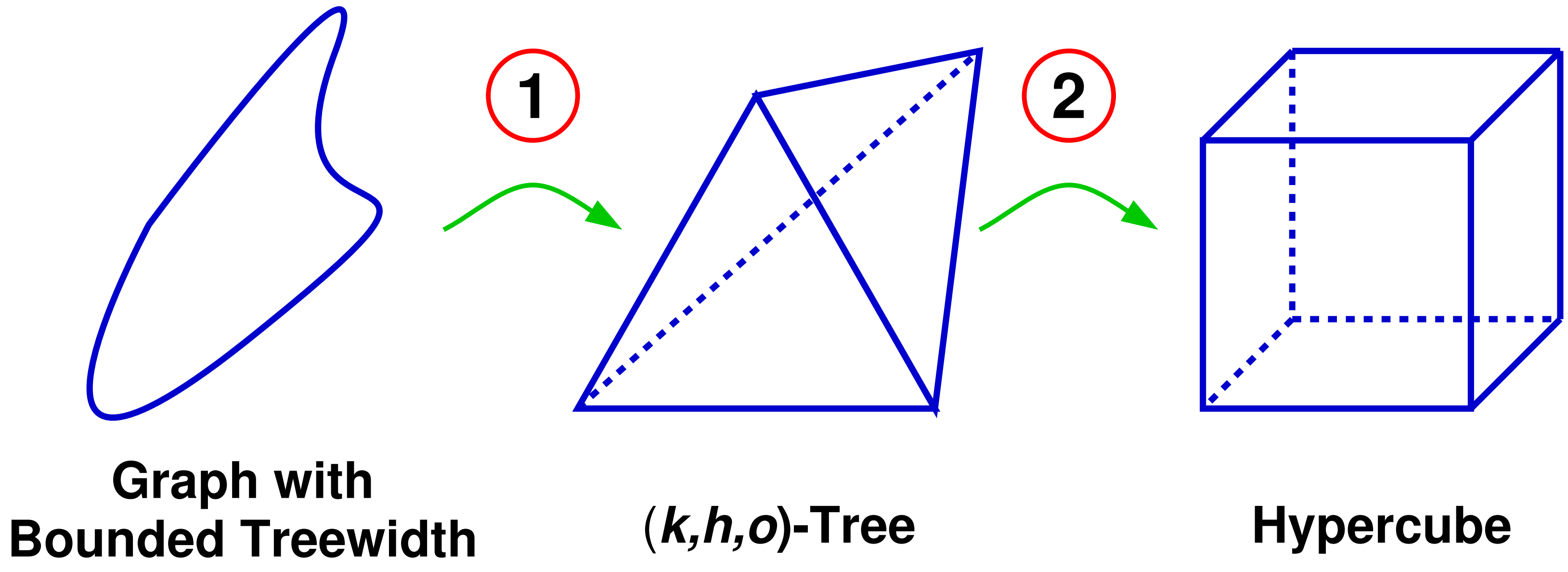
One-legged caterpillars are subgraphs [Havel-Liebl/86]

Special types of caterpillars are subgraphs [Bezrukov-Monien-Unger-Wechsung/93]

Dilation $O(1)$, node-congestion $O(1)$ [Bhatt-Chung-Leighton-Rosenberg/92]

Dilation 8, node-congestion $O(1)$ [Heun-Mayr/93]

