ADFOCS Lectures

- ☐ Asynchronous Crash-Prone Distributed Computing
- ☐ Locality in Distributed Network Computing
- ☐ Congestion-Prone Distributed Network Computing
- ☑ **Other Aspects of Distributed Computing**

Other Aspects of Distributed Computing

1. A Topological Perspective on Distributed Network Computing

2. Distributed Computing with Selfish Agents

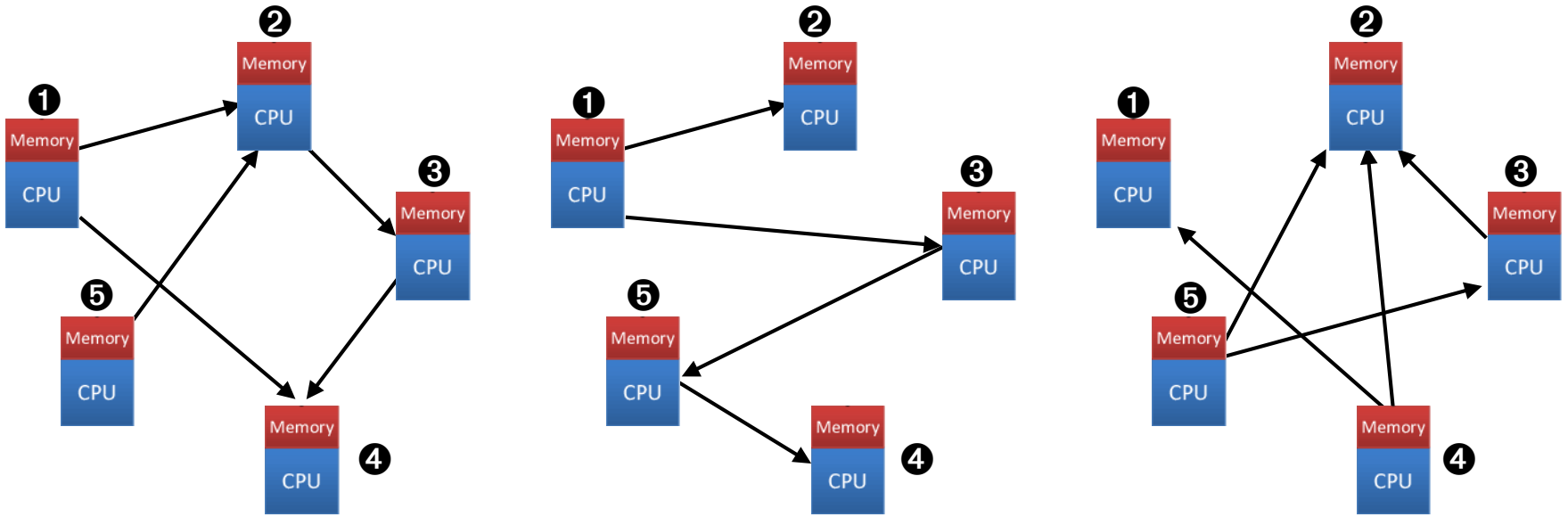
3. Preferential Attachment as a Unique Equilibrium

A Topological Perspective on Distributed Network Computing[1]

[1]Joint work with A. Castañeda, A. Paz, S. Rajsbaum, M. Roy, and C. Travers

KNOW-ALL **Model**

A sequence of labeled digraphs $\mathcal{H} = (H_t)_{1 \le t \le T}$



Round 1 H_1 | Round 2 H_2 | ... | Round T H_T

- Synchronous, failure-free
- Full information communication
- Every process knows its name ❶ and $\mathcal{H} = (H_t)_{1 \le t \le T}$

