Non-blocking networks based on expander graphs

Abhiram Ranade
IIT Bombay

September 22, 2018
ELGGA, BITS Pilani, Hyderabad

Non-blocking Networks

Non-blocking Networks

Design a network for a telephone switch

Non-blocking Networks

Design a network for a telephone switch

- N input nodes, N output nodes, internal nodes and edges.

Non-blocking Networks

Design a network for a telephone switch

- N input nodes, N output nodes, internal nodes and edges.
- Goal: Given a permutation π establish node disjoint connection from input i to output $\pi(i)$

Non-blocking Networks

Design a network for a telephone switch

- N input nodes, N output nodes, internal nodes and edges.
- Goal: Given a permutation π establish node disjoint connection from input i to output $\pi(i)$
- Algorithm for establishing connections should work fast.

Non-blocking Networks

Design a network for a telephone switch

- N input nodes, N output nodes, internal nodes and edges.
- Goal: Given a permutation π establish node disjoint connection from input i to output $\pi(i)$
- Algorithm for establishing connections should work fast.

Useful also in a parallel processor

Non-blocking Networks

Design a network for a telephone switch

- N input nodes, N output nodes, internal nodes and edges.
- Goal: Given a permutation π establish node disjoint connection from input i to output $\pi(i)$
- Algorithm for establishing connections should work fast.

Useful also in a parallel processor

Obvious design: Connect each input to each output.

Non-blocking Networks

Design a network for a telephone switch

- N input nodes, N output nodes, internal nodes and edges.
- Goal: Given a permutation π establish node disjoint connection from input i to output $\pi(i)$
- Algorithm for establishing connections should work fast.

Useful also in a parallel processor

Obvious design: Connect each input to each output.

$O(N^2)$ wires. Too expensive.

Non-blocking Networks

Design a network for a telephone switch

- N input nodes, N output nodes, internal nodes and edges.
- Goal: Given a permutation π establish node disjoint connection from input i to output $\pi(i)$
- Algorithm for establishing connections should work fast.

Useful also in a parallel processor

Obvious design: Connect each input to each output.

$O(N^2)$ wires. Too expensive.

Can we design a network using $O(N \log N)$ edges, vertices?

Non-blocking Networks

Design a network for a telephone switch

- N input nodes, N output nodes, internal nodes and edges.
- Goal: Given a permutation π establish node disjoint connection from input i to output $\pi(i)$
- Algorithm for establishing connections should work fast.

Useful also in a parallel processor

Obvious design: Connect each input to each output.

$O(N^2)$ wires. Too expensive.

Can we design a network using $O(N \log N)$ edges, vertices?

$\Omega(N \log N)$ edges necessary.

Non-blocking Networks

Design a network for a telephone switch

- N input nodes, N output nodes, internal nodes and edges.
- Goal: Given a permutation π establish node disjoint connection from input i to output $\pi(i)$
- Algorithm for establishing connections should work fast.

Useful also in a parallel processor

Obvious design: Connect each input to each output.

$O(N^2)$ wires. Too expensive.

Can we design a network using $O(N \log N)$ edges, vertices?

$\Omega(N \log N)$ edges necessary. Shannon 1950

Outline

Outline

- Some relevant networks

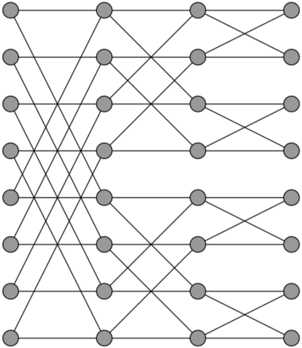
Outline

- Some relevant networks
- The Multibutterfly

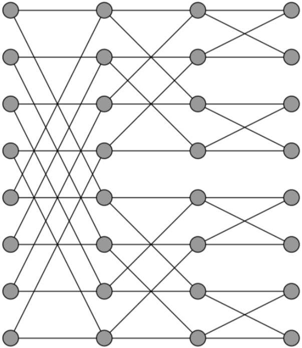
Outline

- Some relevant networks
- The Multibutterfly
- Path selection algorithm

Some relevant networks: Butterfly

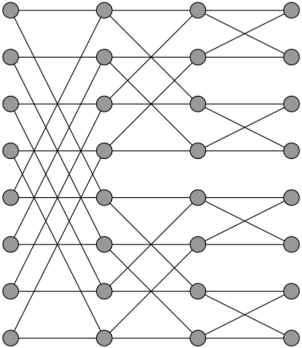


Some relevant networks: Butterfly



Butterfly network $B(N)$

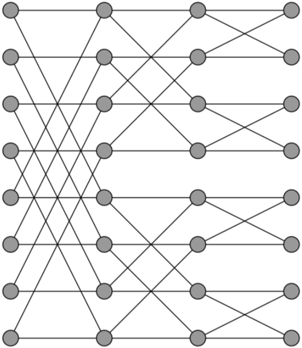
Some relevant networks: Butterfly



Butterfly network $B(N)$

$N = 2^n$ inputs, $N = 2^n$ outputs

Some relevant networks: Butterfly

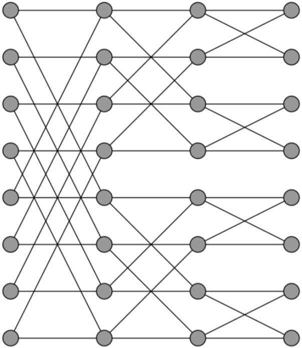


Butterfly network $B(N)$

$N = 2^n$ inputs, $N = 2^n$ outputs

Inputs : X : $\genfrac{}{}{0pt}{}{B(N/2)}{B(N/2)}$: Outputs

Some relevant networks: Butterfly



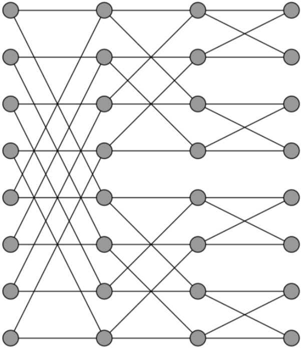
Butterfly network $B(N)$

$N = 2^n$ inputs, $N = 2^n$ outputs

Inputs : X : $\frac{B(N/2)}{B(N/2)}$: Outputs

Unique path from any input i to any output j.

Some relevant networks: Butterfly



Butterfly network $B(N)$

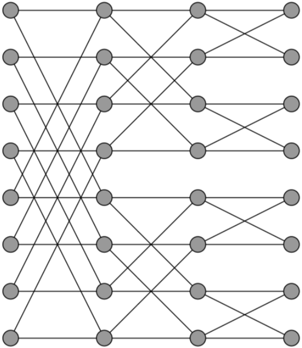
$N = 2^n$ inputs, $N = 2^n$ outputs

Inputs : X : $\frac{B(N/2)}{B(N/2)}$: Outputs

Unique path from any input i to any output j.

1. If each i connects to random $\pi(i)$, then $N/\log N$ vertex disjoint paths can be established with high probability.

Some relevant networks: Butterfly



Butterfly network $B(N)$

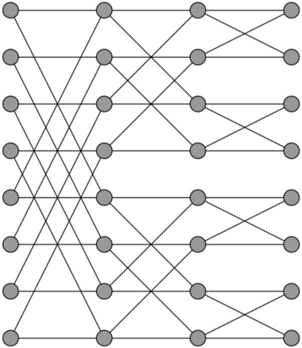
$N = 2^n$ inputs, $N = 2^n$ outputs

Inputs : X : $\frac{B(N/2)}{B(N/2)}$: Outputs

Unique path from any input i to any output j.

