Message Compression on Anonymous Networks and Its Applications

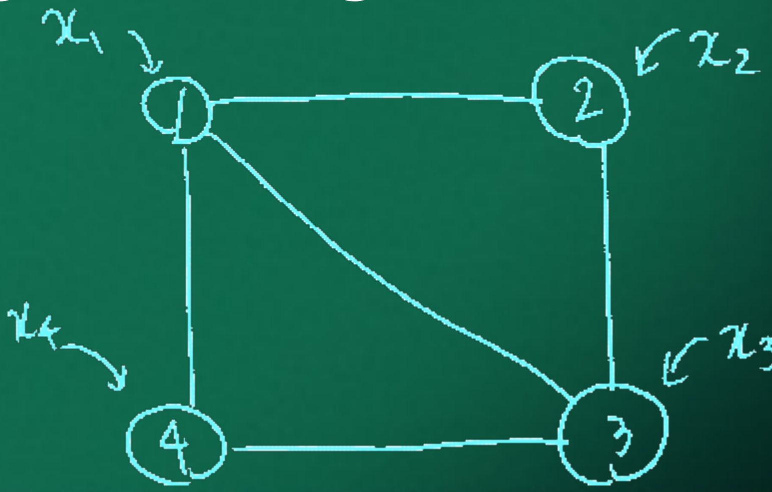
Seiichiro Tani
NTT Communication Science Labs.
NTT Corporation

This talk is based on the following paper: S. Tani. **Compression of View on Anonymous Networks -- Folded View --.** *IEEE Transactions on Parallel and Distributed Systems* **23** 255 - 262 (2012) (A preliminary version is in Sec.5 of arxiv:0712.4213)**.**

Distributed Computing

A model of computation

- There are multiple nodes on a network
- All nodes collaborate to do computational tasks.
- The tasks are normally computing global properties, so message-exchanges are inevitable.



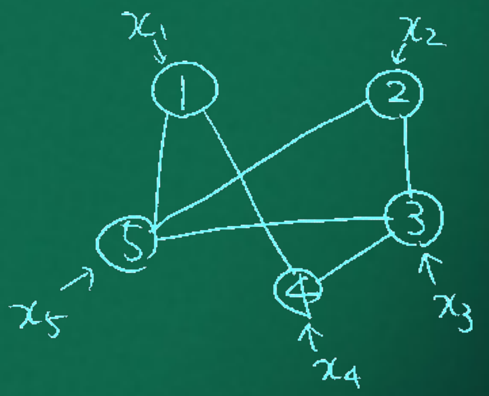
Example:

Every party i gets input $x_i \in \{0,1\}$.

Goal: Compute $MAJ_5(x_1, x_2, x_3, x_4, x_5)$,

where $MAJ_n : \{0,1\}^n \rightarrow \{0,1\}$ is 1 iff the majority of the n bits is 1.

- How many bits are to be communicated?
- How many rounds are required ?



Example (Contd.)

A naïve algorithm:

Every party broadcasts its input attached with its identity.

→Every party will get (possibly multiple copies of) all inputs attached with identities,

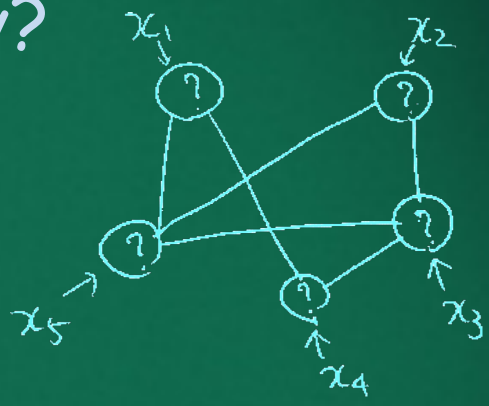
so that

it can locally compute the function.

Note: Broadcast is possible if every party knows an upper bound of the diameter of the underlying graph (without knowing the graph).

Example (Contd.)

Q. What if no party has its identity?



or more formally,

Q: What if all parties with the same number of communication links are identical?

The above naïve algorithm does *not* work!

(Why?) Every party may get multiple copies of a certain input bit, but it cannot tell which bits came from the same party.

Angluin [STOC80]

In fact, many distributed algorithms depend on the fact that

every party has its identity,

as pointed out by Angluin.

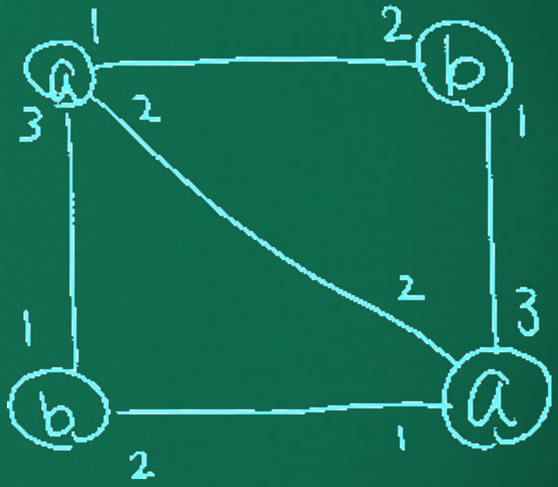
Her question:

How much each party needs to know about

- its identity,
- the identities of other parties, and
- the underlying graph?

The Anonymous Network Model [Ang80]

- It consists of n parties connected by bidirectional communication links, where the underlying graph is an arbitrary connected graph.



- Every party identifies each communication link connected to the party by a local name.