Input-Output Tasks

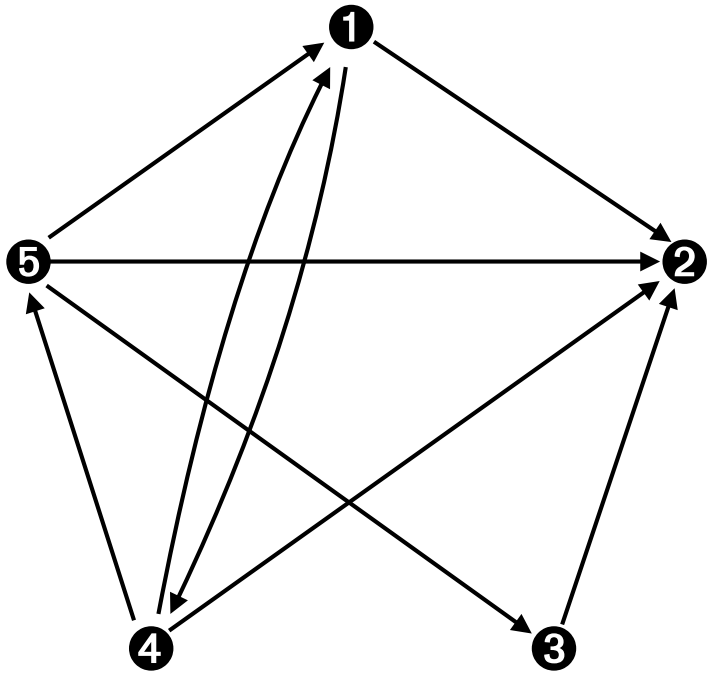
- I = set of input values
- O = set of input values

$$F : I^n \rightarrow 2^{O^n}$$

Example: Binary consensus: $I = O = \{0,1\}$

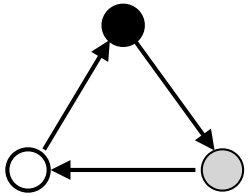
- $F(v_1,\ldots,v_n) = \{(0,\ldots,0),(1,\ldots,1)\}$ if there exists $v_i \neq v_j$
- $F(v,\ldots,v) = \{(v,\ldots,v)\}$ otherwise

Information Flow Graph

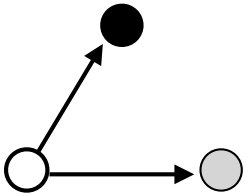


After T rounds:
Information Flow Graph G

Remark: If $\mathcal{H}_1$ and $\mathcal{H}_2$ yields the same information flow graph G, then $\mathcal{H}_1$ and $\mathcal{H}_2$ have the same computational power