1. If each i connects to random $\pi(i)$, then $N/\log N$ vertex disjoint paths can be established with high probability.
2. $\exists$ permutation π such that at most $\sqrt{N}$ vertex disjoint paths can be established.

Some relevant networks: Butterfly



Butterfly network $B(N)$

$N = 2^n$ inputs, $N = 2^n$ outputs

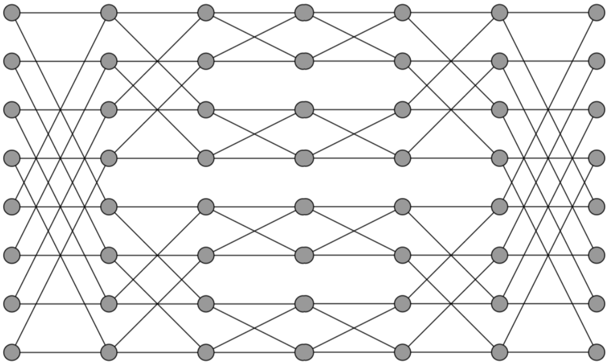
Inputs : X : $\begin{smallmatrix} B(N/2) \\ B(N/2) \end{smallmatrix}$: Outputs

Unique path from any input i to any output j.

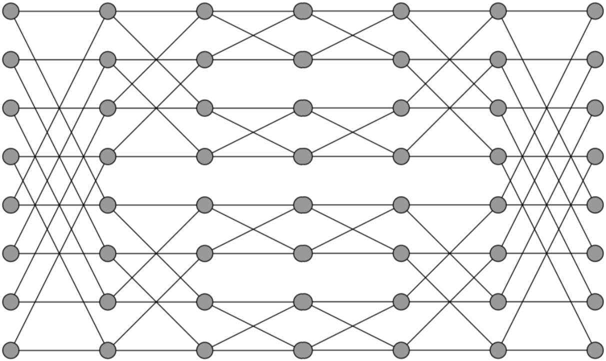
1. If each i connects to random $\pi(i)$, then $N/\log N$ vertex disjoint paths can be established with high probability.
2. $\exists$ permutation π such that at most $\sqrt{N}$ vertex disjoint paths can be established.
3. Edge disjoint paths exist for $\pi(i) = i + \alpha \bmod N$, all α

Benes Network: $Be(n) = B(N) : B(N)^{-1}$

Benes Network: $Be(n) = B(N) : B(N)^{-1}$

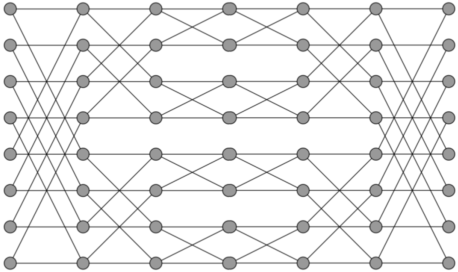


Benes Network: $Be(n) = B(N) : B(N)^{-1}$

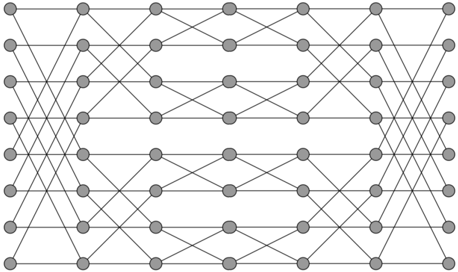


Theorem: We can establish vertex disjoint paths for any π.

Algorithm to make connections in Benes Network

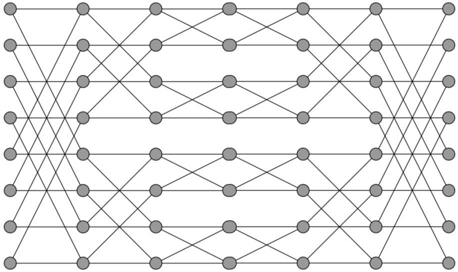


Algorithm to make connections in Benes Network



Algorithm has to make N connections.

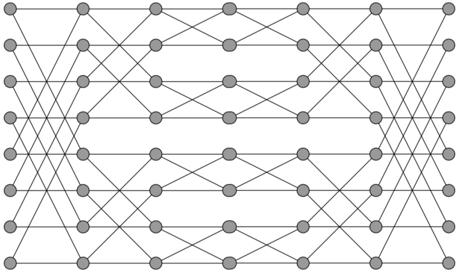
Algorithm to make connections in Benes Network



Algorithm has to make N connections.

Assign $N/2$ connections to top Benes, $N/2$ to bottom Benes s.t.

Algorithm to make connections in Benes Network

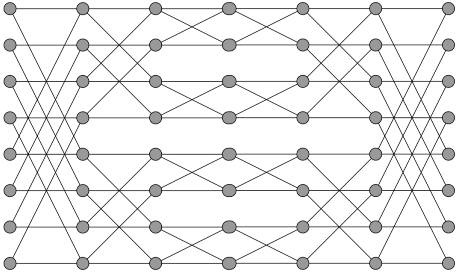


Algorithm has to make N connections.

Assign $N/2$ connections to top Benes, $N/2$ to bottom Benes s.t.

- Top, Bottom Benes are required to recursively realize smaller permutations.

Algorithm to make connections in Benes Network

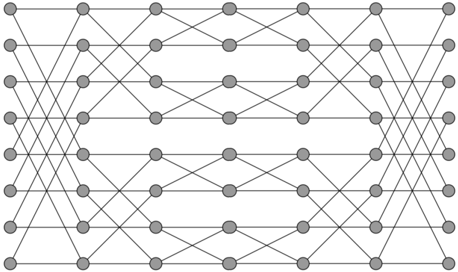


Algorithm has to make N connections.

Assign $N/2$ connections to top Benes, $N/2$ to bottom Benes s.t.

- Top, Bottom Benes are required to recursively realize smaller permutations.
- No conflicts produced in outermost wiring layers.

Algorithm to make connections in Benes Network



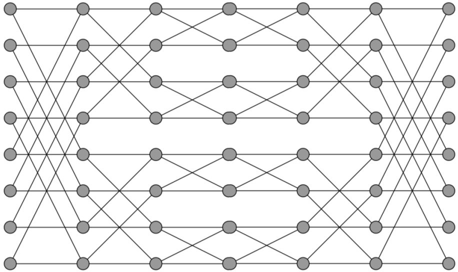
Algorithm has to make N connections.

Assign $N/2$ connections to top Benes, $N/2$ to bottom Benes s.t.

- Top, Bottom Benes are required to recursively realize smaller permutations.
- No conflicts produced in outermost wiring layers.

i —- $\pi(i)$ path uses top $\Leftrightarrow$ $i + \frac{N}{2}$ —- $\pi(i + \frac{N}{2})$ path uses bottom

Algorithm to make connections in Benes Network



Algorithm has to make N connections.

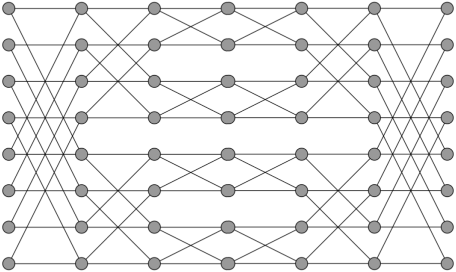
Assign $N/2$ connections to top Benes, $N/2$ to bottom Benes s.t.

- Top, Bottom Benes are required to recursively realize smaller permutations.
- No conflicts produced in outermost wiring layers.

