Title: Graph expansions and simulating one-way quantum computation

Date: Apr 30, 2008 02:50 PM

URL: http://pirsa.org/08040052

Abstract: It is well-known that if a graph G_1 can be obtained from another graph G_2 by removing a degree-2 vertex and combing its two neighbors, the graph state |G_1> can be obtained from |G_2> through LOCC. In this talk, I will describe how to construct a graph G\' from a given graph G so that (a) The maximum degree of G\', Delta(G\'), is no more than 3. (b) G can be obtained from G\' by a sequence of contraction operations described above. (c) The treewidth of G\', tw(G\'), is no more than tw(G)+1. (d) The construction takes exp(O(tw(G))) time. Those properties imply that a one-way quantum computation on the graph state |G> can be simulated by a randomized algorithm in time exp(O(tw(G))). Previously, it was known that such a computation can be simulated in time exp(O(tw(G) Delta(G))) [Markov and Shi, to appear in SICOMP], which is substantially more expensive than our bound when Delta(G) is large. Joint work with Igor Markov, based on arXiv:0707.3622.

Graph expansions and simulating one-way quantum computation

Yaoyun Shi
University of Michigan

Joint work with Igor Markov
based on arXiv:0707.3622

Motivation

Is quantum computation indeed much more powerful than classical computation?

An incrementalist's program: identify largest sets of efficiently simulatable quantum computation.

- If the answer is "No", we are on the right track:
- Otherwise, we know better any non-simulatable quantum computation must be like.

Simulatable classes of quantum computation

- Quantum circuits using a restricted gate set [Gottesman and Knill '98;...]
- Reduction to the Fisher-Kasteleyn-Temperley algorithm [Valiant '02, Terhal and DiVincenzo '02;

computation.

- If the answer is "No", we are on the right track;
- Otherwise, we know better any non-simulatable quantum computation must be like.

Simulatable classes of quantum computation

- Quantum circuits using a restricted gate set [Gottesman and Knill '98;...]
- Reduction to the Fisher-Kasteleyn-Temperley algorithm [Valiant '02, Terhal and DiVincenzo '02; Bravyi and Raussendorf '06;...]

Motivation

Quantum circuit

A quantum circuit describes a quantum algorithm

- Start with a fixed initial state in the computational basis.
- Sequentially apply quantum gates, some on multiple qubits.
- Measure on the computational basis.

Quantum circuit

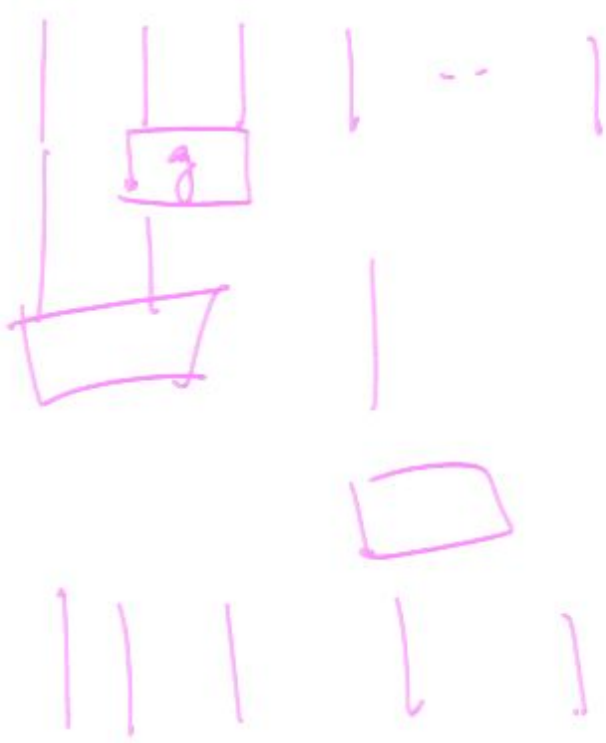
A quantum circuit describes a quantum algorithm

- Start with a fixed initial state in the computational basis.
- Sequentially apply quantum gates, some on multiple qubits.
- Measure on the computational basis.

Quantum circuit

A quantum circuit describes a quantum algorithm

- Start with a fixed initial state in the computational basis.
- Sequentially apply quantum gates, some on multiple qubits.
- Measure on the computational basis.



One-way quantum computation [Raussendorf and Briegel '01]

One-way quantum computation

1. Start with a fixed state $|S\rangle$.
2. Adaptively apply *single* qubit measurements.

One-way quantum computation [Raussendorf and Briegel '01]

One-way quantum computation

1. Start with a fixed state $|S\rangle$.
2. Adaptively apply *single* qubit measurements.

One-way quantum computation [Raussendorf and Briegel '01]

One-way quantum computation

1. Start with a fixed state $|S\rangle$.
2. Adaptively apply *single* qubit measurements.

Graph states [Briegel and Raussendorf '01]

Given an undirected simple graph $G = \langle V, E \rangle$, the graph state $|G\rangle$ is

$$|G\rangle := \frac{1}{2^{|V|\ 2}} \sum_{V' \subseteq V} (-1)^{|E(V')|} |V'\rangle,$$

where $E(V') \subseteq E$ are edges connecting vertices in V'.

Theorem [Raussendorf and Briegel '01] For any

Graph states [Briegel and Raussendorf '01]

Given an undirected simple graph $G = \langle V, E \rangle$, the graph state $|G\rangle$ is

$$|G\rangle := \frac{1}{2^{|V|/2}} \sum_{V' \subseteq V} (-1)^{|E(V')|} |V'\rangle,$$

where $E(V') \subseteq E$ are edges connecting vertices in V'.

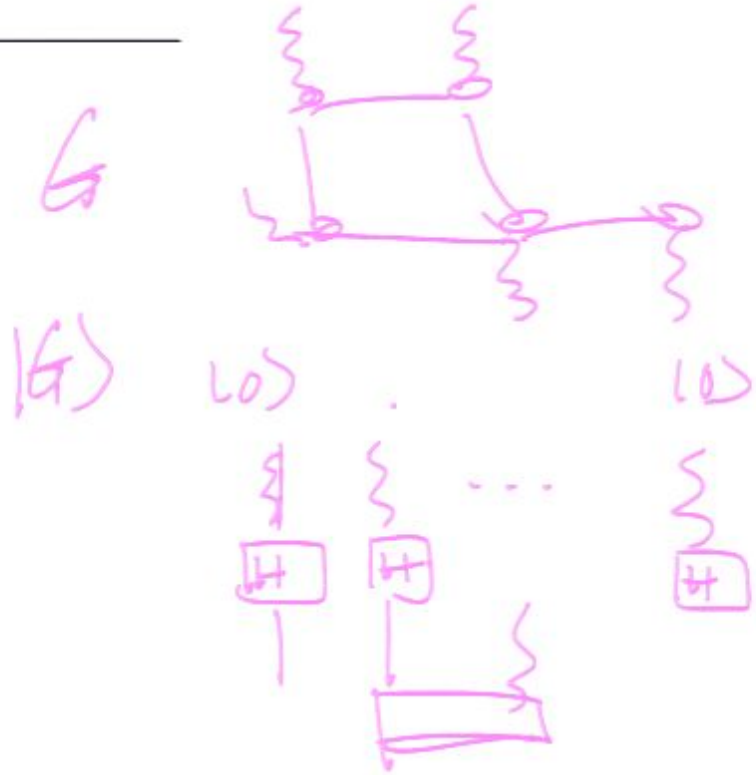
Theorem [Raussendorf and Briegel '01] For any quantum circuit, there exists a one-way quantum computation on the grid state of

Graph states [Briegel and Raussendorf '01]

Given an undirected simple graph $G = \langle V, E \rangle$, the graph state $|G\rangle$ is

$$|G\rangle := \frac{1}{2^{|V|/2}} \sum_{V' \subseteq V} (-1)^{|E(V')|} |V'\rangle,$$

where $E(V') \subseteq E$ are edges connecting vertices in V'.



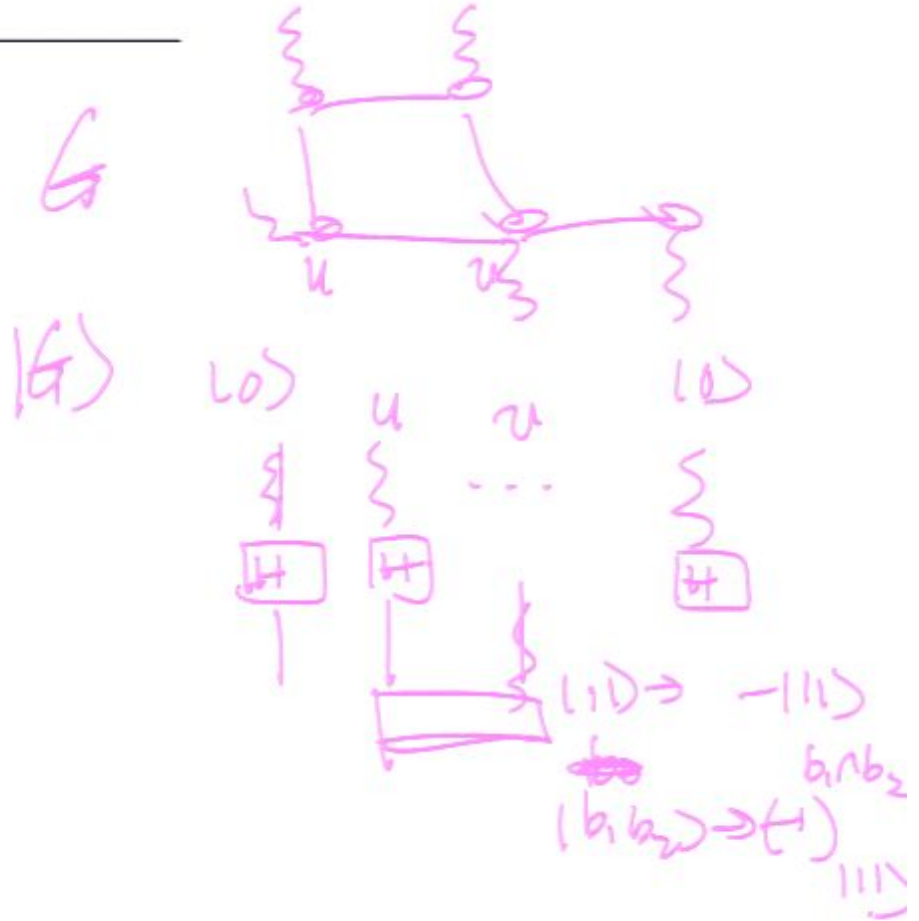
Theorem [Raussendorf and Briegel '01] For any quantum circuit, there exists a one-way quantum computation on the grid state of about the same size that outputs the same final

Graph states [Briegel and Raussendorf '01]

Given an undirected simple graph $G = \langle V, E \rangle$, the graph state $|G\rangle$ is

$$|G\rangle := \frac{1}{2^{|V|\ 2}} \sum_{V' \subseteq V} (-1)^{|E(V')|} |V'\rangle,$$

where $E(V') \subseteq E$ are edges connecting vertices in V'.

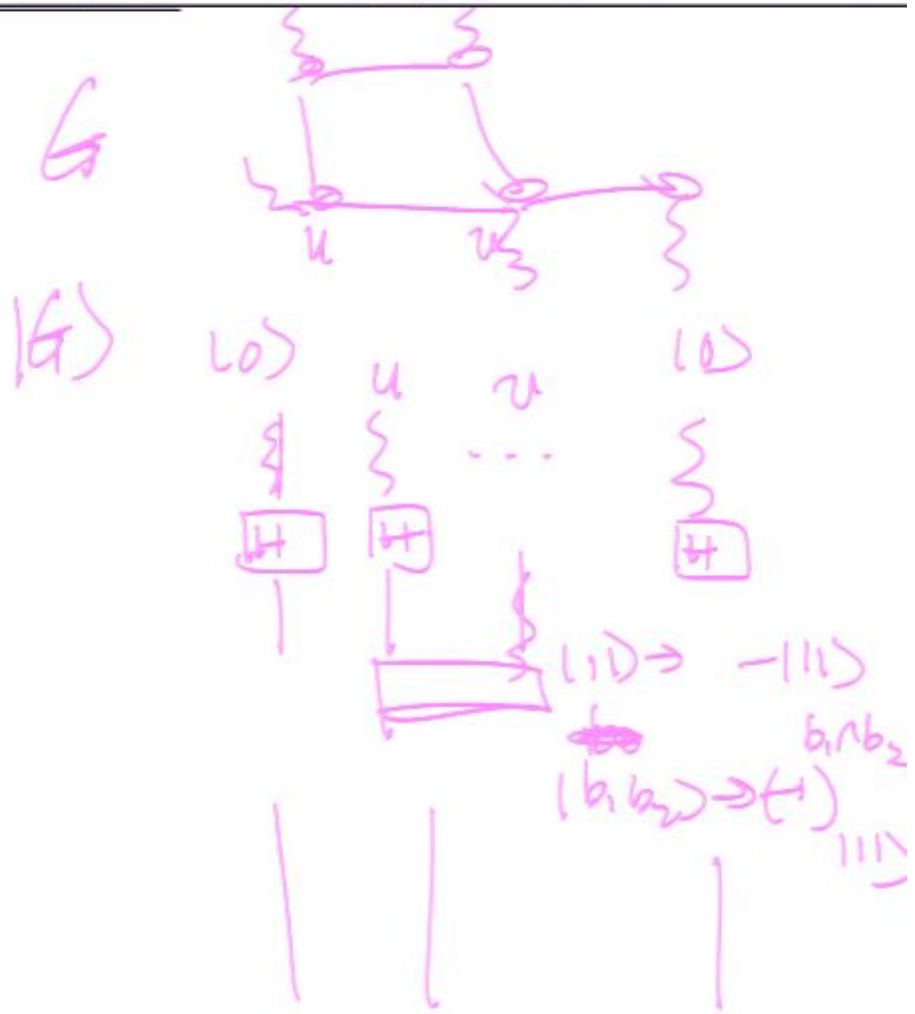


Theorem [Raussendorf and Briegel '01] For any quantum circuit, there exists a one-way quantum computation on the grid state of about the same size that outputs the same final

Given an undirected simple graph $G = \langle V, E \rangle$, the graph state $|G\rangle$ is

$$|G\rangle := \frac{1}{2^{|V|/2}} \sum_{V' \subseteq V} (-1)^{|E(V')|} |V'\rangle,$$

where $E(V') \subseteq E$ are edges connecting vertices in V'.

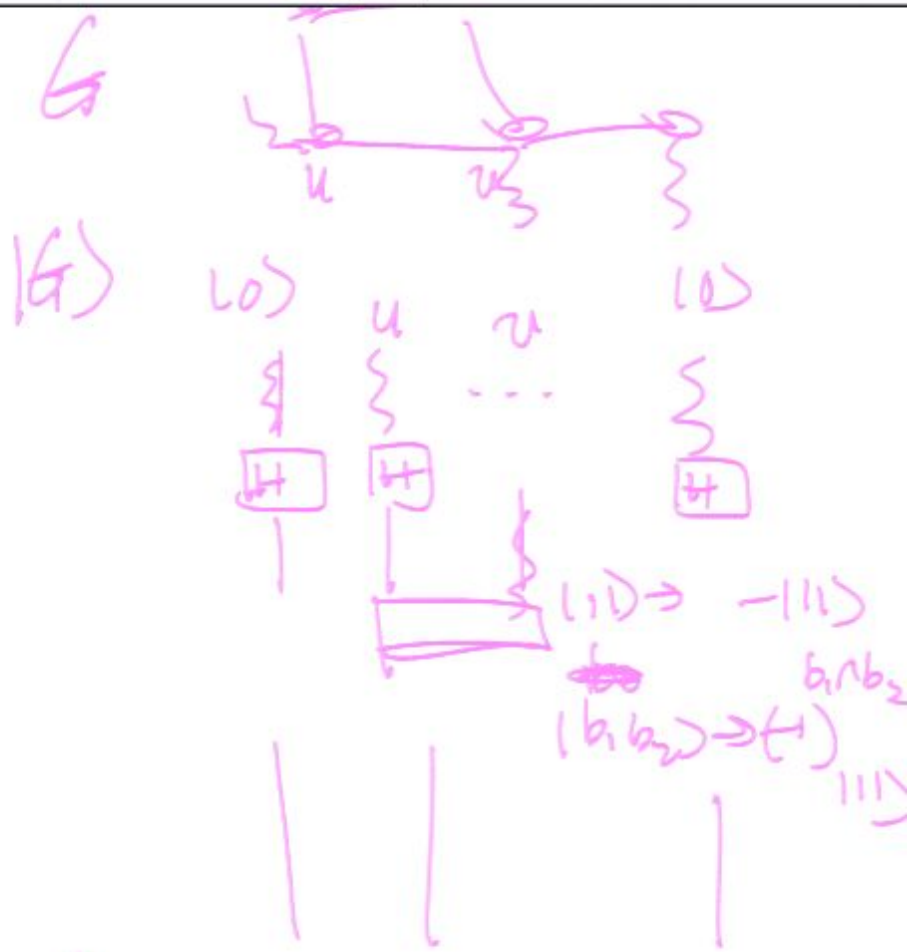


Theorem [Raussendorf and Briegel '01] For any quantum circuit, there exists a one-way quantum computation on the grid state of about the same size that outputs the same final state.

Given an undirected simple graph $G = \langle V, E \rangle$, the graph state $|G\rangle$ is

$$|G\rangle := \frac{1}{2^{|V|\ 2}} \sum_{V' \subseteq V} (-1)^{|E(V')|} |V'\rangle,$$

where $E(V') \subseteq E$ are edges connecting vertices in V'.

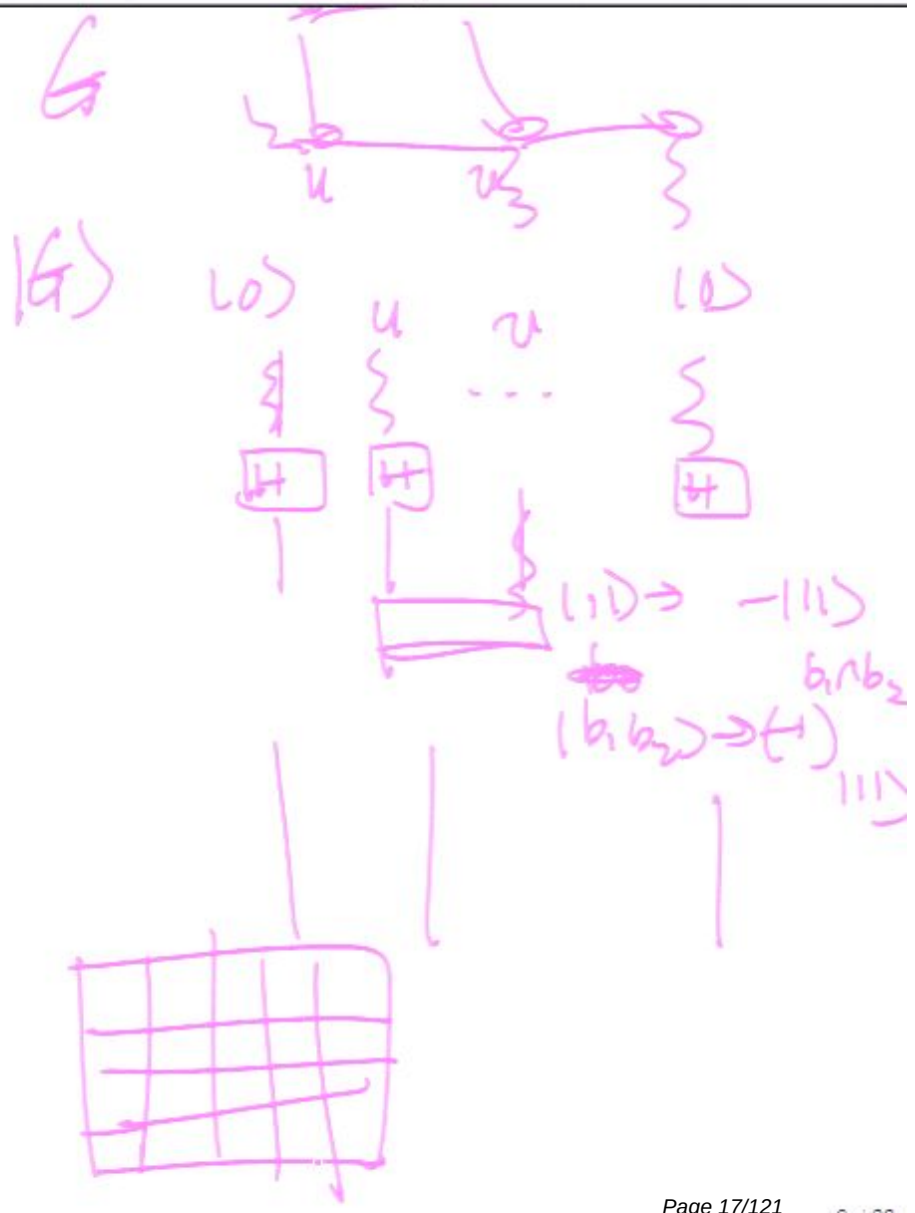


Theorem [Raussendorf and Briegel '01] For any quantum circuit, there exists a one-way quantum computation on the grid state of about the same size that outputs the same final state.

Given an undirected simple graph $G = \langle V, E \rangle$, the graph state $|G\rangle$ is

$$|G\rangle := \frac{1}{2^{|V|\ 2}} \sum_{V' \subseteq V} (-1)^{|E(V')|} |V'\rangle,$$

where $E(V') \subseteq E$ are edges connecting vertices in V'.



Theorem [Raussendorf and Briegel '01] For any quantum circuit, there exists a one-way quantum computation on the grid state of about the same size that outputs the same final state.

Which states are universal?

Definition. Let $|S_n\rangle$, $n = 0, 1, ...$ be a (family of) n-qubit quantum state. We say that $|S_n\rangle$ is *simulatable* if if for any one-way quantum computation on $|S_n\rangle$, there exists a $poly(n)$ time randomized algorithm that outputs an identical distribution of the measurement outcomes.

Main Theorem (Informally): If a graph G is close to a tree then $|G\rangle$ is simulatable.

Tensor and tensor network

A rank-k *tensor* in an m-dimension space

$$g = [g_{i_1.....i_k}]_{1 \le i_1,.....i_k \le m}.$$

is a m^k-dimensional array of complex numbers.

Variants include *Matrix Product States* and Projected Entangled Paired States.

Tensor contraction: generalizes matrix multiplication.

Contracting $g = [g_{i_1....i_s.u_1.....u_k}]$ and $h = [h_{i_1.....i_s.v_1.....v_\ell}]$ gives a rank-$k-\ell$ tensor $f = [f_{u_1.....u_k.v_1.....v_\ell}]$, where

$$f_{u_1.....u_k.v_1.....v_\ell} = \sum_{i_1.....i_s} g_{i_1.....i_s.u_1.....u_k} \cdot h_{i_1.....i_s.v_1.....v_\ell}.$$

Tensor and tensor network

A rank-k *tensor* in an m-dimension space

$$g = [g_{i_1.....i_k}]_{1 \le i_1.....i_k \le m}.$$

is a m^k-dimensional array of complex numbers.

Variants include *Matrix Product States* and Projected Entangled Paired States.

Tensor contraction: generalizes matrix multiplication.

Contracting $g = [g_{i_1.....i_s.u_1.....u_k}]$ and $h = [h_{i_1.....i_s.v_1.....v_\ell}]$ gives a rank-$k-\ell$ tensor $f = [f_{u_1.....u_k.v_1.....v_\ell}]$, where

$$f_{u_1.....u_k.v_1.....v_\ell} = \sum_{i_1.....i_s} g_{i_1.....i_s.u_1.....u_k} \cdot h_{i_1.....i_s.v_1.....v_\ell}.$$

Tensor and tensor network

A rank-k *tensor* in an m-dimension space

$$g = [g_{i_1.....i_k}]_{1 \le i_1.....i_k \le m}.$$

is a m^k-dimensional array of complex numbers.

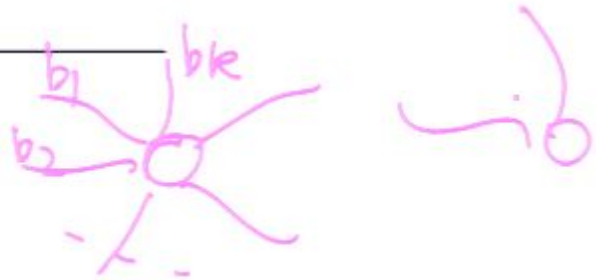
Variants include *Matrix Product States* and Projected Entangled Paired States.

Tensor contraction: generalizes matrix multiplication.

Contracting $g = [g_{i_1.....i_s.u_1.....u_k}]$ and $h = [h_{i_1.....i_s.v_1.....v_\ell}]$ gives a rank-$k-\ell$ tensor $f = [f_{u_1.....u_k.v_1.....v_\ell}]$, where

$$f_{u_1.....u_k.v_1.....v_\ell} = \sum_{i_1.....i_s} g_{i_1.....i_s.u_1.....u_k} \cdot h_{i_1.....i_s.v_1.....v_\ell}.$$

Tensor and tensor network



A rank-k *tensor* in an m-dimension space

$$g = [g_{i_1.....i_k}]_{1 \le i_1.....i_k \le m}.$$

is a m^k-dimensional array of complex numbers.

Variants include *Matrix Product States* and Projected Entangled Paired States.

Tensor contraction: generalizes matrix multiplication.

Contracting $g = [g_{i_1.....i_s.u_1.....u_k}]$ and $h = [h_{i_1.....i_s.v_1.....v_\ell}]$ gives a rank-$k-\ell$ tensor $f = [f_{u_1.....u_k.v_1.....v_\ell}]$, where

$$f_{u_1.....u_k.v_1.....v_\ell} = \sum_{i_1.....i_s} g_{i_1.....i_s.u_1.....u_k} \cdot h_{i_1.....i_s.v_1.....v_\ell}.$$

Tensor and tensor network

A rank-k *tensor* in an m-dimension space

$$g = [g_{i_1.....i_k}]_{1 \le i_1.....i_k \le m}.$$

is a m^k-dimensional array of complex numbers.