Outline of the Embedding

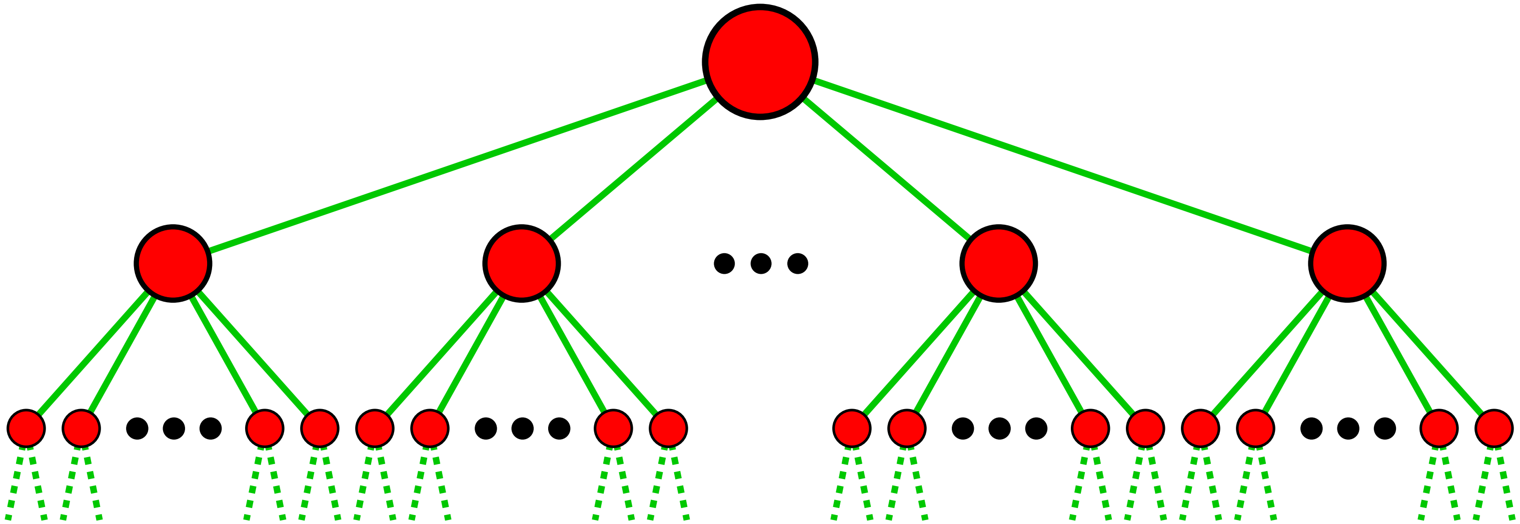


(k,h,o)-Tree

A *(k, h, o)-tree* is a complete 2^k-ary tree of height h with integer node weights, called the *capacities*:

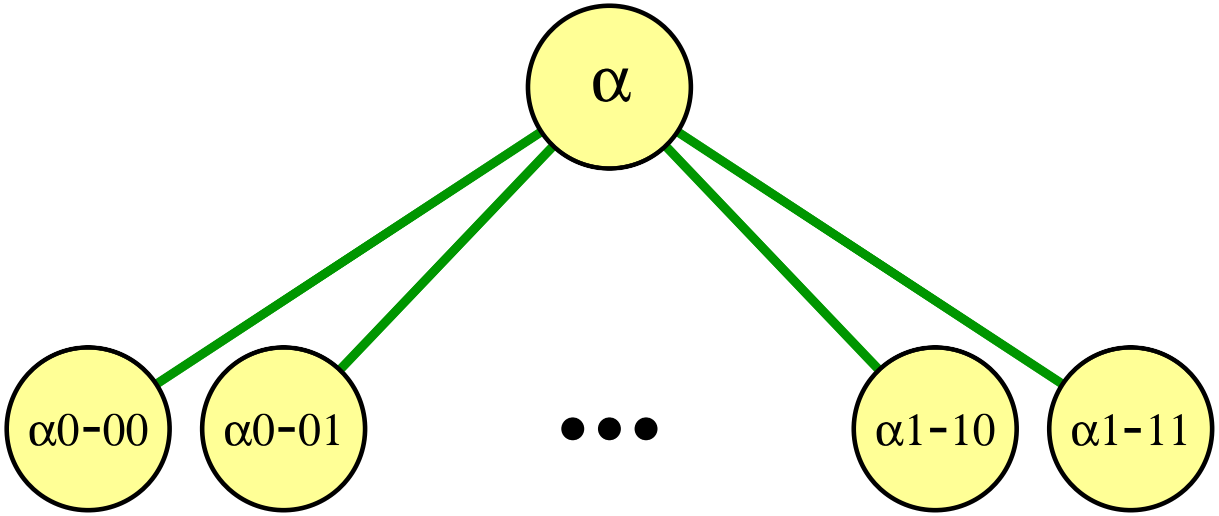
- the root has a capacity of $2^o(2^k(kh-k+1))$,
- a node at level ℓ has a capacity of $2^o((2^k-1)(k(h-\ell+1)-k+1)-k)$.