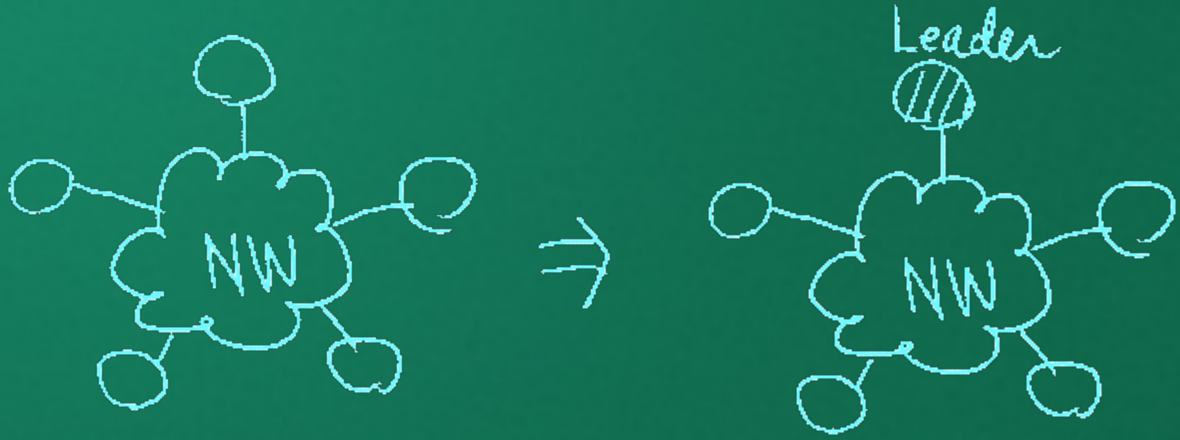
- All parties with the same number of communication links are identical (i.e., they cannot be distinguished).

In other words, they have the same information (before input is given).

Example: Leader Election Problem

Input : n (the number of parties)

Goal : Choose a unique leader from among the n parties.



We consider the exact computation:

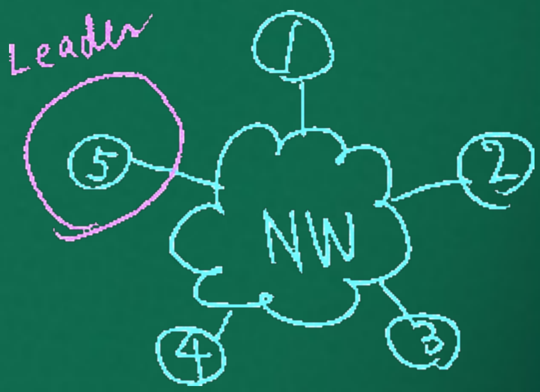
The goal must be achieved without error in a bounded time.

Example (Contd.)

- (Non-anonymous case): Every party has its identity

  Just choose the party with the maximum ID.

  (Note: Numerous works have been improving efficiency)



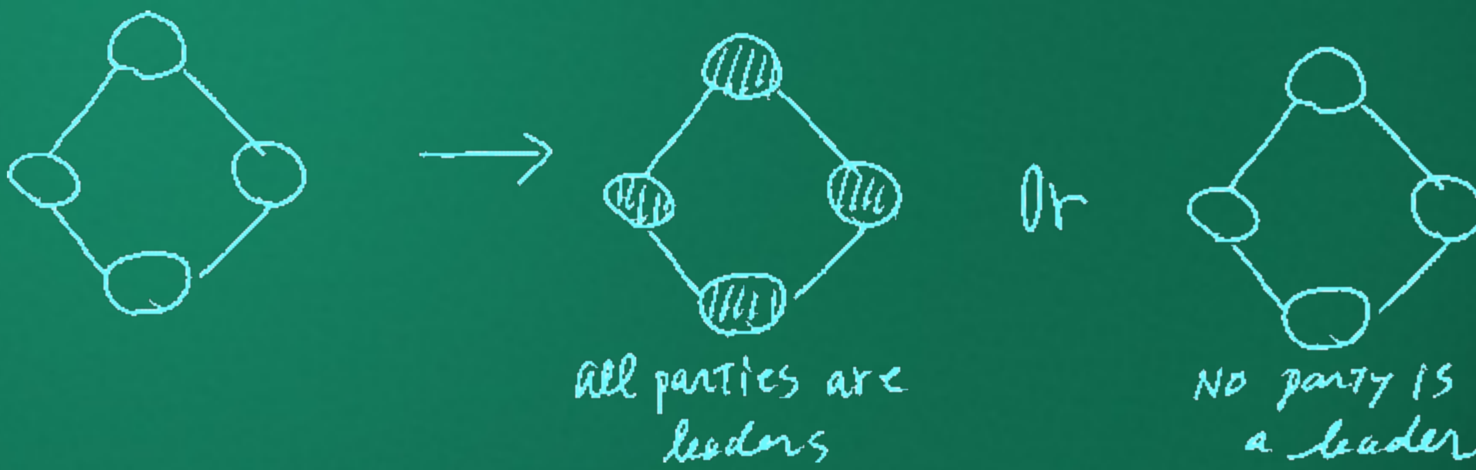
- (Anonymous case):

  Theorem [A80, YK88, BV02]

  No classical algorithm can exactly elect a unique leader for a broad class of graphs.

Idea of Proof (Rings)

- If all parties start with the same state and perform the same deterministic algorithm, then they necessarily end with the same state.

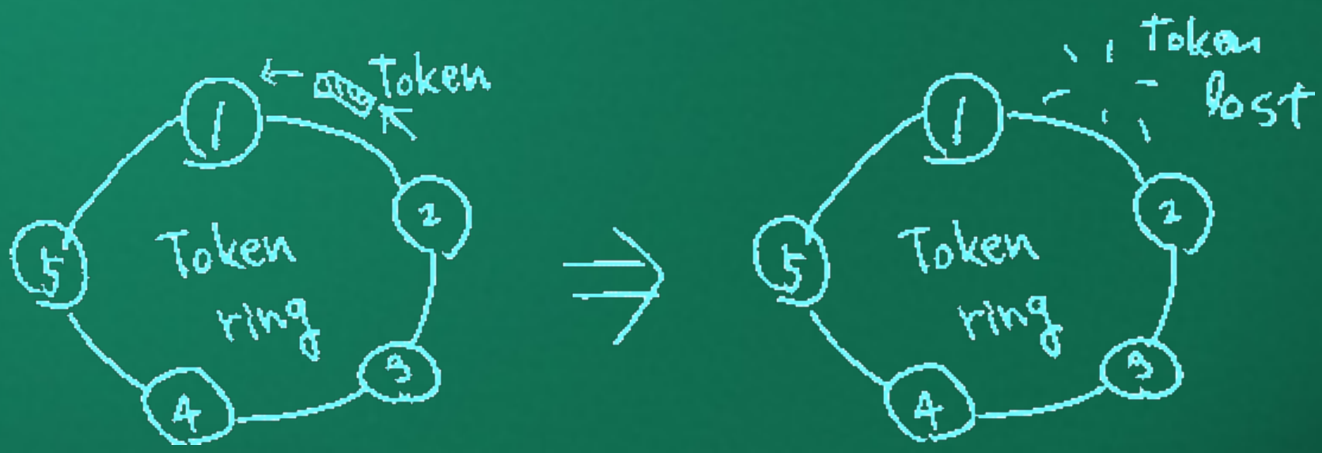


- Even if every party flips coins, there is still non-zero error probability.

