Determining majority in networks with local interactions and very small local memory

George B. Mertzios[1]   Sotiris E. Nikoletseas[2]
Christoforos L. Raptopoulos[2,3]   Paul G. Spirakis[2,4]

[1]School of Engineering and Computing Sciences, Durham University, UK

[2]Computer Technology Institute (CTI) and University of Patras, Greece

[3]Computer Science Department, University of Geneva, Switzerland

[4]Department of Computer Science, University of Liverpool, UK

ICALP 2014

Algorithms and Complexity in Durham (ACiD) Seminar,
Durham University, November 2014

Consensus in distributed systems

In distributed systems:

- a collection of n independent entities (or nodes)
- entities interact / exchange messages to coordinate their actions
- interactions must satisfy some constraints, e.g.:
  - synchronous vs. asynchronous,
  - not every entity can interact with all others (network structure),
  - how often two specific entities may interact, etc.

Consensus in distributed systems

In distributed systems:

- a collection of n independent entities (or nodes)
- entities interact / exchange messages to coordinate their actions
- interactions must satisfy some constraints, e.g.:
  - synchronous vs. asynchronous,
  - not every entity can interact with all others (network structure),
  - how often two specific entities may interact, etc.

A central problem in distributed systems:

Definition (Consensus)

Let each node have an input value. A solution for the consensus problem must guarantee:

- **Termination:** every node eventually decides on some value,
- **Agreement:** all nodes decide on the same value,
- **Validity:** the decided value must be the input of some node.

Consensus in distributed systems

Many applications of the consensus problem, e.g.:

- leader election
- distributed ranking [Jung et al., *ISIT*, 2012]

The majority problem:

- a natural special case of the consensus problem
- the agreed value must be the input value of the majority of the nodes
- two or more different input values (or colors)
  [Angluin et al., *Distributed Computing*, 2008]
  [Becchetti et al., *SPAA*, 2014]
- many applications, e.g.:
  - voting [Kearns et al., *WINE*, 2008]
  - epidemiology and interacting particle systems
    [Liggett, *Interacting Particle Systems*, 2004]
  - social networks [Mizrachi, *MSc thesis*, Ben-Gurion University, 2013]
    [Mossel et al., *Auton. Agents & Multi-Agent Systems*, 2014]

Computing the majority

- To solve the majority problem in a network:
    - we need assumptions on the model of computation

- In the "traditional" settings: "strong" models
    - central authority, unlimited memory, full information about the network
    - efficiently computable
    - the goal is to minimize the number of comparisons
      [Saks et al., *Combinatorica*, 1991]
      [De Marco et al., *Combinatorics, Probability and Computing*, 2006]

Computing the majority

- To solve the majority problem in a network:
  - we need assumptions on the model of computation

- In the "traditional" settings: "strong" models
  - central authority, unlimited memory, full information about the network
  - efficiently computable
  - the goal is to minimize the number of comparisons
    [Saks et al., *Combinatorica*, 1991]
    [De Marco et al., *Combinatorics, Probability and Computing*, 2006]

- In "modern" settings: "weaker" models
  - no central authority, limited memory, partial or no information
  - a node does not know:
    - its own identity
    - the identities of the nodes it can interact with (i.e. its neighbors)
    - when it will interact with other nodes
  - one way to model such systems is using population protocols

Population protocols

- Population V of $|V| = n$ entities (i.e. nodes)
- A population protocol $\mathcal{A}$ consists of:
  - finite input and output alphabets X and Y
  - a finite set of states Q
  - an input function $I : X \to Q$
  - an output function $O : Q \to Y$
  - a transition function $\delta : Q \times Q \to Q \times Q$

Population protocols

- Population V of $|V| = n$ entities (i.e. nodes)
- A population protocol $\mathcal{A}$ consists of:
  - finite input and output alphabets X and Y
  - a finite set of states Q
  - an input function $I : X \to Q$
  - an output function $O : Q \to Y$
  - a transition function $\delta : Q \times Q \to Q \times Q$
- The result of an interaction between nodes u and v depends only on their states q_u and q_v

Population protocols

- Population V of $|V| = n$ entities (i.e. nodes)
- A population protocol $\mathcal{A}$ consists of:
  - finite input and output alphabets X and Y
  - a finite set of states Q
  - an input function $I : X \rightarrow Q$
  - an output function $O : Q \rightarrow Y$
  - a transition function $\delta : Q \times Q \rightarrow Q \times Q$
- The result of an interaction between nodes u and v depends only on their states q_u and q_v
- A population protocol is symmetric if interactions have no "direction":
  - $\delta(q_u, q_v) = (q'_u, q'_v) \iff \delta(q_v, q_u) = (q'_v, q'_u)$,
    for every pair of states $q_u, q_v \in Q$

Population protocols

- Population V of $|V| = n$ entities (i.e. nodes)
- A population protocol $\mathcal{A}$ consists of:
  - finite input and output alphabets X and Y
  - a finite set of states Q
  - an input function $I : X \to Q$
  - an output function $O : Q \to Y$
  - a transition function $\delta : Q \times Q \to Q \times Q$
- The result of an interaction between nodes u and v depends only on their states q_u and q_v
- A population protocol is symmetric if interactions have no "direction":
  - $\delta(q_u, q_v) = (q'_u, q'_v) \iff \delta(q_v, q_u) = (q'_v, q'_u)$,
    for every pair of states $q_u, q_v \in Q$
- Otherwise, for every interaction, one of the nodes is the initiator