The total capacity of a (k, h, o)-tree is 2^{kh+o}.



The Second Embedding Step

Labeling of a (k,h,o)-Tree:

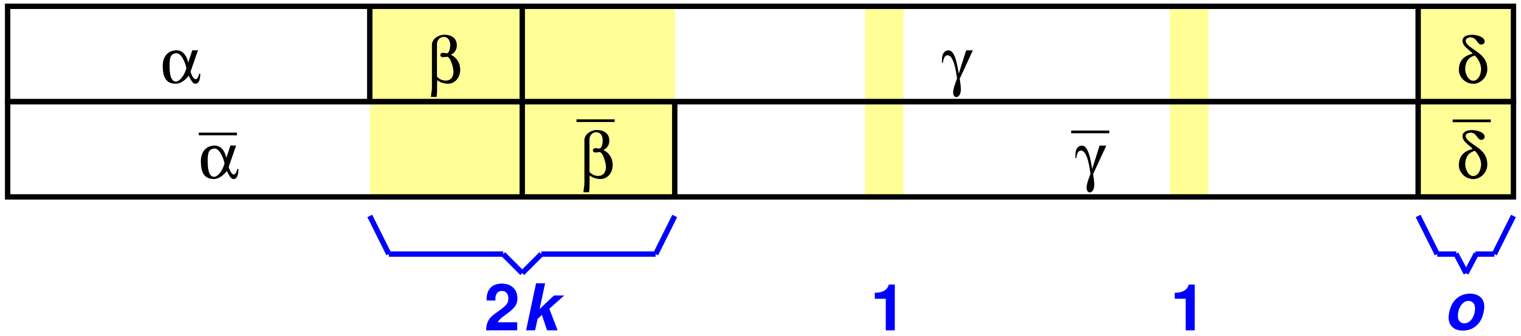


To a node labeled α, the following set S_α of hypercube locations is associated:

$$S_\alpha \quad := \quad \{\alpha\beta\gamma\delta \in \{0,1\}^{kh+o} : \beta{\in}(0{+}1)^*1 \wedge |\beta|{\leq}k \wedge \gamma{\in}0^*10^* \wedge \delta{\in}\{0,1\}^o\}$$