C_3

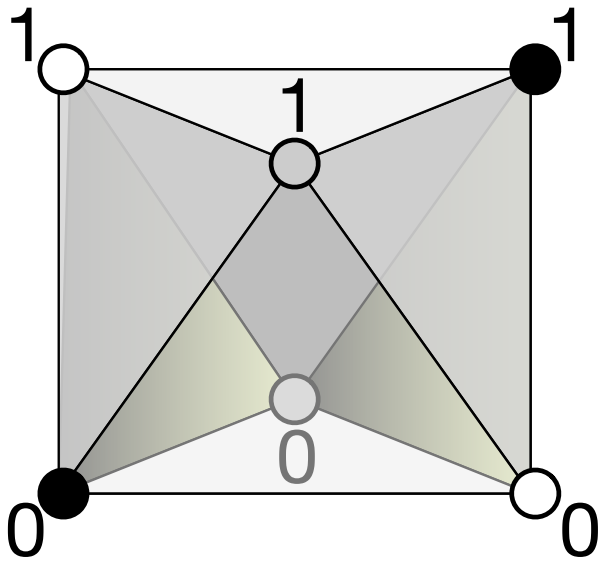


S_3

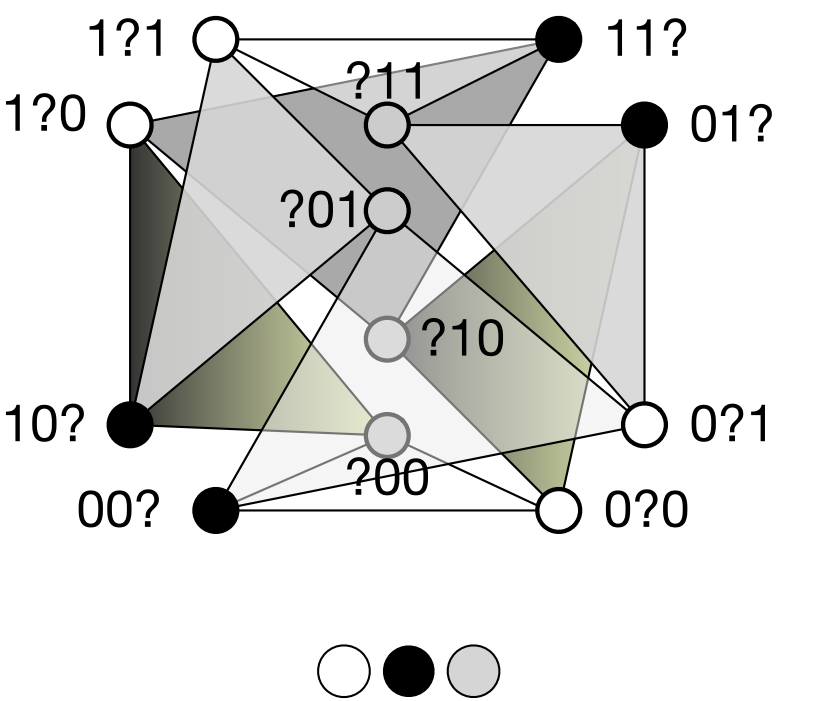
Protocol Complex

Example 1

Input complex



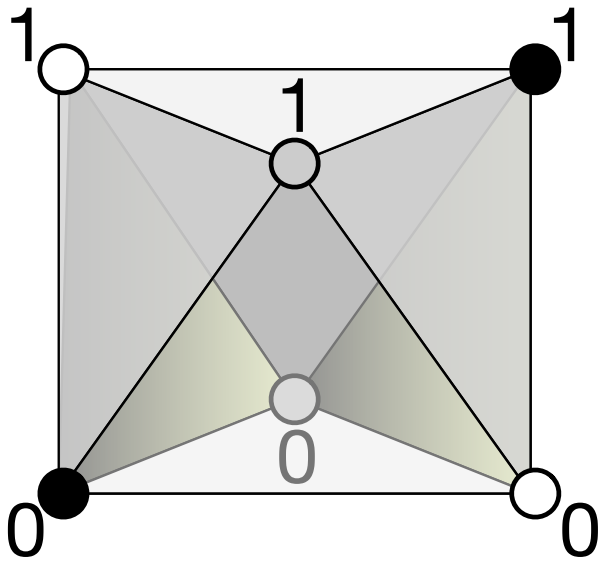
Protocol complex for C_3



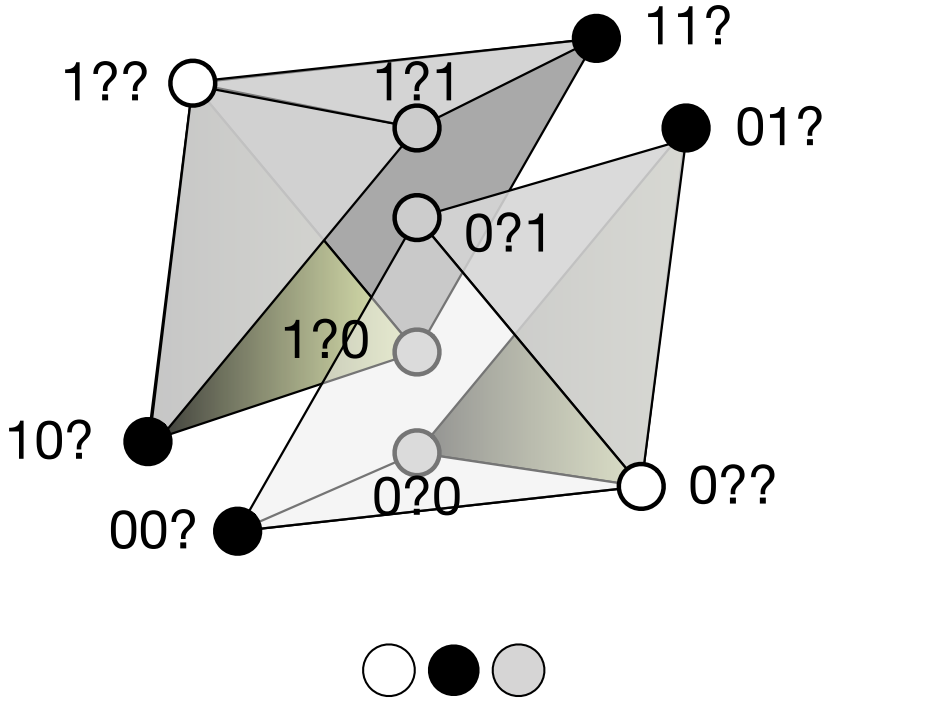