Terminology:

- The interaction order is chosen by an adversary (scheduler)
- To allow meaningful computations: scheduler must be fair
  - we do not allow avoidance of a possible step forever
  - for any two state configurations C_1, C_2, where C_2 is reachable from C_1: if C_1 occurs infinitely often $\Rightarrow$ C_2 also occurs infinitely often

Terminology:

- The interaction order is chosen by an adversary (scheduler)
- To allow meaningful computations: scheduler must be fair
  - we do not allow avoidance of a possible step forever
  - for any two state configurations C_1, C_2, where C_2 is reachable from C_1: if C_1 occurs infinitely often $\Rightarrow$ C_2 also occurs infinitely often
- The interaction graph $G = (V, E)$ of the population:
  - the entities of the population are arranged on the nodes V
  - only neighboring nodes are allowed to interact
- The probabilistic scheduler:
  - a special case of a fair scheduler
  - directed case: every directed edge (u, v) is chosen uniformly at random (u is the initiator)
  - undirected case: replace edge $\{u, v\}$ by directed edges $(u, v), (v, u)$

Terminology:

Definition

Given the probabilistic scheduler, a population protocol $\mathcal{A}$ computes a function g with error probability ε if for every input configuration C_0 the population eventually reaches a configuration C such that with probability at least $1-\varepsilon$:

(a) all nodes have output $g(C_0)$

(b) this remains true for any configuration reachable from C

Terminology:

Definition

Given the probabilistic scheduler, a population protocol $\mathcal{A}$ computes a function g with error probability ε if for every input configuration C_0 the population eventually reaches a configuration C such that with probability at least $1 - \varepsilon$:

(a) all nodes have output $g(C_0)$

(b) this remains true for any configuration reachable from C

Definition

A population protocol $\mathcal{A}$ stably computes a function g if for every fair scheduler the population eventually reaches a configuration C that satisfies both (a) and (b).

Population protocols for computing the majority

- Computing the majority in distributed settings has been mainly studied in homogeneous populations (i.e. the complete graph)
- The following simple 3-state population protocol was introduced in [Angluin et al., *Distributed Computing*, 2008]
  - initially nodes have 2 possible states: **r** and **g**
  - during the execution, a node can have 3 possible states: **r**, **g**, and **b**
  - interactions are dictated by the probabilistic scheduler

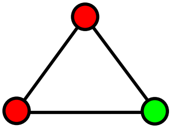
Population protocols for computing the majority

- Computing the majority in distributed settings has been mainly studied in homogeneous populations (i.e. the complete graph)
- The following simple 3-state population protocol was introduced in [Angluin et al., *Distributed Computing*, 2008]
  - initially nodes have 2 possible states: **r** and **g**
  - during the execution, a node can have 3 possible states: **r**, **g**, and **b**
  - interactions are dictated by the probabilistic scheduler
  - the 3×3 transition table can be summarized as follows:
    - node u of state **r** "hits" node v of state **g** $\Rightarrow$ v comes to state **b**
    - node u of state **g** "hits" node v of state **r** $\Rightarrow$ v comes to state **b**
    - node u of state **r** / **g** "hits" node v of state **b** $\Rightarrow$ v comes to state **r** / **g**

Population protocols for computing the majority

- Computing the majority in distributed settings has been mainly studied in homogeneous populations (i.e. the complete graph)
- The following simple 3-state population protocol was introduced in [Angluin et al., *Distributed Computing*, 2008]
  - initially nodes have 2 possible states: **r** and **g**
  - during the execution, a node can have 3 possible states: **r**, **g**, and **b**
  - interactions are dictated by the probabilistic scheduler
  - the 3×3 transition table can be summarized as follows:
    - node u of state **r** "hits" node v of state **g** $\Rightarrow$ v comes to state **b**
    - node u of state **g** "hits" node v of state **r** $\Rightarrow$ v comes to state **b**
    - node u of state **r** / **g** "hits" node v of state **b** $\Rightarrow$ v comes to state **r** / **g**

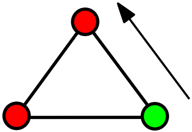
Example:



Population protocols for computing the majority

- Computing the majority in distributed settings has been mainly studied in homogeneous populations (i.e. the complete graph)
- The following simple 3-state population protocol was introduced in [Angluin et al., *Distributed Computing*, 2008]
  - initially nodes have 2 possible states: **r** and **g**
  - during the execution, a node can have 3 possible states: **r**, **g**, and **b**
  - interactions are dictated by the probabilistic scheduler
  - the 3×3 transition table can be summarized as follows:
    - node u of state **r** "hits" node v of state **g** $\Rightarrow$ v comes to state **b**
    - node u of state **g** "hits" node v of state **r** $\Rightarrow$ v comes to state **b**
    - node u of state **r** / **g** "hits" node v of state **b** $\Rightarrow$ v comes to state **r** / **g**

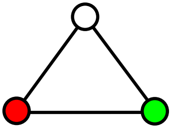
Example:



Population protocols for computing the majority

- Computing the majority in distributed settings has been mainly studied in homogeneous populations (i.e. the complete graph)
- The following simple 3-state population protocol was introduced in [Angluin et al., *Distributed Computing*, 2008]
  - initially nodes have 2 possible states: **r** and **g**
  - during the execution, a node can have 3 possible states: **r**, **g**, and **b**
  - interactions are dictated by the probabilistic scheduler
  - the 3×3 transition table can be summarized as follows:
    - node u of state **r** "hits" node v of state **g** $\Rightarrow$ v comes to state **b**
    - node u of state **g** "hits" node v of state **r** $\Rightarrow$ v comes to state **b**
    - node u of state **r** / **g** "hits" node v of state **b** $\Rightarrow$ v comes to state **r** / **g**

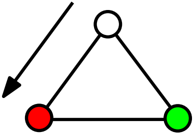
Example:



Population protocols for computing the majority

- Computing the majority in distributed settings has been mainly studied in homogeneous populations (i.e. the complete graph)
- The following simple 3-state population protocol was introduced in [Angluin et al., *Distributed Computing*, 2008]
  - initially nodes have 2 possible states: **r** and **g**
  - during the execution, a node can have 3 possible states: **r**, **g**, and **b**
  - interactions are dictated by the probabilistic scheduler
  - the 3×3 transition table can be summarized as follows:
    - node u of state **r** "hits" node v of state **g** $\Rightarrow$ v comes to state **b**
    - node u of state **g** "hits" node v of state **r** $\Rightarrow$ v comes to state **b**
    - node u of state **r** / **g** "hits" node v of state **b** $\Rightarrow$ v comes to state **r** / **g**

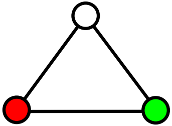
Example:



Population protocols for computing the majority

- Computing the majority in distributed settings has been mainly studied in homogeneous populations (i.e. the complete graph)
- The following simple 3-state population protocol was introduced in [Angluin et al., *Distributed Computing*, 2008]
  - initially nodes have 2 possible states: **r** and **g**
  - during the execution, a node can have 3 possible states: **r**, **g**, and **b**
  - interactions are dictated by the probabilistic scheduler
  - the 3×3 transition table can be summarized as follows:
    - node u of state **r** "hits" node v of state **g** $\Rightarrow$ v comes to state **b**
    - node u of state **g** "hits" node v of state **r** $\Rightarrow$ v comes to state **b**
    - node u of state **r** / **g** "hits" node v of state **b** $\Rightarrow$ v comes to state **r** / **g**

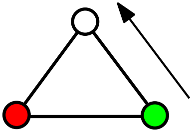
Example:



Population protocols for computing the majority

- Computing the majority in distributed settings has been mainly studied in homogeneous populations (i.e. the complete graph)
- The following simple 3-state population protocol was introduced in [Angluin et al., *Distributed Computing*, 2008]
  - initially nodes have 2 possible states: **r** and **g**
  - during the execution, a node can have 3 possible states: **r**, **g**, and **b**
  - interactions are dictated by the probabilistic scheduler
  - the 3×3 transition table can be summarized as follows:
    - node u of state **r** "hits" node v of state **g** $\Rightarrow$ v comes to state **b**
    - node u of state **g** "hits" node v of state **r** $\Rightarrow$ v comes to state **b**
    - node u of state **r** / **g** "hits" node v of state **b** $\Rightarrow$ v comes to state **r** / **g**

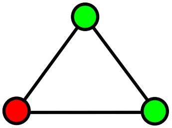
Example:



Population protocols for computing the majority

- Computing the majority in distributed settings has been mainly studied in homogeneous populations (i.e. the complete graph)
- The following simple 3-state population protocol was introduced in [Angluin et al., *Distributed Computing*, 2008]
  - initially nodes have 2 possible states: **r** and **g**
  - during the execution, a node can have 3 possible states: **r**, **g**, and **b**
  - interactions are dictated by the probabilistic scheduler
  - the 3×3 transition table can be summarized as follows:
    - node u of state **r** "hits" node v of state **g** $\Rightarrow$ v comes to state **b**
    - node u of state **g** "hits" node v of state **r** $\Rightarrow$ v comes to state **b**
    - node u of state **r** / **g** "hits" node v of state **b** $\Rightarrow$ v comes to state **r** / **g**

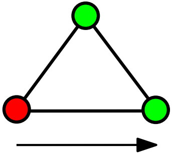
Example:



Population protocols for computing the majority

- Computing the majority in distributed settings has been mainly studied in homogeneous populations (i.e. the complete graph)
- The following simple 3-state population protocol was introduced in [Angluin et al., *Distributed Computing*, 2008]
  - initially nodes have 2 possible states: **r** and **g**
  - during the execution, a node can have 3 possible states: **r**, **g**, and **b**
  - interactions are dictated by the probabilistic scheduler
  - the 3×3 transition table can be summarized as follows:
    - node u of state **r** "hits" node v of state **g** $\Rightarrow$ v comes to state **b**
    - node u of state **g** "hits" node v of state **r** $\Rightarrow$ v comes to state **b**
    - node u of state **r** / **g** "hits" node v of state **b** $\Rightarrow$ v comes to state **r** / **g**