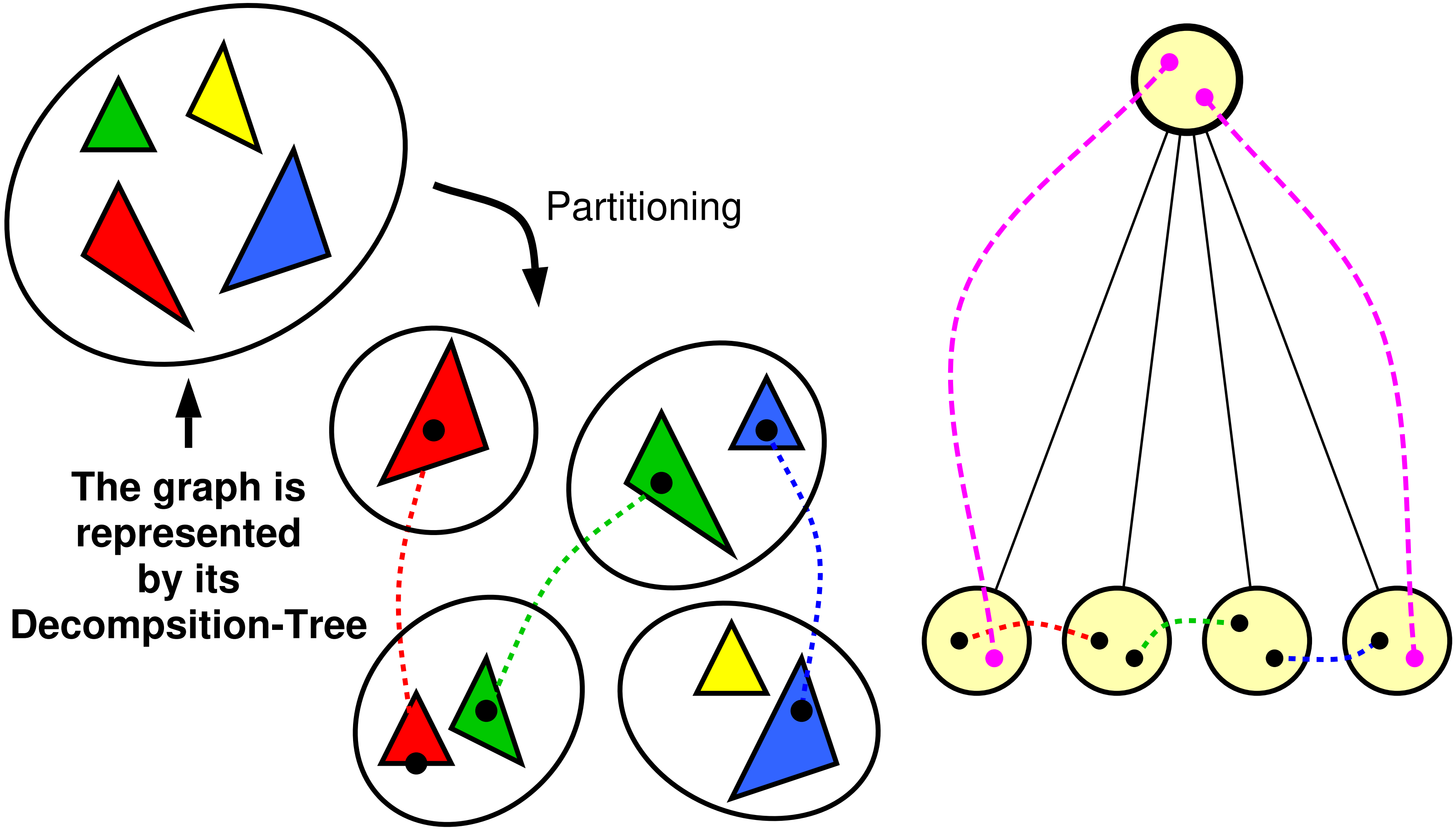
$$T \quad := \quad \{\gamma\delta \in \{0,1\}^{kh+o} : \gamma{\in}(0^*10^*{+}0^*) \wedge \delta{\in}\{0,1\}^o\}$$

Distance in the Hypercube



A pair of adjacent graph vertices which are close in the (k, h, o)-tree (i.e., they are contained in the same node, in two siblings, or in a parent and in its child) are mapped to hypercube locations with Hamming distance at most $2k + o + 2 = O(k)$.

Embedding a Graph into the (k,h,o)-Tree



Results

Theorem: Let G be a graph with treewidth t, maximal degree d, and size n.

There exists a one-to-one embedding of G into its optimal hypercube with dilation $3\lceil\log((d+1)(t+1))\rceil + 8$ and node-congestion $O((d+t)^4)$.