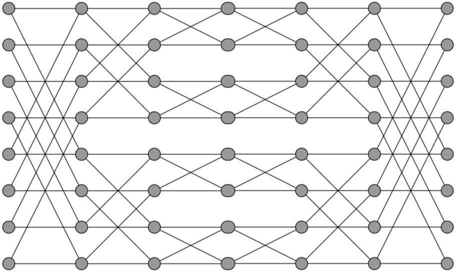
i —- $\pi(i)$ path uses top $\Leftrightarrow$ $i + \frac{N}{2}$ —- $\pi(i + \frac{N}{2})$ path uses bottom

Similarly for the outputs.

$C(i) = i + \frac{N}{2},\ C(i + \frac{N}{2}) = i \quad i, C(i)$: competing pair.

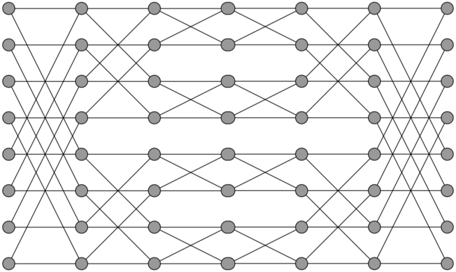


$C(i) = i + \frac{N}{2}$, $C(i + \frac{N}{2}) = i \quad i, C(i)$: competing pair.



Phased algorithm:

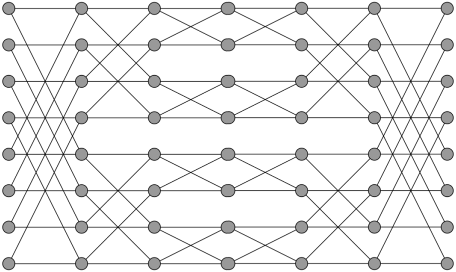
$C(i) = i + \frac{N}{2}$, $C(i + \frac{N}{2}) = i$ $\quad i, C(i)$: competing pair.



Phased algorithm:

Invariant: Competing pairs are both assigned or both not.

$C(i) = i + \frac{N}{2}$, $C(i + \frac{N}{2}) = i$ $\quad i, C(i)$: competing pair.

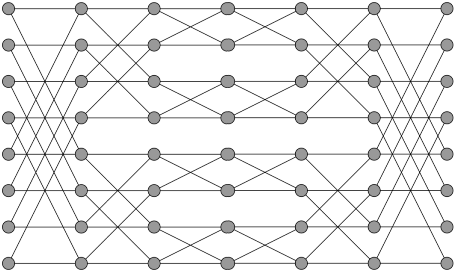


Phased algorithm:

Invariant: Competing pairs are both assigned or both not.

- Consider any unassigned competing pair of inputs $i, C(i)$.

$C(i) = i + \frac{N}{2}$, $C(i + \frac{N}{2}) = i$ $\quad i, C(i)$: competing pair.

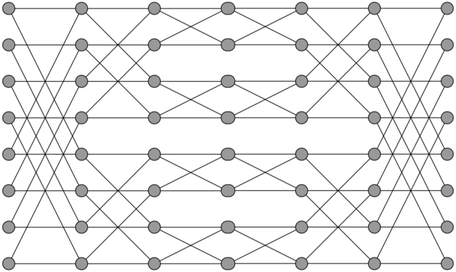


Phased algorithm:

Invariant: Competing pairs are both assigned or both not.

- Consider any unassigned competing pair of inputs $i, C(i)$.
- Connect $i - \pi(i)$ through top, $C(i) - \pi(C(i))$ through bottom.

$C(i) = i + \frac{N}{2}$, $C(i + \frac{N}{2}) = i$ $\quad i, C(i)$: competing pair.

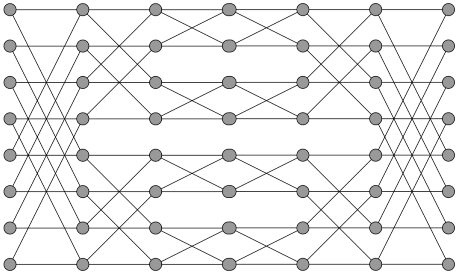


Phased algorithm:

Invariant: Competing pairs are both assigned or both not.

- Consider any unassigned competing pair of inputs $i, C(i)$.
- Connect $i - \pi(i)$ through top, $C(i) - \pi(C(i))$ through bottom.
- If $\pi(i), \pi(C(i))$ are competing pairs, then phase ends.

$C(i) = i + \frac{N}{2}$, $C(i + \frac{N}{2}) = i$ $\quad i, C(i)$: competing pair.

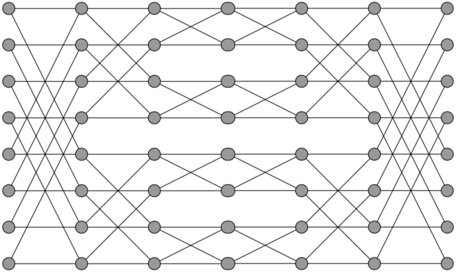


Phased algorithm:

Invariant: Competing pairs are both assigned or both not.

- Consider any unassigned competing pair of inputs $i, C(i)$.
- Connect $i - \pi(i)$ through top, $C(i) - \pi(C(i))$ through bottom.
- If $\pi(i), \pi(C(i))$ are competing pairs, then phase ends.
- Otherwise we have two half touched pairs on outputs.

$C(i) = i + \frac{N}{2}$, $C(i + \frac{N}{2}) = i$ $\quad i, C(i)$: competing pair.

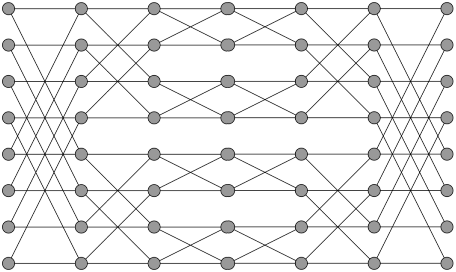


Phased algorithm:

Invariant: Competing pairs are both assigned or both not.

- Consider any unassigned competing pair of inputs $i, C(i)$.
- Connect $i - \pi(i)$ through top, $C(i) - \pi(C(i))$ through bottom.
- If $\pi(i), \pi(C(i))$ are competing pairs, then phase ends.
- Otherwise we have two half touched pairs on outputs.
- Connect $C(\pi(i))$ back using bottom pair.

$C(i) = i + \frac{N}{2},\ C(i+\frac{N}{2}) = i \quad i, C(i)$: competing pair.

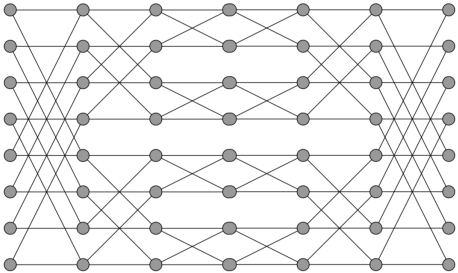


Phased algorithm:

Invariant: Competing pairs are both assigned or both not.

- Consider any unassigned competing pair of inputs $i, C(i)$.
- Connect $i - \pi(i)$ through top, $C(i) - \pi(C(i))$ through bottom.
- If $\pi(i), \pi(C(i))$ are competing pairs, then phase ends.
- Otherwise we have two half touched pairs on outputs.
- Connect $C(\pi(i))$ back using bottom pair.
- Connect $C(\pi(C(i)))$ back using top pair.

$C(i) = i + \frac{N}{2}$, $C(i + \frac{N}{2}) = i \quad i, C(i)$: competing pair.

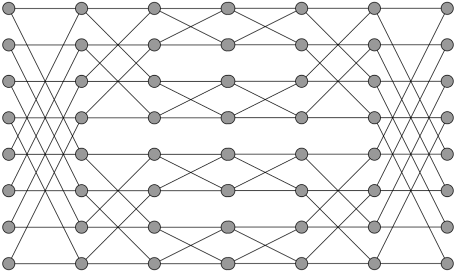


Phased algorithm:

Invariant: Competing pairs are both assigned or both not.