Protocol Complex

Example 2

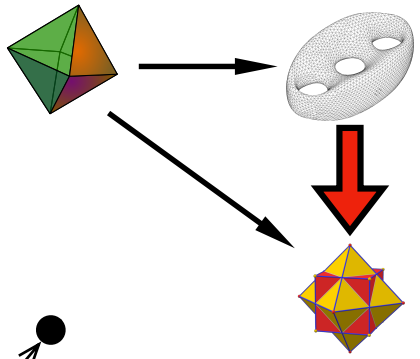
Input complex

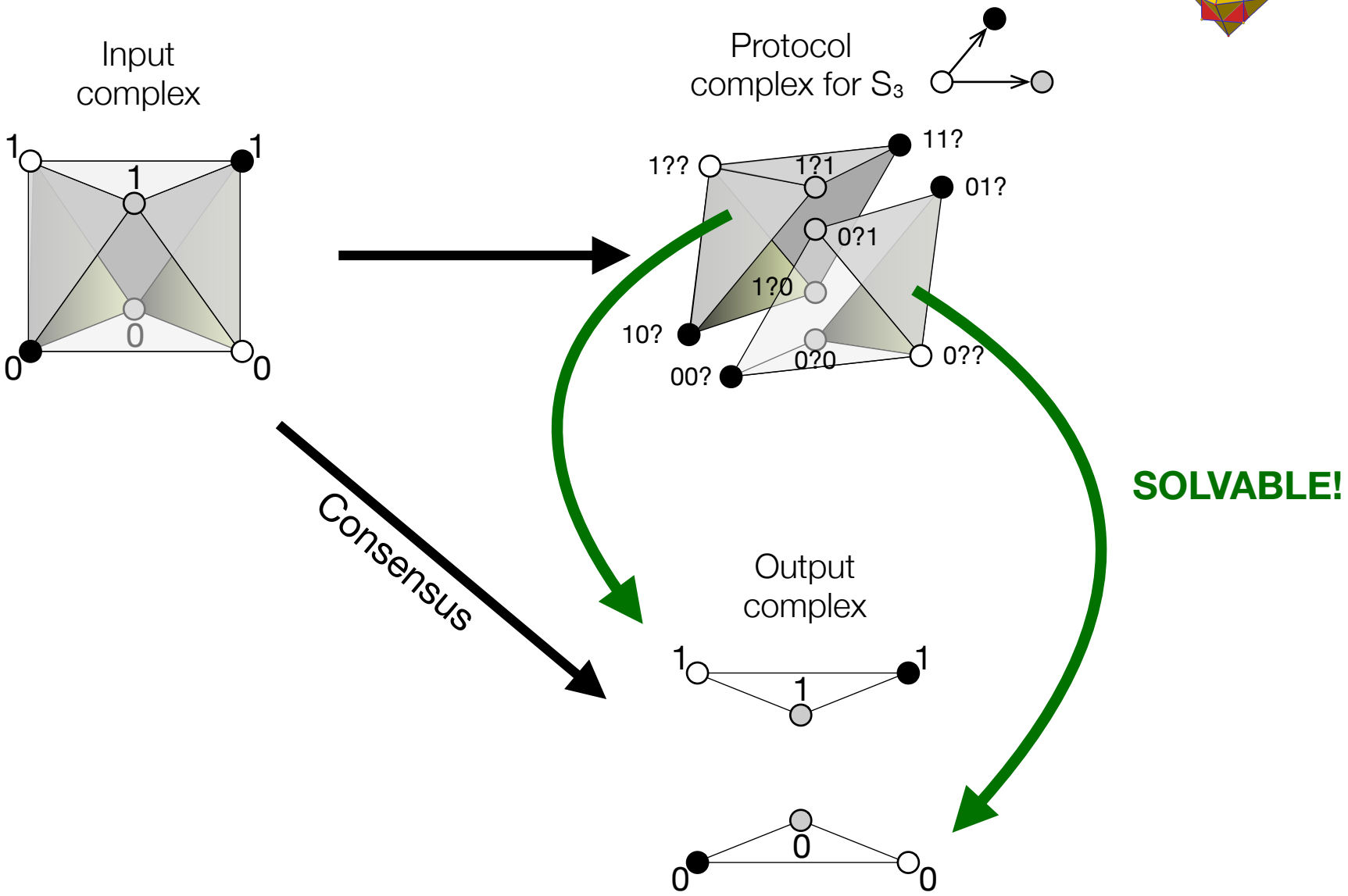


Protocol complex for S_3

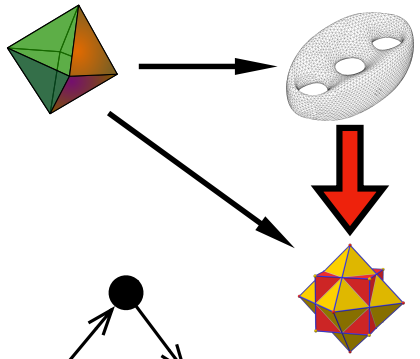


Solvability 1