Remarks on LE

- Historically, LE modeled the situation where:

on a token ring network, we need to exactly choose one party that will be in charge of recovering the token, when a token was lost.



Remarks on LE (Contd.)

- Once a unique leader is elected, it is possible to efficiently
  - Solve a lot of problems (e.g., constructing a spanning tree),
  - assign an identifier to each party, implying

    the leader can make anonymous-NW non-anonymous
- In this sense, LE is not just an example, but the most fundamental problem

  (Indeed, Angluin studied LE in her seminal paper)

View

View is a data structure

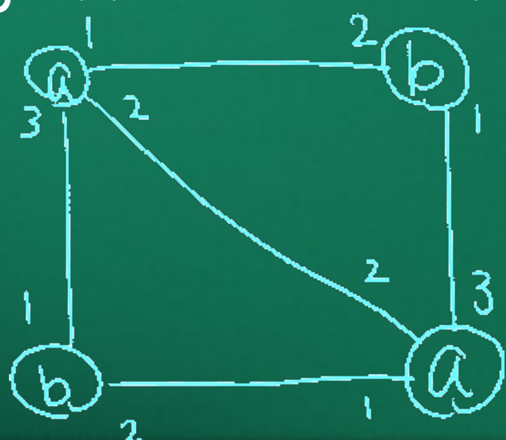
- that contains all the information that every party can obtain by exchanging messages.
- that can be constructed in a distributed way, which yields a generic algorithm.

View (Contd.)

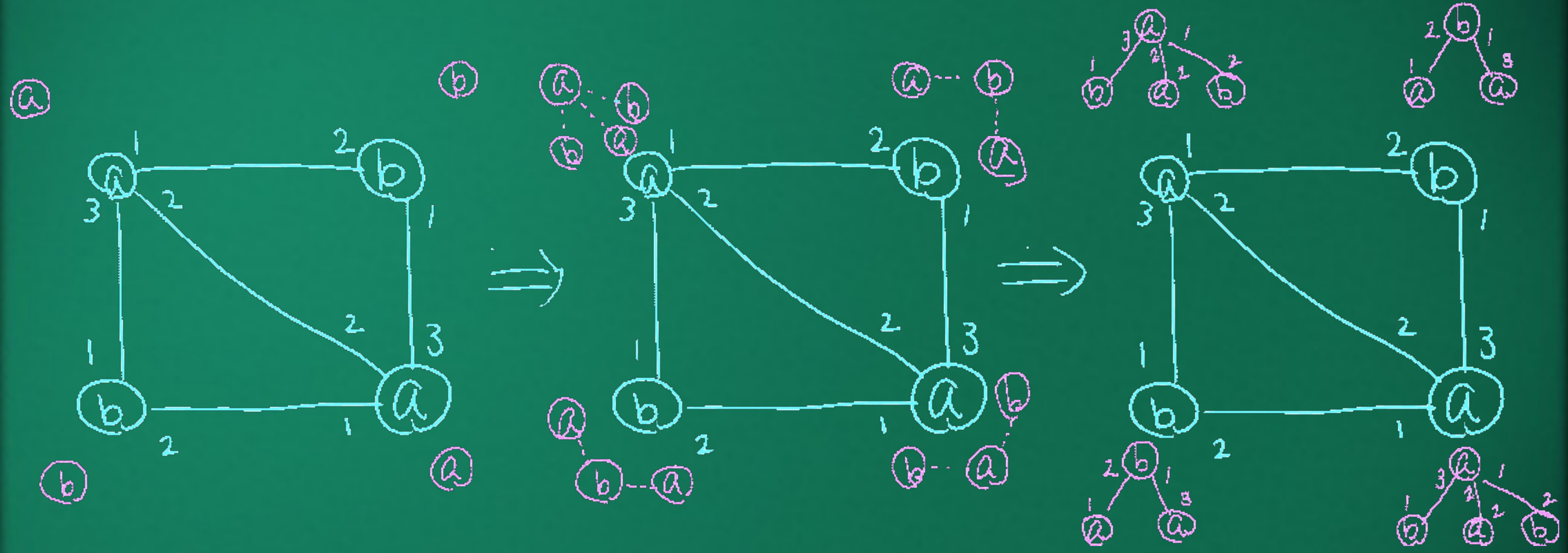
Slightly more formally,

- A view T(v) is an infinite-depth tree defined for each node v,
- It is obtained by sharing the common prefix of any two paths starting from v.

$T^h(v)$ denotes the subgraph of T(v), which obtained by cutting off all nodes of depth more than h and associated edges.