- Consider any unassigned competing pair of inputs $i, C(i)$.
- Connect $i - \pi(i)$ through top, $C(i) - \pi(C(i))$ through bottom.
- If $\pi(i), \pi(C(i))$ are competing pairs, then phase ends.
- Otherwise we have two half touched pairs on outputs.
- Connect $C(\pi(i))$ back using bottom pair.
- Connect $C(\pi(C(i)))$ back using top pair.
- Phase ends, or two half touched pairs on inputs.

$C(i) = i + \frac{N}{2}$, $C(i + \frac{N}{2}) = i \quad i, C(i)$: competing pair.

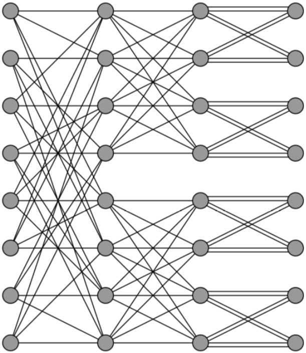


Phased algorithm:

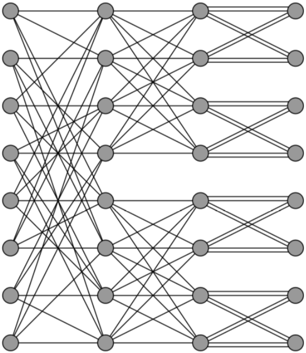
Invariant: Competing pairs are both assigned or both not.

- Consider any unassigned competing pair of inputs $i, C(i)$.
- Connect $i - \pi(i)$ through top, $C(i) - \pi(C(i))$ through bottom.
- If $\pi(i), \pi(C(i))$ are competing pairs, then phase ends.
- Otherwise we have two half touched pairs on outputs.
- Connect $C(\pi(i))$ back using bottom pair.
- Connect $C(\pi(C(i)))$ back using top pair.
- Phase ends, or two half touched pairs on inputs. Repeat..

Multibutterfly $M_d(N)$

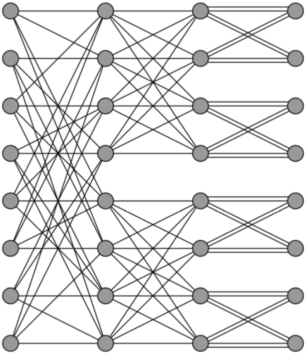


Multibutterfly $M_d(N)$



d edges to top and d edges to bottom from each node.

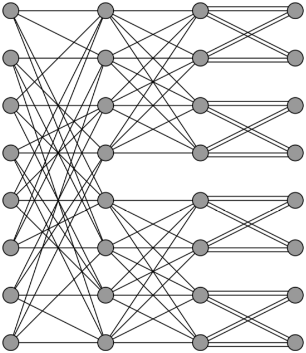
Multibutterfly $M_d(N)$



d edges to top and d edges to bottom from each node.

$d = 2$ shown.

Multibutterfly $M_d(N)$

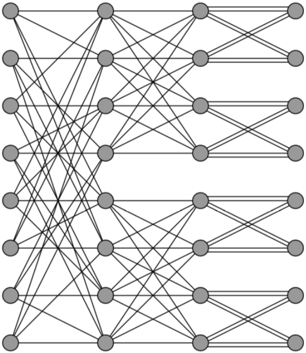


d edges to top and d edges to bottom from each node.

$d = 2$ shown.

Connections are random, s.t. all indegree, outdegree are $2d$.

Multibutterfly $M_d(N)$



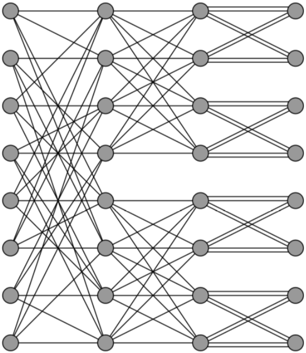
d edges to top and d edges to bottom from each node.

$d = 2$ shown.

Connections are random, s.t. all indegree, outdegree are $2d$.

Produces nice expansion properties.

Multibutterfly $M_d(N)$



d edges to top and d edges to bottom from each node.

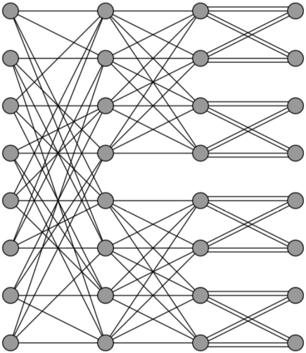
$d = 2$ shown.

Connections are random, s.t. all indegree, outdegree are $2d$.

Produces nice expansion properties.

Effect of random edges: W.h.p. expansion from level to level: k nodes in level 0 connect to βk nodes in top, and βk nodes in bottom, for $k \leq \alpha N$.

Multibutterfly $M_d(N)$



d edges to top and d edges to bottom from each node.

$d = 2$ shown.

Connections are random, s.t. all indegree, outdegree are $2d$.

Produces nice expansion properties.

Effect of random edges: W.h.p. expansion from level to level: k nodes in level 0 connect to βk nodes in top, and βk nodes in bottom, for $k \leq \alpha N$.

Even for $\beta > 1$ $d = O(1)$ suffices.

(α, β, N, d) concentrator

(α, β, N, d) concentrator

is a bipartite graph (U, V, E) where $|U| = N, |V| = N/2$ degree in U, V are resp. at most $d, 2d$, and every subset of $k \leq \alpha N$ nodes of U has edges to at least βk nodes of V.

(α, β, N, d) concentrator

is a bipartite graph (U, V, E) where $|U| = N, |V| = N/2$ degree in U, V are resp. at most $d, 2d$, and every subset of $k \leq \alpha N$ nodes of U has edges to at least βk nodes of V.

Concentrators connect level 0 to top and to bottor $M_d(N/2)$s

(α, β, N, d) concentrator

is a bipartite graph (U, V, E) where $|U| = N, |V| = N/2$ degree in U, V are resp. at most $d, 2d$, and every subset of $k \leq \alpha N$ nodes of U has edges to at least βk nodes of V.

Concentrators connect level 0 to top and to bottor $M_d(N/2)$s

Theorem: Concentrator exists for $\alpha > 0$, $\beta > 1$ s.t. $\alpha\beta < 1$, and

$$d > \beta + 1 + \frac{\beta + 1 + \ln 4\beta}{ln(1/2\alpha\beta)}$$

(α, β, N, d) concentrator

is a bipartite graph (U, V, E) where $|U| = N, |V| = N/2$ degree in U, V are resp. at most $d, 2d$, and every subset of $k \leq \alpha N$ nodes of U has edges to at least βk nodes of V.

Concentrators connect level 0 to top and to bottor $M_d(N/2)$s

Theorem: Concentrator exists for $\alpha > 0$, $\beta > 1$ s.t. $\alpha\beta < 1$, and

$$d > \beta + 1 + \frac{\beta + 1 + \ln 4\beta}{ln(1/2\alpha\beta)}$$

Proof: Randomized construction: Start with vertices U, V

(α, β, N, d) concentrator

is a bipartite graph (U, V, E) where $|U| = N, |V| = N/2$ degree in U, V are resp. at most $d, 2d$, and every subset of $k \leq \alpha N$ nodes of U has edges to at least βk nodes of V.

Concentrators connect level 0 to top and to bottor $M_d(N/2)$s

Theorem: Concentrator exists for $\alpha > 0$, $\beta > 1$ s.t. $\alpha\beta < 1$, and

$$d > \beta + 1 + \frac{\beta + 1 + \ln 4\beta}{ln(1/2\alpha\beta)}$$

Proof: Randomized construction: Start with vertices U, V

- Replace each vertex in U, V resp. by $d, 2d$ vertices.