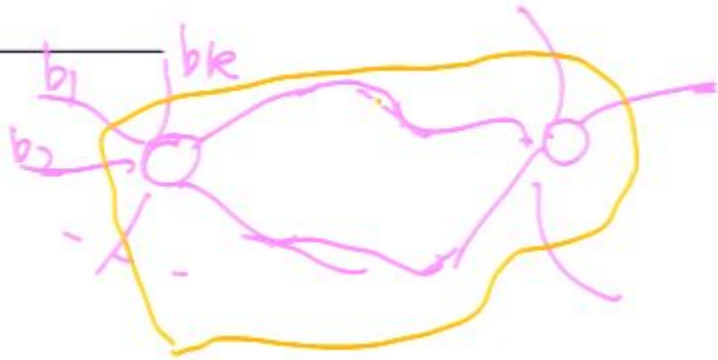
Variants include *Matrix Product States* and Projected Entangled Paired States.

Tensor contraction: generalizes matrix multiplication.

Contracting $g = [g_{i_1.....i_s.u_1.....u_k}]$ and $h = [h_{i_1.....i_s.v_1.....v_\ell}]$ gives a rank-$k-\ell$ tensor $f = [f_{u_1.....u_k.v_1.....v_\ell}]$, where

$$f_{u_1.....u_k.v_1.....v_\ell} = \sum_{i_1.....i_s} g_{i_1.....i_s.u_1.....u_k} \cdot h_{i_1.....i_s.v_1.....v_\ell}.$$

Tensor and tensor network



A rank-k *tensor* in an m-dimension space

$$g = [g_{i_1.....i_k}]_{1 \le i_1.....i_k \le m}.$$

is a m^k-dimensional array of complex numbers.

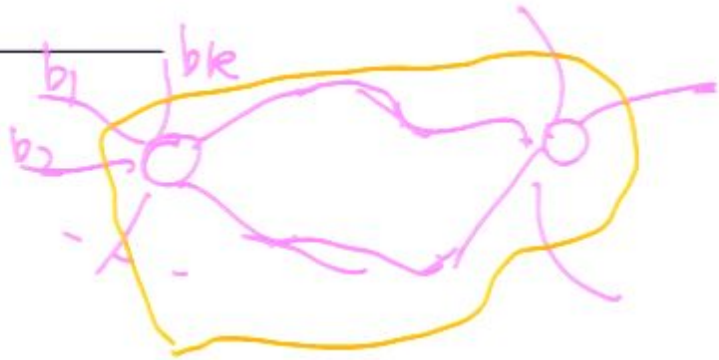
Variants include *Matrix Product States* and Projected Entangled Paired States.

Tensor contraction: generalizes matrix multiplication.

Contracting $g = [g_{i_1.....i_s.u_1.....u_k}]$ and $h = [h_{i_1.....i_s.v_1.....v_\ell}]$ gives a rank-$k-\ell$ tensor $f = [f_{u_1.....u_k.v_1.....v_\ell}]$, where

$$f_{u_1.....u_k.v_1.....v_\ell} = \sum_{i_1.....i_s} g_{i_1.....i_s.u_1.....u_k} \cdot h_{i_1.....i_s.v_1.....v_\ell}.$$

Tensor and tensor network



A rank-k *tensor* in an m-dimension space

$$g = [g_{i_1.....i_k}]_{1 \le i_1.....i_k \le m}.$$

is a m^k-dimensional array of complex numbers.

Variants include *Matrix Product States* and Projected Entangled Paired States.

Tensor contraction: generalizes matrix multiplication.

Contracting $g = [g_{i_1.....i_s.u_1.....u_k}]$ and $h = [h_{i_1.....i_s.v_1.....v_\ell}]$ gives a rank-$k-\ell$ tensor $f = [f_{u_1.....u_k.v_1.....v_\ell}]$, where

$$f_{u_1.....u_k.v_1.....v_\ell} = \sum_{i_1.....i_s} g_{i_1.....i_s.u_1.....u_k} \cdot h_{i_1.....i_s.v_1.....v_\ell}.$$

Tensor and tensor network

A rank-k *tensor* in an m-dimension space

$$g = [g_{i_1.....i_k}]_{1 \le i_1.....i_k \le m}.$$

is a m^k-dimensional array of complex numbers.

Variants include *Matrix Product States* and Projected Entangled Paired States.

Tensor contraction: generalizes matrix multiplication.

Contracting $g = [g_{i_1.....i_s.u_1.....u_k}]$ and $h = [h_{i_1.....i_s.v_1.....v_\ell}]$ gives a rank-$k-\ell$ tensor $f = [f_{u_1.....u_k.v_1.....v_\ell}]$, where

$$f_{u_1.....u_k.v_1.....v_\ell} = \sum_{i_1.....i_s} g_{i_1.....i_s.u_1.....u_k} \cdot h_{i_1.....i_s.v_1.....v_\ell}.$$

Simulation through tensor contraction [Vidal '03; Verstraete et al. '04, Markov and Shi '05]

Given a quantum circuit, an initial (product) state, and a (product) measurement operator

- Attach the tensors for the input state to the input wires.
- Attach the tensors for the measurement operator to the output wires.
- Contract the whole tensor according to a sequence of wires $\pi : e_1, e_2, \ldots, e_T$.
  - For each contraction, write down the entries of the new tensor.
  - Resulting in a degree sequence $d_1, d_2, \ldots, d_T = 0$.
  - Time (and space used): $T \exp(O(\Delta(\pi)))$, where

Simulation through tensor contraction [Vidal '03: Verstraete et al. '04, Markov and Shi '05]

Given a quantum circuit, an initial (product) state, and a (product) measurement operator

- Attach the tensors for the input state to the input wires.
- Attach the tensors for the measurement operator to the output wires.
- Contract the whole tensor according to a sequence of wires $\pi : e_1, e_2, \ldots, e_T$.
  - For each contraction, write down the entries of the new tensor.
  - Resulting in a degree sequence $d_1, d_2, \ldots, d_T = 0$.
  - Time (and space used): $T \exp(O(\Delta(\pi)))$, where $\Delta(\pi) = \max\{d_i : i = 1 \ldots T\}$.

Simulation through tensor contraction [Vidal '03: Verstraete et al. '04, Markov and Shi '05]

Given a quantum circuit, an initial (product) state, and a (product) measurement operator

- Attach the tensors for the input state to the input wires.
- Attach the tensors for the measurement operator to the output wires.
- Contract the whole tensor according to a sequence of wires $\pi : e_1, e_2, \ldots, e_T$.
  - For each contraction, write down the entries of the new tensor.
  - Resulting in a degree sequence $d_1, d_2, \ldots, d_T = 0$.
  - Time (and space used): $T \exp(O(\Delta(\pi)))$, where $\Delta(\pi) = \max\{d_i : i = 1 \ldots T\}$.

Simulation through tensor contraction [Vidal '03: Verstraete et al. '04, Markov and Shi '05]

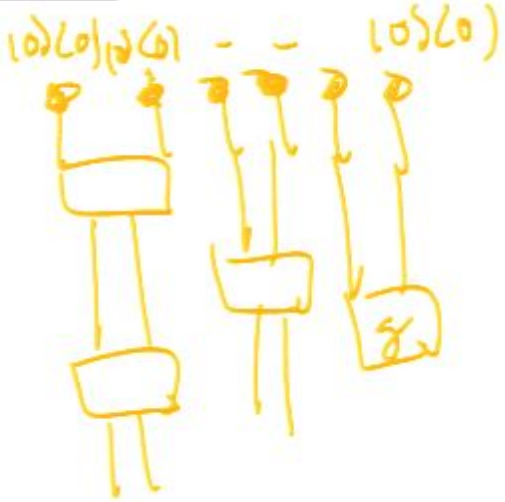
Given a quantum circuit, an initial (product) state, and a (product) measurement operator

- Attach the tensors for the input state to the input wires.
- Attach the tensors for the measurement operator to the output wires.
- Contract the whole tensor according to a sequence of wires $\pi : e_1, e_2, ..., e_T$.
  - For each contraction, write down the entries of the new tensor.
  - Resulting in a degree sequence $d_1, d_2, ..., d_T = 0$.
  - Time (and space used): $T \exp(O(\Delta(\pi)))$, where $\Delta(\pi) = \max\{d_i : i = 1...T\}$.

Simulation through tensor contraction [Vidal '03: Verstraete et al. '04, Markov and Shi '05]

Given a quantum circuit, an initial (product) state, and a (product) measurement operator

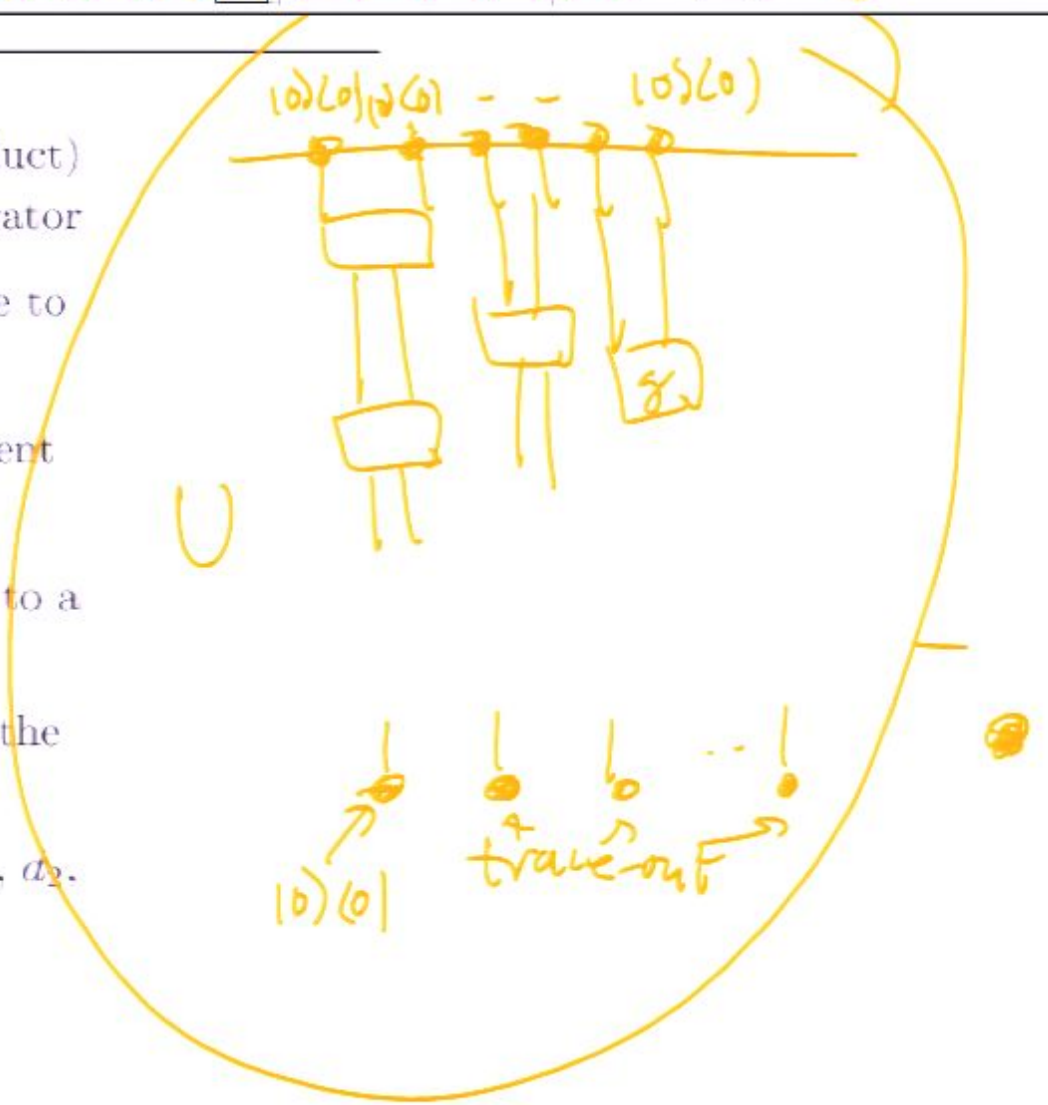
- Attach the tensors for the input state to the input wires.
- Attach the tensors for the measurement operator to the output wires.
- Contract the whole tensor according to a sequence of wires $\pi : e_1, e_2, \ldots, e_T$.
  - For each contraction, write down the entries of the new tensor.
  - Resulting in a degree sequence $d_1, d_2, \ldots, d_T = 0$.
  - Time (and space used): $T \exp(O(\Delta(\pi)))$, where $\Delta(\pi) = \max\{d_i : i = 1 \ldots T\}$.



et al. '04, Markov and Shi '05]

Given a quantum circuit, an initial (product) state, and a (product) measurement operator

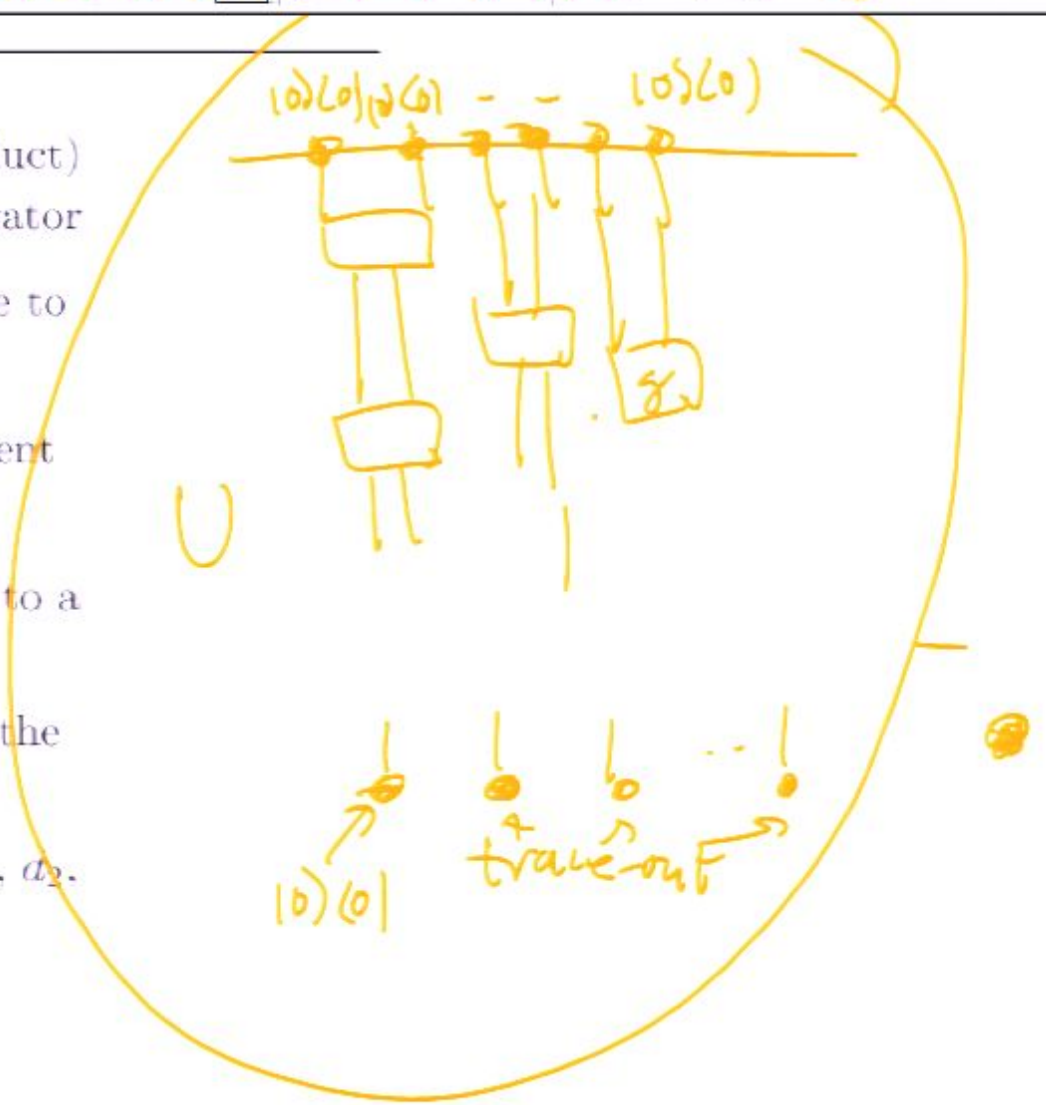
- Attach the tensors for the input state to the input wires.
- Attach the tensors for the measurement operator to the output wires.
- Contract the whole tensor according to a sequence of wires $\pi : e_1, e_2, \ldots, e_T$.
  - For each contraction, write down the entries of the new tensor.
  - Resulting in a degree sequence $d_1, d_2, \ldots, d_T = 0$.
  - Time (and space used): $T \exp(O(\Delta(\pi)))$, where $\Delta(\pi) = \max\{d_i : i = 1 \ldots T\}$.



et al. '04, Markov and Shi '05]

Given a quantum circuit, an initial (product) state, and a (product) measurement operator

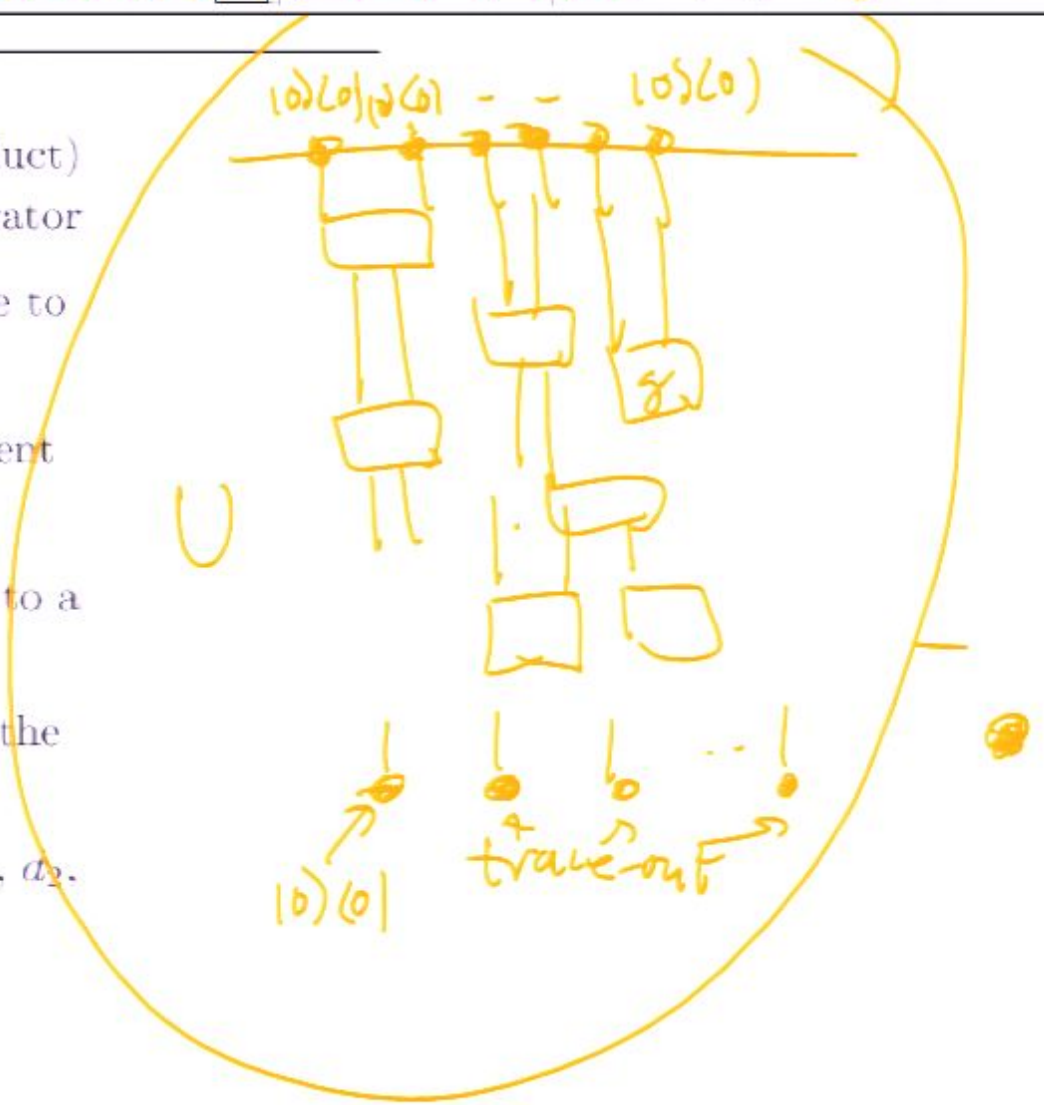
- Attach the tensors for the input state to the input wires.
- Attach the tensors for the measurement operator to the output wires.
- Contract the whole tensor according to a sequence of wires $\pi : e_1, e_2, \ldots, e_T$.
  - For each contraction, write down the entries of the new tensor.
  - Resulting in a degree sequence $d_1, d_2, \ldots, d_T = 0$.
  - Time (and space used): $T \exp(O(\Delta(\pi)))$, where $\Delta(\pi) = \max\{d_i : i = 1 \ldots T\}$.



et al. '04, Markov and Shi '05]

Given a quantum circuit, an initial (product) state, and a (product) measurement operator

- Attach the tensors for the input state to the input wires.
- Attach the tensors for the measurement operator to the output wires.
- Contract the whole tensor according to a sequence of wires $\pi : e_1, e_2, \ldots, e_T$.
  - For each contraction, write down the entries of the new tensor.
  - Resulting in a degree sequence $d_1, d_2, \ldots, d_T = 0$.
  - Time (and space used): $T \exp(O(\Delta(\pi)))$, where $\Delta(\pi) = \max\{d_i : i = 1 \ldots T\}$.



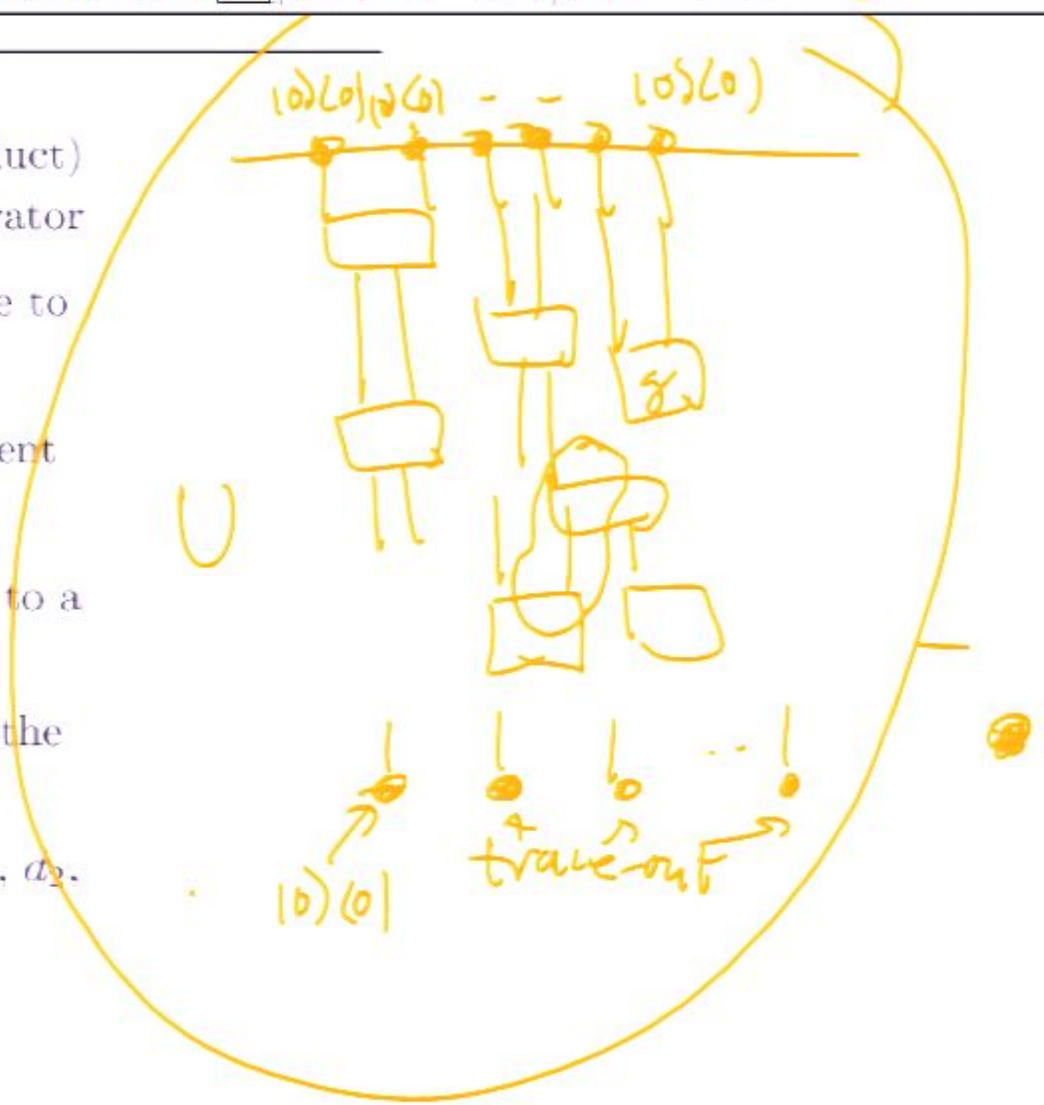
expansion_talk-a - Windows Journal

Edit View Insert Actions Tools Help

et al. '04, Markov and Shi '05]

Given a quantum circuit, an initial (product) state, and a (product) measurement operator