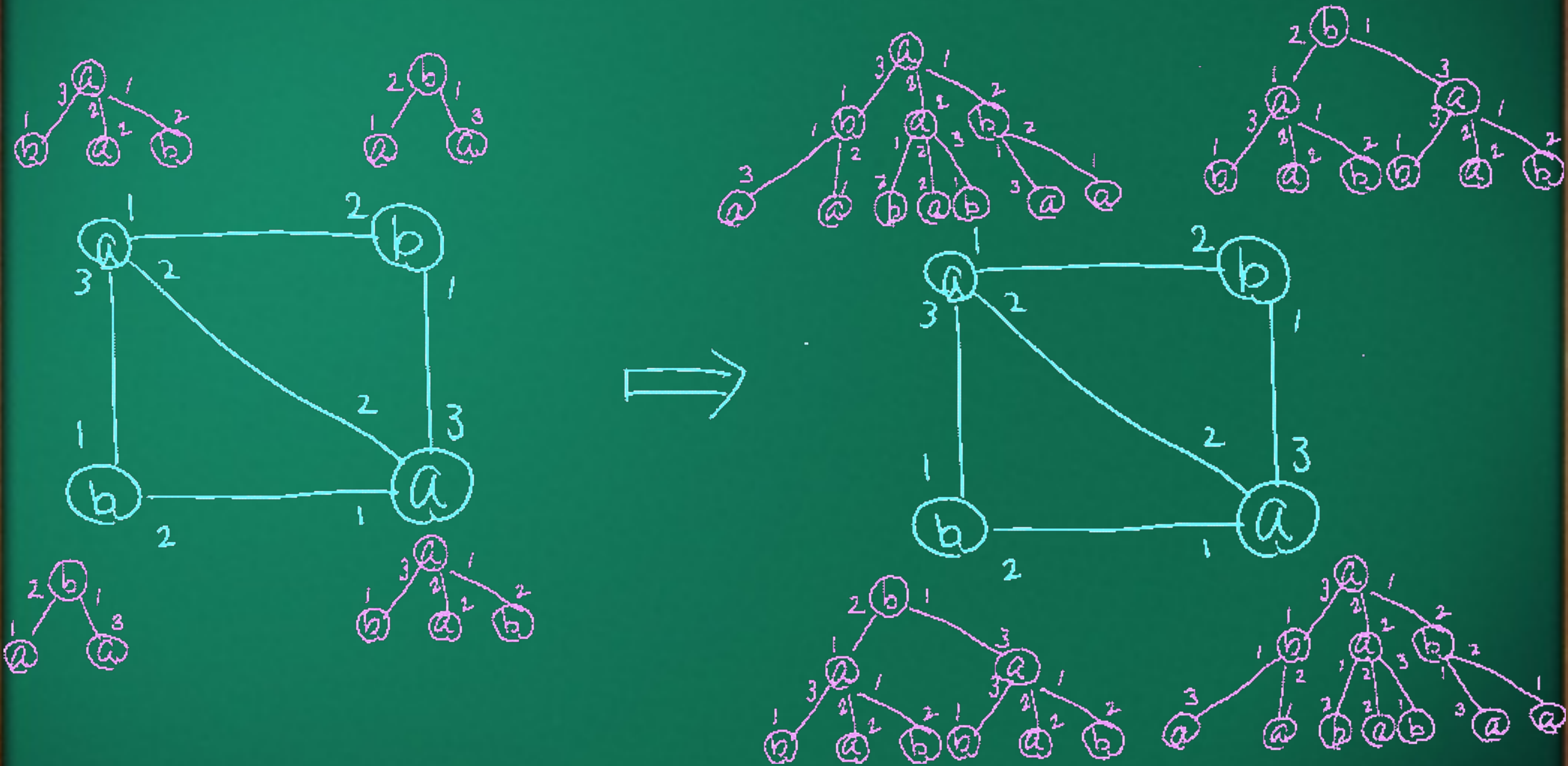


Construction of View



Note: By exchanging one message with each neighbor, every party can know the labels of both ends of each communication link incident to the party.

Construction of View



After h rounds, every party v obtains $T^h(v)$.

Constructing View

Every party v performs the following steps.

1. Create $T^0(v)$ and send it to each neighbor.
2. For k:=1 to h
   2.1 Receive $T^{k-1}(w)$ from each neighbor w.
   2.2 Construct $T^k(v)$ from the $T^{k-1}(w)$'s
   2.3 If k=h, then halt; Otherwise send $T^k(v)$ to each neighbor.

Generic Algorithm

Claim: $T^h(v)$ has all the information v can obtain in h rounds.

Once every party v obtains $T^h(v)$, v can simulate (locally) any deterministic distributed algorithm that works in h rounds.

Q. How deep should view be to cover all deterministic algorithms? Infinitely deep?

A. No.

Norris's Theorem

Theorem (Norris 95)

If the underlying graph G has n nodes, it holds that for any two nodes v and v', $T^{n-1}(v) \equiv T^{n-1}(v')$ iff $T(v) \equiv T(v')$.

Thus the subgraph $T^{n-1}(v)$ up to $(n-1)$ depth suffices to simulate every deterministic algorithms

Example: LE

Recall that $T^{n-1}(v)$ contains all the information that v can gather.

If $T^{n-1}(v) \equiv T^{n-1}(v')$, then v and v' can not be distinguished,

which means:

v is elected as a leader iff so is v'

(i.e., leader election fails).

Therefore:

If LE can exactly be solved,

there exists a party v such that $T^{n-1}(v) \not\equiv T^{n-1}(v')$ for any other party v'

Example (LE) Contd.

Conversely, suppose that there exists v such that $T^{n-1}(v) \not\equiv T^{n-1}(v')$ for any other party v′.

Then a unique leader can be elected as one with lexicographically small view among such v (we will see how to identify such a view).

$\boxed{\text{LE can exactly be elected iff there exists v such that } T^{n-1}(v) \not\equiv T^{n-1}(v') \text{ for any other v′.}}$

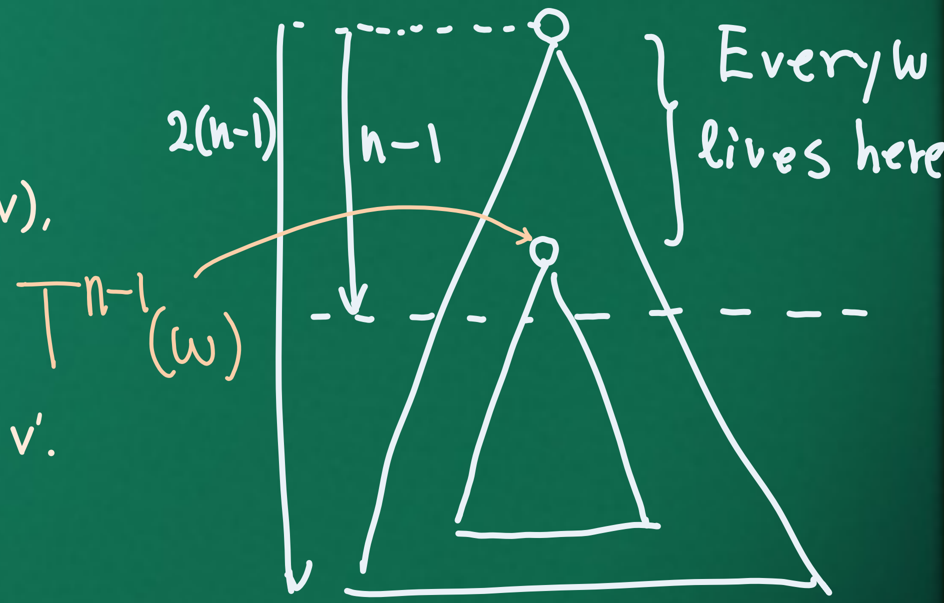
Example (LE) Contd.