(α, β, N, d) concentrator

is a bipartite graph (U, V, E) where $|U| = N, |V| = N/2$ degree in U, V are resp. at most $d, 2d$, and every subset of $k \leq \alpha N$ nodes of U has edges to at least βk nodes of V.

Concentrators connect level 0 to top and to bottor $M_d(N/2)$s

Theorem: Concentrator exists for $\alpha > 0$, $\beta > 1$ s.t. $\alpha\beta < 1$, and

$$d > \beta + 1 + \frac{\beta + 1 + \ln 4\beta}{ln(1/2\alpha\beta)}$$

Proof: Randomized construction: Start with vertices U, V

- Replace each vertex in U, V resp. by $d, 2d$ vertices.
- Pick a random matching on the two pairs of Nd vertices.

(α, β, N, d) concentrator

is a bipartite graph (U, V, E) where $|U| = N, |V| = N/2$ degree in U, V are resp. at most $d, 2d$, and every subset of $k \leq \alpha N$ nodes of U has edges to at least βk nodes of V.

Concentrators connect level 0 to top and to bottor $M_d(N/2)$s

Theorem: Concentrator exists for $\alpha > 0$, $\beta > 1$ s.t. $\alpha\beta < 1$, and

$$d > \beta + 1 + \frac{\beta + 1 + \ln 4\beta}{ln(1/2\alpha\beta)}$$

Proof: Randomized construction: Start with vertices U, V

- Replace each vertex in U, V resp. by $d, 2d$ vertices.
- Pick a random matching on the two pairs of Nd vertices.
- Collapse vertices back.

(α, β, N, d) concentrator

is a bipartite graph (U, V, E) where $|U| = N, |V| = N/2$ degree in U, V are resp. at most $d, 2d$, and every subset of $k \le \alpha N$ nodes of U has edges to at least βk nodes of V.

Concentrators connect level 0 to top and to bottor $M_d(N/2)$s

Theorem: Concentrator exists for $\alpha > 0$, $\beta > 1$ s.t. $\alpha\beta < 1$, and

$$d > \beta + 1 + \frac{\beta + 1 + \ln 4\beta}{ln(1/2\alpha\beta)}$$

Proof: Randomized construction: Start with vertices U, V

- Replace each vertex in U, V resp. by $d, 2d$ vertices.
- Pick a random matching on the two pairs of Nd vertices.
- Collapse vertices back.
- U have degree $\le d$, V have degree $\le 2d$

(α, β, N, d) concentrator

is a bipartite graph (U, V, E) where $|U| = N, |V| = N/2$ degree in U, V are resp. at most $d, 2d$, and every subset of $k \leq \alpha N$ nodes of U has edges to at least βk nodes of V.

Concentrators connect level 0 to top and to bottor $M_d(N/2)$s

Theorem: Concentrator exists for $\alpha > 0$, $\beta > 1$ s.t. $\alpha\beta < 1$, and

$$d > \beta + 1 + \frac{\beta + 1 + \ln 4\beta}{ln(1/2\alpha\beta)}$$

Proof: Randomized construction: Start with vertices U, V

- Replace each vertex in U, V resp. by $d, 2d$ vertices.
- Pick a random matching on the two pairs of Nd vertices.
- Collapse vertices back.
- U have degree $\leq d$, V have degree $\leq 2d$

Remove parallel edges if any

(α, β, N, d) concentrator

is a bipartite graph (U, V, E) where $|U| = N, |V| = N/2$ degree in U, V are resp. at most $d, 2d$, and every subset of $k \leq \alpha N$ nodes of U has edges to at least βk nodes of V.

Concentrators connect level 0 to top and to bottor $M_d(N/2)$s

Theorem: Concentrator exists for $\alpha > 0$, $\beta > 1$ s.t. $\alpha\beta < 1$, and

$$d > \beta + 1 + \frac{\beta + 1 + \ln 4\beta}{ln(1/2\alpha\beta)}$$

Proof: Randomized construction: Start with vertices U, V

- Replace each vertex in U, V resp. by $d, 2d$ vertices.
- Pick a random matching on the two pairs of Nd vertices.
- Collapse vertices back.
- U have degree $\leq d$, V have degree $\leq 2d$

Remove parallel edges if any

Resulting graph satisfies definition with probability > 0.

(α, β, N, d) concentrator

is a bipartite graph (U, V, E) where $|U| = N, |V| = N/2$ degree in U, V are resp. at most $d, 2d$, and every subset of $k \leq \alpha N$ nodes of U has edges to at least βk nodes of V.

Concentrators connect level 0 to top and to bottor $M_d(N/2)$s

Theorem: Concentrator exists for $\alpha > 0$, $\beta > 1$ s.t. $\alpha\beta < 1$, and

$$d > \beta + 1 + \frac{\beta + 1 + \ln 4\beta}{ln(1/2\alpha\beta)}$$

Proof: Randomized construction: Start with vertices U, V

- Replace each vertex in U, V resp. by $d, 2d$ vertices.
- Pick a random matching on the two pairs of Nd vertices.
- Collapse vertices back.
- U have degree $\leq d$, V have degree $\leq 2d$

Remove parallel edges if any

Resulting graph satisfies definition with probability > 0. (Next)

(α, β, N, d) concentrator

is a bipartite graph (U, V, E) where $|U| = N, |V| = N/2$ degree in U, V are resp. at most $d, 2d$, and every subset of $k \leq \alpha N$ nodes of U has edges to at least βk nodes of V.

Concentrators connect level 0 to top and to bottor $M_d(N/2)$s

Theorem: Concentrator exists for $\alpha > 0$, $\beta > 1$ s.t. $\alpha\beta < 1$, and

$$d > \beta + 1 + \frac{\beta + 1 + \ln 4\beta}{ln(1/2\alpha\beta)}$$

Proof: Randomized construction: Start with vertices U, V

- Replace each vertex in U, V resp. by $d, 2d$ vertices.
- Pick a random matching on the two pairs of Nd vertices.
- Collapse vertices back.
- U have degree $\leq d$, V have degree $\leq 2d$

Remove parallel edges if any

Resulting graph satisfies definition with probability > 0. (Next)

If you increase d, probability increases very rapidly.

Probability of expansion

Probability of expansion

Assume not concentrator.

Probability of expansion

Assume not concentrator.

$\exists S \subseteq U$ s.t. $|S| = k \leq \alpha N$ and $|T = Nbr(S)| < \beta k$.

Probability of expansion

Assume not concentrator.

$\exists S \subseteq U$ s.t. $|S| = k \leq \alpha N$ and $|T = Nbr(S)| < \beta k$.

k can range between 1 and αN

Probability of expansion

Assume not concentrator.
$\exists S \subseteq U$ s.t. $|S| = k \leq \alpha N$ and $|T = Nbr(S)| < \beta k$.

k can range between 1 and αN
For each k there are $\binom{N}{k}$ choices for S, and $\binom{N/2}{\beta k}$ choices for T.

Probability of expansion

Assume not concentrator.
$\exists S \subseteq U$ s.t. $|S| = k \leq \alpha N$ and $|T = Nbr(S)| < \beta k$.

k can range between 1 and αN
For each k there are $\binom{N}{k}$ choices for S, and $\binom{N/2}{\beta k}$ choices for T.