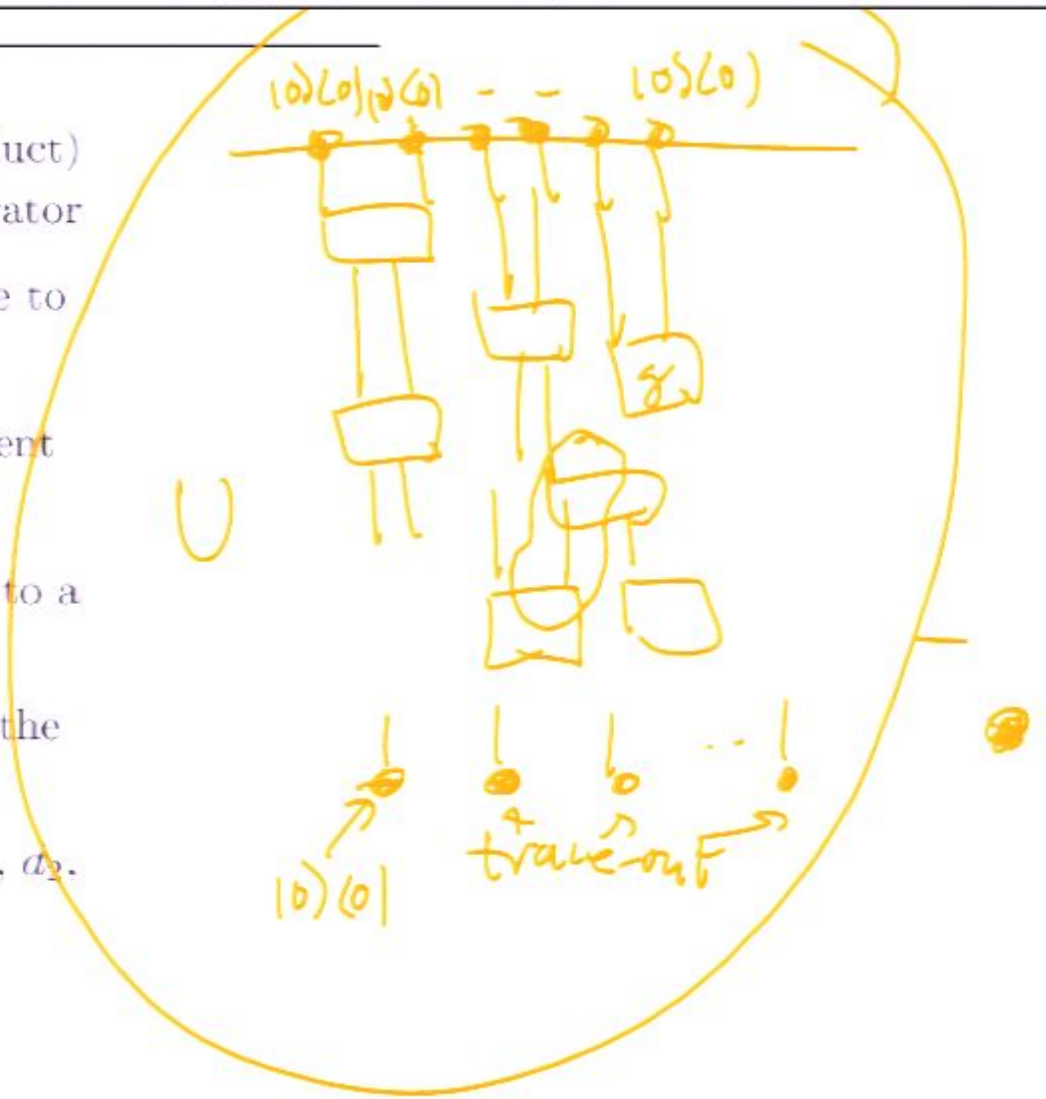
- Attach the tensors for the input state to the input wires.
- Attach the tensors for the measurement operator to the output wires.
- Contract the whole tensor according to a sequence of wires $\pi : e_1, e_2, \ldots, e_T$.
  - For each contraction, write down the entries of the new tensor.
  - Resulting in a degree sequence $d_1, d_2, \ldots, d_T = 0$.
  - Time (and space used): $T \exp(O(\Delta(\pi)))$, where $\Delta(\pi) = \max\{d_i : i = 1 \ldots T\}$.



et al. '04, Markov and Shi '05]

Given a quantum circuit, an initial (product) state, and a (product) measurement operator

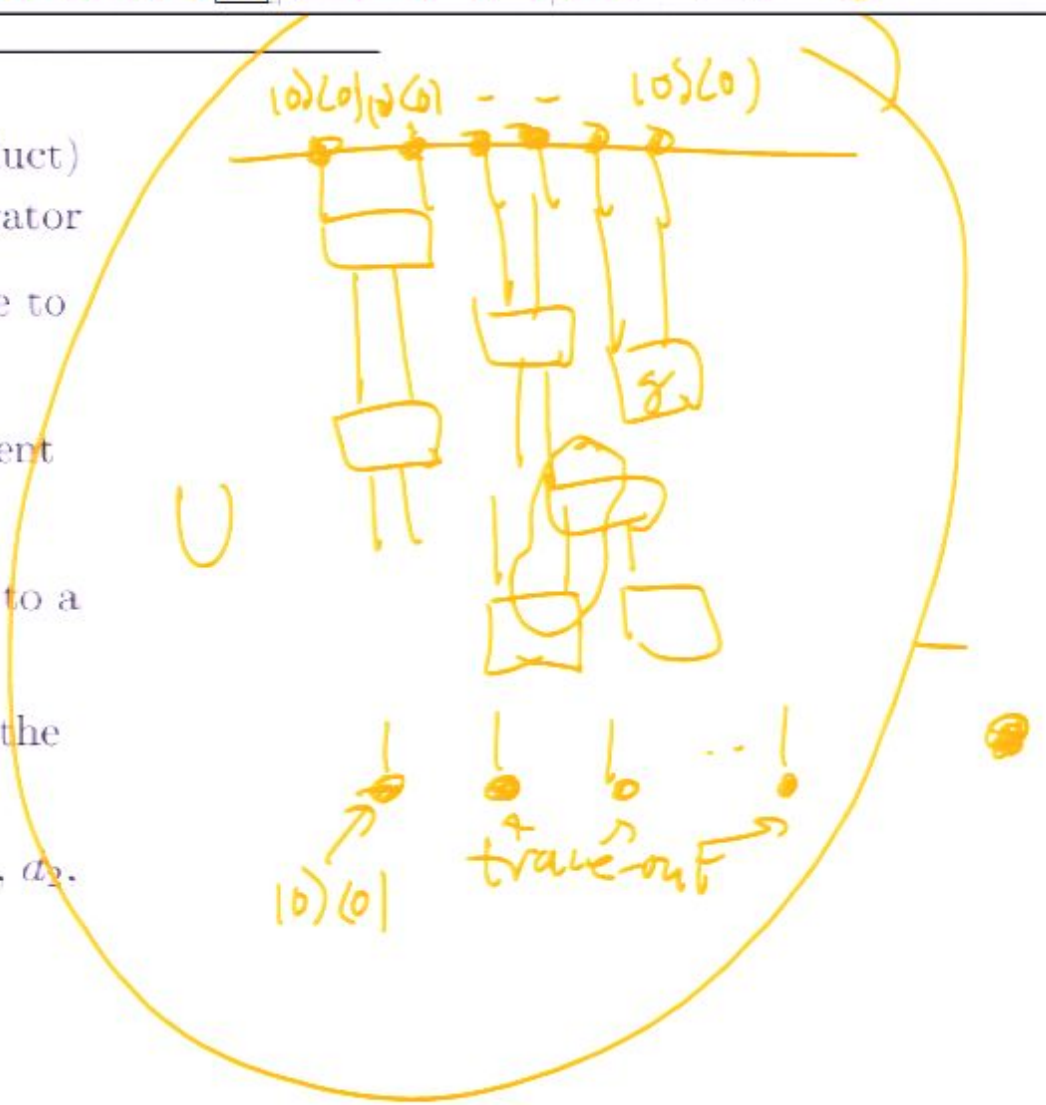
- Attach the tensors for the input state to the input wires.
- Attach the tensors for the measurement operator to the output wires.
- Contract the whole tensor according to a sequence of wires $\pi : e_1, e_2, \ldots, e_T$.
  - For each contraction, write down the entries of the new tensor.
  - Resulting in a degree sequence $d_1, d_2, \ldots, d_T = 0$.
  - Time (and space used): $T \exp(O(\Delta(\pi)))$, where $\Delta(\pi) = \max\{d_i : i = 1 \ldots T\}$.



et al. '04, Markov and Shi '05]

Given a quantum circuit, an initial (product) state, and a (product) measurement operator

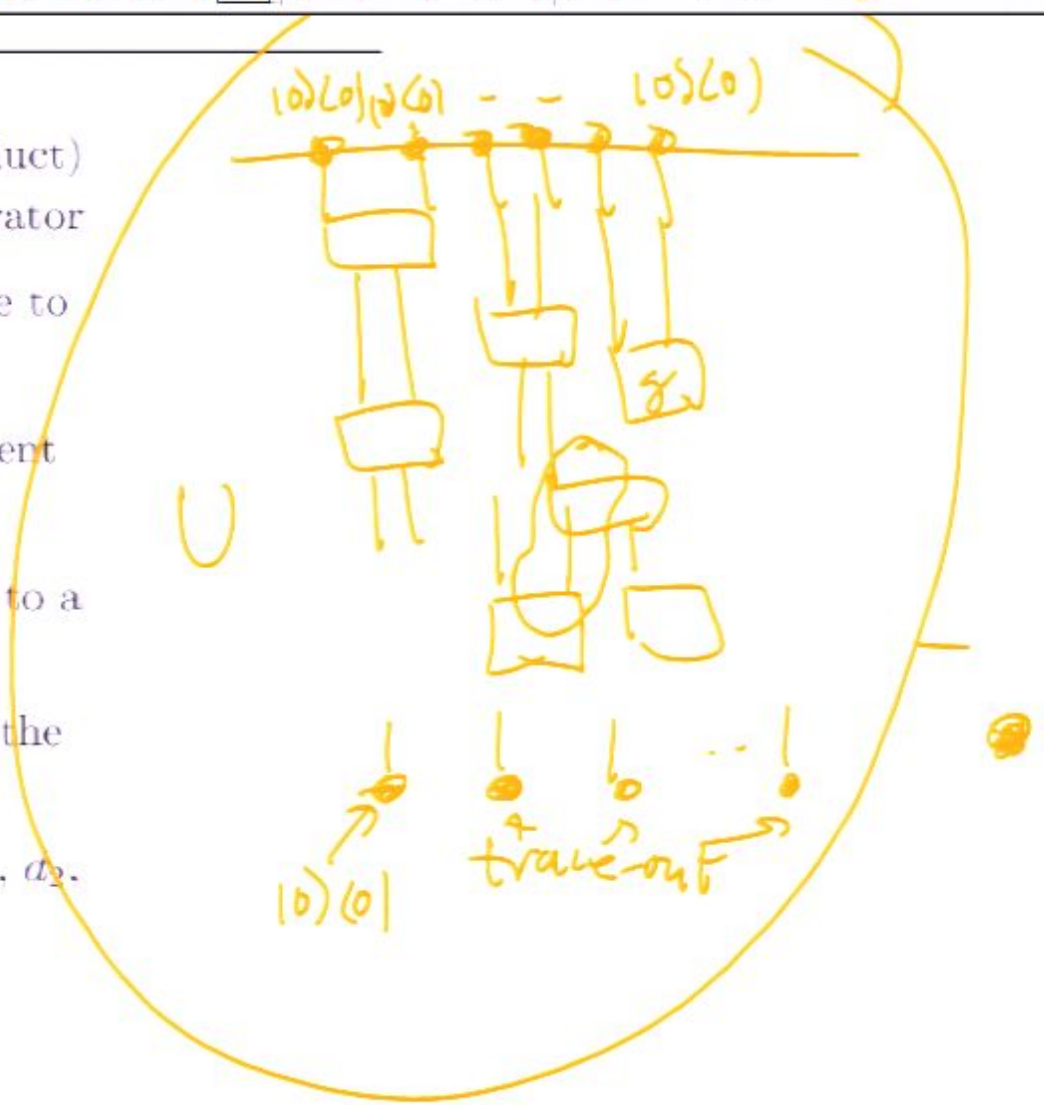
- Attach the tensors for the input state to the input wires.
- Attach the tensors for the measurement operator to the output wires.
- Contract the whole tensor according to a sequence of wires $\pi : e_1, e_2, \ldots, e_T$.
  - For each contraction, write down the entries of the new tensor.
  - Resulting in a degree sequence $d_1, d_2, \ldots, d_T = 0$.
  - Time (and space used): $T \exp(O(\Delta(\pi)))$, where $\Delta(\pi) = \max\{d_i : i = 1 \ldots T\}$.



et al. '04, Markov and Shi '05]

Given a quantum circuit, an initial (product) state, and a (product) measurement operator

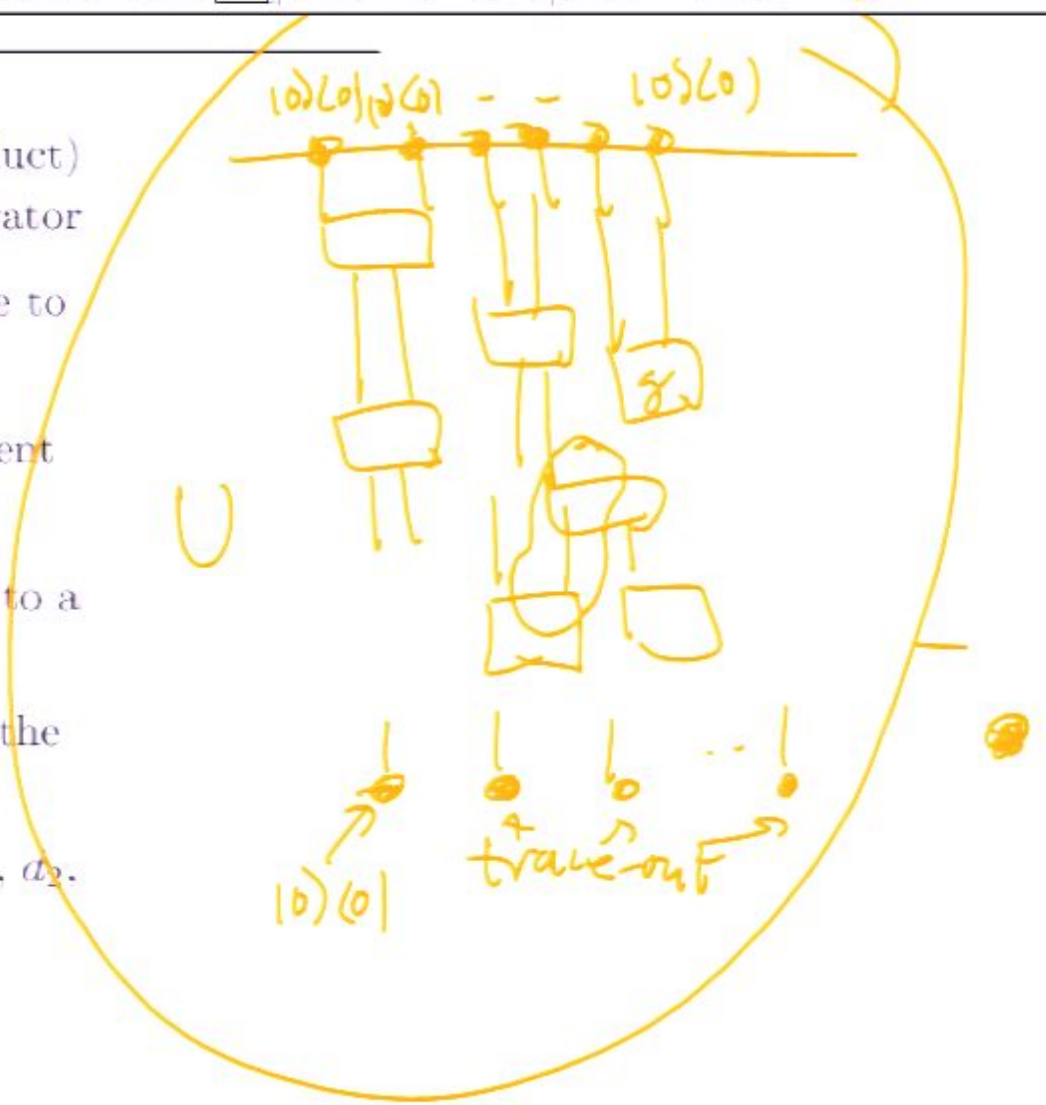
- Attach the tensors for the input state to the input wires.
- Attach the tensors for the measurement operator to the output wires.
- Contract the whole tensor according to a sequence of wires $\pi : e_1, e_2, \ldots, e_T$.
  - For each contraction, write down the entries of the new tensor.
  - Resulting in a degree sequence $d_1, d_2, \ldots, d_T = 0$.
  - Time (and space used): $T \exp(O(\Delta(\pi)))$, where $\Delta(\pi) = \max\{d_i : i = 1 \ldots T\}$.



et al. '04, Markov and Shi '05]

Given a quantum circuit, an initial (product) state, and a (product) measurement operator

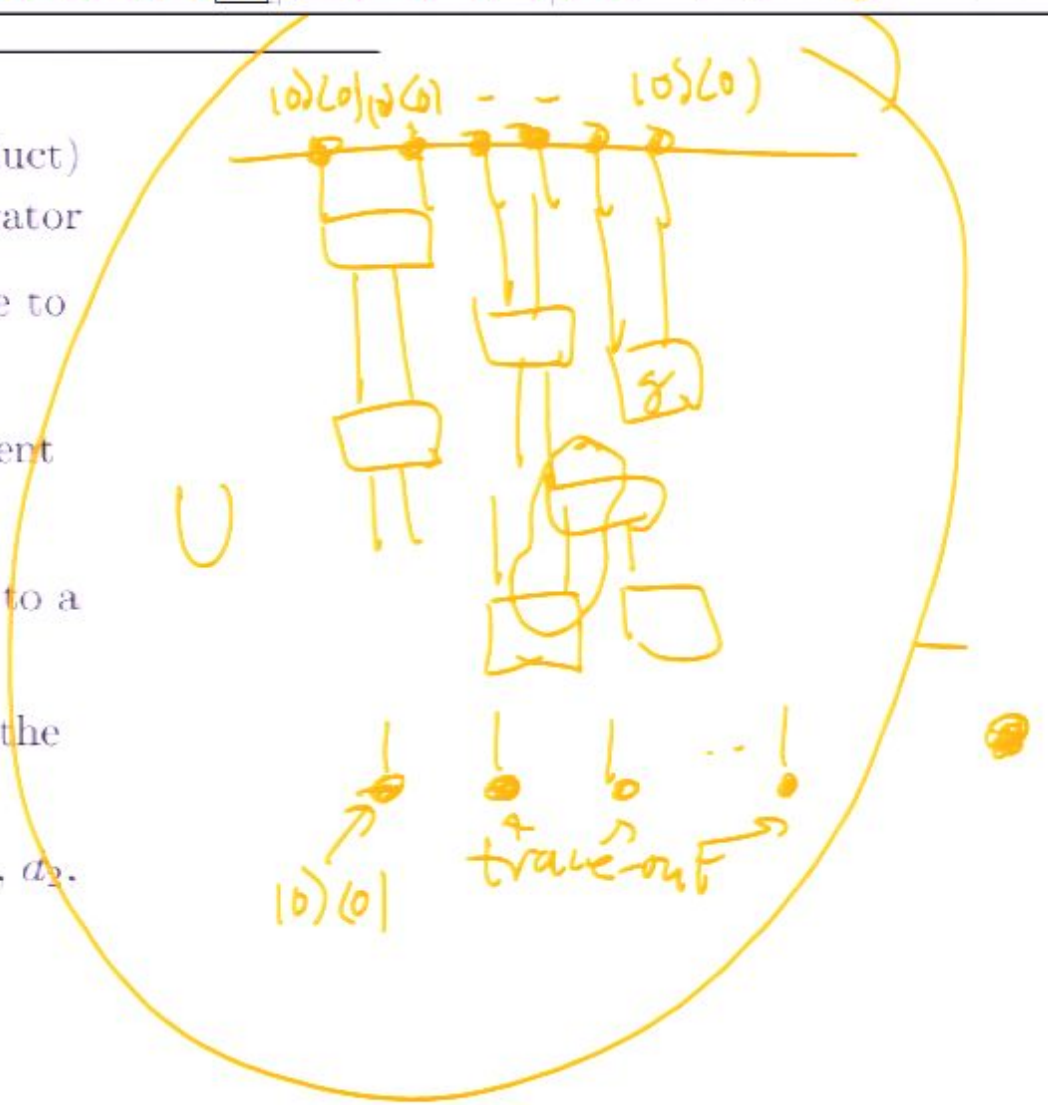
- Attach the tensors for the input state to the input wires.
- Attach the tensors for the measurement operator to the output wires.
- Contract the whole tensor according to a sequence of wires $\pi : e_1, e_2, \ldots, e_T$.
  - For each contraction, write down the entries of the new tensor.
  - Resulting in a degree sequence $d_1, d_2, \ldots, d_T = 0$.
  - Time (and space used): $T \exp(O(\Delta(\pi)))$, where $\Delta(\pi) = \max\{d_i : i = 1 \ldots T\}$.



et al. '04, Markov and Shi '05]

Given a quantum circuit, an initial (product) state, and a (product) measurement operator

- Attach the tensors for the input state to the input wires.
- Attach the tensors for the measurement operator to the output wires.
- Contract the whole tensor according to a sequence of wires $\pi : e_1, e_2, \ldots, e_T$.
  - For each contraction, write down the entries of the new tensor.
  - Resulting in a degree sequence $d_1, d_2, \ldots, d_T = 0$.
  - Time (and space used): $T \exp(O(\Delta(\pi)))$, where $\Delta(\pi) = \max\{d_i : i = 1 \ldots T\}$.



Contraction complexity and treewidth

Definition [Markov and Shi '05]. The *contraction complexity* of a graph G is

$$\mathrm{CC}(G) := \min_{\pi} \Delta(\pi).$$

Definition. The line graph of $G = \langle V, E \rangle$ is the graph G^* whose vertex set is E and $e_1 e_2$ is an edge if they are incident to a common vertex in G.

Proposition [Markov and Shi '05]. The contraction complexity of G is identical to the

Contraction complexity and treewidth

Definition [Markov and Shi '05]. The *contraction complexity* of a graph G is

$$\mathrm{CC}(G) := \min_{\pi} \Delta(\pi).$$

Definition. The line graph of $G = \langle V, E \rangle$ is the graph G^* whose vertex set is E and $e_1 e_2$ is an edge if they are incident to a common vertex in G.

Proposition [Markov and Shi '05]. The contraction complexity of G is identical to the

Contraction complexity and treewidth

Definition [Markov and Shi '05]. The *contraction complexity* of a graph G is

$$\mathrm{CC}(G) := \min_{\pi} \Delta(\pi).$$

Definition. The line graph of $G = \langle V, E \rangle$ is the graph G^* whose vertex set is E and $e_1 e_2$ is an edge if they are incident to a common vertex in G.

Proposition [Markov and Shi '05]. The contraction complexity of G is identical to the

Contraction complexity and treewidth

Definition [Markov and Shi '05]. The *contraction complexity* of a graph G is

$$\mathrm{CC}(G) := \min_{\pi} \Delta(\pi).$$

Definition. The line graph of $G = \langle V, E \rangle$ is the graph G^* whose vertex set is E and $e_1 e_2$ is an edge if they are incident to a common vertex in G.

Proposition [Markov and Shi '05]. The

Contraction complexity and treewidth

Definition [Markov and Shi '05]. The *contraction complexity* of a graph G is

$$\mathrm{CC}(G) := \min_{\pi} \Delta(\pi).$$

Definition. The line graph of $G = \langle V, E \rangle$ is the graph G^* whose vertex set is E and $e_1 e_2$ is an edge if they are incident to a common vertex in G.

Proposition [Markov and Shi '05]. The

Contraction complexity and treewidth

Definition [Markov and Shi '05]. The *contraction complexity* of a graph G is

$$\mathrm{CC}(G) := \min_{\pi} \Delta(\pi).$$

Definition. The line graph of $G = \langle V, E \rangle$ is the graph G^* whose vertex set is E and $e_1 e_2$ is an edge if they are incident to a common vertex in G.

Proposition [Markov and Shi '05]. The

Contraction complexity and treewidth

Definition [Markov and Shi '05]. The *contraction complexity* of a graph G is

$$\mathrm{CC}(G) := \min_{\pi} \Delta(\pi).$$

Definition. The line graph of $G = \langle V, E \rangle$ is the graph G^* whose vertex set is E and $e_1 e_2$ is an edge if they are incident to a common vertex in G.

Proposition [Markov and Shi '05]. The contraction complexity of G is identical to the

Definition [Markov and Shi '05]. The *contraction complexity* of a graph G is

$$\mathrm{CC}(G) := \min_{\pi} \Delta(\pi).$$

Definition. The line graph of $G = \langle V, E \rangle$ is the graph G^* whose vertex set is E and $e_1 e_2$ is an edge if they are incident to a common vertex in G.

Proposition [Markov and Shi '05]. The contraction complexity of G is identical to the *treewidth* of G^*.

Treewidth

The treewidth of a graph G, $\mathrm{tw}(G)$, describes how close G is to a tree.

- If G is a (nonempty) tree, $\mathrm{tw}(G) = 1$.
- If G is a series parallel graph, $\mathrm{tw}(G) \leq 2$.
- If G is fully connected, $\mathrm{tw}(G) = |V(G)| - 1$.
- If G' is a *minor* of G, i.e., G' can be obtained from a subgraph of G through edge contractions, then $\mathrm{tw}(G') \leq \mathrm{tw}(G)$.

Treewidth

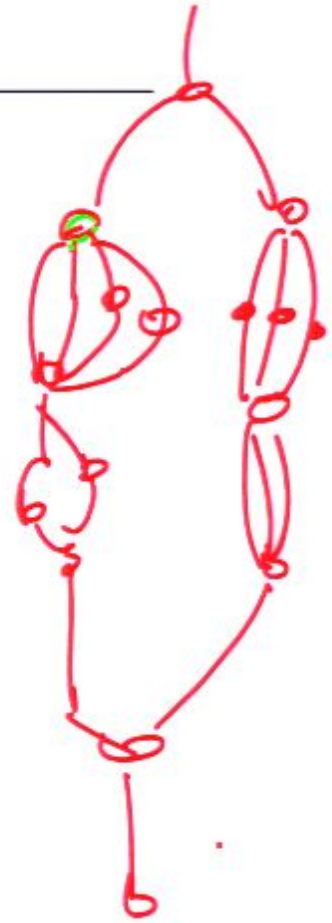
The treewidth of a graph G, $\mathrm{tw}(G)$, describes how close G is to a tree.