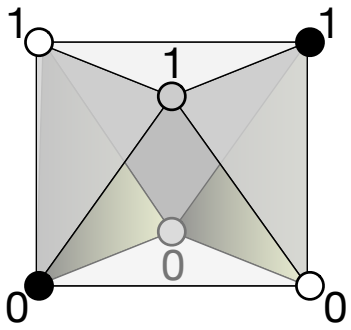




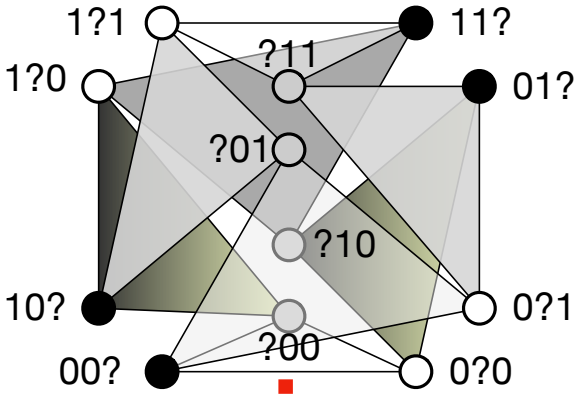
Solvability 2



Input complex



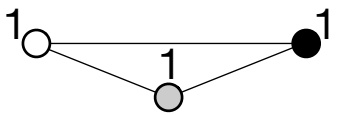
Protocol complex for C_3

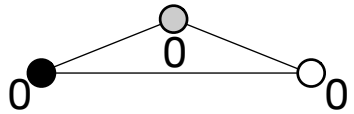


Consensus

?