Probability that construction does not give a concentrator:

$$\leq \sum_{k=1}^{k=\alpha N} \binom{N}{k} \binom{N/2}{\beta k} \Pr[\text{fixed } T \supseteq \text{nbrhood of fixed } S\ ]$$

Probability of expansion

Assume not concentrator.
$\exists S \subseteq U$ s.t. $|S| = k \leq \alpha N$ and $|T = Nbr(S)| < \beta k$.

k can range between 1 and αN
For each k there are $\binom{N}{k}$ choices for S, and $\binom{N/2}{\beta k}$ choices for T.

Probability that construction does not give a concentrator:

$$\leq \sum_{k=1}^{k=\alpha N} \binom{N}{k}\binom{N/2}{\beta k} \Pr[\text{fixed } T \supseteq \text{nbrhood of fixed } S\ ]$$

In the construction we first explode S to S' with $kd = s$ vertices and T to T' $2\beta kd = t$ vertices.

Probability of expansion

Assume not concentrator.
$\exists S \subseteq U$ s.t. $|S| = k \leq \alpha N$ and $|T = Nbr(S)| < \beta k$.

k can range between 1 and αN
For each k there are $\binom{N}{k}$ choices for S, and $\binom{N/2}{\beta k}$ choices for T.

Probability that construction does not give a concentrator:

$$\leq \sum_{k=1}^{k=\alpha N} \binom{N}{k} \binom{N/2}{\beta k} \Pr[\text{fixed } T \supseteq \text{nbrhood of fixed } S\ ]$$

In the construction we first explode S to S' with $kd = s$ vertices and T to T' $2\beta kd = t$ vertices.

$\Pr[\text{fixed } T \supseteq \text{nbrhood of fixed } S\ ] = \frac{t \cdot t-1 \cdots t-s+1}{Nd \cdots Nd-1 \cdots Nd-s+1}$

Probability of expansion

Assume not concentrator.
$\exists S \subseteq U$ s.t. $|S| = k \leq \alpha N$ and $|T = Nbr(S)| < \beta k$.

k can range between 1 and αN
For each k there are $\binom{N}{k}$ choices for S, and $\binom{N/2}{\beta k}$ choices for T.

Probability that construction does not give a concentrator:

$$\leq \sum_{k=1}^{k=\alpha N} \binom{N}{k} \binom{N/2}{\beta k} \Pr[\text{fixed } T \supseteq \text{nbrhood of fixed } S\ ]$$

In the construction we first explode S to S' with $kd = s$ vertices and T to T' $2\beta kd = t$ vertices.

$\Pr[\text{fixed } T \supseteq \text{nbrhood of fixed } S\ ] = \frac{t \cdot t-1 \cdots t-s+1}{Nd \cdots Nd-1 \cdots Nd-s+1}$

$\leq \left(\frac{t}{Nd}\right)^s$

Probability of expansion

Assume not concentrator.
$\exists S \subseteq U$ s.t. $|S| = k \leq \alpha N$ and $|T = Nbr(S)| < \beta k$.

k can range between 1 and αN
For each k there are $\binom{N}{k}$ choices for S, and $\binom{N/2}{\beta k}$ choices for T.

Probability that construction does not give a concentrator:

$$\leq \sum_{k=1}^{k=\alpha N} \binom{N}{k}\binom{N/2}{\beta k} \Pr[\text{fixed } T \supseteq \text{nbrhood of fixed } S\ ]$$

In the construction we first explode S to S' with $kd = s$ vertices and T to T' $2\beta kd = t$ vertices.

$\Pr[\text{fixed } T \supseteq \text{nbrhood of fixed } S\ ] = \frac{t \cdot t-1 \cdots t-s+1}{Nd \cdots Nd-1 \cdots Nd-s+1}$

$\leq \left(\frac{t}{Nd}\right)^s = \left(\frac{2\beta k}{N}\right)^{kd}$

Probability calculation continued

Probability calculation continued

Probability that construction does not give a concentrator:

$$\leq \sum_{k=1}^{k=\alpha N} \binom{N}{k} \binom{N/2}{\beta k} \left(\frac{2\beta k}{N}\right)^{kd}$$

Probability calculation continued

Probability that construction does not give a concentrator:

$$\leq \sum_{k=1}^{k=\alpha N} \binom{N}{k} \binom{N/2}{\beta k} \left(\frac{2\beta k}{N}\right)^{kd} \leq \sum_{k=1}^{k=\alpha N} \left(\frac{Ne}{k}\right)^{k} \left(\frac{Ne/2}{\beta k}\right)^{\beta k} \left(\frac{2\beta k}{N}\right)^{kd}$$

Probability calculation continued

Probability that construction does not give a concentrator:

$$\leq \sum_{k=1}^{k=\alpha N} \binom{N}{k} \binom{N/2}{\beta k} \left(\frac{2\beta k}{N}\right)^{kd} \leq \sum_{k=1}^{k=\alpha N} \left(\frac{Ne}{k}\right)^{k} \left(\frac{Ne/2}{\beta k}\right)^{\beta k} \left(\frac{2\beta k}{N}\right)^{kd}$$

$$\leq \sum_{k=1}^{k=\alpha N} \left(\left(\frac{K}{N}\right)^{d-\beta-1} e^{\beta+1}(2\beta)^{d-\beta} \right)^{k}$$

Probability calculation continued

Probability that construction does not give a concentrator:

$$\leq \sum_{k=1}^{k=\alpha N} \binom{N}{k} \binom{N/2}{\beta k} \left(\frac{2\beta k}{N}\right)^{kd} \leq \sum_{k=1}^{k=\alpha N} \left(\frac{Ne}{k}\right)^{k} \left(\frac{Ne/2}{\beta k}\right)^{\beta k} \left(\frac{2\beta k}{N}\right)^{kd}$$

$$\leq \sum_{k=1}^{k=\alpha N} \left(\left(\frac{K}{N}\right)^{d-\beta-1} e^{\beta+1}(2\beta)^{d-\beta}\right)^{k} \leq \sum_{k=1}^{k=\alpha N} \left(\alpha^{d-\beta-1} e^{\beta+1}(2\beta)^{d-\beta}\right)^{k}$$

Probability calculation continued

Probability that construction does not give a concentrator:

$$\leq \sum_{k=1}^{k=\alpha N} \binom{N}{k} \binom{N/2}{\beta k} \left(\frac{2\beta k}{N}\right)^{kd} \leq \sum_{k=1}^{k=\alpha N} \left(\frac{Ne}{k}\right)^{k} \left(\frac{Ne/2}{\beta k}\right)^{\beta k} \left(\frac{2\beta k}{N}\right)^{kd}$$

$$\leq \sum_{k=1}^{k=\alpha N} \left(\left(\frac{K}{N}\right)^{d-\beta-1} e^{\beta+1}(2\beta)^{d-\beta}\right)^{k} \leq \sum_{k=1}^{k=\alpha N} \left(\alpha^{d-\beta-1} e^{\beta+1}(2\beta)^{d-\beta}\right)^{k}$$

Using $\binom{n}{r} \leq (ne/r)^r, k/N \leq \alpha$

Probability calculation continued

Probability that construction does not give a concentrator:

$$\leq \sum_{k=1}^{k=\alpha N} \binom{N}{k} \binom{N/2}{\beta k} \left(\frac{2\beta k}{N}\right)^{kd} \leq \sum_{k=1}^{k=\alpha N} \left(\frac{Ne}{k}\right)^{k} \left(\frac{Ne/2}{\beta k}\right)^{\beta k} \left(\frac{2\beta k}{N}\right)^{kd}$$

$$\leq \sum_{k=1}^{k=\alpha N} \left(\left(\frac{K}{N}\right)^{d-\beta-1} e^{\beta+1}(2\beta)^{d-\beta}\right)^{k} \leq \sum_{k=1}^{k=\alpha N} \left(\alpha^{d-\beta-1} e^{\beta+1}(2\beta)^{d-\beta}\right)^{k}$$