Provided that the graph is given by a tree-decomposition of size $O(n)$, this embedding can be computed on the optimal hypercube itself in time

$$O\left(\max\left\{T_{\mathrm{LR}}(n), (d+t^2)\log^2(n)\right\}\right) = \begin{cases} O(\log^2(n)\operatorname{logloglog}(n)\log^*(n)) & \text{if } d+t^2 = O(\operatorname{logloglog}(n)\log^*(n)) \\ O((d+t^2)\log^2(n)) & \text{if } d+t^2 = \Omega(\operatorname{logloglog}(n)\log^*(n)) \end{cases}$$

Note: Using [Bodlaender-Hagerup/95], a minimal tree-decomposition for a graph with constant treewidth can be computed on the hypercube in time $O(\log^3(n)\operatorname{loglog}^2(n))$.

Call Scheduling in Fast Networks

(based on work by Th. Erlebach and K. Jansen

- ATM Networks (ATM = *Asynchronous Transfer Mode*)
  - high bandwidth
  - guaranteed quality of service
  - bandwidth reservation schemes
  - *virtual* connections (calls) share physical links
- ATM can improve communication performance in workstation clusters and make them suitable for parallel computation.
- Problem: Given a set of connection requests, how do we schedule these calls in order to minimize the completion time?

An instance of the call-scheduling problem is given by:

(a) a graph $G = (V, E)$ with edge capacities $c : E \to \mathbb{N}$

(unit bandwidth: $\forall e \in E : c(e) = 1$)

- undirected graph (bidirectional calls, bidirectional links)
- symmetric directed graph (directed calls, bidirectional links)
- directed graph (directed calls, directed links)

(b) a set R of call requests $r = (u, v, b, d)$, where

- u is the sender
- v is the receiver
- b is the requested bandwidth (unit bandwidth: $b = 1$)
- d is the duration of the call (unit duration: $d = 1$)

A feasible schedule assigns to each request $r = (u, v, b, d)$:

(1) a (directed) path P_r from u to v in G

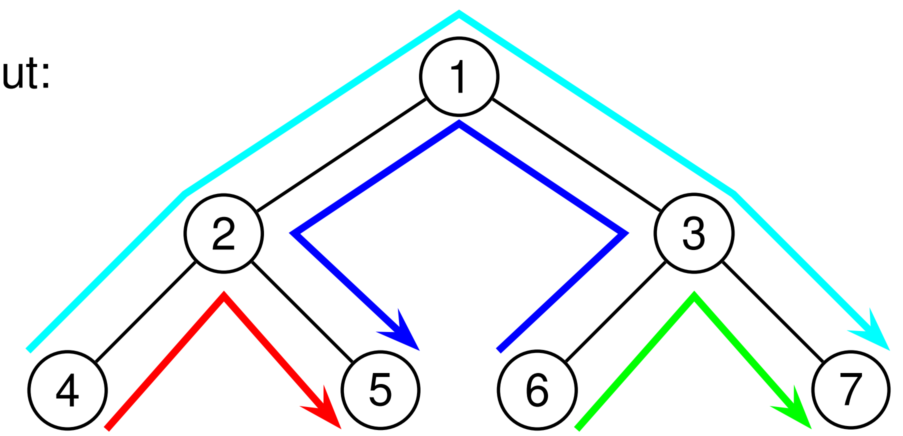
(2) a starting time t_r (preemption is not allowed!)

satisfying:

The sum of bandwidths of simultaneously active calls using an edge e must not exceed $c(e)$.