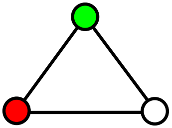
Example:



Population protocols for computing the majority

- Computing the majority in distributed settings has been mainly studied in homogeneous populations (i.e. the complete graph)
- The following simple 3-state population protocol was introduced in [Angluin et al., *Distributed Computing*, 2008]
  - initially nodes have 2 possible states: **r** and **g**
  - during the execution, a node can have 3 possible states: **r**, **g**, and **b**
  - interactions are dictated by the probabilistic scheduler
  - the 3×3 transition table can be summarized as follows:
    - node u of state **r** "hits" node v of state **g** $\Rightarrow$ v comes to state **b**
    - node u of state **g** "hits" node v of state **r** $\Rightarrow$ v comes to state **b**
    - node u of state **r** / **g** "hits" node v of state **b** $\Rightarrow$ v comes to state **r** / **g**

Example:



Population protocols for computing the majority

- In the protocol of [Angluin et al., *Distributed Computing*, 2008]:
  - if the underlying interaction graph is complete (with n vertices)
  - and the initial difference between majority and minority is $\Omega(\sqrt{n}\log n)$
  - then it converges to the initial majority in $O(n \log n)$ time w.h.p.

- A similar protocol for the complete graph has been studied in [Perron et al., *INFOCOM*, 2009]

Population protocols for computing the majority

- In the protocol of [Angluin et al., *Distributed Computing*, 2008]:
  - if the underlying interaction graph is complete (with n vertices)
  - and the initial difference between majority and minority is $\Omega(\sqrt{n}\log n)$
  - then it converges to the initial majority in $O(n \log n)$ time w.h.p.
- A similar protocol for the complete graph has been studied in [Perron et al., *INFOCOM*, 2009]

In the case of arbitrary interaction graphs:

- how fast can such protocols terminate?
- do they compute the correct initial majority with high probability?
- is it possible to compute majority with probability 1?
- how many states (per node) do we need to compute majority?
- how large should be the difference between initial majority / minority?

Our results

First result: the ambassador protocol

Theorem

- *There exists a 4-state protocol, the ambassador protocol, which stably computes the initial majority value:*
    - *for any interaction graph G,*
    - *for any initial difference between majority / minority,*
    - *with probability 1.*
- *There does not exist any 3-state protocol with these properties*

Our results

First result: the ambassador protocol

Theorem

- *There exists a 4-state protocol, the ambassador protocol, which stably computes the initial majority value:*
  - *for any interaction graph G,*
  - *for any initial difference between majority / minority,*
  - *with probability 1.*
- *There does not exist any 3-state protocol with these properties*

Theorem

Under the probabilistic scheduler:

- *The 4-state ambassador protocol runs in expected polynomial time.*
- *If the interaction graph G is complete and the initial difference is $\Theta(n)$, then the protocol terminates in expected time $O(n \log n)$.*

Our results

Second result: a detailed analysis of the protocol of *Angluin et al.* on an arbitrary interaction graph G (under the probabilistic scheduler)

Theorem

If the types $\mathbf{r}$ *and* $\mathbf{g}$ *are distributed uniformly at random on the vertices of* G*, the protocol converges to the initial majority with probability* $\geq \frac{1}{2}$.

Our results

Second result: a detailed analysis of the protocol of *Angluin et al.* on an arbitrary interaction graph G (under the probabilistic scheduler)

Theorem

If the types $\mathbf{r}$ *and* $\mathbf{g}$ *are distributed uniformly at random on the vertices of* G*, the protocol converges to the initial majority with probability* $\geq \frac{1}{2}$.

Theorem

There exists an infinite family $\{G_n\}_{n \in \mathbb{N}}$ *of interaction graphs where the protocol fails with high probability, even when the initial difference between majority / minority is* $n - \Theta(\log n)$.

Our results

Second result: a detailed analysis of the protocol of *Angluin et al.* on an arbitrary interaction graph G (under the probabilistic scheduler)

Theorem

If the types **r** *and* **g** *are distributed uniformly at random on the vertices of* G*, the protocol converges to the initial majority with probability* $\geq \frac{1}{2}$.

Theorem

There exists an infinite family $\{G_n\}_{n \in \mathbb{N}}$ *of interaction graphs where the protocol fails with high probability, even when the initial difference between majority / minority is* $n - \Theta(logn)$.

Theorem

There exists an infinite family $\{G'_n\}_{n \in \mathbb{N}}$ *of interaction graphs where the protocol terminates in exponential expected time.*

The 4-state ambassador protocol

The symmetric 4-state ambassador protocol:

- every node always has a color (**r** or **g**)
- every node may (or may not) have an extra token (called ambassador)

$\Rightarrow$ every node has 4 possible states: (**r**,0), (**r**,1), (**g**,0), (**g**,1)

- having an ambassador, a node can promote its color to a neighbor
- initially every node is at state (**r**,1) or (**g**,1), i.e. with an ambassador

The 4-state ambassador protocol

The symmetric 4-state ambassador protocol:

- every node always has a color ($\mathbf{r}$ or $\mathbf{g}$)
- every node may (or may not) have an extra token (called ambassador)