Using $\binom{n}{r} \leq (ne/r)^r, k/N \leq \alpha$

Geometric series is < 1 if:

$$\alpha^{d-\beta-1} e^{\beta+1}(2\beta)^{d-\beta} < 1/2$$

Probability calculation continued

Probability that construction does not give a concentrator:

$$\leq \sum_{k=1}^{k=\alpha N} \binom{N}{k} \binom{N/2}{\beta k} \left(\frac{2\beta k}{N}\right)^{kd} \leq \sum_{k=1}^{k=\alpha N} \left(\frac{Ne}{k}\right)^{k} \left(\frac{Ne/2}{\beta k}\right)^{\beta k} \left(\frac{2\beta k}{N}\right)^{kd}$$

$$\leq \sum_{k=1}^{k=\alpha N} \left(\left(\frac{K}{N}\right)^{d-\beta-1} e^{\beta+1}(2\beta)^{d-\beta}\right)^{k} \leq \sum_{k=1}^{k=\alpha N} \left(\alpha^{d-\beta-1} e^{\beta+1}(2\beta)^{d-\beta}\right)^{k}$$

Using $\binom{n}{r} \leq (ne/r)^r, k/N \leq \alpha$

Geometric series is < 1 if:

$$\alpha^{d-\beta-1} e^{\beta+1}(2\beta)^{d-\beta} < 1/2$$

Simplifying gives the result.

What we have at this point..

What we have at this point..

Consider some submultibutterfly with N' inputs.

What we have at this point..

Consider some submultibutterfly with N' inputs.
Each set of $\alpha N'$ inputs will have βk neighbours in top of next.

What we have at this point..

Consider some submultibutterfly with N' inputs.
Each set of $\alpha N'$ inputs will have βk neighbours in top of next.

Can we extend paths in this submultibutterfly?

What we have at this point..

Consider some submultibutterfly with N' inputs.
Each set of $\alpha N'$ inputs will have βk neighbours in top of next.

Can we extend paths in this submultibutterfly?
Only if we are asking to extend at most $\alpha N'$ paths.

What we have at this point..

Consider some submultibutterfly with N' inputs.
Each set of $\alpha N'$ inputs will have βk neighbours in top of next.

Can we extend paths in this submultibutterfly?
Only if we are asking to extend at most $\alpha N'$ paths.

Only consider packets going to destinations 0 mod $L = 1/2\alpha$.

What we have at this point..

Consider some submultibutterfly with N' inputs.
Each set of $\alpha N'$ inputs will have βk neighbours in top of next.

Can we extend paths in this submultibutterfly?
Only if we are asking to extend at most $\alpha N'$ paths.

Only consider packets going to destinations 0 mod $L = 1/2\alpha$.
In our submulti at most $(N'/2)/L = N'\alpha$ top destinations are valid.

What we have at this point..

Consider some submultibutterfly with N' inputs.
Each set of $\alpha N'$ inputs will have βk neighbours in top of next.

Can we extend paths in this submultibutterfly?
Only if we are asking to extend at most $\alpha N'$ paths.

Only consider packets going to destinations 0 mod $L = 1/2\alpha$.
In our submulti at most $(N'/2)/L = N'\alpha$ top destinations are valid.
So number of paths requesting will be at most $\alpha N'$. top.

What we have at this point..

Consider some submultibutterfly with N' inputs.
Each set of $\alpha N'$ inputs will have βk neighbours in top of next.

Can we extend paths in this submultibutterfly?
Only if we are asking to extend at most $\alpha N'$ paths.

Only consider packets going to destinations 0 mod $L = 1/2\alpha$.
In our submulti at most $(N'/2)/L = N'\alpha$ top destinations are valid.
So number of paths requesting will be at most $\alpha N'$. top.

By Hall's Theorem: all paths will extend forward.

What we have at this point..

Consider some submultibutterfly with N' inputs.
Each set of $\alpha N'$ inputs will have βk neighbours in top of next.

Can we extend paths in this submultibutterfly?
Only if we are asking to extend at most $\alpha N'$ paths.

Only consider packets going to destinations 0 mod $L = 1/2\alpha$.
In our submulti at most $(N'/2)/L = N'\alpha$ top destinations are valid.
So number of paths requesting will be at most $\alpha N'$. top.

By Hall's Theorem: all paths will extend forward.

So we overdesign the network by a factor $L = 1/2\alpha = O(1)$.

What we have at this point..

Consider some submultibutterfly with N' inputs.
Each set of $\alpha N'$ inputs will have βk neighbours in top of next.

Can we extend paths in this submultibutterfly?
Only if we are asking to extend at most $\alpha N'$ paths.

Only consider packets going to destinations 0 mod $L = 1/2\alpha$.
In our submulti at most $(N'/2)/L = N'\alpha$ top destinations are valid.
So number of paths requesting will be at most $\alpha N'$. top.

By Hall's Theorem: all paths will extend forward.

So we overdesign the network by a factor $L = 1/2\alpha = O(1)$.

But will extending paths take time = O(finding matching)?

Parallel algorithm to find the matching

Parallel algorithm to find the matching

- S nodes wish to extend path, send request to neighbours.

Parallel algorithm to find the matching

- S nodes wish to extend path, send request to neighbours.
- Requests go to set T, with $|T| \geq \beta |S|$

Parallel algorithm to find the matching

- S nodes wish to extend path, send request to neighbours.
- Requests go to set T, with $|T| \geq \beta|S|$
- Those T' that get just 1 request, send back acknowledgement.

Parallel algorithm to find the matching

- S nodes wish to extend path, send request to neighbours.
- Requests go to set T, with $|T| \geq \beta|S|$
- Those T' that get just 1 request, send back acknowledgement.
- Requesters getting acknowledgement extend their path.

Parallel algorithm to find the matching

- S nodes wish to extend path, send request to neighbours.
- Requests go to set T, with $|T| \geq \beta|S|$
- Those T' that get just 1 request, send back acknowledgement.
- Requesters getting acknowledgement extend their path.
- Repeat.

Parallel algorithm to find the matching

- S nodes wish to extend path, send request to neighbours.
- Requests go to set T, with $|T| \geq \beta|S|$
- Those T' that get just 1 request, send back acknowledgement.
- Requesters getting acknowledgement extend their path.
- Repeat.

Number of requests sent out from S is $d|S|$.

Parallel algorithm to find the matching

- S nodes wish to extend path, send request to neighbours.
- Requests go to set T, with $|T| \geq \beta |S|$
- Those T' that get just 1 request, send back acknowledgement.
- Requesters getting acknowledgement extend their path.
- Repeat.

Number of requests sent out from S is $d|S|$.
Number of requests received is at least $|T'| + 2(|T| - |T'|)$

Parallel algorithm to find the matching

- S nodes wish to extend path, send request to neighbours.
- Requests go to set T, with $|T| \geq \beta|S|$
- Those T' that get just 1 request, send back acknowledgement.
- Requesters getting acknowledgement extend their path.
- Repeat.

Number of requests sent out from S is $d|S|$.
Number of requests received is at least $|T'| + 2(|T| - |T'|)$

$$d|S| \geq |T'| + 2(|T| - |T'|) = 2|T| - |T'|$$

Parallel algorithm to find the matching

- S nodes wish to extend path, send request to neighbours.
- Requests go to set T, with $|T| \geq \beta|S|$
- Those T' that get just 1 request, send back acknowledgement.
- Requesters getting acknowledgement extend their path.
- Repeat.

Number of requests sent out from S is $d|S|$.
Number of requests received is at least $|T'| + 2(|T| - |T'|)$

$$d|S| \geq |T'| + 2(|T| - |T'|) = 2|T| - |T'|$$

But $|T| \geq \beta|S|$.