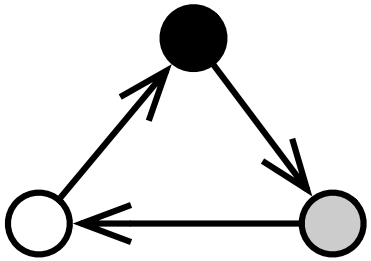
Output complex

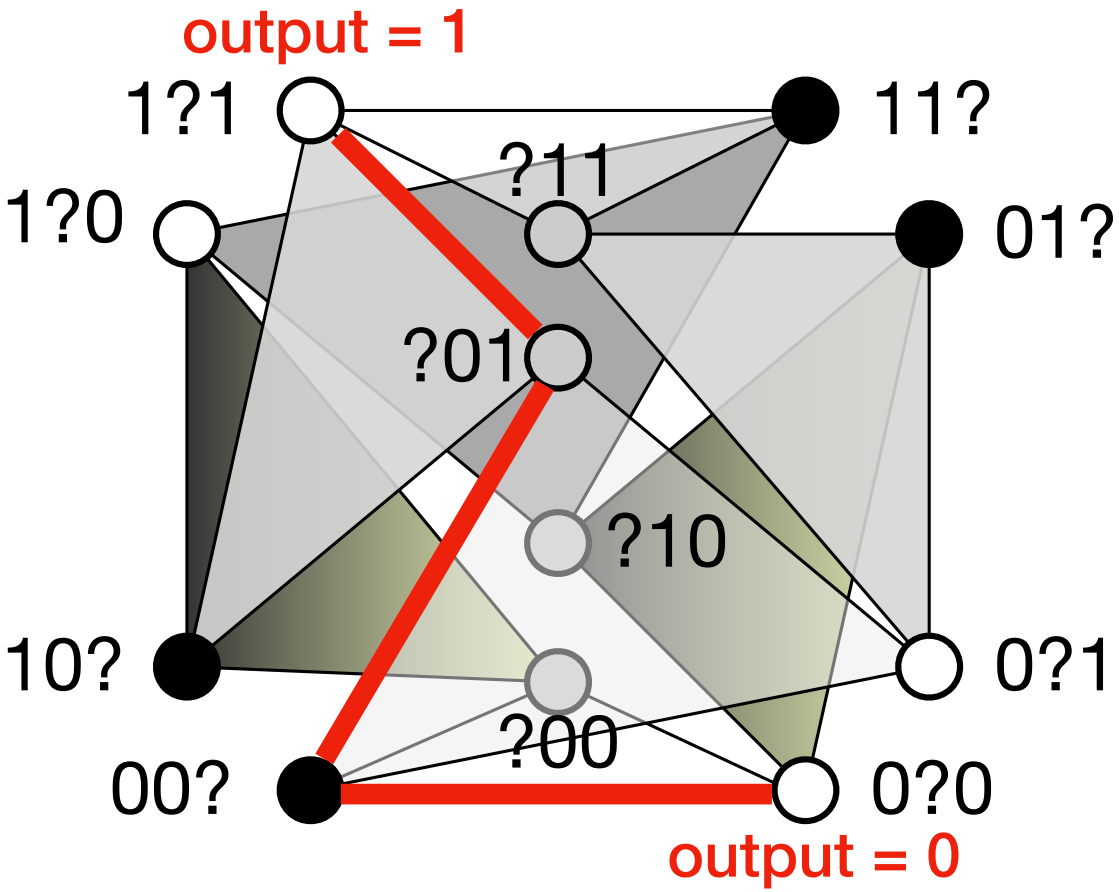




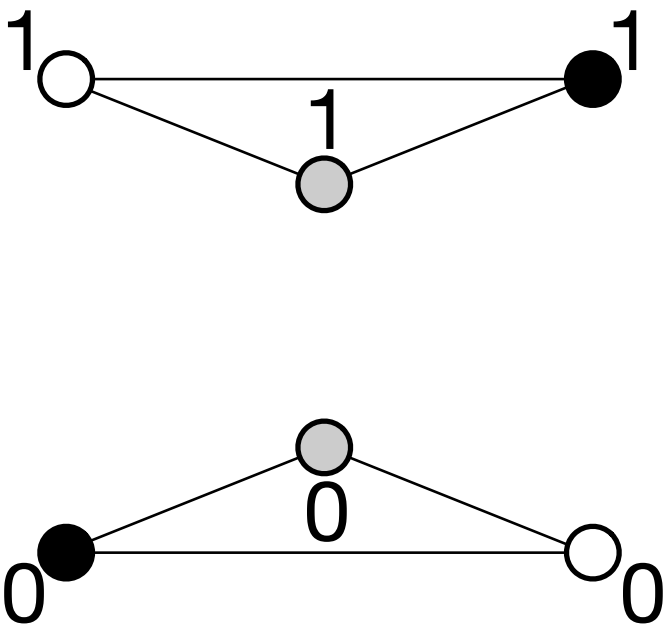
Path-Connectivity

Protocol complex for C_3





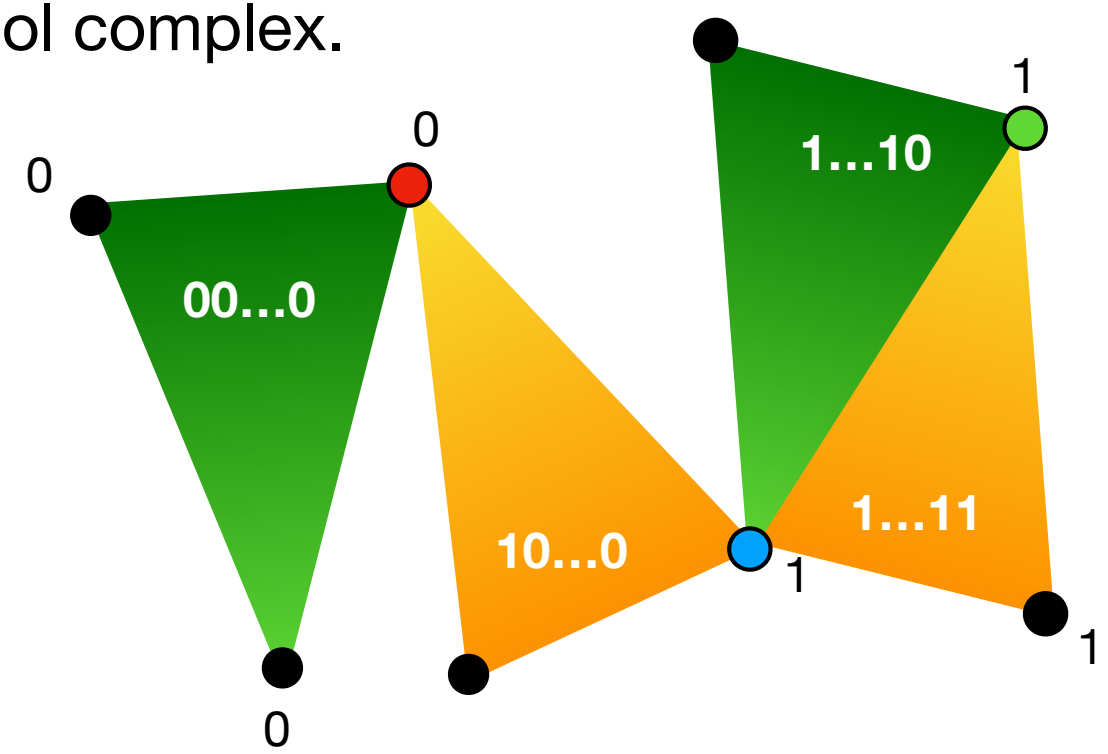
Output complex



Necessary condition for Consensus

Theorem 1. If the information flow graph G associated to $\mathcal{H}$ has no dominating vertex, then consensus is impossible under $\mathcal{H}$

Proof. We show that there is a path between $S_{0\ldots0}$ and $S_{1\ldots1}$ in the protocol complex.



□

NSC for k-Set-Agreement

$\gamma(G)$ = minimum size of a dominating set of G

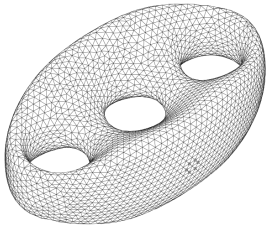
Property If the information flow graph G associated to $\mathcal{H}$ satisfies $\gamma(G) \leq k$, then k-set-agreement is possible under $\mathcal{H}$