- If G is a (nonempty) tree, $\mathrm{tw}(G) = 1$.
- If G is a series parallel graph, $\mathrm{tw}(G) \leq 2$.
- If G is fully connected, $\mathrm{tw}(G) = |V(G)| - 1$.
- If G' is a *minor* of G, i.e., G' can be obtained from a subgraph of G through edge contractions, then $\mathrm{tw}(G') \leq \mathrm{tw}(G)$.

Treewidth

The treewidth of a graph G, $\mathrm{tw}(G)$, describes how close G is to a tree.

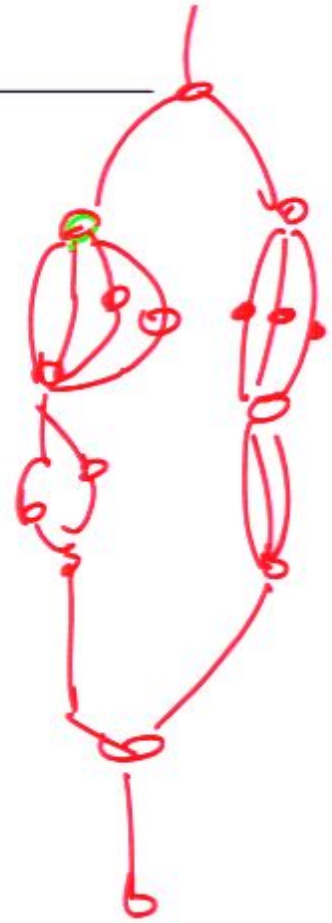
- If G is a (nonempty) tree, $\mathrm{tw}(G) = 1$.
- If G is a series parallel graph, $\mathrm{tw}(G) \leq 2$.
- If G is fully connected, $\mathrm{tw}(G) = |V(G)| - 1$.
- If G' is a *minor* of G, i.e., G' can be obtained from a subgraph of G through edge contractions, then $\mathrm{tw}(G') \leq \mathrm{tw}(G)$.



Treewidth

The treewidth of a graph G, $\mathrm{tw}(G)$, describes how close G is to a tree.

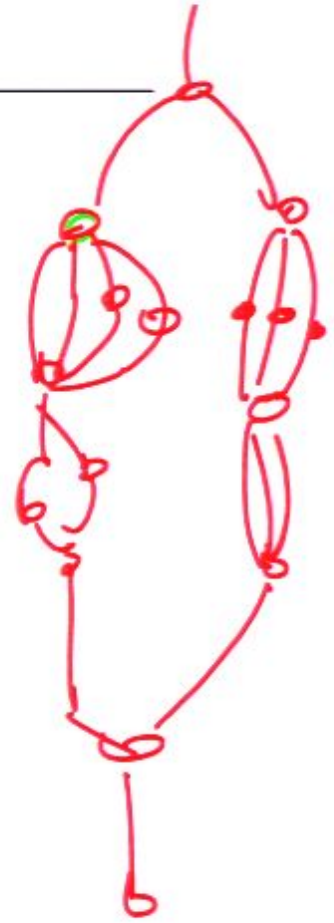
- If G is a (nonempty) tree, $\mathrm{tw}(G) = 1$.
- If G is a series parallel graph, $\mathrm{tw}(G) \leq 2$.
- If G is fully connected, $\mathrm{tw}(G) = |V(G)| - 1$.
- If G' is a *minor* of G, i.e., G' can be obtained from a subgraph of G through edge contractions, then $\mathrm{tw}(G') \leq \mathrm{tw}(G)$.



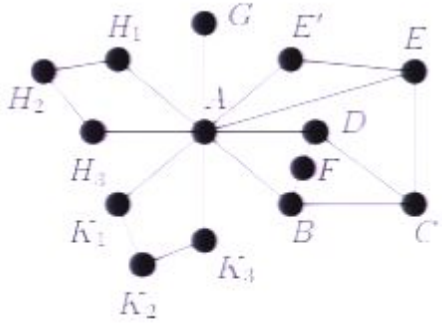
Treewidth

The treewidth of a graph G, $\mathrm{tw}(G)$, describes how close G is to a tree.

- If G is a (nonempty) tree, $\mathrm{tw}(G) = 1$.
- If G is a series parallel graph, $\mathrm{tw}(G) \leq 2$.
- If G is fully connected, $\mathrm{tw}(G) = |V(G)| - 1$.
- If G' is a *minor* of G, i.e., G' can be obtained from a subgraph of G through edge contractions, then $\mathrm{tw}(G') \leq \mathrm{tw}(G)$.

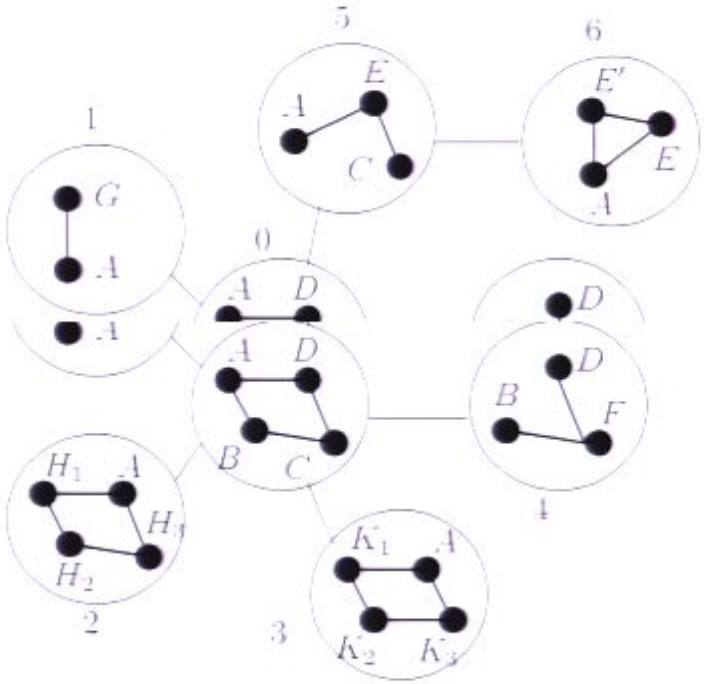


Definition [Robertson and Seymour '84]. A *tree decomposition* of G is a tree $\mathcal{T}$, together with a function that maps each vertex $w \in V(\mathcal{T})$ to a subset $B_w \subseteq V(G)$, called *bags*. In addition, the following conditions must hold.



(T1) $\bigcup_{v \in V(\mathcal{T})} B_v = V(G)$, i.e., each vertex must appear in at least one bag.

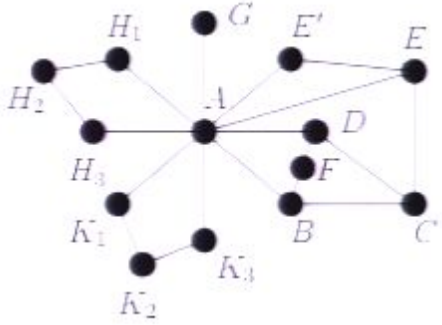
(T2) $\forall \{u, v\} \in E(G)$, $\exists w \in V(\mathcal{T})$, $\{u, v\} \subseteq B_w$, i.e., for each edge, at least one bag must contain both of its end vertices.



(T3) $\forall\, u \in V(G)$, the set of vertices $w \in V(\mathcal{T})$ with $u \in B_w$ form a connected subtree, i.e., all bags containing a given vertex must be connected in $\mathcal{T}$.

The *width* of a tree decomposition is $\max_{w \in V(\mathcal{T})} |B_w| - 1$. The *treewidth* of G is the minimum width over its tree decompositions.

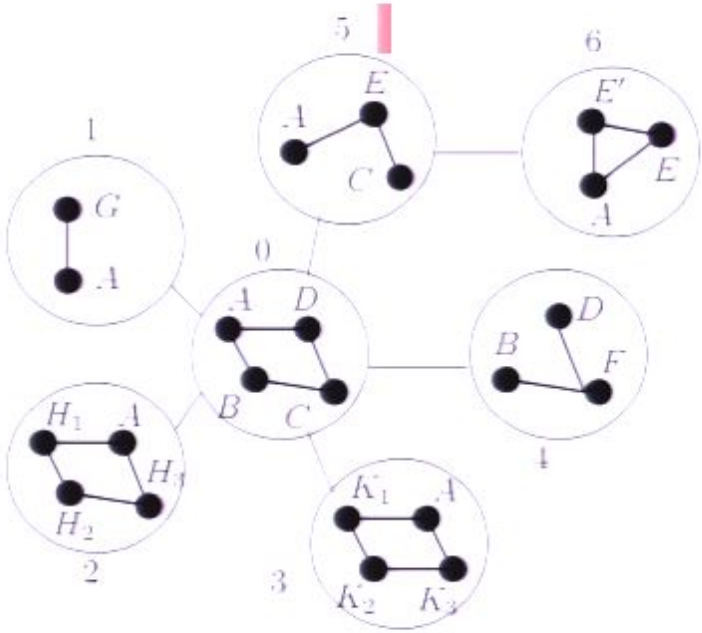
Definition [Robertson and Seymour '84]. A *tree decomposition* of G is a tree $\mathcal{T}$, together with a function that maps each vertex $w \in V(\mathcal{T})$ to a subset $B_w \subseteq V(G)$, called *bags*. In addition, the following conditions must hold.



(T1) $\bigcup_{v \in V(\mathcal{T})} B_v = V(G)$, i.e., each vertex must appear in at least one bag.

(T2) $\forall \{u, v\} \in E(G)$, $\exists w \in V(\mathcal{T})$, $\{u, v\} \subseteq B_w$, i.e., for each edge, at least one bag must contain both of its end vertices.

(T3) $\forall u \in V(G)$, the set of vertices $w \in V(\mathcal{T})$ with $u \in B_w$ form a connected subtree, i.e., all bags containing a given vertex must be connected in $\mathcal{T}$.



The *width* of a tree decomposition is $\max_{w \in V(\mathcal{T})} |B_w| - 1$. The *treewidth* of G is the minimum width over its tree decompositions.

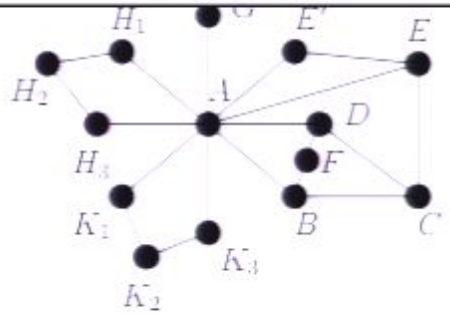
decomposition of G is a tree $\mathcal{T}$, together with a function that maps each vertex $w \in V(\mathcal{T})$ to a subset $B_w \subseteq V(G)$, called *bags*. In addition, the following conditions must hold.

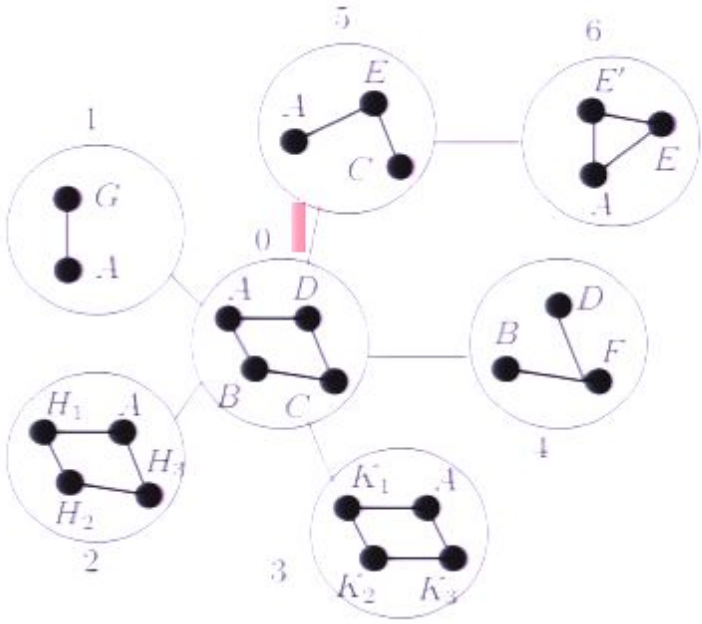
(T1) $\bigcup_{v \in V(\mathcal{T})} B_v = V(G)$, i.e., each vertex must appear in at least one bag.

(T2) $\forall \{u, v\} \in E(G), \exists w \in V(\mathcal{T})$. $\{u, v\} \subseteq B_w$, i.e., for each edge, at least one bag must contain both of its end vertices.

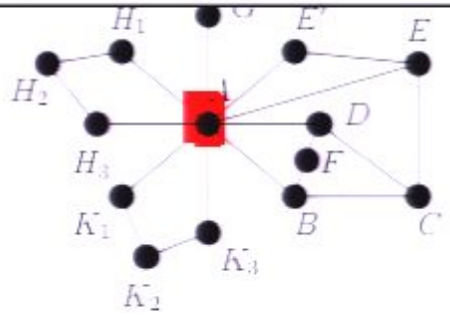
(T3) $\forall u \in V(G)$, the set of vertices $w \in V(\mathcal{T})$ with $u \in B_w$ form a connected subtree. i.e., all bags containing a given vertex must be connected in $\mathcal{T}$.

The *width* of a tree decomposition is $\max_{w \in V(\mathcal{T})} |B_w| - 1$. The *treewidth* of G is the minimum width over its tree decompositions.





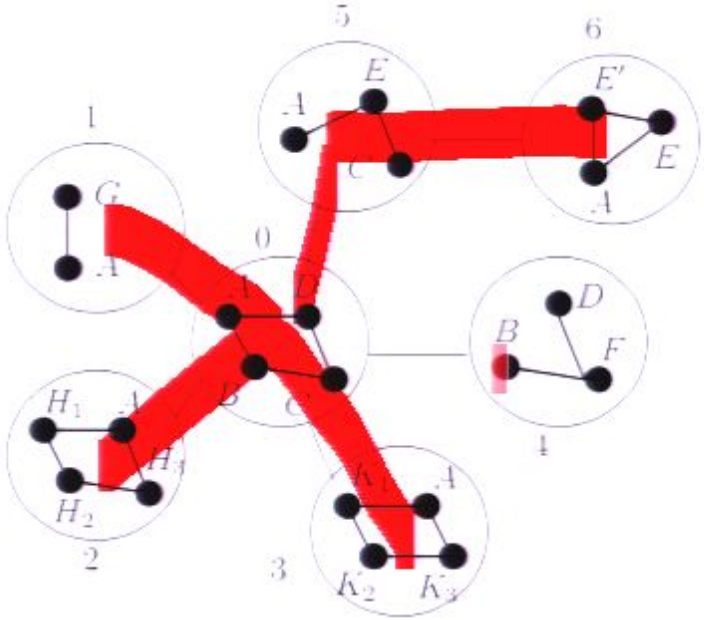
decomposition of G is a tree $\mathcal{T}$, together with a function that maps each vertex $w \in V(\mathcal{T})$ to a subset $B_w \subseteq V(G)$, called *bags*. In addition, the following conditions must hold.



(T1) $\bigcup_{v \in V(\mathcal{T})} B_v = V(G)$, i.e., each vertex must appear in at least one bag.

(T2) $\forall \{u, v\} \in E(G)$, $\exists w \in V(\mathcal{T})$. $\{u, v\} \subseteq B_w$, i.e., for each edge, at least one bag must contain both of its end vertices.

(T3) $\forall u \in V(G)$, the set of vertices $w \in V(\mathcal{T})$ with $u \in B_w$ form a connected subtree. i.e., all bags containing a given vertex must be connected in $\mathcal{T}$.



The *width* of a tree decomposition is $\max_{w \in V(\mathcal{T})} |B_w| - 1$. The *treewidth* of G is the minimum width over its tree decompositions.

Denote the maximum degree of G by $\Delta(G)$.

Proposition.

$(\mathrm{tw}(G)-1)\ \ 2 \leq \mathrm{tw}(G^*) \leq \Delta(G)(\mathrm{tw}(G)+1)-1).$

The treewidths of the graph and its line graph

Denote the maximum degree of G by $\Delta(G)$.

Proposition.

$(\mathrm{tw}(G)-1)\ 2 \leq \mathrm{tw}(G^*) \leq \Delta(G)(\mathrm{tw}(G)+1)-1)$.

The treewidths of the graph and its line graph

Denote the maximum degree of G by $\Delta(G)$.

Proposition.

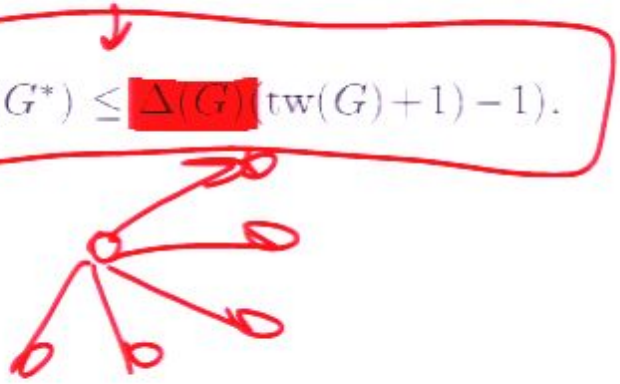
$(\mathrm{tw}(G)-1)\ 2 \leq \mathrm{tw}(G^*) \leq \Delta(G)(\mathrm{tw}(G)+1)-1)$.

The treewidths of the graph and its line graph

Denote the maximum degree of G by $\Delta(G)$.

Proposition.

$$(\mathrm{tw}(G)-1)\ 2 \leq \mathrm{tw}(G^*) \leq \Delta(G)(\mathrm{tw}(G)+1)-1).$$



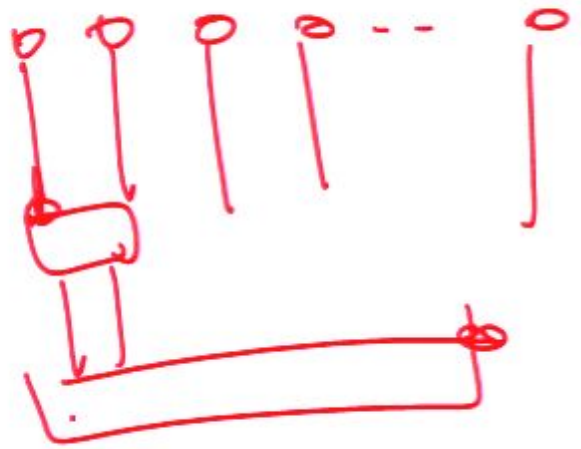
Back to the simulation of one-way computation

Fix a circuit C_G creating $|G\rangle$. A simulation algorithm:

- Select the next qubit to be measured according to the one-way algorithm.
- Attach the corresponding measurement element to the circuit.
- Compute the probability p_0 of outputting 0 through tensor contraction.
- Toss a coin with identical distribution.
- Attach the measurement element corresponding to the coin outcome b_1 to the circuit.
- Repeat till the end of the one-way algorithm.
  - The t'th contraction computes the probability of outputting $b_1 b_2 \cdots b_{t-1} 0$.
  - Should remember the prob. of

Back to the simulation of one-way computation

G

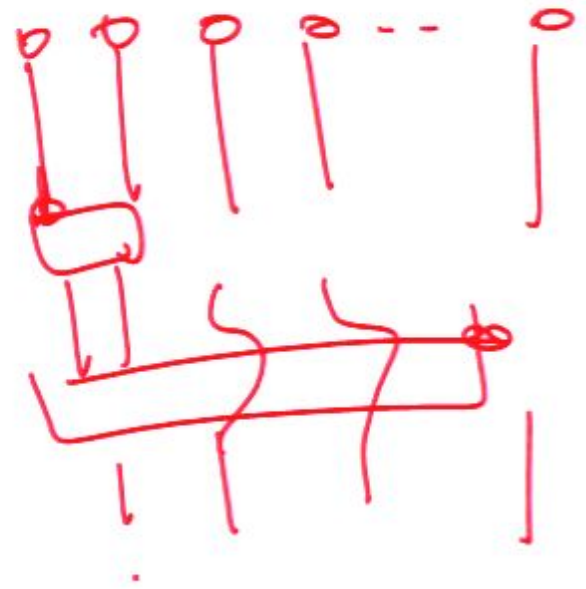


Fix a circuit C_G creating $|G\rangle$. A simulation algorithm:

- Select the next qubit to be measured according to the one-way algorithm.
- Attach the corresponding measurement element to the circuit.
- Compute the probability p_0 of outputting 0 through tensor contraction.
- Toss a coin with identical distribution.
- Attach the measurement element corresponding to the coin outcome b_1 to the circuit.
- Repeat till the end of the one-way algorithm.
  - The t'th contraction computes the

Back to the simulation of one-way computation

G



Fix a circuit C_G creating $|G\rangle$. A simulation algorithm:

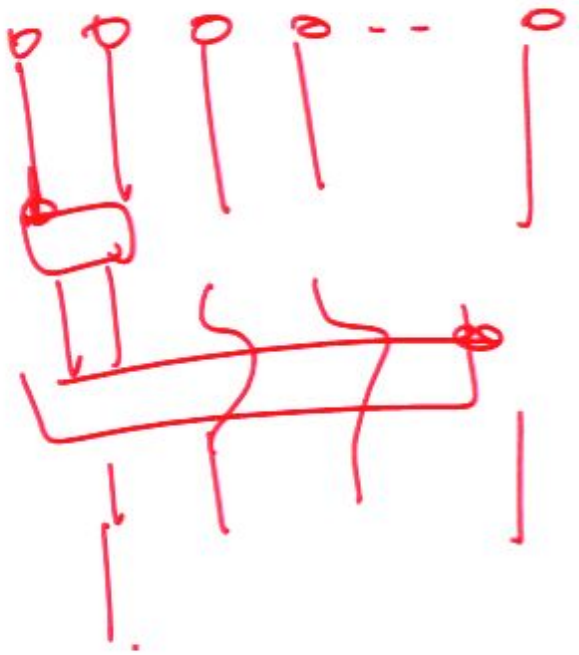
- Select the next qubit to be measured according to the one-way algorithm.
- Attach the corresponding measurement element to the circuit.
- Compute the probability p_0 of outputting 0 through tensor contraction.
- Toss a coin with identical distribution.
- Attach the measurement element corresponding to the coin outcome b_1 to the circuit.
- Repeat till the end of the one-way algorithm.
  - The t'th contraction computes the

Back to the simulation of one-way computation

G

Fix a circuit C_G creating $|G\rangle$. A simulation algorithm:

- Select the next qubit to be measured according to the one-way algorithm.
- Attach the corresponding measurement element to the circuit.
- Compute the probability p_0 of outputting 0 through tensor contraction.
- Toss a coin with identical distribution.
- Attach the measurement element corresponding to the coin outcome b_1 to the circuit.
- Repeat till the end of the one-way algorithm.
  - The t'th contraction computes the

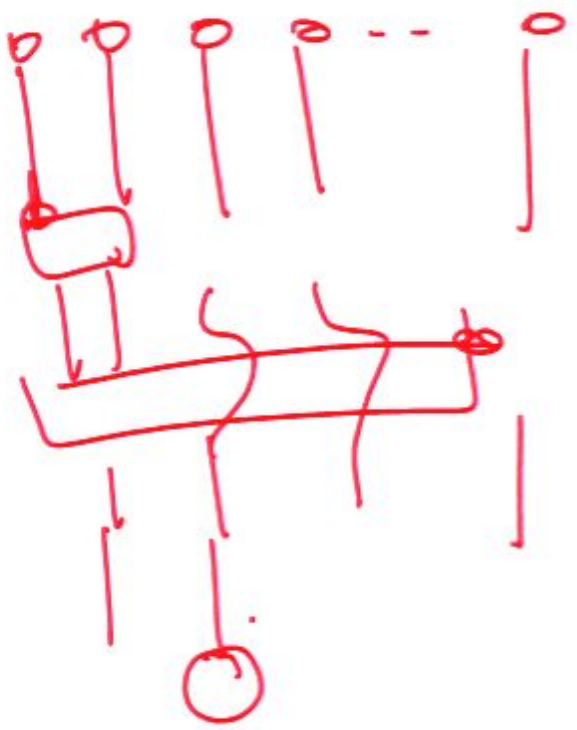


Back to the simulation of one-way computation

G

Fix a circuit C_G creating $|G\rangle$. A simulation algorithm:

- Select the next qubit to be measured according to the one-way algorithm.
- Attach the corresponding measurement element to the circuit.
- Compute the probability p_0 of outputting 0 through tensor contraction.
- Toss a coin with identical distribution.
- Attach the measurement element corresponding to the coin outcome b_1 to the circuit.
- Repeat till the end of the one-way algorithm.
  - The t'th contraction computes the

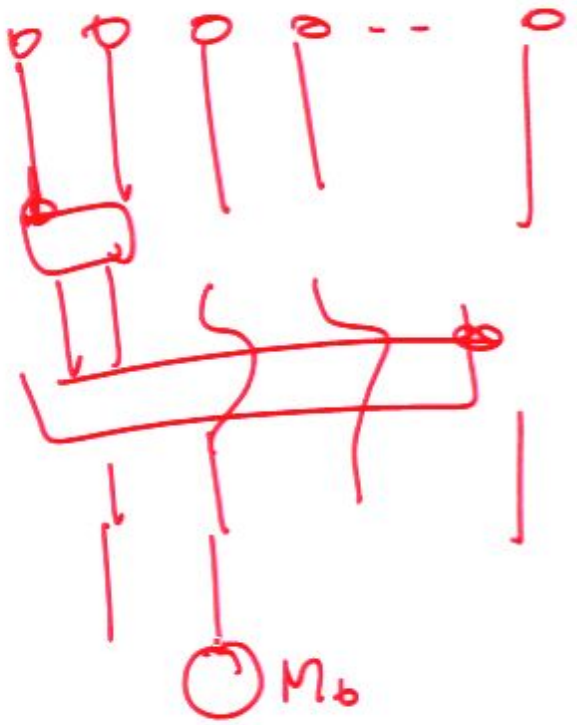


Back to the simulation of one-way computation

G

Fix a circuit C_G creating $|G\rangle$. A simulation algorithm:

- Select the next qubit to be measured according to the one-way algorithm.
- Attach the corresponding measurement element to the circuit.
- Compute the probability p_0 of outputting 0 through tensor contraction.
- Toss a coin with identical distribution.
- Attach the measurement element corresponding to the coin outcome b_1 to the circuit.
- Repeat till the end of the one-way algorithm.
  - The t'th contraction computes the