Let us construct $T^{2(n-1)}(v)$, which contains $T^{n-1}(w)$ for all w in the underlying graph G.

Bad idea:

(1) Compute a multi-set $\{T^{n-1}(w)$: w in $G\}$ from $T^{2(n-1)}(v)$,

(2) Check if there is a party v such that $T^{n-1}(v) \not\equiv T^{n-1}(v')$ for any other v'.



∵ we do not know how to pick up exactly one view for each party.

Example (LE) Contd.

Theorem [YK88] The multi-set $\{T^{n-1}(w)$: w in $G\}$, can be partitioned into equivalence classes of the same size, where the equivalence relation is defined by isomorphism.

⇩

There exists v such that $T^{n-1}(v) \not\equiv T^{n-1}(v')$ for all $v' \neq v$
iff $T^{n-1}(v) \not\equiv T^{n-1}(v')$ for every two parties v and v' with $v \neq v'$.

⇩

It suffices to count the number of non-isomorphic views of depth n-1 among those contained in $T^{2(n-1)}(v)$.

Example: LE (Contd.)

Once every party has constructed $T^{2(n-1)}(v)$, they can solve LE in such a way that

- Elect a unique leader if it is possible
- Declare ``impossible'' otherwise.

in O(n) rounds and exp(n) bit complexity.

Any anonymous network on which LE can be solved can be made non-anonymous with the above cost.

Improvements

- [KKvdB94]: $O(n^2)$ rounds, poly(n) bit complexity.
  - Note for specific graphs, there are more efficient algorithms

same order

- Ours: Optimal $O(n)$ rounds, poly (n) bit complexity.

Our Idea

Recall a view is the tree obtained by sharing the common prefix of every pair of paths starting at a fixed node.

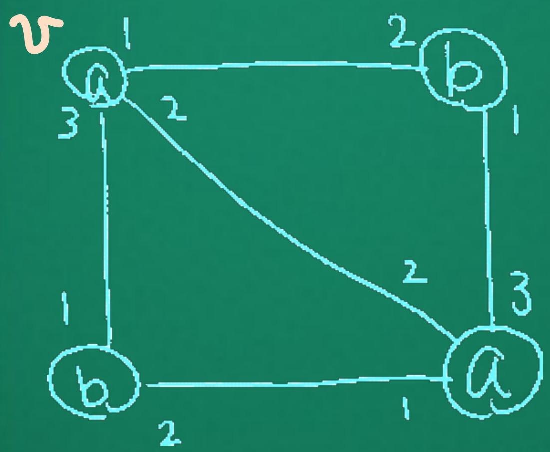
Let us regard view as a rooted directed tree (every edge is destined toward leaves).

Our idea:

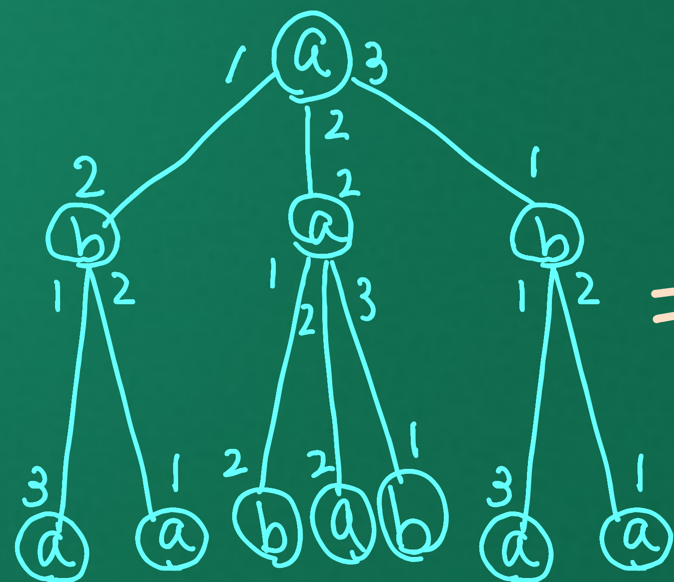
Maximally share all isomorphic subtrees rooted at nodes of the same level.

Our Idea (Contd.)

A straight-forward generalization of reducing the BT representing a Boolean function to the quasi-reduced OBDD.