Algorithm:

- $D = \{v_1,\ldots,v_k\}$ dominating set of G
- Adopt input value of any v_i

Theorem 2. If the information flow graph G associated to $\mathcal{H}$ satisfies $\gamma(G) > k$, then k-set-agreement is impossible under $\mathcal{H}$

Proof:



□

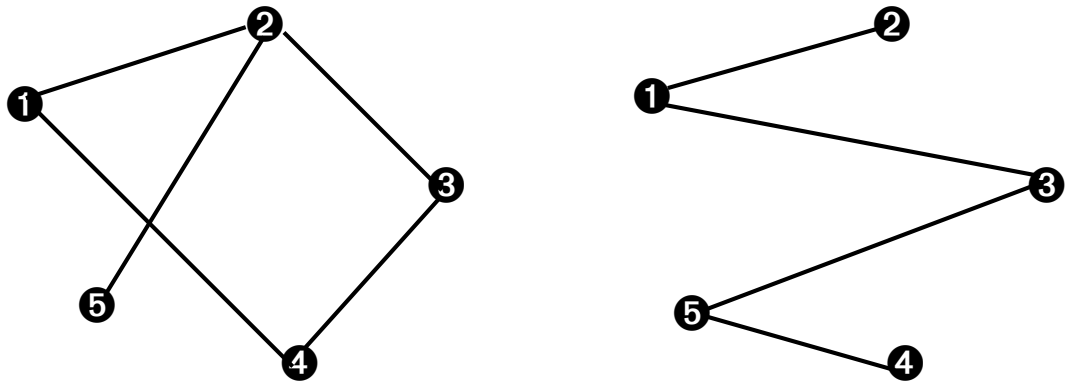
Application 1
LOCAL model

LOCAL model: synchronous rounds in a fixed graph H, no failures

Corollary 1 For any $k \geq 1$, k-set-agreement in network H requires at least r rounds, where r is the smallest integer such that $\gamma(H^r) \leq k$.

Application 2
Dynamic Networks

DYNAMIC networks: synchronous, no failure; A sequence of labeled digraphs $\mathcal{H} = (H_t)_{t\geq 1}$



Round 1 H_1 Round 2 H_2 ...