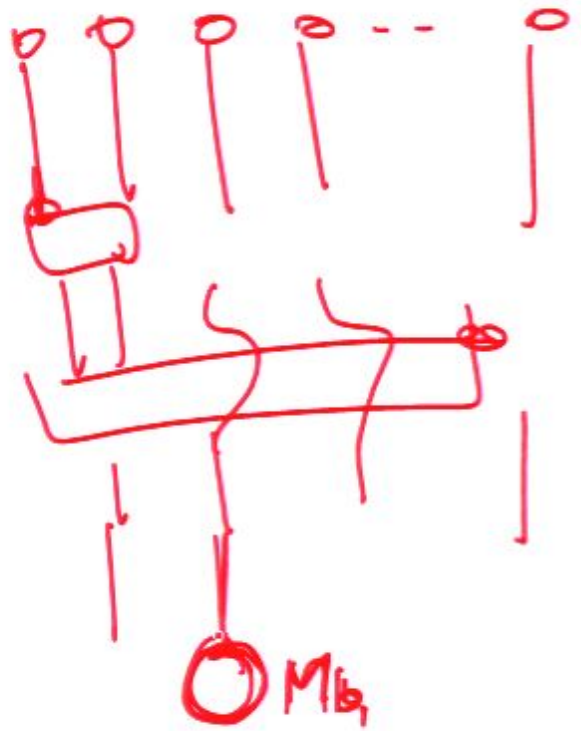


Back to the simulation of one-way computation

G

Fix a circuit C_G creating $|G\rangle$. A simulation algorithm:

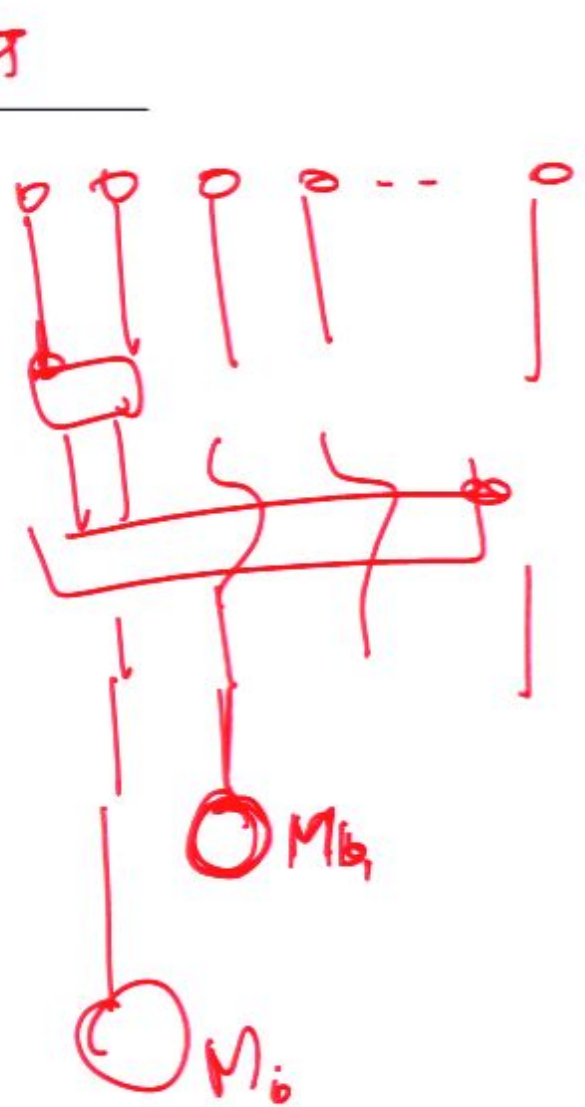
- Select the next qubit to be measured according to the one-way algorithm.
- Attach the corresponding measurement element to the circuit.
- Compute the probability p_0 of outputting 0 through tensor contraction.
- Toss a coin with identical distribution.
- Attach the measurement element corresponding to the coin outcome b_1 to the circuit.
- Repeat till the end of the one-way algorithm.
  - The t'th contraction computes the



Back to the simulation of one-way computation

Fix a circuit C_G creating $|G\rangle$. A simulation algorithm:

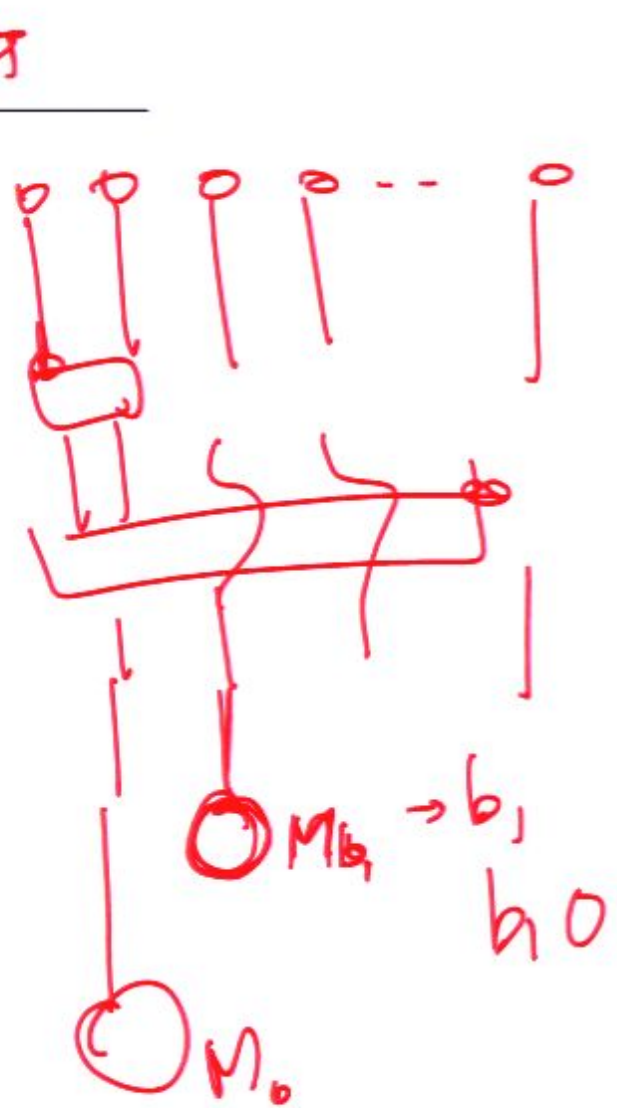
- Select the next qubit to be measured according to the one-way algorithm.
- Attach the corresponding measurement element to the circuit.
- Compute the probability p_0 of outputting 0 through tensor contraction.
- Toss a coin with identical distribution.
- Attach the measurement element corresponding to the coin outcome b_1 to the circuit.
- Repeat till the end of the one-way algorithm.
  - The t'th contraction computes the



Back to the simulation of one-way computation

Fix a circuit C_G creating $|G\rangle$. A simulation algorithm:

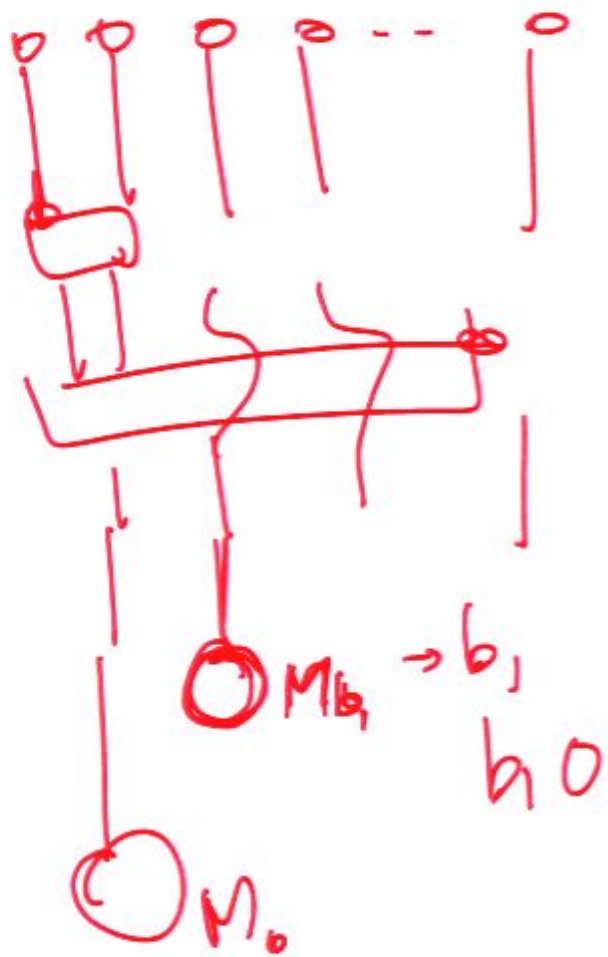
- Select the next qubit to be measured according to the one-way algorithm.
- Attach the corresponding measurement element to the circuit.
- Compute the probability p_0 of outputting 0 through tensor contraction.
- Toss a coin with identical distribution.
- Attach the measurement element corresponding to the coin outcome b_1 to the circuit.
- Repeat till the end of the one-way algorithm.
  - The t'th contraction computes the



Back to the simulation of one-way computation

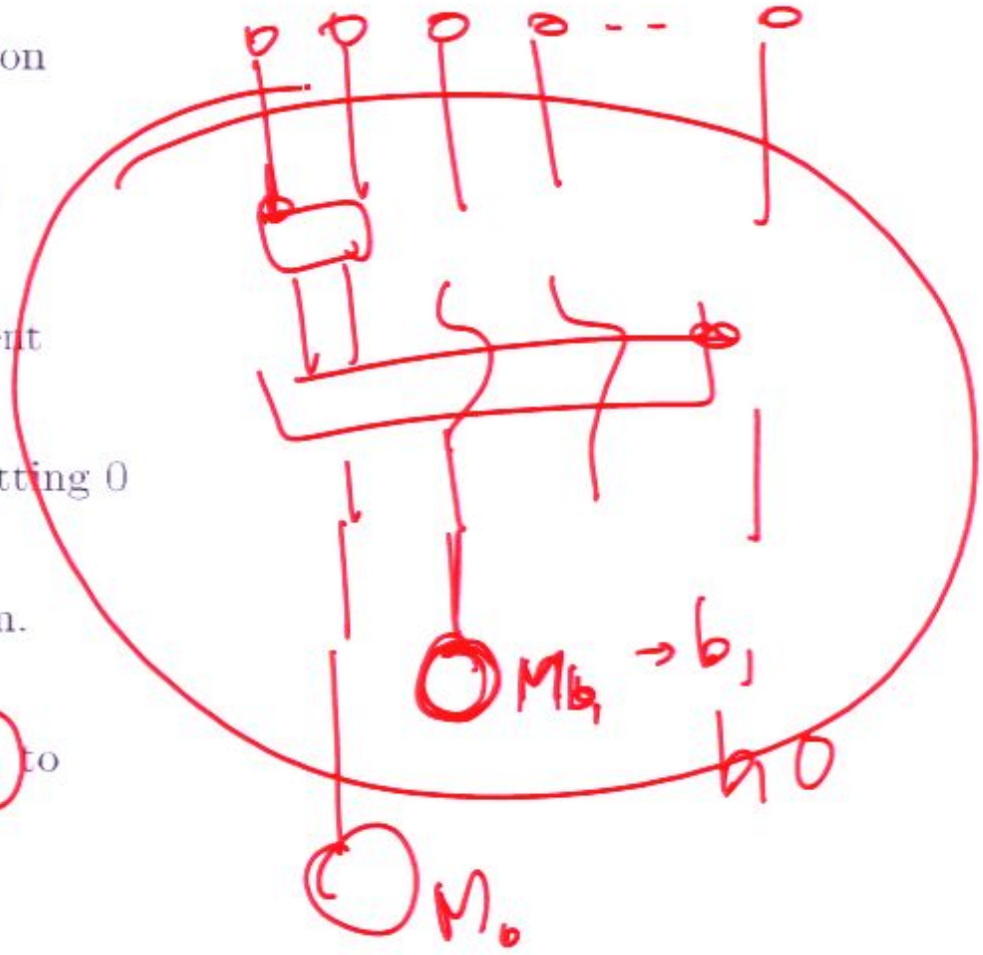
Fix a circuit C_G creating $|G\rangle$. A simulation algorithm:

- Select the next qubit to be measured according to the one-way algorithm.
- Attach the corresponding measurement element to the circuit.
- Compute the probability p_0 of outputting 0 through tensor contraction.
- Toss a coin with identical distribution.
- Attach the measurement element corresponding to the coin outcome b_1 to the circuit.
- Repeat till the end of the one-way algorithm.
  - The t'th contraction computes the



Fix a circuit C_G creating $|G\rangle$. A simulation algorithm:

- Select the next qubit to be measured according to the one-way algorithm.
- Attach the corresponding measurement element to the circuit.
- Compute the probability p_0 of outputting 0 through tensor contraction.
- Toss a coin with identical distribution.
- Attach the measurement element corresponding to the coin outcome b_1 to the circuit.
- Repeat till the end of the one-way algorithm.
  - The t'th contraction computes the probability of outputting $b_1 b_2 \cdots b_{t-1} 0$.
  - Should remember the prob. of



Back to the simulation of one-way computation

outputting $b_1 b_2 ... b_{t-1}$ to compute the conditional probability of outputting 0 for the current measurement.

Complexity: $O(T)\exp(\mathrm{CC}(C_G))$.

But $\mathrm{CC}(C_G) = O(\Delta(G)\mathrm{tw}(G))$.

When $\Delta(G)$ is large this bound is bad.

Example: a star

Back to the simulation of one-way computation

outputting $b_1 b_2 ... b_{t-1}$ to compute the conditional probability of outputting 0 for the current measurement.

Complexity: $O(T)\exp(\mathrm{CC}(C_G))$.

But $\mathrm{CC}(C_G) = O(\Delta(G)\mathrm{tw}(G))$.

When $\Delta(G)$ is large this bound is bad.

Example: a star

Back to the simulation of one-way computation

outputting $b_1 b_2 ... b_{t-1}$ to compute the conditional probability of outputting 0 for the current measurement.

Complexity: $O(T)\exp(\mathrm{CC}(C_G))$.

But $\mathrm{CC}(C_G) = O(\Delta(G)\mathrm{tw}(G))$.

When $\Delta(G)$ is large this bound is bad.

Example: a star

$\mathrm{tw}(G^*) \leq \Delta(G) \cdot \mathrm{tw}(G)$

$\cdot \exp(\Delta(G) \cdot \mathrm{tw}$

Back to the simulation of one-way computation

outputting $b_1 b_2 ... b_{t-1}$ to compute the conditional probability of outputting 0 for the current measurement.

Complexity: $O(T)\exp(\mathrm{CC}(C_G))$.

But $\mathrm{CC}(C_G) = O(\Delta(G)\mathrm{tw}(G))$.

When $\Delta(G)$ is large this bound is bad.

Example: a star

$tw(G^*) \leq \Delta(G) \cdot tw(G)$

$\cdot \exp(\Delta(G) \cdot tw(G))$

Back to the simulation of one-way computation

outputting $b_1 b_2 ... b_{t-1}$ to compute the conditional probability of outputting 0 for the current measurement.

Complexity: $O(T)\exp(\mathrm{CC}(C_G))$.

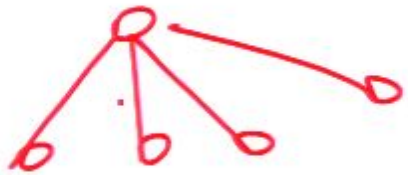
But $\mathrm{CC}(C_G) = O(\Delta(G)\mathrm{tw}(G))$.

When $\Delta(G)$ is large this bound is bad.

Example: a star

$tw(G^*) \leq \Delta(G) \cdot tw(G)$

$\cdot \exp(\Delta(G) \cdot tw(G))$



Back to the simulation of one-way computation

outputting $b_1 b_2 ... b_{t-1}$ to compute the conditional probability of outputting 0 for the current measurement.

Complexity: $O(T)\exp(\mathrm{CC}(C_G))$.

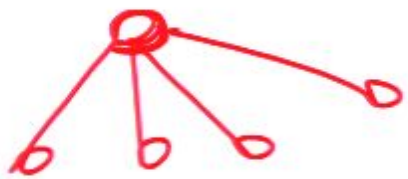
But $\mathrm{CC}(C_G) = O(\Delta(G)\mathrm{tw}(G))$.

When $\Delta(G)$ is large this bound is bad.

Example: a star

$tw(G^*) \leq \Delta(G) \cdot tw(G)$

$\cdot \exp(\Delta(G) \cdot tw(G))$



Back to the simulation of one-way computation

outputting $b_1 b_2 ... b_{t-1}$ to compute the conditional probability of outputting 0 for the current measurement.

Complexity: $O(T)\exp(\mathrm{CC}(C_G))$.

But $\mathrm{CC}(C_G) = O(\Delta(G)\mathrm{tw}(G))$.

When $\Delta(G)$ is large this bound is bad.

Example: a star

$tw(G^*) \leq \Delta(G) \cdot tw(G)$

$\cdot \exp(\Delta(G) \cdot tw(G))$

Back to the simulation of one-way computation

outputting $b_1 b_2 ... b_{t-1}$ to compute the conditional probability of outputting 0 for the current measurement.

Complexity: $O(T)\exp(\mathrm{CC}(C_G))$.

But $\mathrm{CC}(C_G) = O(\Delta(G)\mathrm{tw}(G))$.

When $\Delta(G)$ is large this bound is bad.

Example: a star

$tw(G^*) \leq \Delta(G) \cdot tw(G)$

$\cdot \exp(\Delta(G) \cdot tw(G))$

Back to the simulation of one-way computation

outputting $b_1 b_2 ... b_{t-1}$ to compute the conditional probability of outputting 0 for the current measurement.

Complexity: $O(T)\exp(\mathrm{CC}(C_G))$.

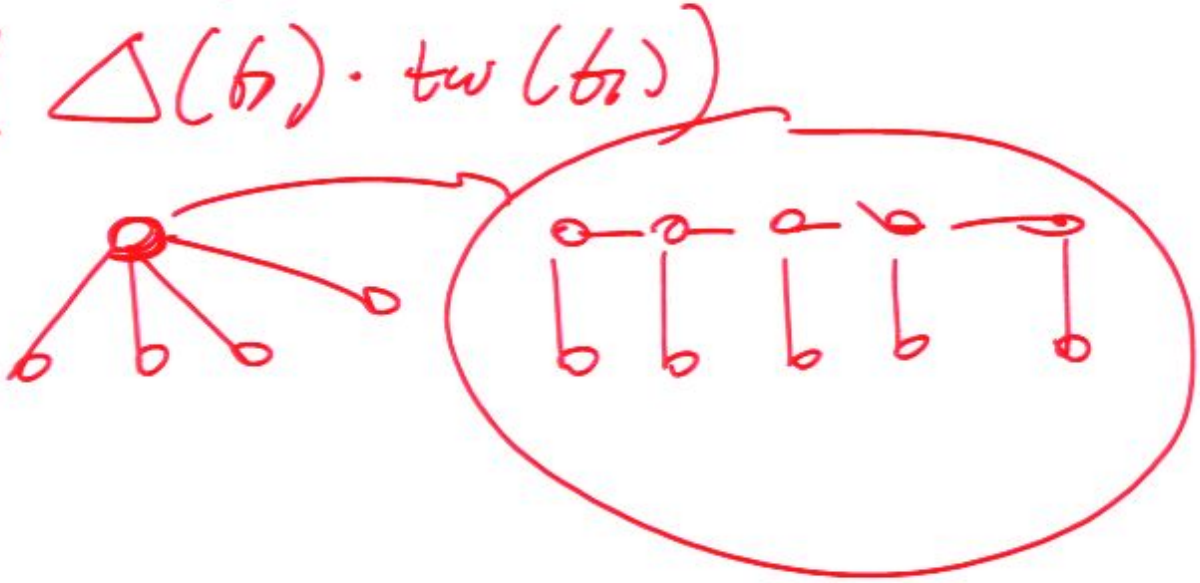
But $\mathrm{CC}(C_G) = O(\Delta(G)\mathrm{tw}(G))$.

When $\Delta(G)$ is large this bound is bad.

Example: a star

$tw(G^*) \leq \Delta(G) \cdot tw(G)$

$\cdot \exp(\Delta(G) \cdot tw(G))$



Back to the simulation of one-way computation

outputting $b_1 b_2 ... b_{t-1}$ to compute the conditional probability of outputting 0 for the current measurement.

Complexity: $O(T)\exp(\mathrm{CC}(C_G))$.

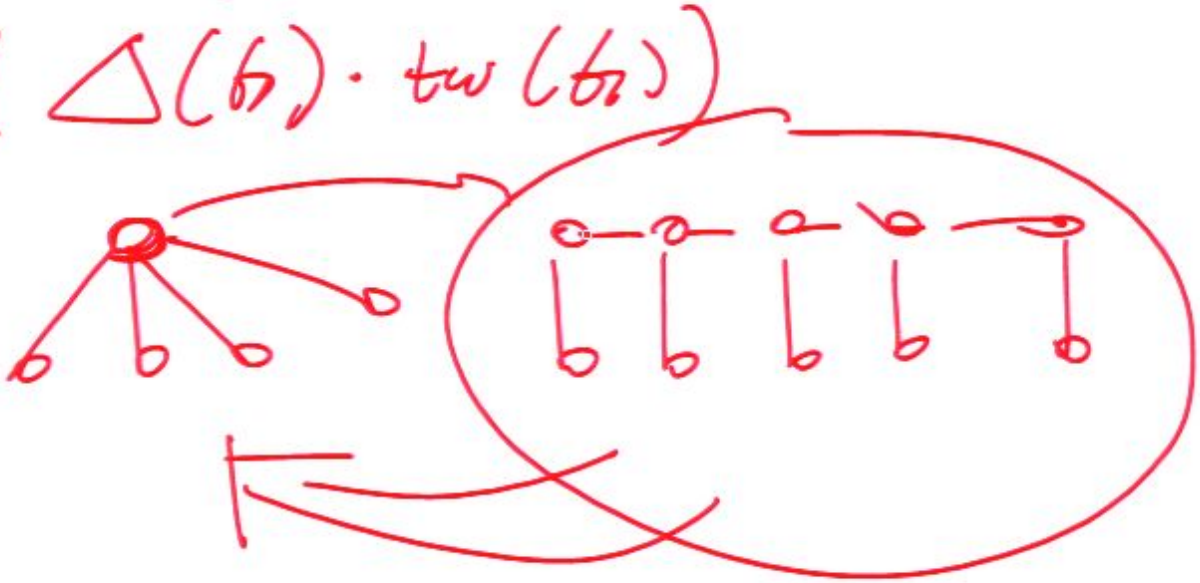
But $\mathrm{CC}(C_G) = O(\Delta(G)\mathrm{tw}(G))$.

When $\Delta(G)$ is large this bound is bad.

Example: a star

$tw(G^*) \leq \Delta(G) \cdot tw(G)$

$\cdot \exp(\Delta(G) \cdot tw(G))$



Back to the simulation of one-way computation

outputting $b_1 b_2 ... b_{t-1}$ to compute the conditional probability of outputting 0 for the current measurement.

Complexity: $O(T)\exp(\mathrm{CC}(C_G))$.

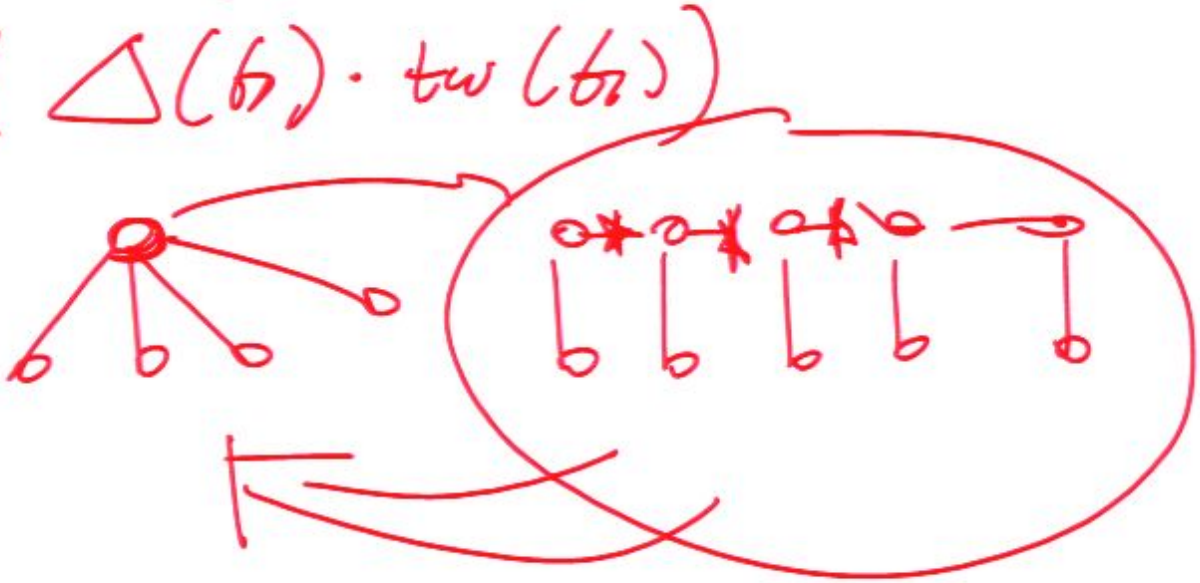
But $\mathrm{CC}(C_G) = O(\Delta(G)\mathrm{tw}(G))$.

When $\Delta(G)$ is large this bound is bad.

Example: a star

$tw(G^*) \leq \Delta(G) \cdot tw(G)$

$\cdot \exp(\Delta(G) \cdot tw(G))$



Back to the simulation of one-way computation

outputting $b_1 b_2 ... b_{t-1}$ to compute the conditional probability of outputting 0 for the current measurement.

Complexity: $O(T)\exp(\mathrm{CC}(C_G))$.