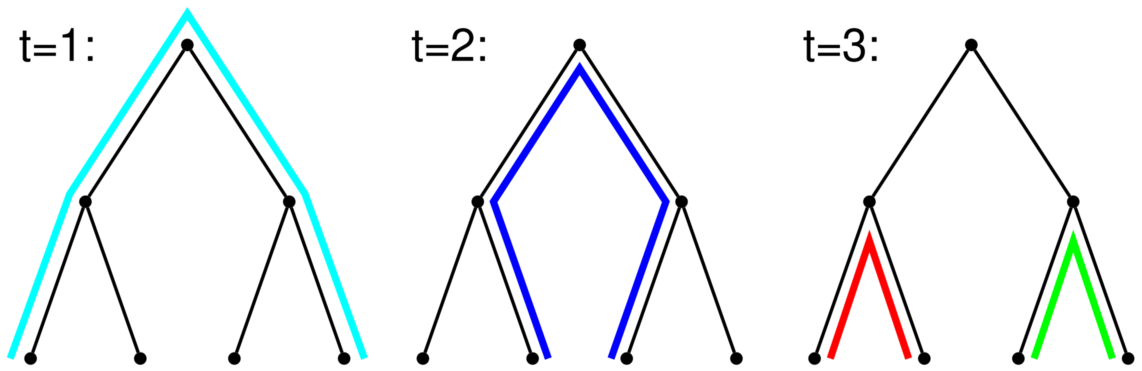
Example $(b = 1, d = 1)$

Input:

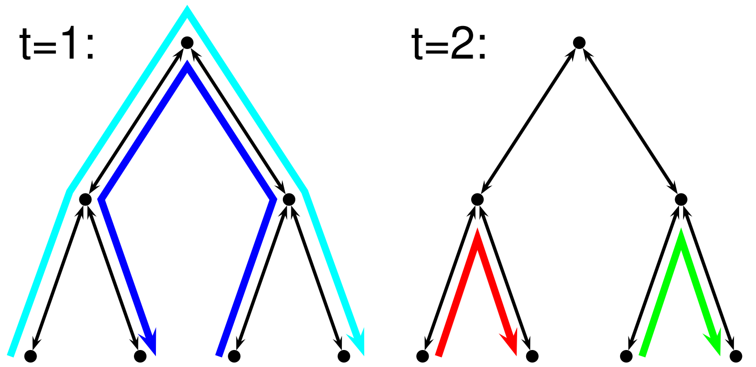


R={(4,7),(4,5),(6,7),(6,5)}

Schedule for Bidirectional Calls:



Schedule for Directed Calls:



Summary: Unit Bandwidth and Unit Duration

	bidirectional	directed
linear arrays	polynomial	polynomial
binary trees	polynomial	NP–hard $\frac{5}{3}$–approximation
trees	NP–hard 1.1–approximation	NP–hard $\frac{5}{3}$–approximation
rings	NP–hard 2–approximation	NP–hard 2–approximation

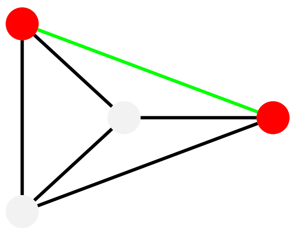
Scheduling Directed Calls in Tree Networks

Theorem: [Erlebach, Jansen, Kaklamanis, Persiano 1997] There is a polynomial time $\frac{5}{3}$–approximation algorithm for scheduling directed calls in tree networks.

Theorem: The scheduling problem for directed calls in tree networks is NP–hard.