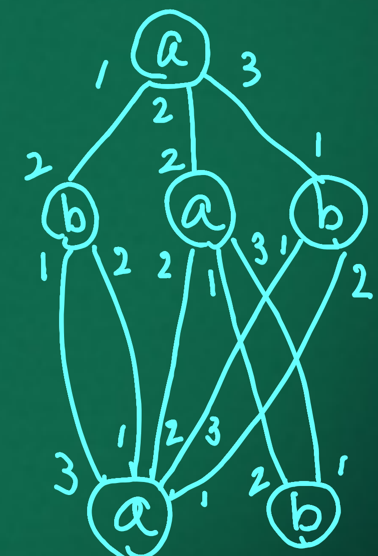


Underlying graph G



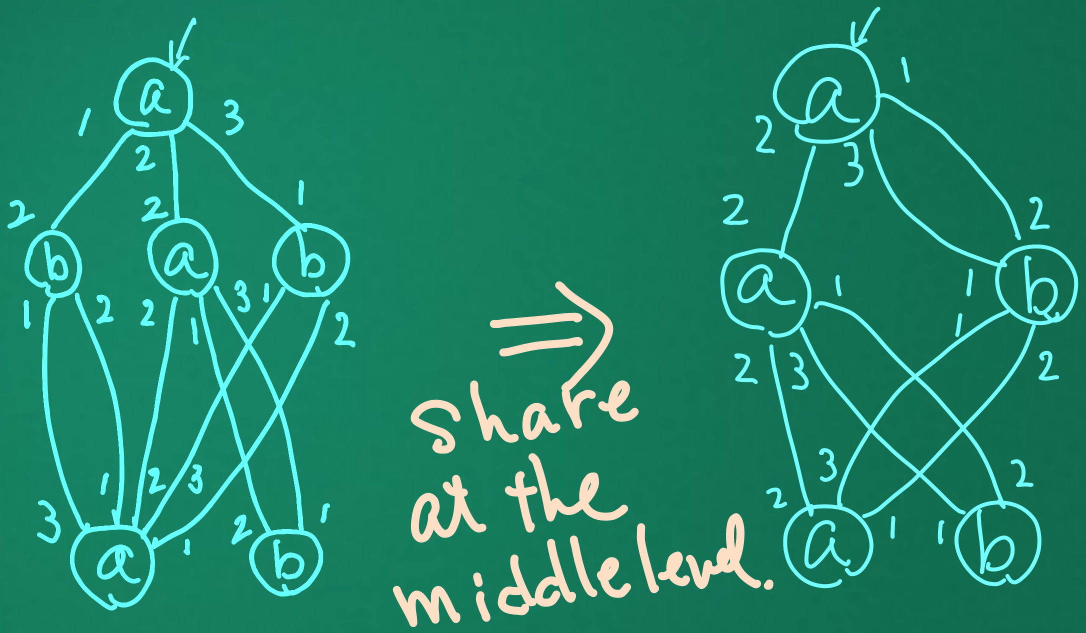
View $T^2(v)$

⇒



Share at the bottom level

Our Idea (Contd.)



A folded-view (f-view) is defined as a directed graph obtained by sharing isomorphic subgraphs rooted at the same level of a view in the way above.

Uniqueness and Size of f-view

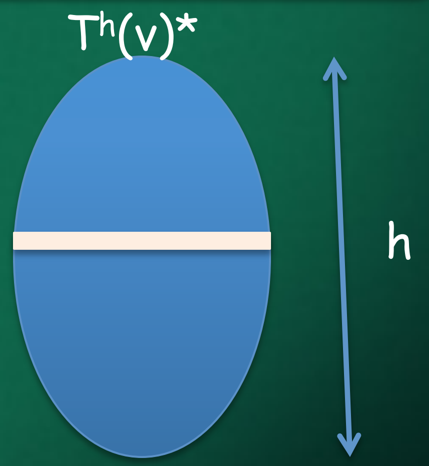
Theorem:

For any view $T^h(v)$, there exists a minimal f-view $T^h(v)^*$ that is unique up to isomorphism.

There are at most $O(hn)$ nodes and $O(h\Delta n)$ edges in $T^h(v)^*$, where Δ is the maximum degree over all nodes of the underlying graph.

Proof (Sketch):

The number of nodes at each level is at most n.

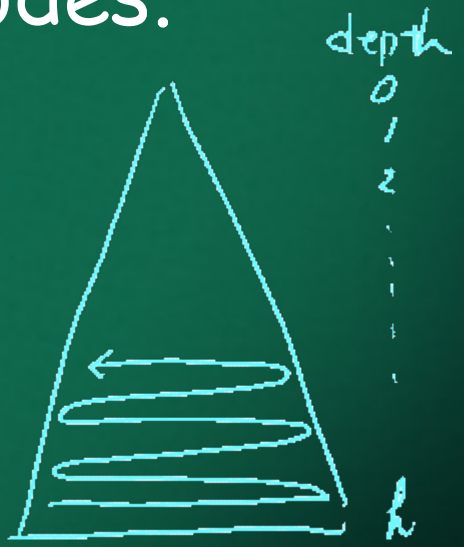


The minimization algorithm

For d:=h down to 1

Repeatedly merge every two nodes at depth d that are the roots of isomorphic subgraphs,

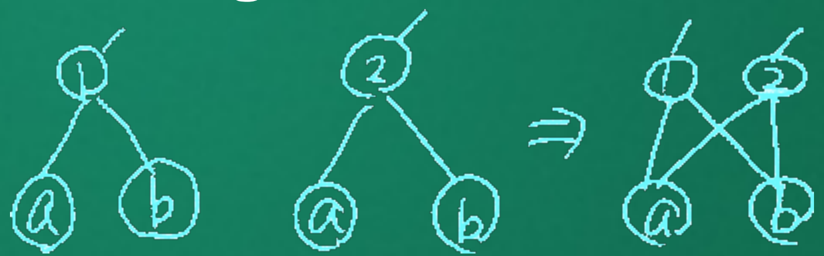
until there are no such two nodes.

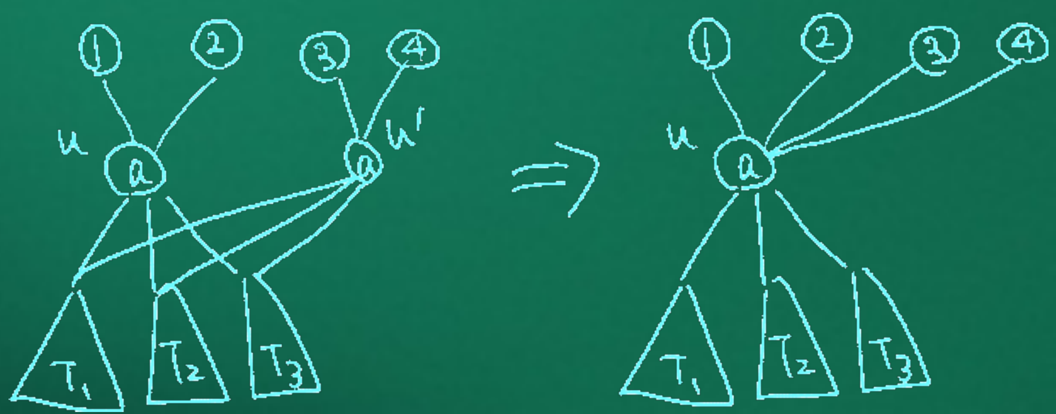


Merging Operation

(Assume we proceed level by level in a bottom-up manner)

If nodes u and u′ at the same depth of a (f-)view are the roots of two isomorphic subgraphs, then merge u and u′.