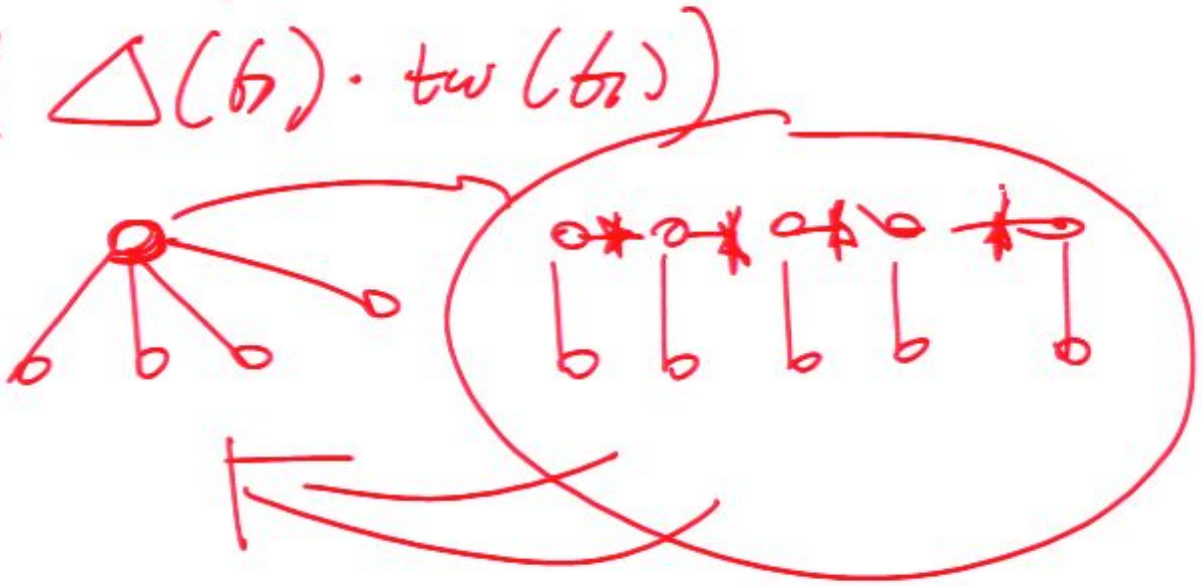
But $\mathrm{CC}(C_G) = O(\Delta(G)\mathrm{tw}(G))$.

When $\Delta(G)$ is large this bound is bad.

Example: a star

$tw(G^*) \leq \Delta(G) \cdot tw(G)$

$\cdot \exp(\Delta(G) \cdot tw(G))$



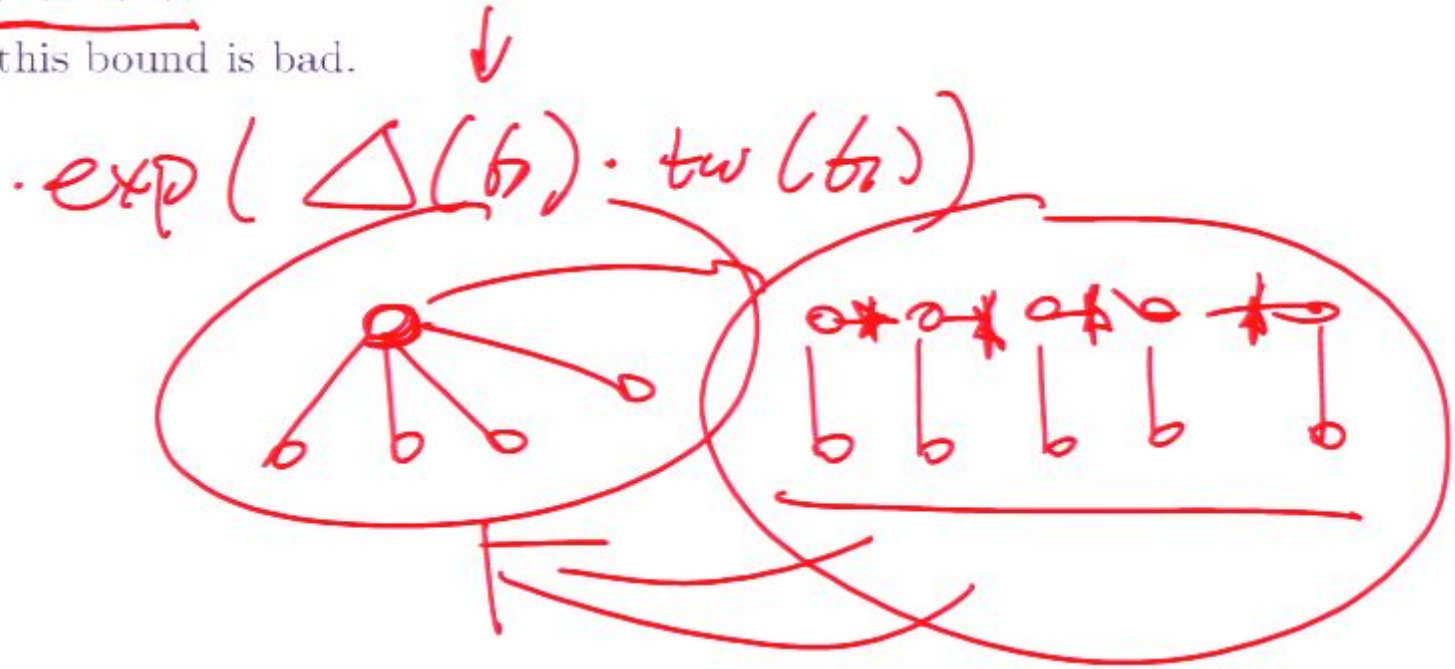
expansion_talk-a - Windows Journal

Edit View Insert Actions Tools Help

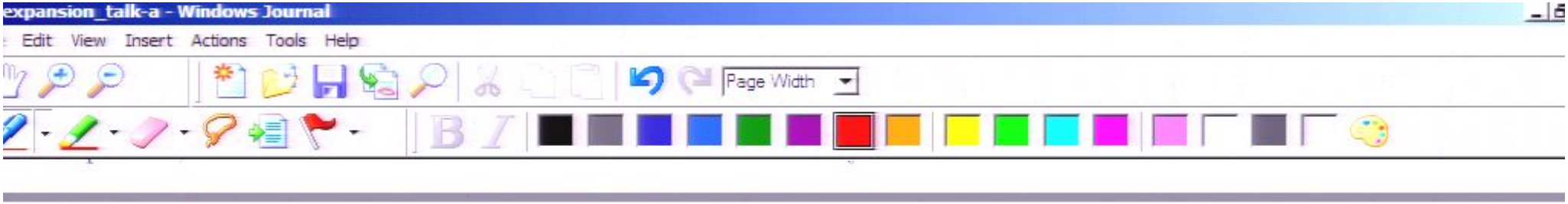
But $\mathrm{CC}(C_G) = O(\Delta(G)\mathrm{tw}(G))$.

When $\Delta(G)$ is large this bound is bad.

Example: a star



Graph expansion



Graph expansion

Splitting a vertex v of a graph $G = \langle V. E \rangle$:

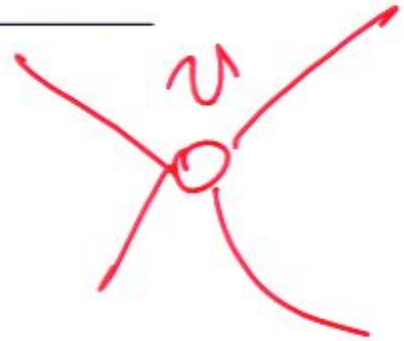
- Replace v by two new vertices v_1 and v_2:
- Connect $v_1 v_2$;
- Connect each neighbor of v to either v_1 or v_2.

Definition. An *expansion* of G is a graph obtained from G through a sequence of vertex splitting.

Graph expansion

Splitting a vertex v of a graph $G = \langle V. E \rangle$:

- Replace v by two new vertices v_1 and v_2:
- Connect $v_1 v_2$;
- Connect each neighbor of v to either v_1 or v_2.



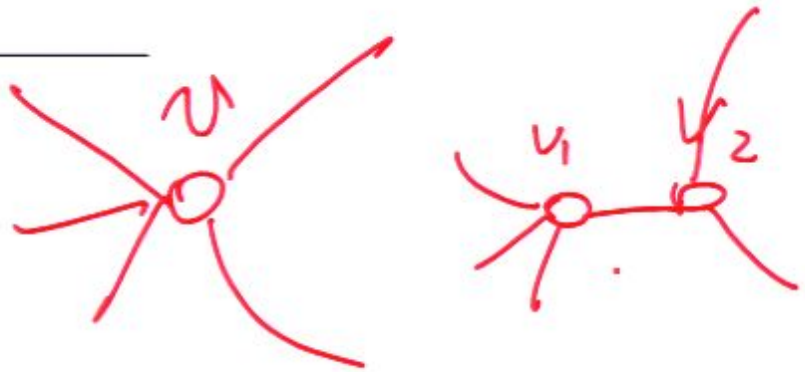
Definition. An *expansion* of G is a graph obtained from G through a sequence of vertex splitting.

Graph expansion

Splitting a vertex v of a graph $G = \langle V, E \rangle$:

- Replace v by two new vertices v_1 and v_2:
- Connect $v_1 v_2$;
- Connect each neighbor of v to either v_1 or v_2.

Definition. An *expansion* of G is a graph obtained from G through a sequence of vertex splitting.

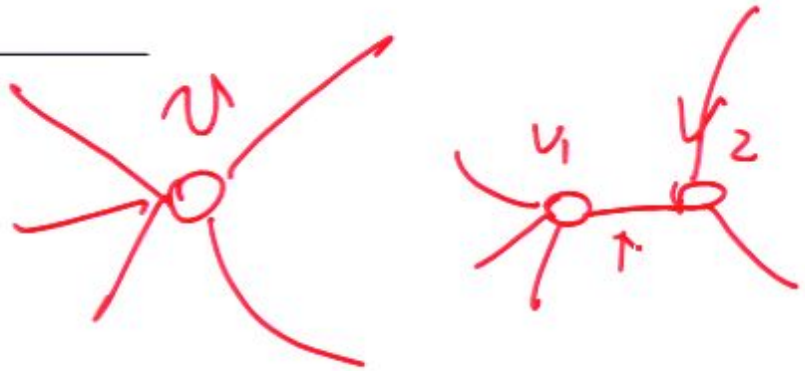


Graph expansion

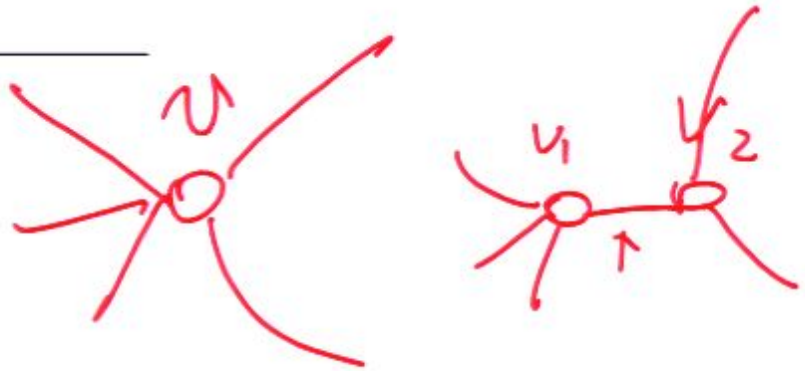
Splitting a vertex v of a graph $G = \langle V, E \rangle$:

- Replace v by two new vertices v_1 and v_2:
- Connect $v_1 v_2$;
- Connect each neighbor of v to either v_1 or v_2.

Definition. An *expansion* of G is a graph obtained from G through a sequence of vertex splitting.



Graph expansion



Splitting a vertex v of a graph $G = \langle V, E \rangle$:

- Replace v by two new vertices v_1 and v_2:
- Connect $v_1 v_2$;
- Connect each neighbor of v to either v_1 or v_2.

Definition. An *expansion* of G is a graph obtained from G through a sequence of vertex splitting.

Use a graph expansion for one-way computation

Fact [RB'01]. Suppose that G' is an expansion of G. and G'' is obtained from G' by inserting a new vertex on each edge. Then there exists a one-way computation on $|G''\rangle$ that produces $|G\rangle$.

Thus a T-step one-way computation on $|G\rangle$ can be simulated in time (and space) $O((T - |V(G'')|)\exp(\Delta(G'')\mathrm{tw}(G'')))$.

Question: How to expand G so that $\Delta(G'') \leq 3$ and $\mathrm{tw}(G'') \approx \mathrm{tw}(G)$?

Use a graph expansion for one-way computation

Fact [RB'01]. Suppose that G' is an expansion of G, and G'' is obtained from G' by inserting a new vertex on each edge. Then there exists a one-way computation on $|G''\rangle$ that produces $|G\rangle$.

Thus a T-step one-way computation on $|G\rangle$ can be simulated in time (and space) $O((T - |V(G'')|)\exp(\Delta(G'')\mathrm{tw}(G'')))$.

Question: How to expand G so that $\Delta(G'') \leq 3$ and $\mathrm{tw}(G'') \approx \mathrm{tw}(G)$?

Use a graph expansion for one-way computation

Fact [RB'01]. Suppose that G' is an expansion of G, and G'' is obtained from G' by inserting a new vertex on each edge. Then there exists a one-way computation on $|G''\rangle$ that produces $|G\rangle$.

Thus a T-step one-way computation on $|G\rangle$ can be simulated in time (and space) $O((T - |V(G'')|)\exp(\Delta(G'')\mathrm{tw}(G'')))$.

Question: How to expand G so that $\Delta(G'') \leq 3$ and $\mathrm{tw}(G'') \approx \mathrm{tw}(G)$?

Use a graph expansion for one-way computation

Fact [RB'01]. Suppose that G' is an expansion of G, and G'' is obtained from G' by inserting a new vertex on each edge. Then there exists a one-way computation on $|G''\rangle$ that produces $|G\rangle$.

Thus a T-step one-way computation on $|G\rangle$ can be simulated in time (and space) $O((T - |V(G'')|)\exp(\Delta(G'')\mathrm{tw}(G'')))$.

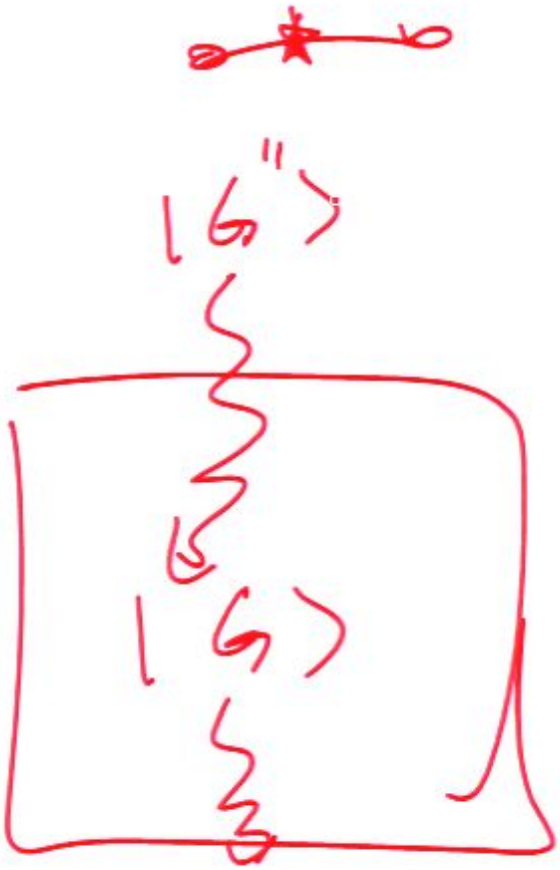
Question: How to expand G so that $\Delta(G'') \leq 3$ and $\mathrm{tw}(G'') \approx \mathrm{tw}(G)$?

Use a graph expansion for one-way computation

Fact [RB'01]. Suppose that G' is an expansion of G, and G'' is obtained from G' by inserting a new vertex on each edge. Then there exists a one-way computation on $|G''\rangle$ that produces $|G\rangle$.

Thus a T-step one-way computation on $|G\rangle$ can be simulated in time (and space) $O((T - |V(G'')|)\exp(\Delta(G'')\mathrm{tw}(G'')))$.

Question: How to expand G so that $\Delta(G'') \leq 3$ and $\mathrm{tw}(G'') \approx \mathrm{tw}(G)$?

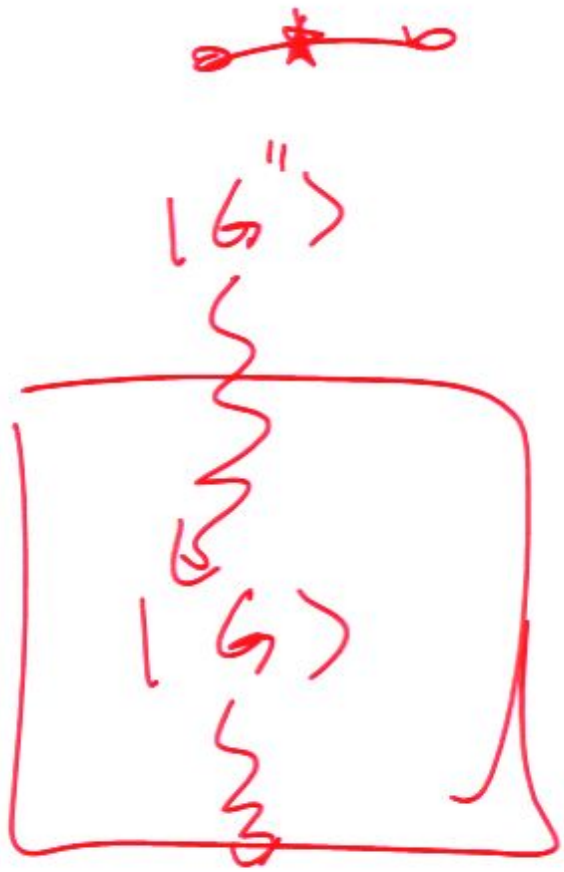


Use a graph expansion for one-way computation

Fact [RB'01]. Suppose that G' is an expansion of G, and G'' is obtained from G' by inserting a new vertex on each edge. Then there exists a one-way computation on $|G''\rangle$ that produces $|G\rangle$.

Thus a T-step one-way computation on $|G\rangle$ can be simulated in time (and space) $O((T - |V(G'')|)\exp(\Delta(G'')\mathrm{tw}(G'')))$.

Question: How to expand G so that $\Delta(G'') \leq 3$ and $\mathrm{tw}(G'') \approx \mathrm{tw}(G)$?

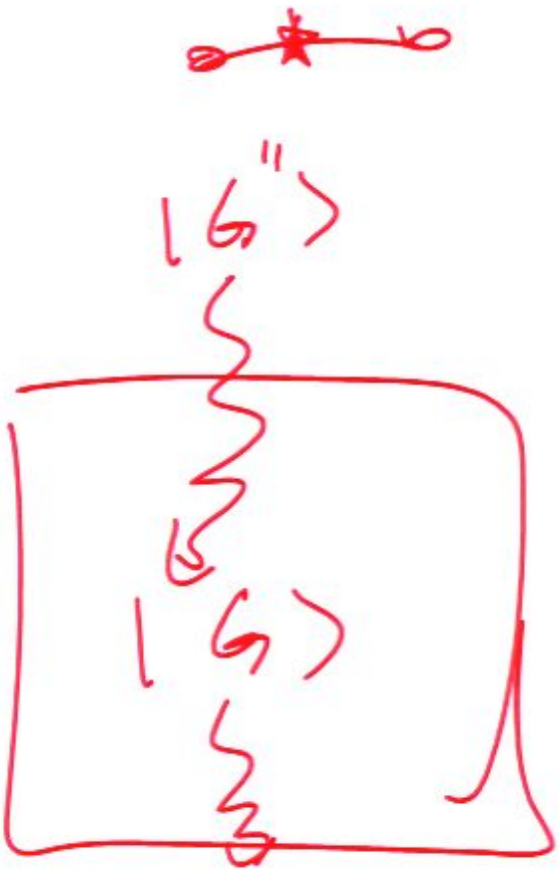


Use a graph expansion for one-way computation

Fact [RB'01]. Suppose that G' is an expansion of G, and G'' is obtained from G' by inserting a new vertex on each edge. Then there exists a one-way computation on $|G''\rangle$ that produces $|G\rangle$.

Thus a T-step one-way computation on $|G\rangle$ can be simulated in time (and space) $O((T - |V(G'')|)\exp(\Delta(G'')\mathrm{tw}(G'')))$.

Question: How to expand G so that $\Delta(G'') \leq 3$ and $\mathrm{tw}(G'') \approx \mathrm{tw}(G)$?

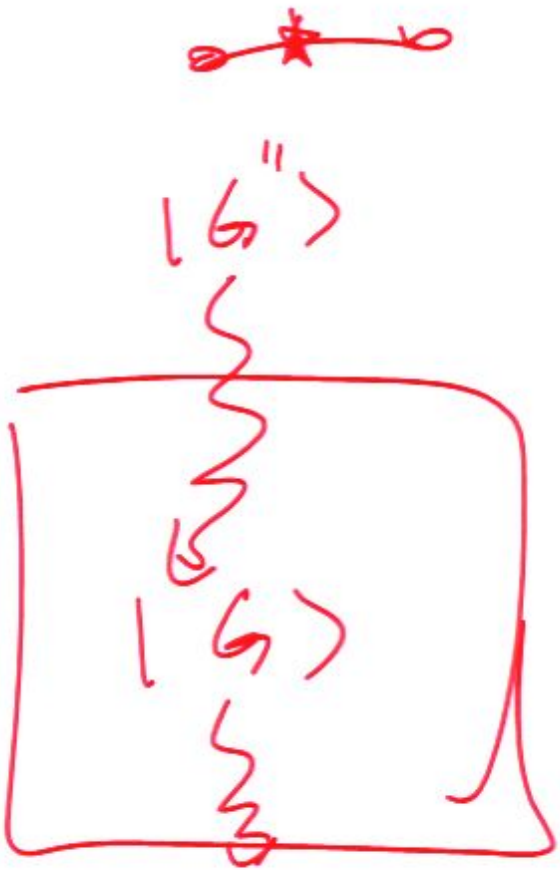


Use a graph expansion for one-way computation

Fact [RB'01]. Suppose that G' is an expansion of G, and G'' is obtained from G' by inserting a new vertex on each edge. Then there exists a one-way computation on $|G''\rangle$ that produces $|G\rangle$.

Thus a T-step one-way computation on $|G\rangle$ can be simulated in time (and space) $O((T - |V(G'')|)\exp(\Delta(G'')\mathrm{tw}(G'')))$.

Question: How to expand G so that $\Delta(G'') \leq 3$ and $\mathrm{tw}(G'') \approx \mathrm{tw}(G)$?



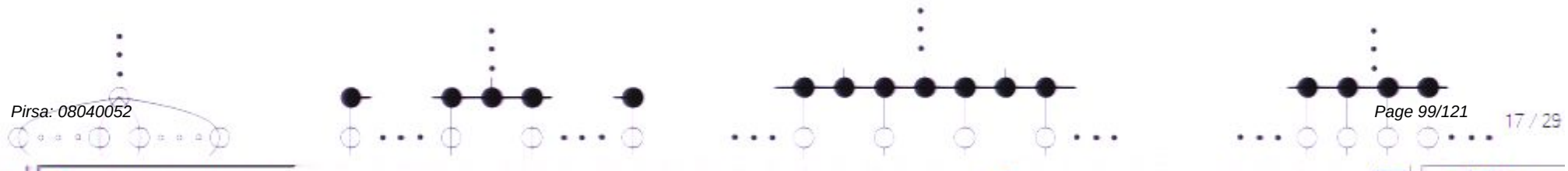
expansion_talk-a - Windows Journal

Edit View Insert Actions Tools Help

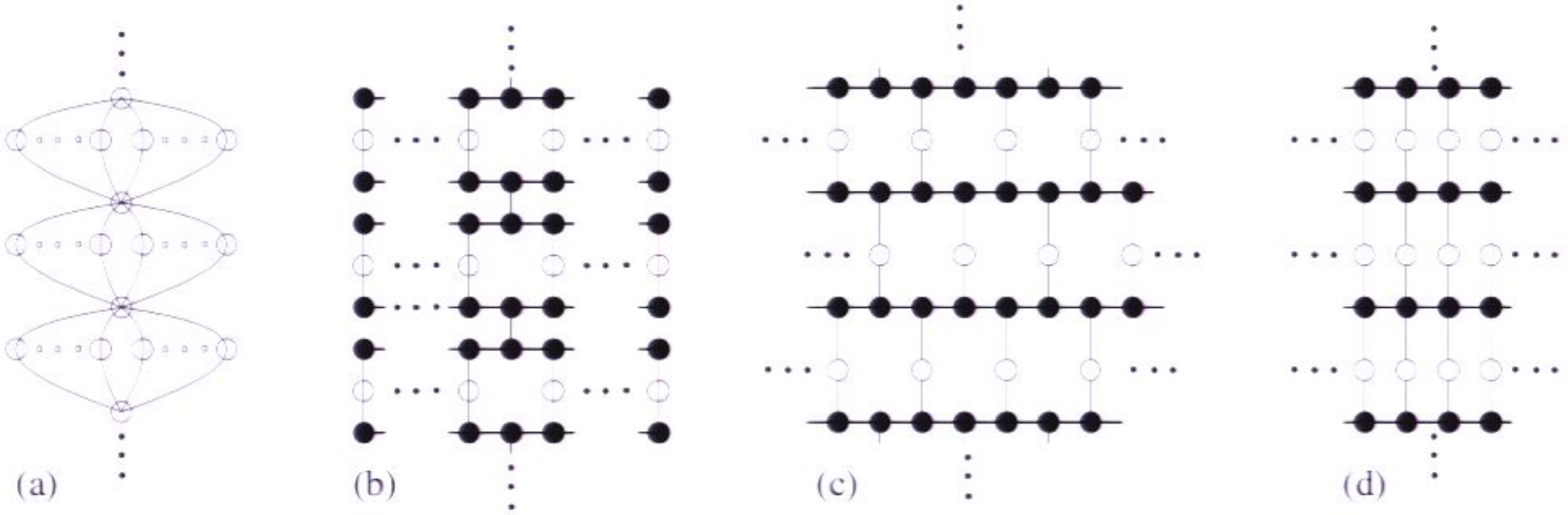
Page Width

Question: How to expand G so that $\Delta(G'') \leq 3$ and $\mathrm{tw}(G'') \approx \mathrm{tw}(G)$?