$\Rightarrow$ every node has 4 possible states: ($\mathbf{r}$,0), ($\mathbf{r}$,1), ($\mathbf{g}$,0), ($\mathbf{g}$,1)

- having an ambassador, a node can promote its color to a neighbor
- initially every node is at state ($\mathbf{r}$,1) or ($\mathbf{g}$,1), i.e. with an ambassador

When two nodes u and v interact, then:

- if both u and v have an ambassador:
  - if u and v have the same color, nothing happens
  - if u and v have different color, they both lose their ambassadors

The 4-state ambassador protocol

The symmetric 4-state ambassador protocol:

- every node always has a color ($\mathbf{r}$ or $\mathbf{g}$)
- every node may (or may not) have an extra token (called ambassador)

$\Rightarrow$ every node has 4 possible states: ($\mathbf{r}$,0), ($\mathbf{r}$,1), ($\mathbf{g}$,0), ($\mathbf{g}$,1)

- having an ambassador, a node can promote its color to a neighbor
- initially every node is at state ($\mathbf{r}$,1) or ($\mathbf{g}$,1), i.e. with an ambassador

When two nodes u and v interact, then:

- if both u and v have an ambassador:
  - if u and v have the same color, nothing happens
  - if u and v have different color, they both lose their ambassadors
- if u has an ambassador and v does not:
  - the ambassador of u moves to v
  - v takes the color of u

The 4-state ambassador protocol

The symmetric 4-state ambassador protocol:

- every node always has a color (**r** or **g**)
- every node may (or may not) have an extra token (called ambassador)

$\Rightarrow$ every node has 4 possible states: (**r**,0), (**r**,1), (**g**,0), (**g**,1)

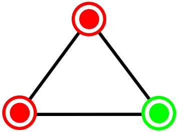
- having an ambassador, a node can promote its color to a neighbor
- initially every node is at state (**r**,1) or (**g**,1), i.e. with an ambassador

When two nodes u and v interact, then:

- if both u and v have an ambassador:
  - if u and v have the same color, nothing happens
  - if u and v have different color, they both lose their ambassadors
- if u has an ambassador and v does not:
  - the ambassador of u moves to v
  - v takes the color of u
- if neither u nor v have an ambassador:
  - nothing happens

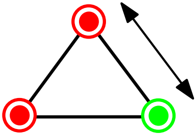
The 4-state ambassador protocol

Example:



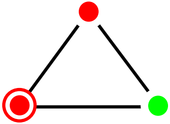
The 4-state ambassador protocol

Example:



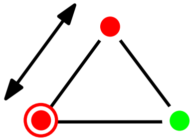
The 4-state ambassador protocol

Example:



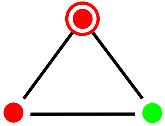
The 4-state ambassador protocol

Example:



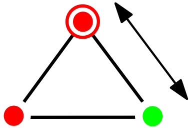
The 4-state ambassador protocol

Example:



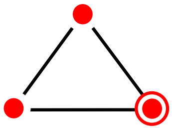
The 4-state ambassador protocol

Example:



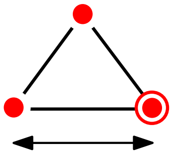
The 4-state ambassador protocol

Example:



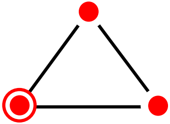
The 4-state ambassador protocol

Example:



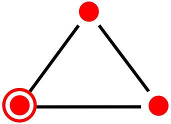
The 4-state ambassador protocol

Example:



The 4-state ambassador protocol

Example:

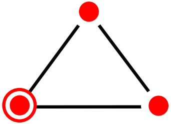


For any fair scheduler:

- the ambassadors of the minority will eventually all die out
- the remaining ambassadors will eventually color all the graph

The 4-state ambassador protocol

Example:



For any fair scheduler:

- the ambassadors of the minority will eventually all die out
- the remaining ambassadors will eventually color all the graph

Theorem (correctness)

- *The 4-state ambassador protocol stably computes the initial majority:*
  - *for any interaction graph G,*
  - *for any initial difference between majority / minority,*
  - *with probability 1.*

Lower bound on the number of states

Theorem

Let P be a population protocol that stably computes the majority function in an arbitrary 2-type population and for an arbitrary interaction graph. Then P needs at least 4 states.

Lower bound on the number of states

Theorem

Let P be a population protocol that stably computes the majority function in an arbitrary 2-type population and for an arbitrary interaction graph. Then P needs at least 4 states.

Proof (sketch, by contradiction).

- Assume P has 3 states **r**, **g**, **b**
- For at least one of the two input colors (say **r**):
  - starting with a majority of **r**,
  - eventually all nodes have the same state $q \in \{\mathbf{r},\mathbf{g},\mathbf{b}\}$
- We construct two instances C_1, C_2 on the same population such that:
  - C_1 and C_2 have different initial majorities
  - there exists a fair scheduler that brings both C_1 and C_2 to the same intermediate configuration
  - contradiction

□

The 4-state ambassador protocol

For the probabilistic scheduler:

Theorem

If $\Delta > 0$ is the initial difference between majority / minority, the 4-state ambassador protocol converges in expected:

- $O(n^6)$ *time for an arbitrary connected graph* G
- $O\left(\frac{\ln n}{\Delta} n^2\right)$ *time for the complete graph* K_n.