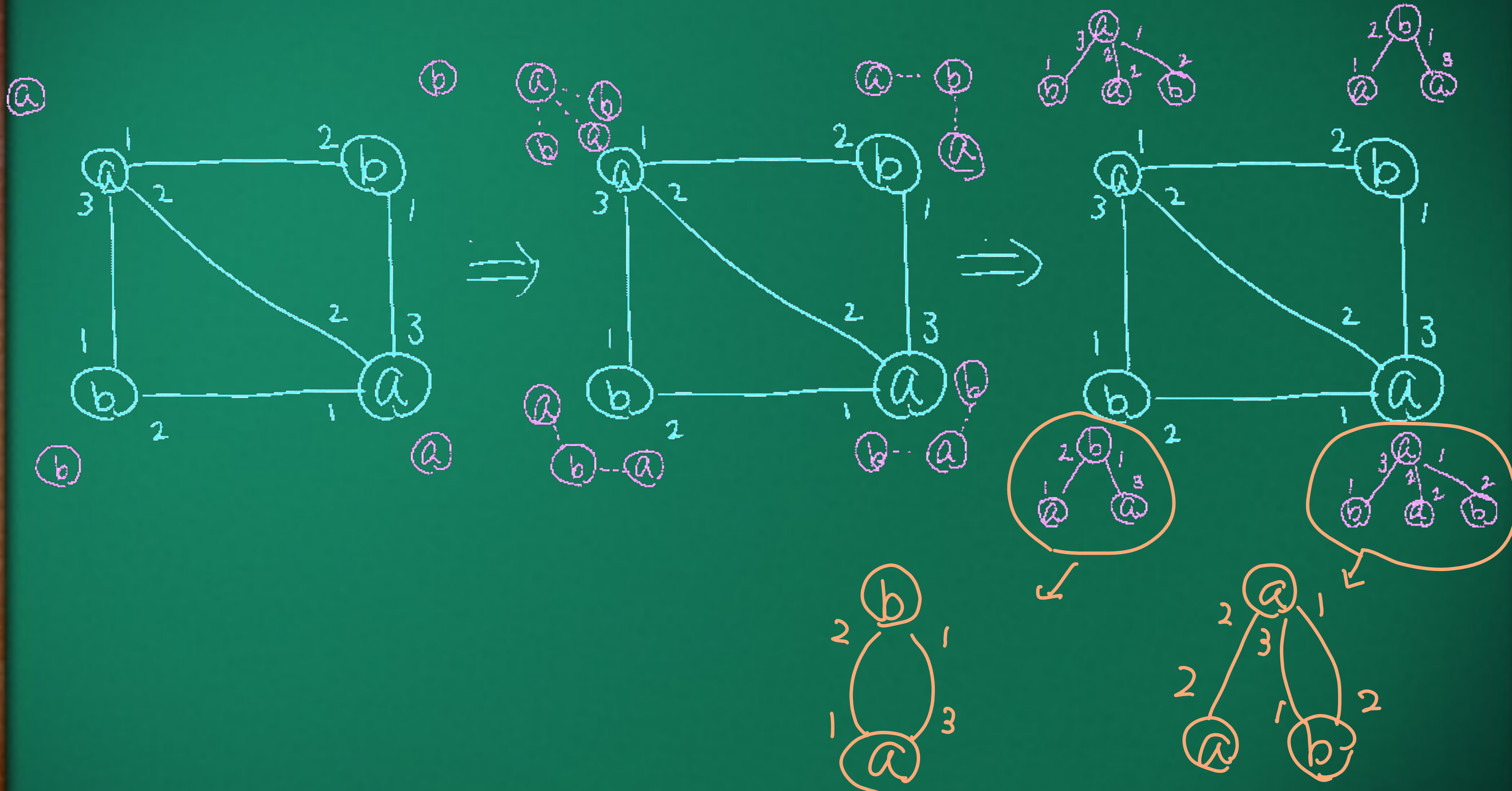
(port number omitted.)

Complexity of Minimizing View

Lemma

For a given (f-)view, the minimization algorithm outputs the (unique) minimal view, with time complexity $O(|V|(\log |V|)(\log |U| + \Delta \log(n|V|))$ where V is the node set, U is the set of node labels, Δ is the maximum node degree, of the underlying graph.

Distributed f-view Construction Algorithm



In each round d, send minimized f-views of depth d.

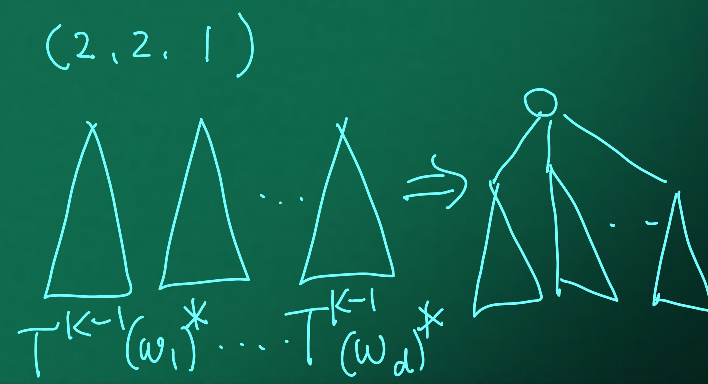
Distributed f-view Construction Algorithm (contd)

Every party v performs the following steps.

1. Create $T^0(v)^*$ and send it to each neighbor.
2. For k:=1 to h
   - 2.1 Receive $T^{k-1}(w)^*$ from each neighbor w.
   - 2.2 Construct $T^k(v)^*$ from the $T^{k-1}(w)^*$'s
   - 2.3 If k=h, then halt; Otherwise send $T^k(v)^*$ to each neighbor.

2.2.1 Construct an f-view T of depth k from $T^{k-1}(w)^*$'s

2.2.2 Minimize T to get $T^k(v)^*$



Distributed f-view Construction Algorithm (contd)

The point:

In each round,

1. every party sends copies of a minimized f-view to achieve polynomial bit complexity.

2. $T^k(v)^*$ is constructed from the $T^{k-1}(w)^*$'s without unfolding $T^{k-1}(w)^*$'s to achieve polynomial time complexity.