An example for careless expansion

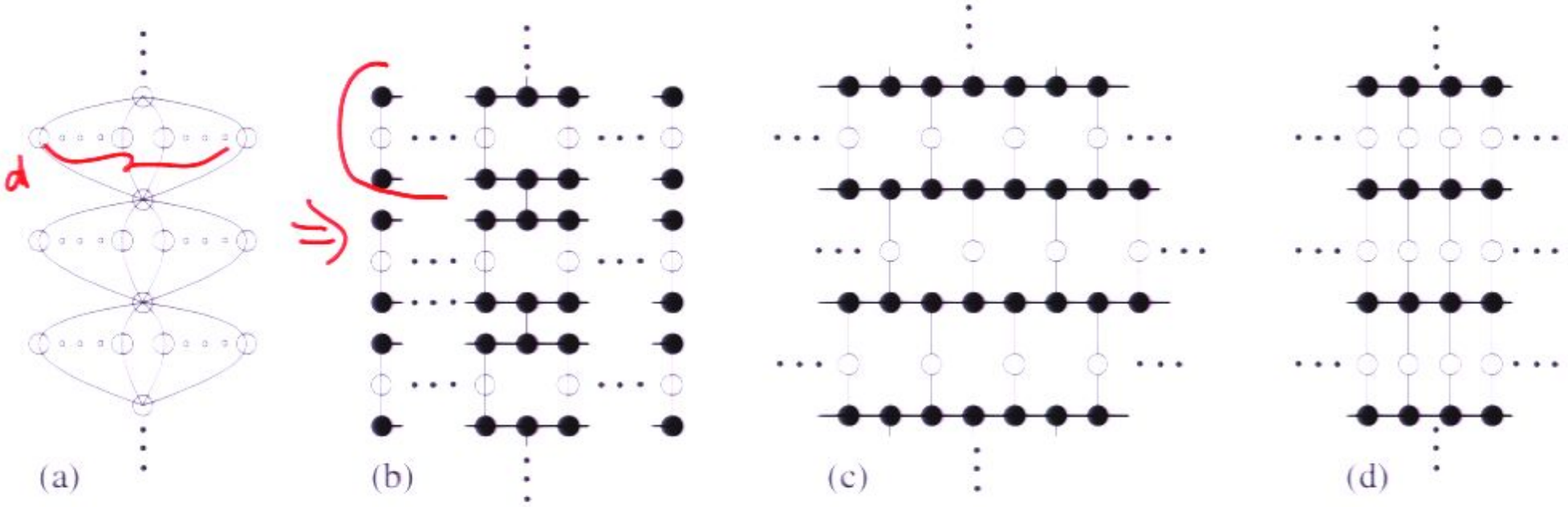


An example for careless expansion



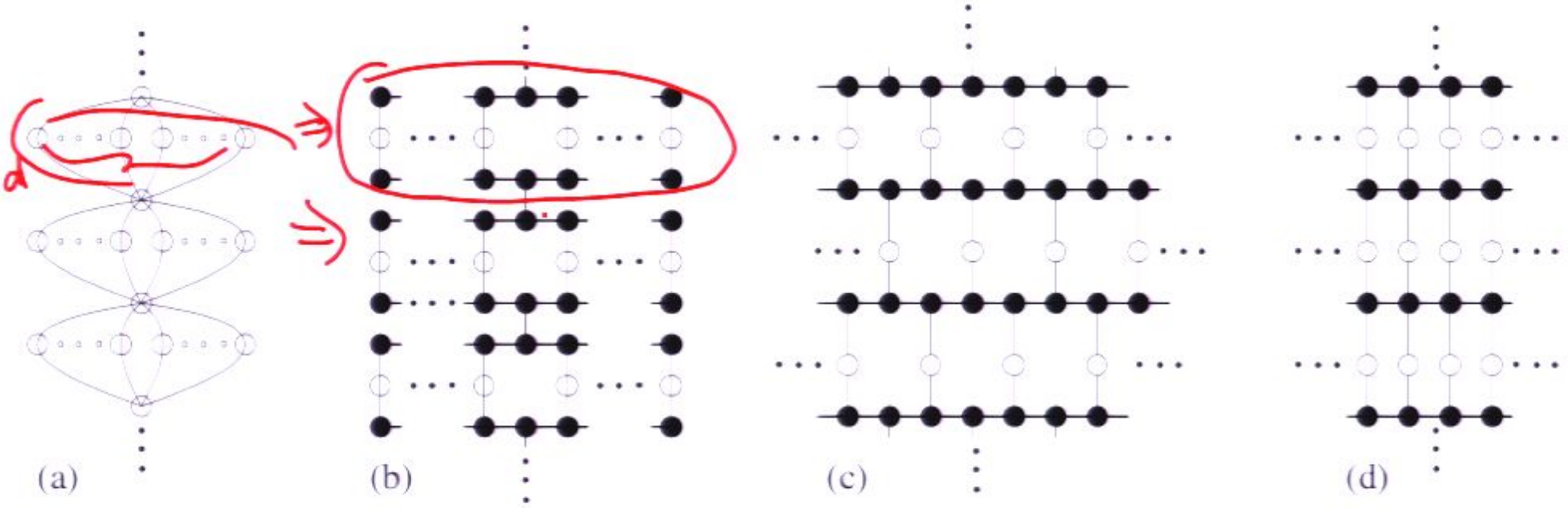
- The graph in (a) has treewidth 2, and
- has a ternary expansion (b) of the same treewidth.
- Another expansion (c) contains a grid

An example for careless expansion



- The graph in (a) has treewidth 2, and
- has a ternary expansion (b) of the same treewidth.
- Another expansion (c) contains a grid

An example for careless expansion



- The graph in (a) has treewidth 2, and
- has a ternary expansion (b) of the same treewidth.
- Another expansion (c) contains a grid

Main Result

- For any graph G there exists an expansion G' such that $\Delta(G') \leq 3$ and $\mathrm{tw}(G') \leq \mathrm{tw}(G) + 1$. Such an expansion can be computed efficiently from an optimal tree decomposition of G.
- There exists a family of graphs so that the increase in treewidth is necessary for any ternary expansions.

Main Result

- For any graph G there exists an expansion G' such that $\Delta(G') \leq 3$ and $\mathrm{tw}(G') \leq \mathrm{tw}(G) + 1$. Such an expansion can be computed efficiently from an optimal tree decomposition of G.
- There exists a family of graphs so that the increase in treewidth is necessary for any ternary expansions.

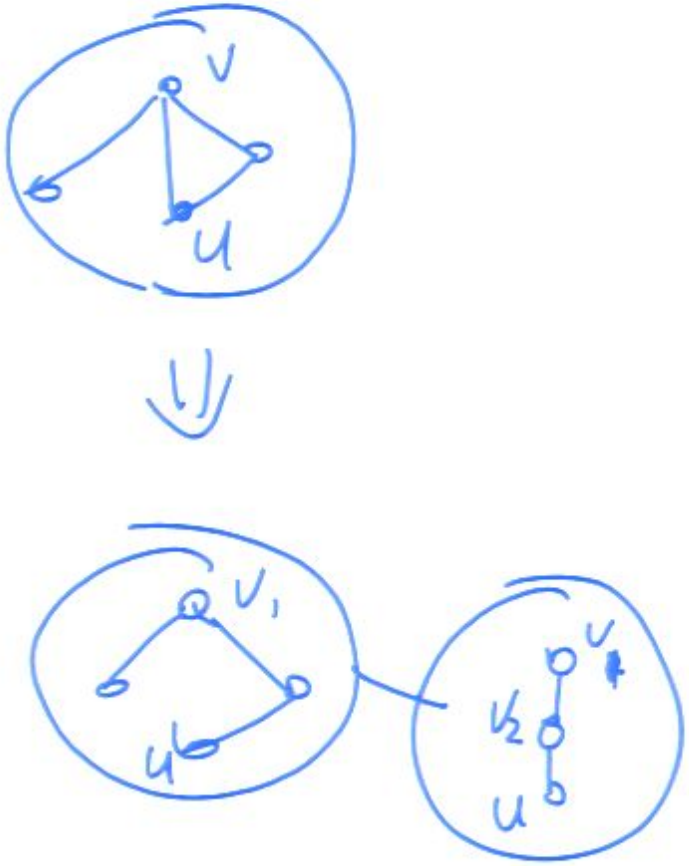
Reducing in-bag max degree

Fix a tree decomposition $\mathcal{T}$ of G.

For each bag B and each $v \in B$, if v has ≥ 3 neighbors in B.

- Split $v \to v_1 v_2$.
- Attach a neighbor $u \in B$ to v_2, and all other neighbors to v_1.
- Create a new bag $\{v_2, v_1, u\}$ and attach it to B.

We can now assume that each vertex has ≤ 2 neighbors in each bag.



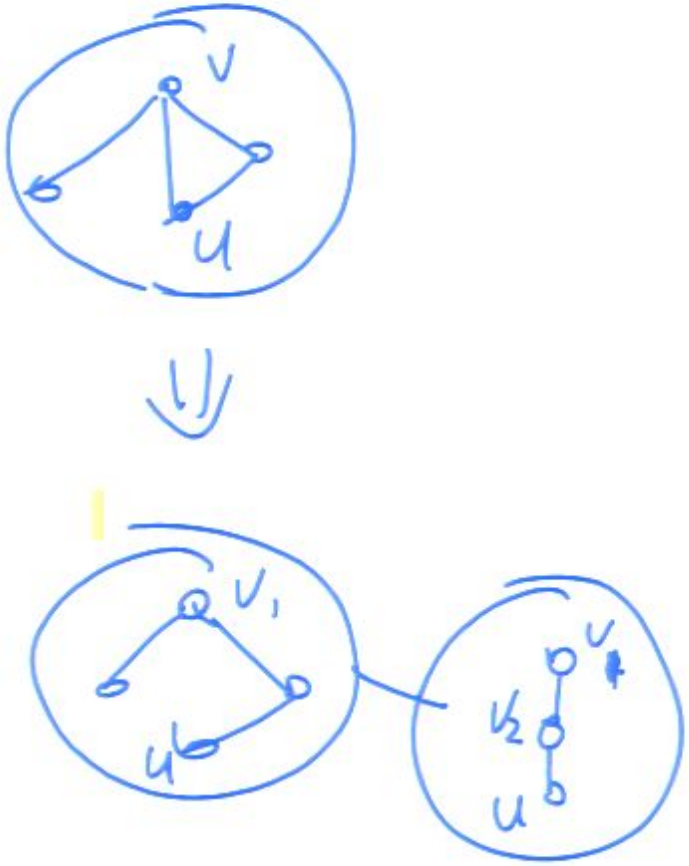
Reducing in-bag max degree

Fix a tree decomposition $\mathcal{T}$ of G.

For each bag B and each $v \in B$, if v has ≥ 3 neighbors in B.

- Split $v \to v_1 v_2$.
- Attach a neighbor $u \in B$ to v_2, and all other neighbors to v_1.
- Create a new bag $\{v_2, v_1, u\}$ and attach it to B.

We can now assume that each vertex has ≤ 2 neighbors in each bag.



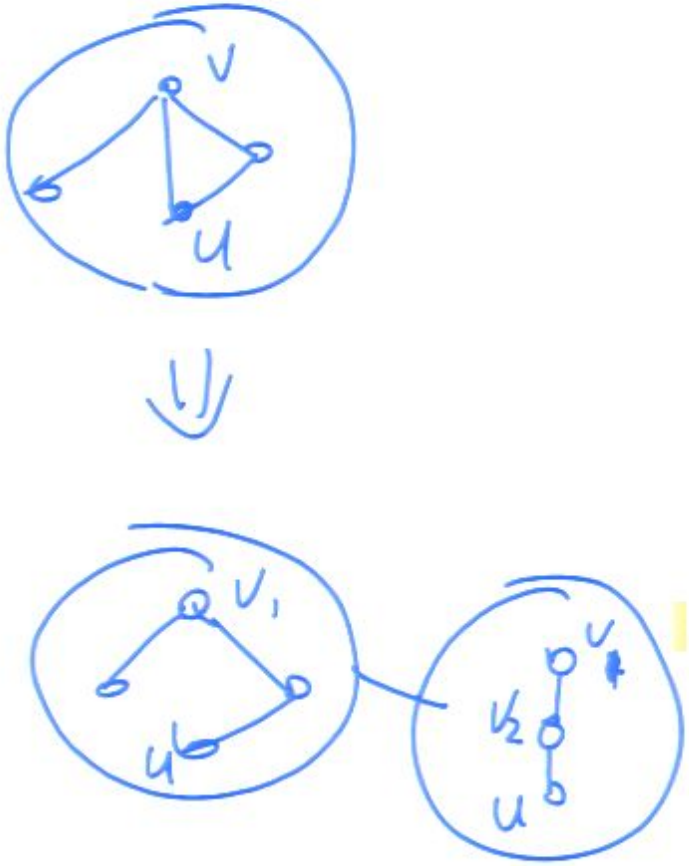
Reducing in-bag max degree

Fix a tree decomposition $\mathcal{T}$ of G.

For each bag B and each $v \in B$, if v has ≥ 3 neighbors in B.

- Split $v \longrightarrow v_1 v_2$.
- Attach a neighbor $u \in B$ to v_2, and all other neighbors to v_1.
- Create a new bag $\{v_2, v_1, u\}$ and attach it to B.

We can now assume that each vertex has ≤ 2 neighbors in each bag.



Constructing the expansion

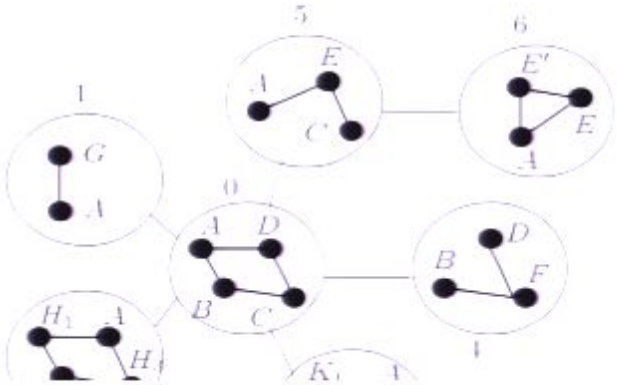
For each bag B_0 with adjacent bags B_1, B_2, ... B_t, and each $v \in B_0$, create a path graph

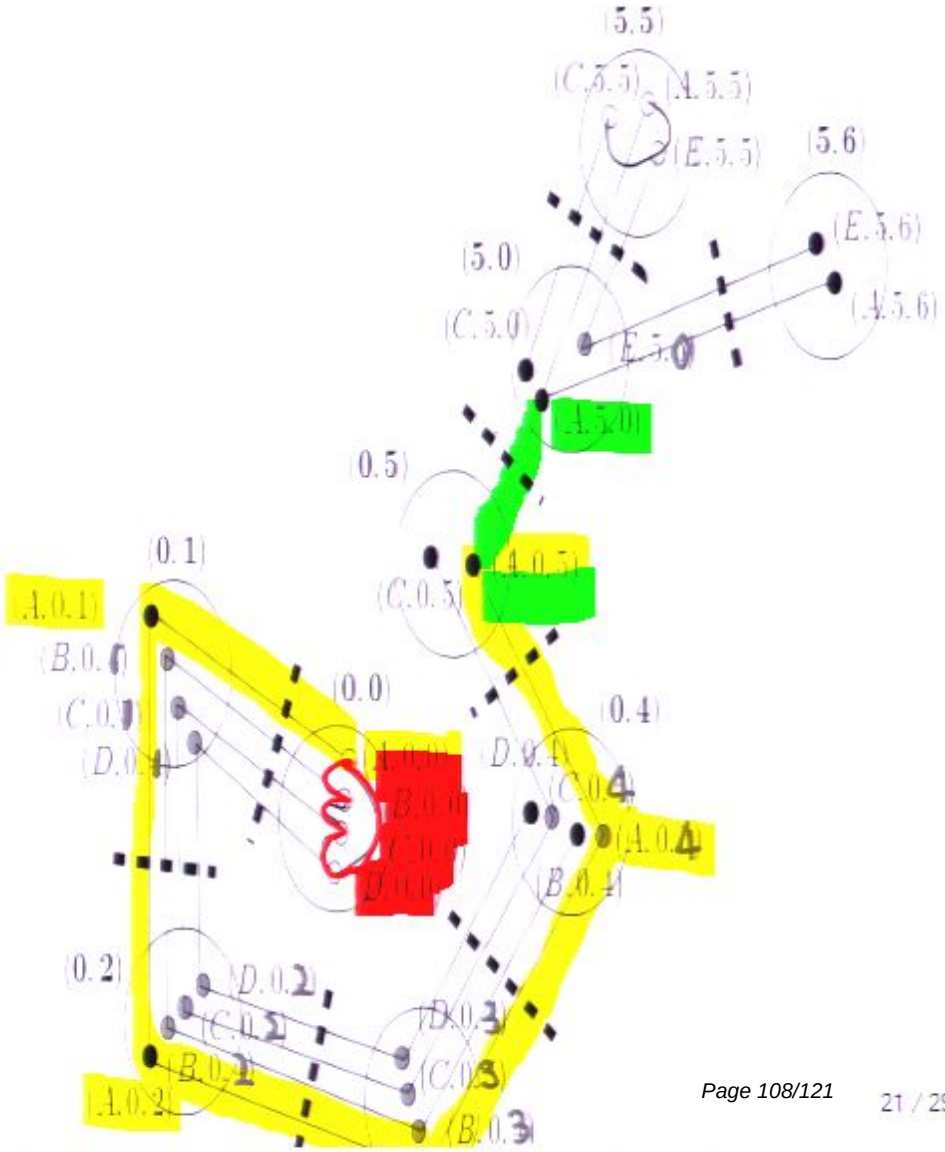
$$(v, B_0, B_0) - (v, B_0, B_1) - \cdots - (v, B_0, B_t).$$

Add the following edges

- If $v \in B \cap B'$, connect $(v, B, B') - (v, B', B)$.
- For each edge $uv \in E(G)$, fix a bag B_0 with $u, v \in B_0$, connect $(u, B_0, B_0) - (v, B_0, B_0)$.

Call the resulting graph G'.





expansion_talk-a - Windows Journal

Edit View Insert Actions Tools Help

Page Width

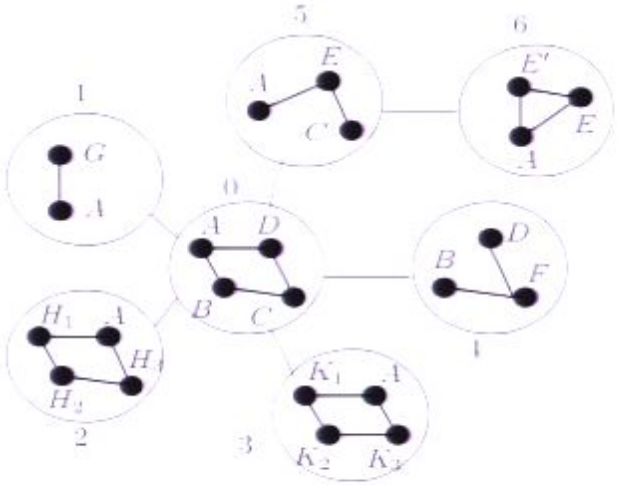
B_t, and each $v \in B_0$, create a path graph

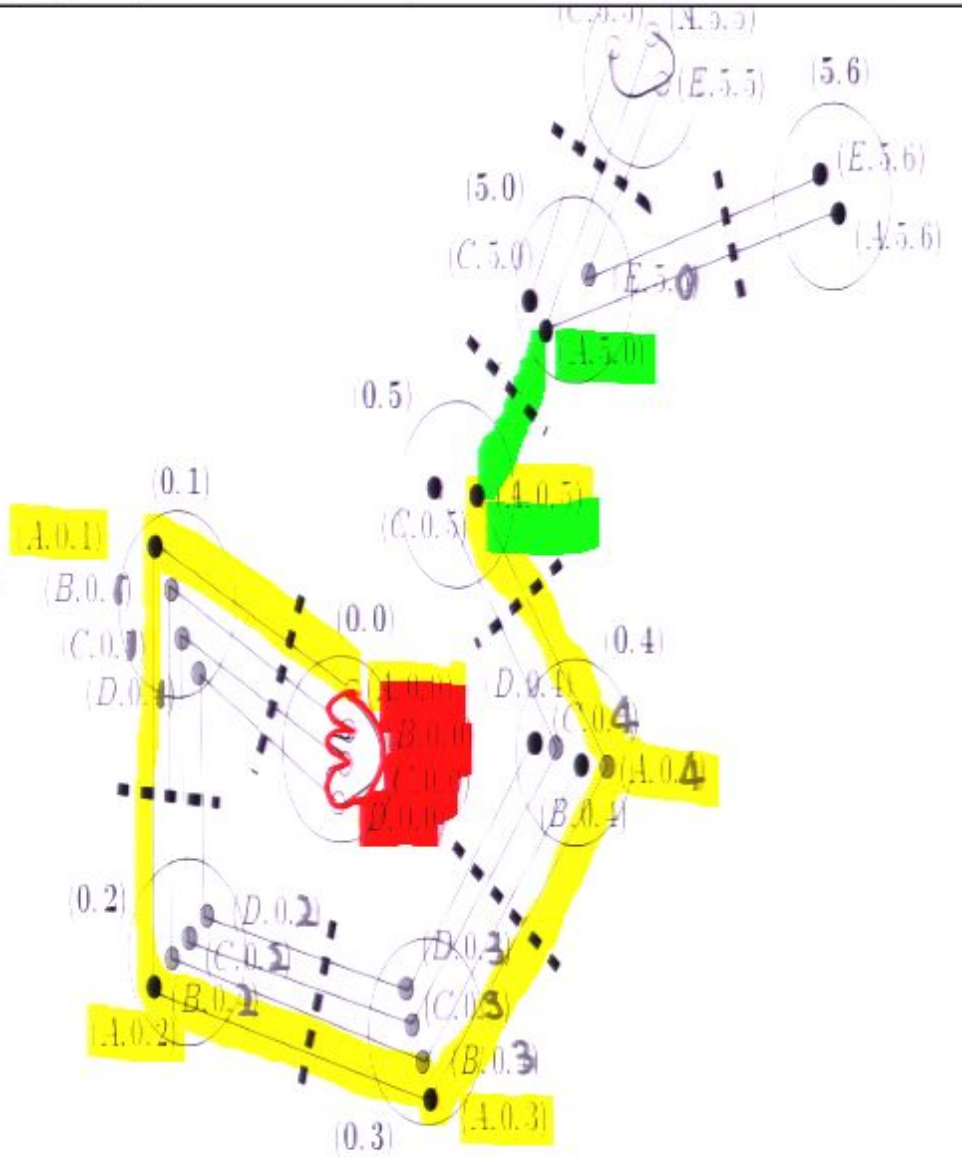
$$(v, B_0, B_0) - (v, B_0, B_1) - \cdots - (v, B_0, B_t).$$

Add the following edges

- If $v \in B \cap B'$, connect $(v, B, B') - (v, B', B)$.
- For each edge $uv \in E(G)$, fix a bag B_0 with $u, v \in B_0$, connect $(u, B_0, B_0) - (v, B_0, B_0)$.

Call the resulting graph G'.





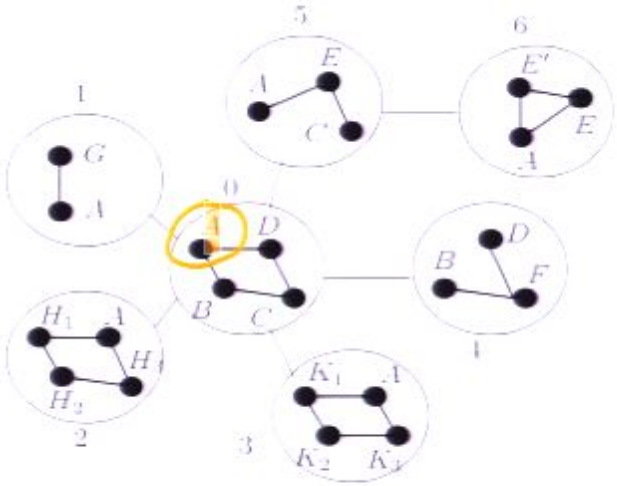
For each bag B_0 with adjacent bags B_1, B_2, ... B_t, and each $v \in B_0$, create a path graph

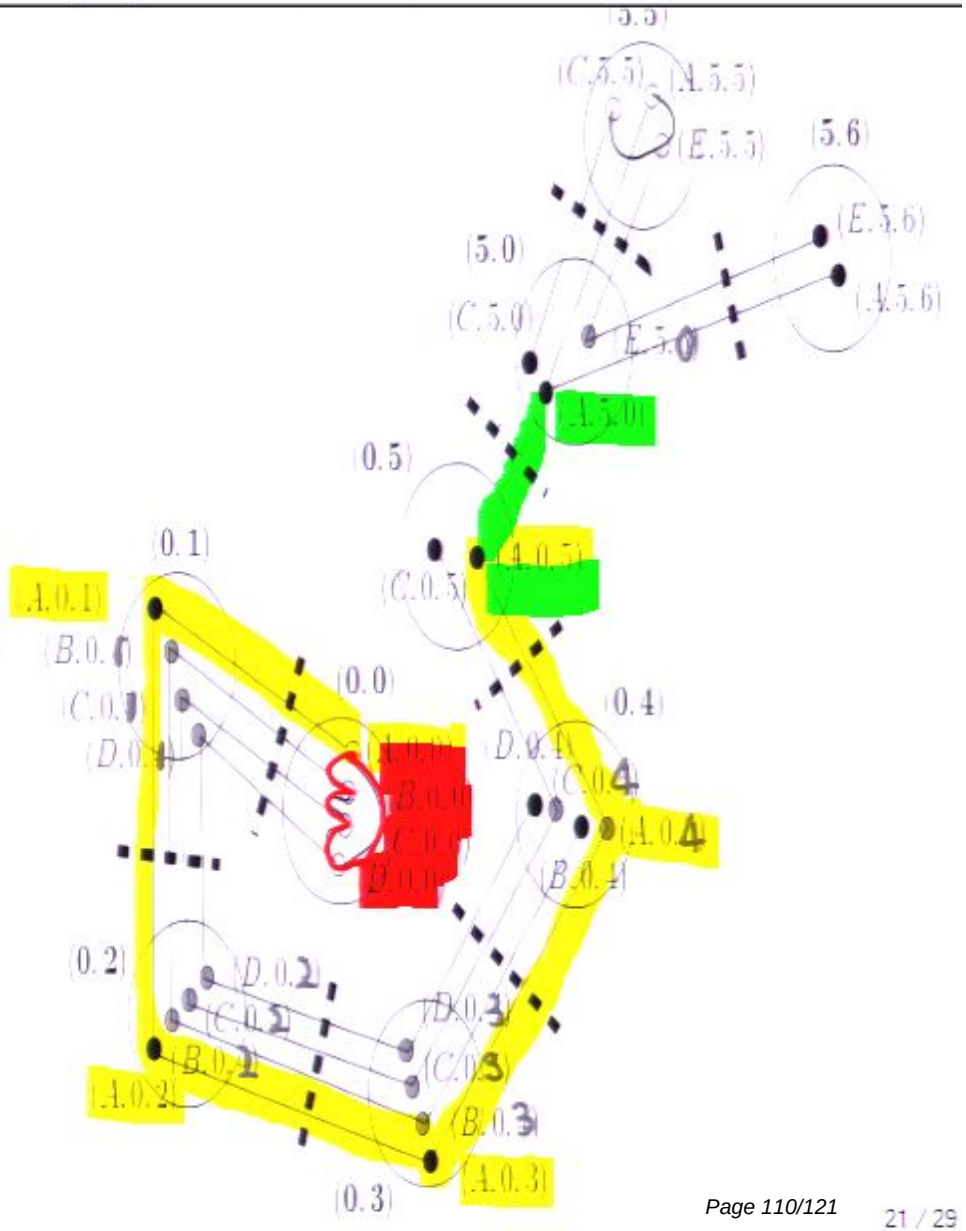
$$(v, B_0, B_0) - (v, B_0, B_1) - \cdots - (v, B_0, B_t).$$

Add the following edges

- If $v \in B \cap B'$, connect $(v, B, B') - (v, B', B)$.
- For each edge $uv \in E(G)$, fix a bag B_0 with $u, v \in B_0$, connect $(u, B_0, B_0) - (v, B_0, B_0)$.