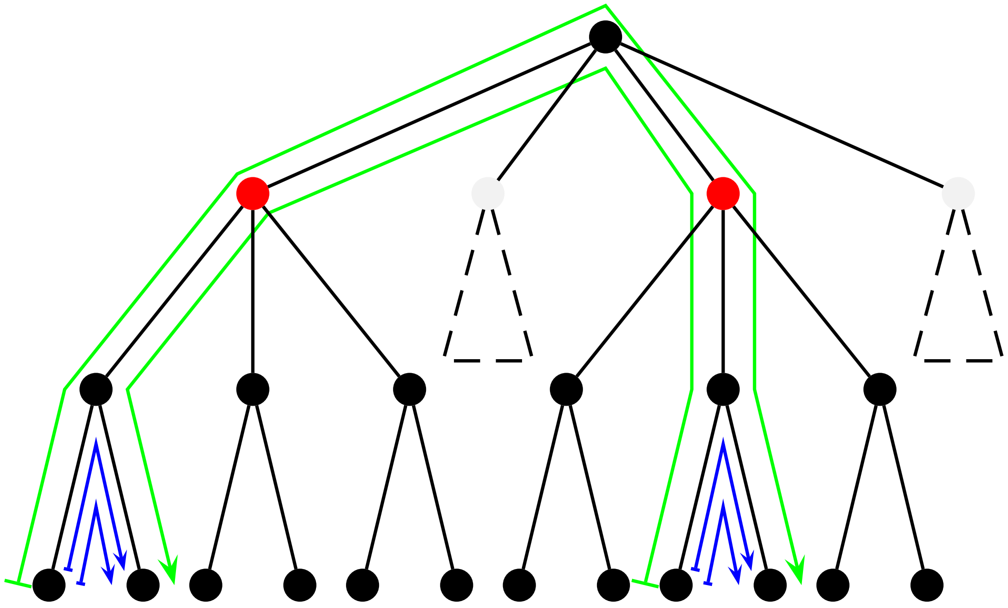
Proof: by reduction from EDGE-COLORING

3–regular graph



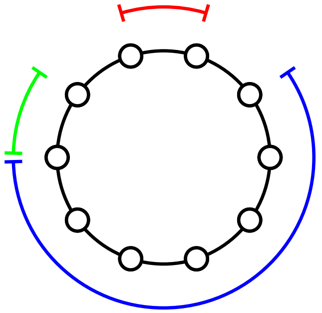
chromatic index:
3 or 4 ?

$\Rightarrow$



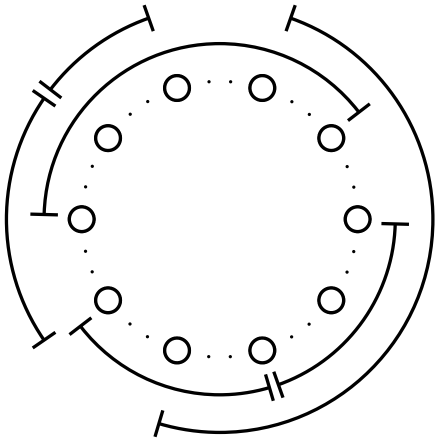
minimum schedule length: 3 or 4 ?

Scheduling Bidirectional Calls in Ring Networks



- Each call can be routed clockwise or counterclockwise.
- If the routing is determined by ignoring some arbitrary link of the ring, the resulting schedule is a 2–approximation.

Coloring circular arcs: A related problem



ARC-COLORING: Given a family F of circular arcs and a positive integer k, can F be colored with k colors?

Theorem: ARC-COLORING is NP–complete.
(in [Garey, Johnson, Miller, Papadimitriou 1980])

Theorem: Scheduling bidirectional calls in ring networks optimally is NP–hard.

Proof: by reduction from ARC-COLORING

Let a family F of circular arcs and a positive integer k be given.

(1) extend F so that every part of the circle is spanned by exactly k arcs

(2) select two intervals of the circle lying opposite each other, and split each arc overlapping one of these intervals into three separate arcs which must be assigned the same color in any k–coloring of F

(3) interpret each arc as a call between its endpoints

Scheduling Directed Calls in Ring Networks

Optimization Problem:

- can be shown NP–hard by a reduction from ARC-COLORING
  (similar to the bidirectional case, with each arc being viewed as two directed calls in opposite direction between its endpoints)

Approximation Algorithm:

- ignore some arbitrary link of the ring $\Rightarrow$ 2–approximation

Conclusion and Open Problems

- more dynamic embeddings
- more efficient algorithms for embeddings
- algorithms for on-line call-scheduling
- schedules for special communication patterns
- efficient on-line scheduling algorithms
- (work)efficient $\mathcal{NC}$ scheduling algorithms
- *intelligent* load balancing algorithms