Complexity of f-view Construction

Theorem (Main)

Let U be the set of node labels.

For a distributed network with n parties,

there is an algorithm that constructs the minimal f-view of depth $h \in O(n)$

- in $h+O(1)$ rounds and
- $O(\Delta h^2 n (\log n) \log(|U|n^\Delta))$ time for each party
- with $O(mh^2 n \log(|U|\Delta^\Delta)$-bit communication over all parties,

where m and Δ are the numbers of edge and the maximum node degree, respectively, of the underlying graph.

Application of f-view

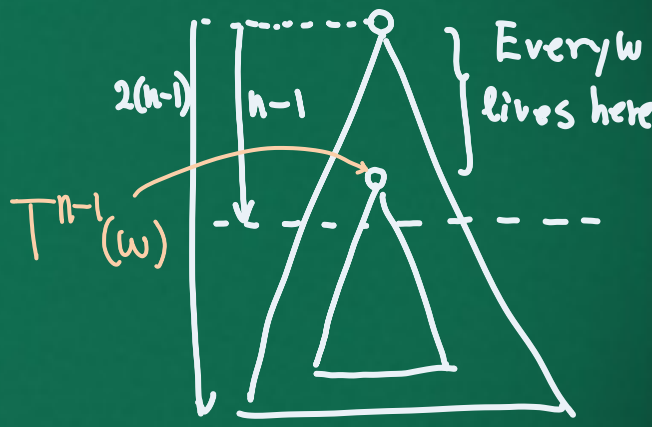
Recall that view $T^{(n-1)}(v)$ of depth n-1 has all the information that party v can gather.

So is the corresponding f-view $T^{(n-1)}(v)^*$, since it is a ``loss-less'' compressed form of $T^{(n-1)}(v)$.

Q. How efficiently can we extract what we want from an f-view?

Application to LE

Recall that it suffices to count the number of non-isomorphic views of depth n-1 among those in $T^{2(n-1)}(v)$.



Q. How efficiently can we count the number for given f-view?

A naïve way is to unfold the f-view.

- Exponential time.

Application to LE

Let $\Gamma^{n-1} \equiv \{T^{n-1}(v)$: v in $G\}$ be the set of non-isomorphic views of depth n-1.

Theorem: There is an algorithm that outputs $|\Gamma^{n-1}|$ in poly(n) time for a given minimal f-view $T^{2(n-1)}(v)$.

Corollary: There is an algorithm that, for the given number n of parties,

elects a unique leader if it is possible,

and declares ``impossible" othewise,

in O(n) rounds and poly(n) time with poly(n) bit-complexity.

Application to Symmetric Boolean Functions

A Boolean function is said to be symmetric if it is invariant under any permutation over its variables.

Equivalently, it depends only on the Hamming weight of its input bit-string.

Let $x \in \{0,1\}^n$.

e.g.) $OR(x)=1$ iff $|x|\geq 1$. $Maj(x)=1$ iff $|x|\geq\lceil n/2\rceil$.

$PARITY(x) \equiv x_1 \oplus \cdots \oplus x_n = 1$ iff $|x|$ is odd.

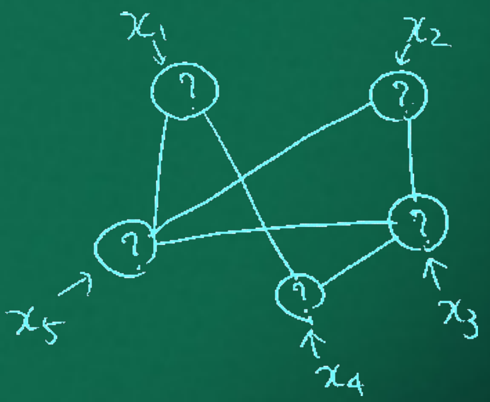
Application to Symmetric Boolean Functions

Suppose that on an anonymous network, every party *i* is given $x_i \in \{0,1\}$ as input.

(Note *i* is used just for explanation, and it is not an identity.)

Goal: Compute $f(x_1,\ldots,x_n)$ for any symmetric Boolean function.

(Note that f is well-defined, since
f does not depend how to index variables.)



Application to Symmetric Boolean Functions

Theorem[Yk88] There is a distributed algorithm that exactly computes any symmetric Boolean function on any anonymous network, if the number n of parties is given to every party:
Constructing $T^{2(n-1)}(v)$ for each party v in O(n) rounds with exp(n) bit-complexity.

By using folded views, we have:

Constructing $T^{2(n-1)}(v)$ for each party v in O(n) rounds with ~~exp(n)~~ bit-complexity.
poly(n)

Quantum network model

The difference is

- Every party can send quantum messages,
- Every party can locally performs quantum computation.

Note that classical communication and computation can be simulated by quantum equivalents.

A major goal of quantum information researchers is to exhibit the advantages of quantum models over classical counterparts.

Separation between quantum and classical

Theorem [A80, YK88, BV02]
No classical algorithm can exactly elect a unique leader for a broad class of graphs, even if the number n of parties is given to every party.

In the quantum case, the situation is remarkably different.

Theorem[TKM05, KMT10, TKM12] There is a quantum algorithm that exactly elects a unique leader on any anonymous network in $O(n \log n)$ rounds with poly(n) time and bit complexity, if the number n of parties is given.

Complexity of Quantum LE

- Undirected graph:
  - $O(n)$ rounds
  - $O(n^4)$ bit complexity.

  The algorithm does not use f-view,

  but it does not work for directed graphs.

- General graph (directed or undirected):
  - $O(n \log n)$ rounds
  - $O(n^6 (\log n)^2)$ bit complexity

  The algorithm needs f-view.

How is f-view used for LE?

How does the algorithm proceed?

- Initially, every party is a candidate of the unique leader.
- The algorithm has log n stages: candidates are reduced by a factor $\frac{1}{2}$ or less.

Where does f-view appear?

- The algorithm uses f-view to compute MAJ and PARITY.
- It is unknown how to compute MAJ and PARITY on arbitrary directed graphs without using (f-)views.

Summary

- View is a fundamental tool for studying distributed computing on anonymous networks.
- The idea of ROBDDs can be applied to compressing view, improving complexity.
- The compressed view plays an important role in quantum as well as classical settings.