Call the resulting graph G'.





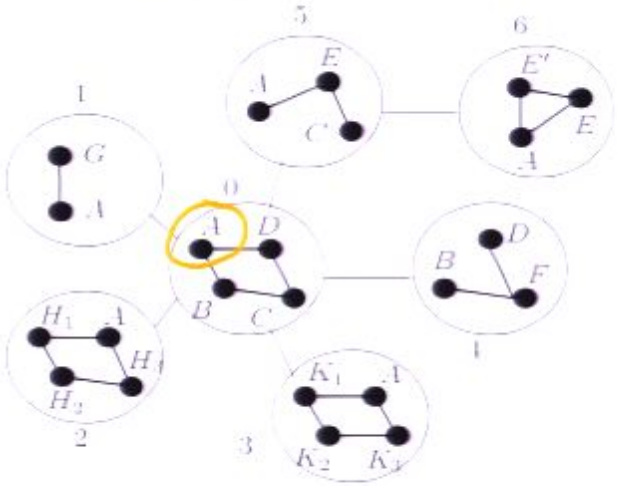
For each bag B_0 with adjacent bags B_1, B_2, ... B_t, and each $v \in B_0$, create a path graph

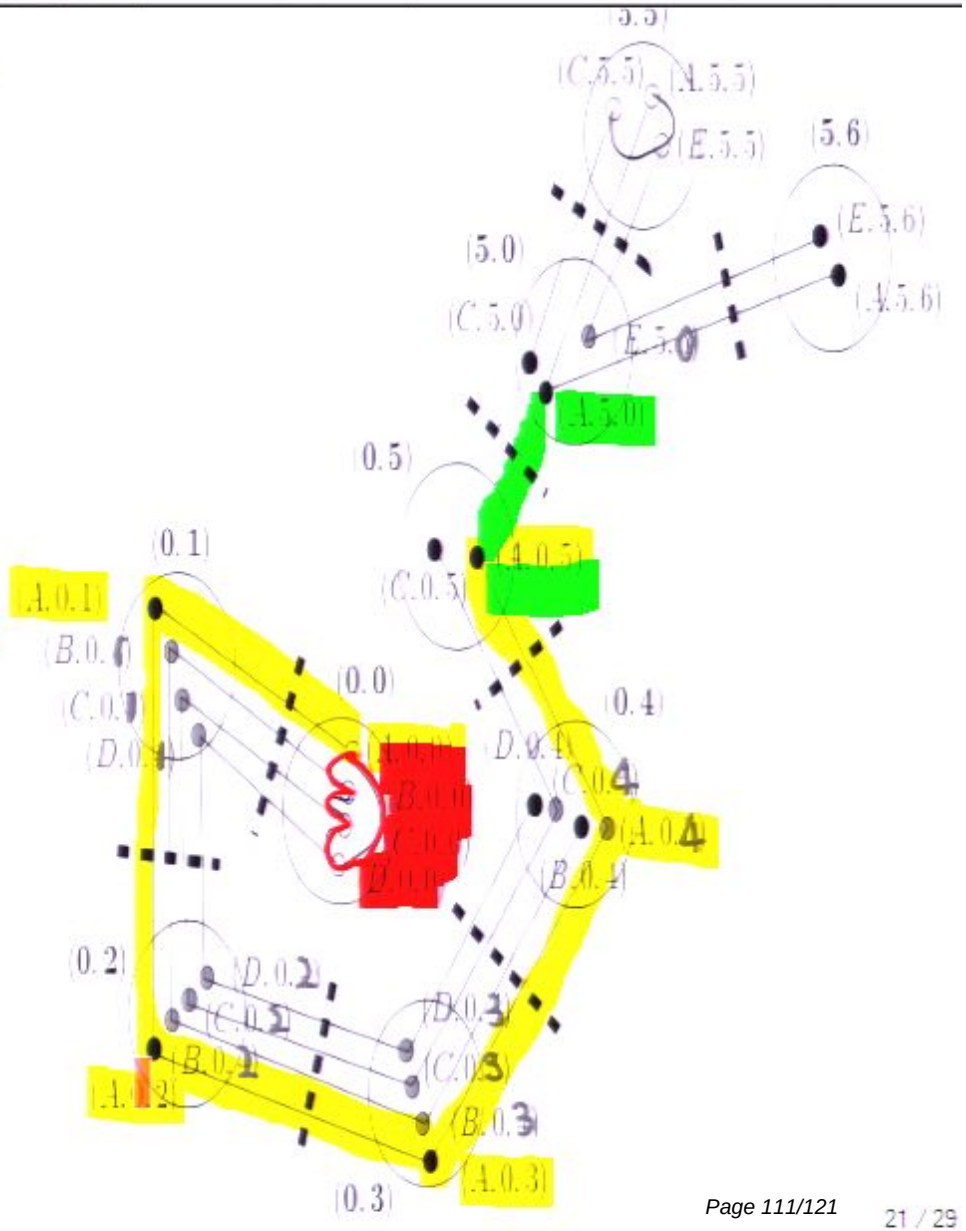
$$(v, B_0, B_0) - (v, B_0, B_1) - \cdots - (v, B_0, B_t).$$

Add the following edges

- If $v \in B \cap B'$, connect $(v, B, B') - (v, B', B)$.
- For each edge $uv \in E(G)$, fix a bag B_0 with $u, v \in B_0$, connect $(u, B_0, B_0) - (v, B_0, B_0)$.

Call the resulting graph G'.





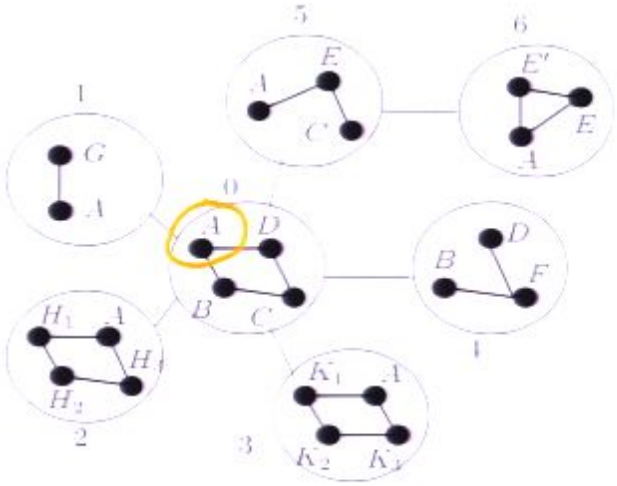
D_t, and each $v \in B_0$, create a path graph

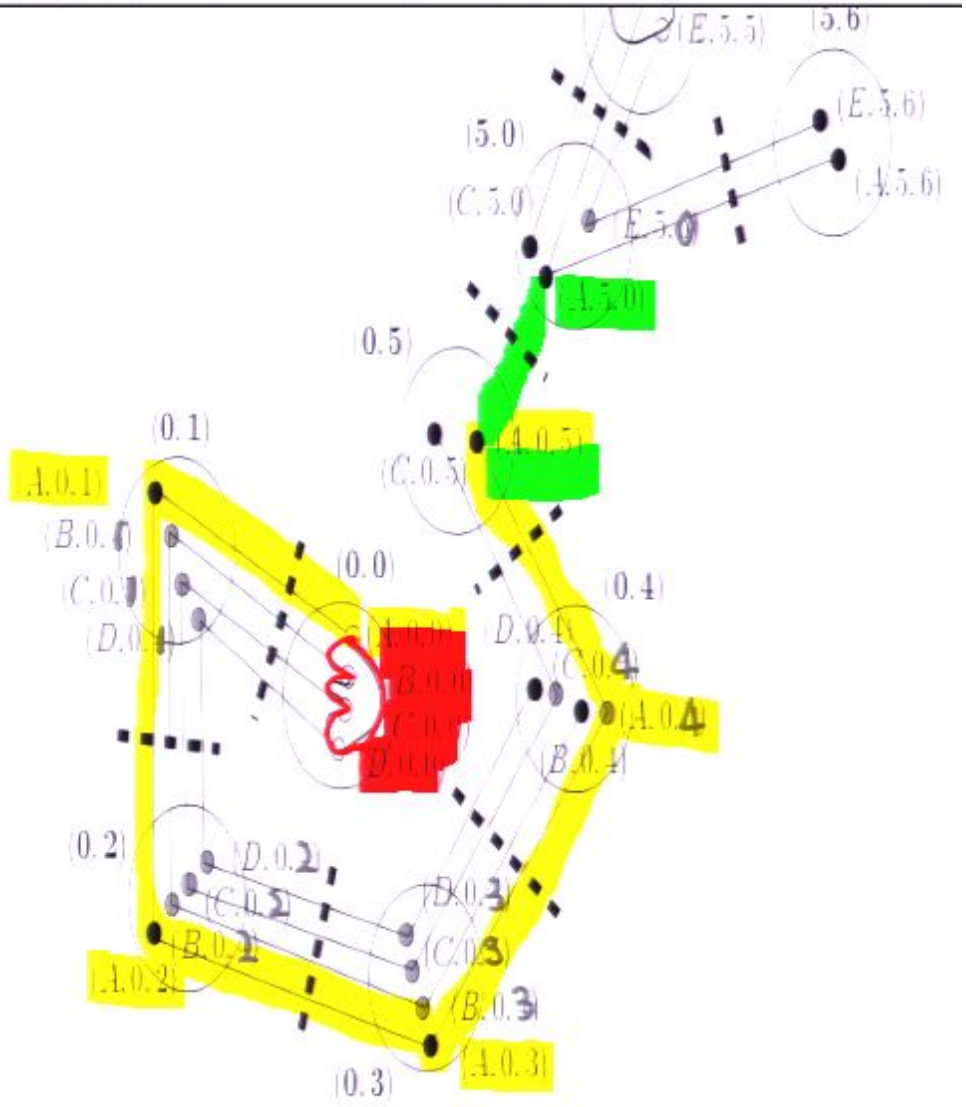
$$(v, B_0, B_0) - (v, B_0, B_1) - \cdots - (v, B_0, B_t).$$

Add the following edges

- If $v \in B \cap B'$, connect $(v, B, B') - (v, B', B)$.
- For each edge $uv \in E(G)$, fix a bag B_0 with $u, v \in B_0$, connect $(u, B_0, B_0) - (v, B_0, B_0)$.

Call the resulting graph G'.





Three types of bags:

- For each bag B of $\mathcal{T}$, create a bag

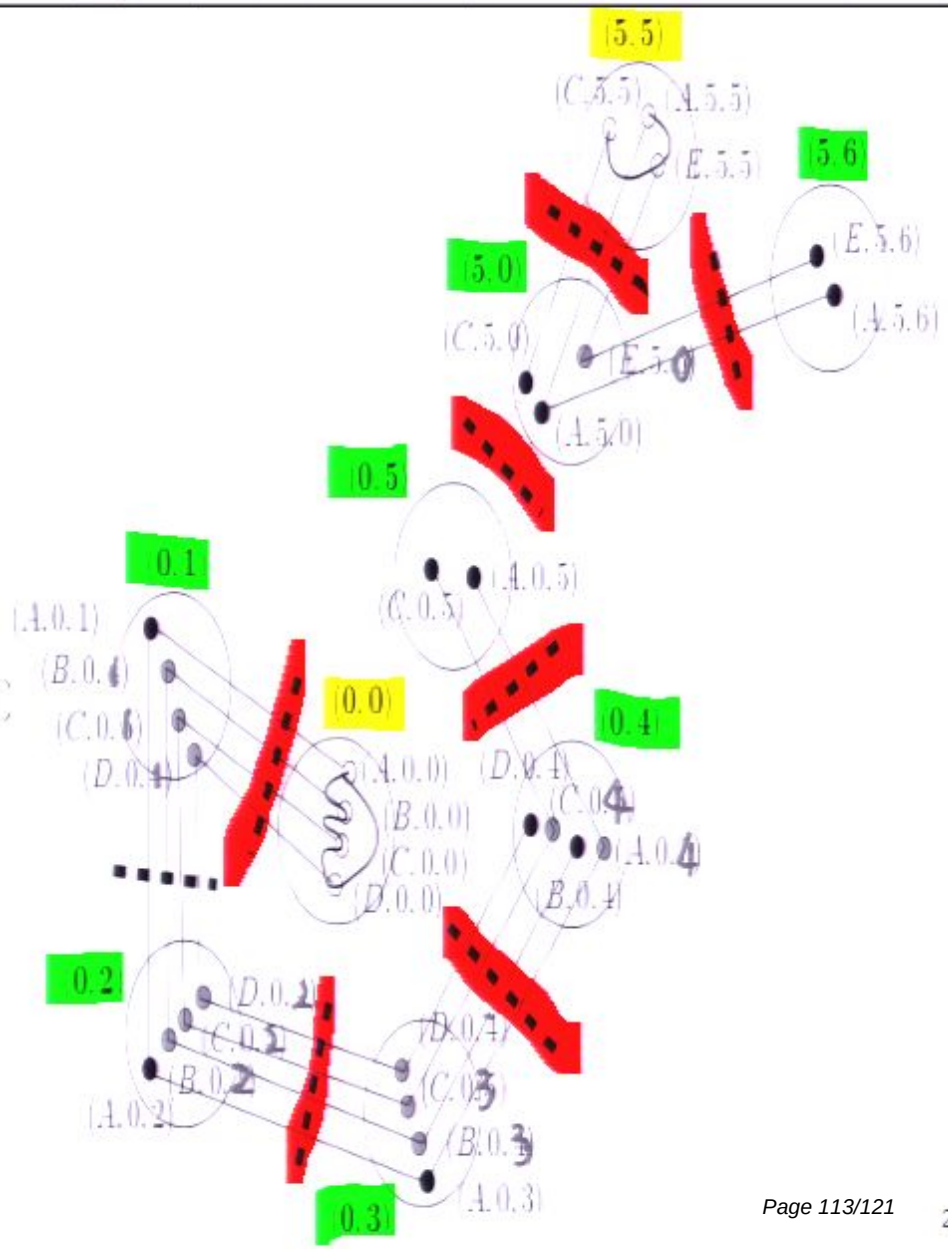
  $$(B, B) = \{(v, B, B) : v \in B\}.$$

  This bag covers all edges between vertex trees.

- For adjacent bags $B_0 B_i$ in $\mathcal{T}$, create

  $$(B_0, B_i) = \{(v, B_0, B_i) : v \in B\}.$$

- "Bridge" the bags (B_0, B_i) and (B_0, B_{i-1}) and the bags (B, B') and (B', B) by a chain of bags.



- $A \cap B = \{u_1, \cdots, u_s\}$, and
- $E_{AB} \stackrel{\text{def}}{=} \{v_i v_i' : 1 \le i \le t\}$, are edges in G'.

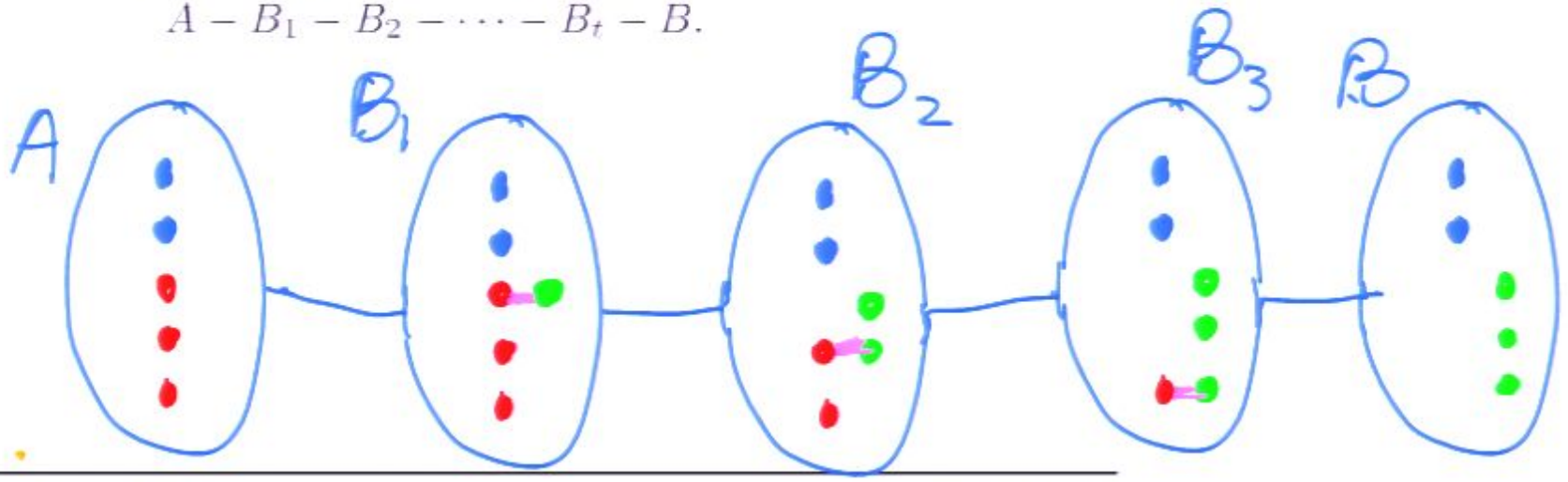
Bridging the two bags to cover E_{AB}:

(Step 1) Create bags $B_1, B_2, \cdots, B_t$, where

$$B_i = (A \cap B) \cup \{v_1', v_2', \cdots, v_{i-1}', v_i', v_i, v_{i+1}, \cdots, v_t\}.$$

(Step 2) Form a path graph of bags

$$A - B_1 - B_2 - \cdots - B_t - B.$$



Concluding the proof

expansion_talk-a - Windows Journal

Edit View Insert Actions Tools Help

- $A \cap B = \{u_1, \cdots, u_s\}$, and
- $E_{AB} \stackrel{\text{def}}{=} \{v_i v_i' : 1 \le i \le t\}$, are edges in G'.

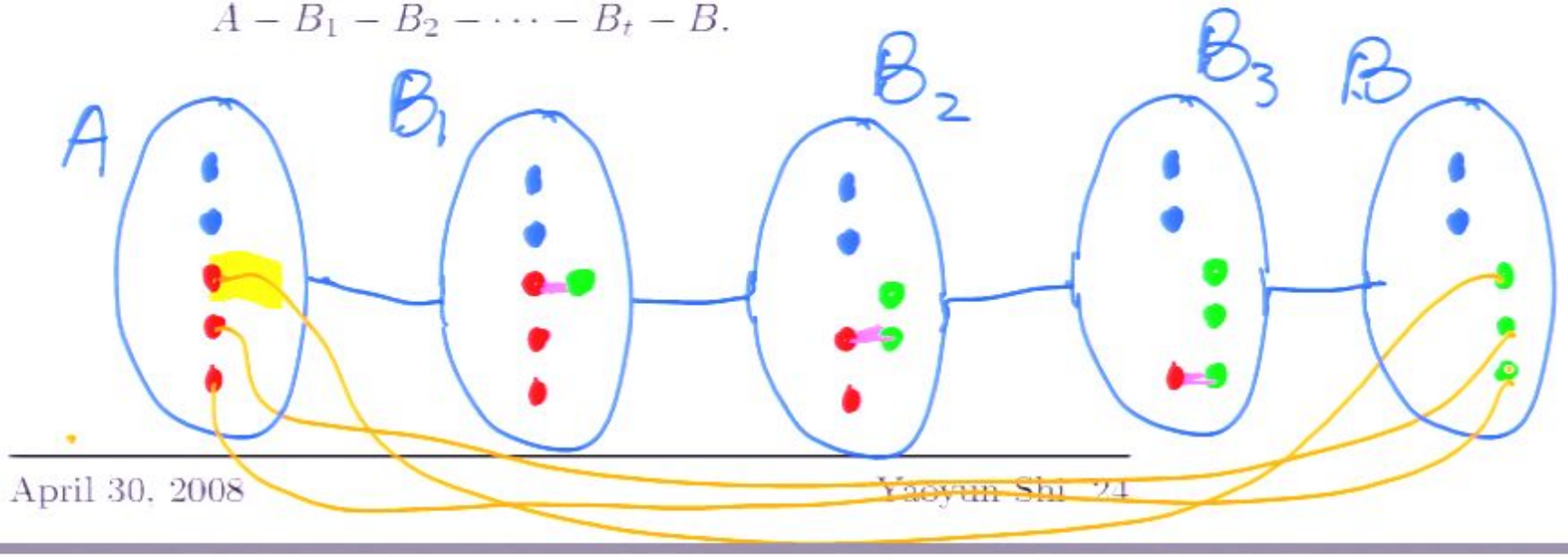
Bridging the two bags to cover E_{AB}:

(Step 1) Create bags $B_1, B_2, \cdots, B_t$, where

$$B_i = (A \cap B) \cup \{v_1', v_2', \cdots, v_{i-1}', v_i', v_i, v_{i+1}, \cdots, v_t\}.$$

(Step 2) Form a path graph of bags

$$A - B_1 - B_2 - \cdots - B_t - B.$$



Concluding the proof

Open problems

- Which states are universal for measurement-based computation?
  - A universal state does not need to be a graph state [Gross and Eisert '06...]
  - A definition of universality.
    A family of states $|S_n\rangle$ on n qubits is universal if for any quantum circuit of size m, there exists a measurement-based quantum computation on $|S_n\rangle$, for some $n = poly(m)$, that creates the same state as the final state of the circuit.
- Sharpen the classical upper bound of BQP (the class of functions computable by a polynomial time quantum algorithm).
  - A reduction from a BQP problem to $\exists\forall$ SAT for a constant number of

measurement-based computation?

- A universal state does not need to be a graph state [Gross and Eisert '06...]
- A definition of universality.
  A family of states $|S_n\rangle$ on n qubits is universal if for any quantum circuit of size m, there exists a measurement-based quantum computation on $|S_n\rangle$, for some $n = poly(m)$, that creates the same state as the final state of the circuit.

- Sharpen the classical upper bound of BQP (the class of functions computable by a polynomial time quantum algorithm).
  - A reduction from a BQP problem to $\exists\forall\cdots SAT$, for a constant number of alternations of the quantifiers?

measurement-based computation?

- A universal state does not need to be a graph state [Gross and Eisert '06...]
- A definition of universality.
  A family of states $|S_n\rangle$ on n qubits is universal if for any quantum circuit of size m, there exists a measurement-based quantum computation on $|S_n\rangle$, for some $n = poly(m)$, that creates the same state as the final state of the circuit.

- Sharpen the classical upper bound of BQP (the class of functions computable by a polynomial time quantum algorithm).
  - A reduction from a BQP problem to $\exists\forall\cdots SAT$, for a constant number of alternations of the quantifiers?

measurement-based computation?

- A universal state does not need to be a graph state [Gross and Eisert '06...]
- A definition of universality.
  A family of states $|S_n\rangle$ on n qubits is universal if for any quantum circuit of size m, there exists a measurement-based quantum computation on $|S_n\rangle$, for some $n = poly(m)$, that creates the same state as the final state of the circuit.

$\exists x_1 x_2 \forall y_1 y_2 \cdots \phi(x_1 \cdots x_n)$

- Sharpen the classical upper bound of BQP (the class of functions computable by a polynomial time quantum algorithm).
  - A reduction from a BQP problem to $\exists\forall\cdots SAT$, for a constant number of alternations of the quantifiers?

measurement-based computation?

- A universal state does not need to be a graph state [Gross and Eisert '06...]
- A definition of universality.
  A family of states $|S_n\rangle$ on n qubits is universal if for any quantum circuit of size m, there exists a measurement-based quantum computation on $|S_n\rangle$, for some $n = poly(m)$, that creates the same state as the final state of the circuit.

$\exists x_1 x_2 \forall y_1 y_2 \cdots \phi(x_1 \cdots x_n)$

constant

- Sharpen the classical upper bound of BQP (the class of functions computable by a polynomial time quantum algorithm).
  - A reduction from a BQP problem to $\exists\forall\cdots SAT$, for a constant number of alternations of the quantifiers?

measurement-based computation?

- A universal state does not need to be a graph state [Gross and Eisert '06...]
- A definition of universality.
  A family of states $|S_n\rangle$ on n qubits is universal if for any quantum circuit of size m, there exists a measurement-based quantum computation on $|S_n\rangle$, for some $n = poly(m)$, that creates the same state as the final state of the circuit.

$\exists$

$\exists x_1 x_2 \forall y_1 y_2 \cdots \exists \cdots \phi(x_1 \cdots x_n)$

constant

- Sharpen the classical upper bound of BQP (the class of functions computable by a polynomial time quantum algorithm).
  - A reduction from a BQP problem to $\exists\forall\cdots SAT$, for a constant number of alternations of the quantifiers?