Parallel algorithm to find the matching

- S nodes wish to extend path, send request to neighbours.
- Requests go to set T, with $|T| \geq \beta|S|$
- Those T' that get just 1 request, send back acknowledgement.
- Requesters getting acknowledgement extend their path.
- Repeat.

Number of requests sent out from S is $d|S|$.
Number of requests received is at least $|T'| + 2(|T| - |T'|)$

$$d|S| \geq |T'| + 2(|T| - |T'|) = 2|T| - |T'|$$

But $|T| \geq \beta|S|$.

$|T'| \geq 2|T| - d|S|$

Parallel algorithm to find the matching

- S nodes wish to extend path, send request to neighbours.
- Requests go to set T, with $|T| \geq \beta|S|$
- Those T' that get just 1 request, send back acknowledgement.
- Requesters getting acknowledgement extend their path.
- Repeat.

Number of requests sent out from S is $d|S|$.
Number of requests received is at least $|T'| + 2(|T| - |T'|)$

$$d|S| \geq |T'| + 2(|T| - |T'|) = 2|T| - |T'|$$

But $|T| \geq \beta|S|$.

$|T'| \geq 2|T| - d|S| \geq 2\beta|S| - d|S|$

Parallel algorithm to find the matching

- S nodes wish to extend path, send request to neighbours.
- Requests go to set T, with $|T| \geq \beta|S|$
- Those T' that get just 1 request, send back acknowledgement.
- Requesters getting acknowledgement extend their path.
- Repeat.

Number of requests sent out from S is $d|S|$.
Number of requests received is at least $|T'| + 2(|T| - |T'|)$

$$d|S| \geq |T'| + 2(|T| - |T'|) = 2|T| - |T'|$$

But $|T| \geq \beta|S|$.

$|T'| \geq 2|T| - d|S| \geq 2\beta|S| - d|S| \geq |S|(2\beta - d)$.

Parallel algorithm to find the matching

- S nodes wish to extend path, send request to neighbours.
- Requests go to set T, with $|T| \geq \beta|S|$
- Those T' that get just 1 request, send back acknowledgement.
- Requesters getting acknowledgement extend their path.
- Repeat.

Number of requests sent out from S is $d|S|$.
Number of requests received is at least $|T'| + 2(|T| - |T'|)$

$$d|S| \geq |T'| + 2(|T| - |T'|) = 2|T| - |T'|$$

But $|T| \geq \beta|S|$.

$|T'| \geq 2|T| - d|S| \geq 2\beta|S| - d|S| \geq |S|(2\beta - d)$.

The $|T'|$ acknowledgements must go to at least $|T'|/d$ requesters.

Parallel algorithm to find the matching

- S nodes wish to extend path, send request to neighbours.
- Requests go to set T, with $|T| \geq \beta|S|$
- Those T' that get just 1 request, send back acknowledgement.
- Requesters getting acknowledgement extend their path.
- Repeat.

Number of requests sent out from S is $d|S|$.
Number of requests received is at least $|T'| + 2(|T| - |T'|)$

$$d|S| \geq |T'| + 2(|T| - |T'|) = 2|T| - |T'|$$

But $|T| \geq \beta|S|$.

$|T'| \geq 2|T| - d|S| \geq 2\beta|S| - d|S| \geq |S|(2\beta - d)$.

The $|T'|$ acknowledgements must go to at least $|T'|/d$ requesters.

Number of successful requesters $\geq |S|(2\beta/d - 1)$.

Parallel algorithm to find the matching

- S nodes wish to extend path, send request to neighbours.
- Requests go to set T, with $|T| \geq \beta|S|$
- Those T' that get just 1 request, send back acknowledgement.
- Requesters getting acknowledgement extend their path.
- Repeat.

Number of requests sent out from S is $d|S|$.
Number of requests received is at least $|T'| + 2(|T| - |T'|)$

$$d|S| \geq |T'| + 2(|T| - |T'|) = 2|T| - |T'|$$

But $|T| \geq \beta|S|$.

$|T'| \geq 2|T| - d|S| \geq 2\beta|S| - d|S| \geq |S|(2\beta - d)$.

The $|T'|$ acknowledgements must go to at least $|T'|/d$ requesters.

Number of successful requesters $\geq |S|(2\beta/d - 1)$.

Constant fraction progress if $2\beta > d$

Parallel algorithm to find the matching

- S nodes wish to extend path, send request to neighbours.
- Requests go to set T, with $|T| \geq \beta|S|$
- Those T' that get just 1 request, send back acknowledgement.
- Requesters getting acknowledgement extend their path.
- Repeat.

Number of requests sent out from S is $d|S|$.
Number of requests received is at least $|T'| + 2(|T| - |T'|)$

$$d|S| \geq |T'| + 2(|T| - |T'|) = 2|T| - |T'|$$

But $|T| \geq \beta|S|$.

$|T'| \geq 2|T| - d|S| \geq 2\beta|S| - d|S| \geq |S|(2\beta - d)$.

The $|T'|$ acknowledgements must go to at least $|T'|/d$ requesters.

Number of successful requesters $\geq |S|(2\beta/d - 1)$.

Constant fraction progress if $2\beta > d$

Time per level $O(\log N)$, overall $O(\log^2 N)$.

Concluding remarks

Concluding remarks

- Random graphs are very powerful.

Concluding remarks

- Random graphs are very powerful.
- Note however that we know that with high probability we will have expansion, but there is not good way to decide how much expansion there is.

Concluding remarks

- Random graphs are very powerful.
- Note however that we know that with high probability we will have expansion, but there is not good way to decide how much expansion there is.
- Eigenvalues of adjacency or similar matrices give approximate estimate of expansion.

Concluding remarks

- Random graphs are very powerful.
- Note however that we know that with high probability we will have expansion, but there is not good way to decide how much expansion there is.
- Eigenvalues of adjacency or similar matrices give approximate estimate of expansion.
- Deterministic algorithms exist for constructing expanders, but they give smaller β for same d, α.

Concluding remarks

- Random graphs are very powerful.
- Note however that we know that with high probability we will have expansion, but there is not good way to decide how much expansion there is.
- Eigenvalues of adjacency or similar matrices give approximate estimate of expansion.
- Deterministic algorithms exist for constructing expanders, but they give smaller β for same d, α.
- Multibenes is also similarly defined.

Concluding remarks

- Random graphs are very powerful.
- Note however that we know that with high probability we will have expansion, but there is not good way to decide how much expansion there is.
- Eigenvalues of adjacency or similar matrices give approximate estimate of expansion.
- Deterministic algorithms exist for constructing expanders, but they give smaller β for same d, α.
- Multibenes is also similarly defined.
  - Enables requests to be added while others are in progress.

Concluding remarks

- Random graphs are very powerful.
- Note however that we know that with high probability we will have expansion, but there is not good way to decide how much expansion there is.
- Eigenvalues of adjacency or similar matrices give approximate estimate of expansion.
- Deterministic algorithms exist for constructing expanders, but they give smaller β for same d, α.
- Multibenes is also similarly defined.
  - Enables requests to be added while others are in progress.
- Do not know what happens with higher radix Butterfly.

S. Arora, T. Leighton, and B. Maggs, *On-line algorithms for path selection in a nonblocking network*, Proceedings of the ACM Annual Symposium on Theory of Computing, May 1990, pp. 149–158.

F. T. Leighton and B. M. Maggs, *Fast algorithms for routing around faults in multibutterflies and randomly-wired splitter networks*, IEEE Transactions on Computers **41** (1992), no. 5, 578–587.