Proof based on:

- random walks on graphs and coupon collector arguments

Therefore:

- in the complete graph K_n, when $\Delta = \omega(\sqrt{n} \log n)$, the ambassador protocol converges in expected $O(n\sqrt{n})$ time
- a bit slower than $O(n \log n)$ of the 3-state protocol of [Angluin et al., *Distributed Computing*, 2008]
- but always correct

The protocol of Angluin et al. in arbitrary graphs

Assuming the probabilistic scheduler:

- What can we achieve with a 3-state protocol?
  - it cannot stably compute majority on arbitrary graphs
  - but it might compute majority with large enough probability.

The 3-state protocol of Angluin et al.:

- Converges quickly to the correct initial majority whp in the clique (for sufficiently large majority).
- What about arbitrary graphs?

The protocol of Angluin et al. in arbitrary graphs

Assuming the probabilistic scheduler:

- What can we achieve with a 3-state protocol?
  - it cannot stably compute majority on arbitrary graphs
  - but it might compute majority with large enough probability.

The 3-state protocol of Angluin et al.:

- Converges quickly to the correct initial majority whp in the clique (for sufficiently large majority).
- What about arbitrary graphs?

Theorem

If the types $\mathbf{r}$ *and* $\mathbf{g}$ *are distributed uniformly at random on the vertices of* G*, the protocol converges to the initial majority with probability* $\geq \frac{1}{2}$.

- Proof based on Hall's Marriage Theorem.

The protocol of Angluin et al. in arbitrary graphs

- The model of Angluin et al. can be abstracted by a Markov chain $\mathcal{M}$:
  - $\mathcal{M}$ has states (R_t, G_t), where R_t (resp. G_t) is the set of nodes of type **r** (resp. **g**) at time t
  - symmetries of the interaction graph can reduce the size of the state space; e.g. in the clique K_n, the set of states is just $(|R_t|, |G_t|)$.
  - The analysis of $\mathcal{M}$ on arbitrary graphs is complicated; for the clique exact formulae can be found [Perron et al., *INFOCOM*, 2009].

The protocol of Angluin et al. in arbitrary graphs

- The model of Angluin et al. can be abstracted by a Markov chain $\mathcal{M}$:
  - $\mathcal{M}$ has states (R_t, G_t), where R_t (resp. G_t) is the set of nodes of type **r** (resp. **g**) at time t
  - symmetries of the interaction graph can reduce the size of the state space; e.g. in the clique K_n, the set of states is just $(|R_t|, |G_t|)$.
  - The analysis of $\mathcal{M}$ on arbitrary graphs is complicated; for the clique exact formulae can be found [Perron et al., *INFOCOM*, 2009].

- We define 2 stochastic processes that filter the information from $\mathcal{M}$:

Definition (The Blank Process $\mathcal{W}$)

$$\mathcal{W}(t) \stackrel{def}{=} \langle \# \text{ nodes of type } \mathbf{b} \text{ at time } t \rangle$$

The protocol of Angluin et al. in arbitrary graphs

Definition (The Contest Process $\mathcal{C}$)

- We recursively pair the state changing transitions in $\mathcal{M}$ as follows:
  - each transition that increases the **blanks** ($\mathbf{g} \to \mathbf{r}$ or $\mathbf{r} \to \mathbf{g}$)
  - with the earliest subsequent transition that decreases the **blanks** ($\mathbf{g} \to \mathbf{b}$ or $\mathbf{r} \to \mathbf{b}$) and is not paired yet.
- define $\tau(t) \stackrel{def}{=} \langle \#\text{ pairs until time } t \rangle$
- $\mathcal{C}$ is defined over time scale τ
- Initially set $\mathcal{C}(0) = |R_0|$, and recursively:

$$\mathcal{C}(\tau) = \begin{cases} \mathcal{C}(\tau - 1) + 1, & \text{if } \tau\text{-th pair is } (\mathbf{r} \to \mathbf{g}, \mathbf{r} \to \mathbf{b}) \\ \mathcal{C}(\tau - 1) - 1, & \text{if } \tau\text{-th pair is } (\mathbf{g} \to \mathbf{r}, \mathbf{g} \to \mathbf{b}) \text{ and} \\ \mathcal{C}(\tau - 1), & \text{otherwise.} \end{cases}$$

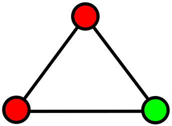
The protocol of Angluin et al. in arbitrary graphs

- The Contest Process keeps track of the battle between **g** and **r**
- $\mathcal{C}(\tau)$ counts the number of:
  - nodes of type **r** and
  - nodes of type **b** that were previously of type **r**

The protocol of Angluin et al. in arbitrary graphs

- The Contest Process keeps track of the battle between **g** and **r**
- $\mathcal{C}(\tau)$ counts the number of:
  - nodes of type **r** and
  - nodes of type **b** that were previously of type **r**

Example:

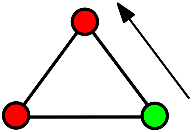


t	$\tau = \tau(t)$	$\mathcal{W}(t)$	$\mathcal{C}(\tau)$	transitions
0	0	0	2	-

The protocol of Angluin et al. in arbitrary graphs

- The Contest Process keeps track of the battle between **g** and **r**
- $\mathcal{C}(\tau)$ counts the number of:
  - nodes of type **r** and
  - nodes of type **b** that were previously of type **r**

Example:

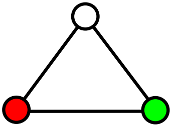


t	$\tau = \tau(t)$	$\mathcal{W}(t)$	$\mathcal{C}(\tau)$	transitions
0	0	0	2	-
1	0	1	2	**g** $\rightarrow$ **r**

The protocol of Angluin et al. in arbitrary graphs

- The Contest Process keeps track of the battle between **g** and **r**
- $\mathcal{C}(\tau)$ counts the number of:
  - nodes of type **r** and
  - nodes of type **b** that were previously of type **r**

Example:

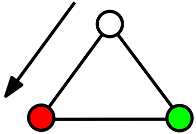


t	$\tau = \tau(t)$	$\mathcal{W}(t)$	$\mathcal{C}(\tau)$	transitions
0	0	0	2	-
1	0	1	2	**g** $\rightarrow$ **r**

The protocol of Angluin et al. in arbitrary graphs

- The Contest Process keeps track of the battle between **g** and **r**
- $\mathcal{C}(\tau)$ counts the number of:
  - nodes of type **r** and
  - nodes of type **b** that were previously of type **r**

Example:

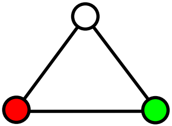


t	$\tau = \tau(t)$	$\mathcal{W}(t)$	$\mathcal{C}(\tau)$	transitions
0	0	0	2	-
1	0	1	2	**g** $\rightarrow$ **r**
2	0	1	2	**b** $\rightarrow$ **r**

The protocol of Angluin et al. in arbitrary graphs

- The Contest Process keeps track of the battle between **g** and **r**
- $\mathcal{C}(\tau)$ counts the number of:
  - nodes of type **r** and
  - nodes of type **b** that were previously of type **r**

Example:

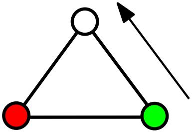


t	$\tau = \tau(t)$	$\mathcal{W}(t)$	$\mathcal{C}(\tau)$	transitions
0	0	0	2	-
1	0	1	2	**g** $\rightarrow$ **r**
2	0	1	2	**b** $\rightarrow$ **r**

The protocol of Angluin et al. in arbitrary graphs

- The Contest Process keeps track of the battle between **g** and **r**
- $\mathcal{C}(\tau)$ counts the number of:
  - nodes of type **r** and
  - nodes of type **b** that were previously of type **r**

Example:

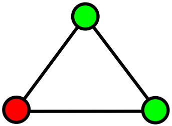


t	$\tau = \tau(t)$	$\mathcal{W}(t)$	$\mathcal{C}(\tau)$	transitions
0	0	0	2	-
1	0	1	2	**g** $\rightarrow$ **r**
2	0	1	2	**b** $\rightarrow$ **r**
3	1	0	1	**g** $\rightarrow$ **b**

The protocol of Angluin et al. in arbitrary graphs

- The Contest Process keeps track of the battle between **g** and **r**
- $\mathcal{C}(\tau)$ counts the number of:
  - nodes of type **r** and
  - nodes of type **b** that were previously of type **r**

Example:



t	$\tau = \tau(t)$	$\mathcal{W}(t)$	$\mathcal{C}(\tau)$	transitions
0	0	0	2	-
1	0	1	2	**g** $\rightarrow$ **r**
2	0	1	2	**b** $\rightarrow$ **r**
3	1	0	1	**g** $\rightarrow$ **b**

The protocol of Angluin et al. in arbitrary graphs

- $\mathcal{W}$ and $\mathcal{C}$ are dependent and not Markov chains
- $\mathcal{C}$ is defined on different time scale than $\mathcal{W}$ and $\mathcal{M}$
- $\mathcal{W}$ decreases $\Rightarrow$ pair of transitions in $\mathcal{M}$ $\Rightarrow$ transition step in $\mathcal{C}$
- Under assumptions on $|R_t|$ and $|G_t|$, we can dominate both $\mathcal{W}$ and $\mathcal{C}$ in the clique by appropriate birth-death processes
- Combining the above, we can prove that under the probabilistic scheduler the protocol of Angluin et al. in the clique is robust:

Theorem

For every constant $\epsilon < 1/7$ in the complete graph K_n:

- *if we initially have at most ϵn type* **r** *nodes*
- *then the probability that the minority* **r** *wins is exponentially small in n.*

The protocol of Angluin et al. in arbitrary graphs

Convergence to minority whp

Theorem

There exists an infinite family $\{G_n\}_{n\in\mathbb{N}}$ of interaction graphs where the protocol fails with high probability, even when the initial difference between majority / minority is $n - \Theta(log n)$.

The protocol of Angluin et al. in arbitrary graphs

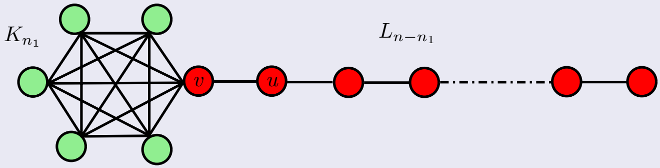
Convergence to minority whp

Theorem

There exists an infinite family $\{G_n\}_{n\in\mathbb{N}}$ of interaction graphs where the protocol fails with high probability, even when the initial difference between majority / minority is $n - \Theta(log n)$.

Proof (sketch).

- Let $n_1 \geq 100 \ln n$ and consider the lollipop graph:
  - line L_{n-n_1} with leftmost vertex u connected to vertex v of clique K_{n_1}
  - $L_{n-n_1} \cup \{v\}$ is of type **r** and $K_{n_1} \setminus \{v\}$ is of type **g**



Proof sketch. (cntd.)

- Define similarly Blank and Contest processes $\mathcal{W}'$ and $\mathcal{C}'$ on K_{n_1}
- These are slightly different than before, because of the edge $\{u, v\}$.
- Using $\mathcal{W}'$ and $\mathcal{C}'$ we first show that:

$$\Pr(\text{all } K_{n_1} \text{ becomes } \mathbf{r}) = e^{-\Omega(n_1)}$$

Proof sketch. (cntd.)

- Define similarly Blank and Contest processes $\mathcal{W}'$ and $\mathcal{C}'$ on K_{n_1}
- These are slightly different than before, because of the edge $\{u, v\}$.
- Using $\mathcal{W}'$ and $\mathcal{C}'$ we first show that:

$$\Pr(\text{all } K_{n_1} \text{ becomes } \mathbf{r}) = e^{-\Omega(n_1)}$$

- Second, we prove that in a line L_{n-n_1} with a single vertex of type $\mathbf{g}$ and the rest of type $\mathbf{r}$:

$$\Pr(\text{all } L_{n-n_1} \text{ becomes } \mathbf{g}) = \Omega\left(\frac{1}{n-n_1}\right)$$

Proof sketch. (cntd.)

- Define similarly Blank and Contest processes $\mathcal{W}'$ and $\mathcal{C}'$ on K_{n_1}
- These are slightly different than before, because of the edge $\{u, v\}$.
- Using $\mathcal{W}'$ and $\mathcal{C}'$ we first show that:

$$\Pr(\text{all } K_{n_1} \text{ becomes } \mathbf{r}) = e^{-\Omega(n_1)}$$

- Second, we prove that in a line L_{n-n_1} with a single vertex of type **g** and the rest of type **r**:

$$\Pr(\text{all } L_{n-n_1} \text{ becomes } \mathbf{g}) = \Omega\left(\frac{1}{n-n_1}\right)$$

- The above imply that, for $n_1 \geq 100 \ln n$, the minority **g** in the clique K_{n_1} has enough attempts to take over the whole graph.

$\square$

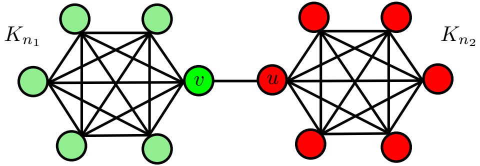
The protocol of Angluin et al. in arbitrary graphs

Exponential expected convergence time

Theorem

There exists an infinite family $\{G'_n\}_{n\in\mathbb{N}}$ of interaction graphs where the protocol terminates in exponential expected time.

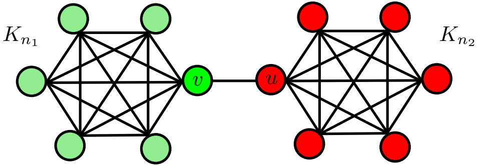
- We consider the family of graphs consisting of a clique K_{n_1} of type **g** and a clique K_{n_2} of type **r**, connected with an edge.



- The proof builds upon the proof ideas for the robustness of the protocol in the clique.

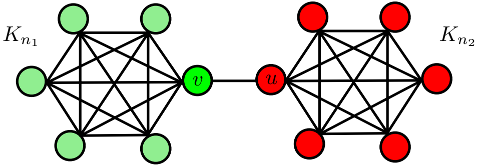
The protocol of Angluin et al. in arbitrary graphs

Exponential expected convergence time



Main idea:

- if vertex v becomes **r**:
  - K_{n_1} needs expected exponential time in n_1 to become of type **r**



Main idea:

- if vertex v becomes **r**:
  - K_{n_1} needs expected exponential time in n_1 to become of type **r**
- if vertex u becomes **g**:
  - K_{n_2} needs expected exponential time in n_2 to become of type **g**

Summary and Open Problems

- A 4-state symmetric (ambassador) protocol that always computes the majority
  - this is not possible with 3 states per node
- A detailed analysis of the majority protocol of *Angluin et al.* on arbitrary graphs
  - although it converges correctly and quickly whp in the clique,
  - this is not the case for arbitrary graphs

Summary and Open Problems

Open problems:

- A "good" 3-state protocol for majority on arbitrary graphs (under the probabilistic scheduler) ?
- Other computations than majority ?
  - average value, median, . . .
- What can we compute by allowing more powerful agents?
  - Non-deterministic interactions
  - More memory; what kind of functions can we (stably) compute with (say) 10, or $\log \log n$ states per vertex?
- What if every interaction involves more than 2 agents?

Thank you for your attention!