Corollary 2 For any $k\geq 1$, k-set-agreement in dynamic network $(H_t)_{t\geq 1}$ requires at least r rounds, where r is the smallest integer such that $(H_t)_{1\leq t\leq r}$ has *temporal* domination number $\leq k$

Conclusion

- Topology is in the *genes of distributed computing*.
- Usefully applied to *crash-prone asynchronous shared-memory computing*.
- Can also be usefully applied to *failure-free synchronous network computing*, as far as input-output tasks are concerned.

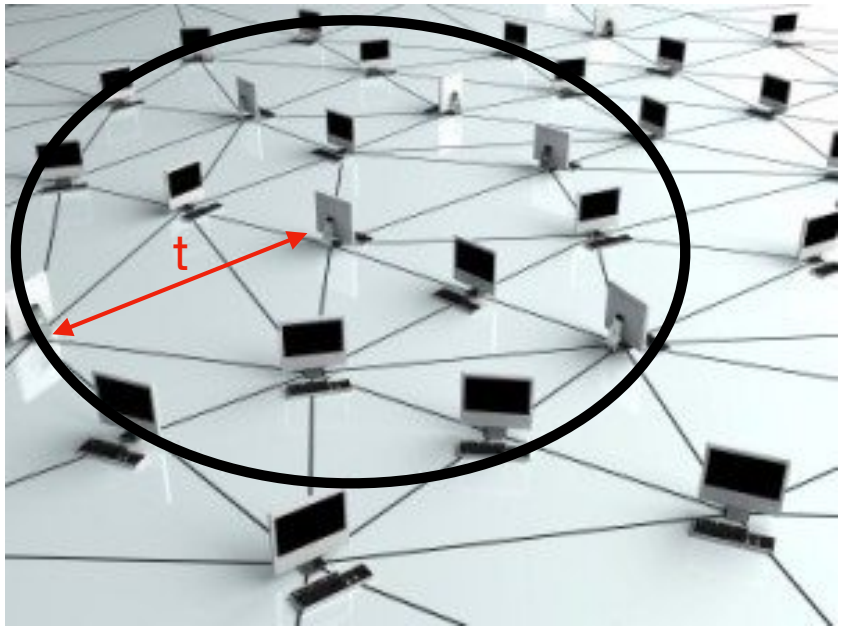
Open Problems

- **What about graph problems?** (coloring, max independent set, LCL tasks, etc.)
- **What about constrained models?** (e.g., CONGEST)

Distributed Computing with Selfish Agents[1]

[1] Joint work with Simon Collet and Paolo Penna

Local Tasks



Tasks that can be solved locally in networks.

Every node outputs after having consulted information stored at nodes in its vicinity.

Ideally radius $t = O(1)$ or $t = O(\log^{O(1)} n)$

t = #rounds performed by the algorithm

Examples:

- Vertex-coloring (e.g., for frequency assignment)
- Independent set (e.g., for scheduling)
- Dominating set (e.g., to serve as cluster head)
- etc.

LCL Tasks

- Locally checkable labelings (LCL) are tasks whose solution can be *checked* locally.
- Coloring, MIS, dominating set, etc., are LCL tasks.
- An LCL is characterized by
  - a set $\mathcal{L}$ of node labels
  - a set $\mathcal{B}$ of labeled balls with radius r
- **Task:** Every node of network G must compute a label in $\mathcal{L}$ such that, for every node v, the ball $B_G(v,r)$ is in $\mathcal{B}$.

A generic randomized algorithm for LCL tasks

Algorithm of node $u \in V(G)$

Repeat

observe $B_G(u,r)$

select a label $\ell(u) \in \mathcal{L}$ at random according to $D(u)$

observe $B_G(u,r)$

if $B_G(u,r) \in \mathcal{B}$ **then** commit with label $\ell(u)$ and stop

here we need LCL

The distribution $D(\cdot)$ can be uniform, but is often biased to increase the probability of constructing a good ball.

Also, $D(\cdot)$ can be different at different nodes, and may vary along with the execution of the algorithm.

Examples

1. Maximal Independent Set (MIS)

Luby's algorithm performs in $O(\log n)$ rounds w.h.p.

- Pr[u proposes itself to enter the MIS] ≈ $1/\deg(u)$

2. $(\Delta+1)$-coloring in max-degree-Δ networks

Barenboim & Elkin's algorithm performs in $O(\log n)$ rounds w.h.p.

- Pr[u participates in the phase] = $1/2$
- Pr[u proposes color c] ≈ uniform among available colors

Limits of the Generic Algorithm

- MIS or dominating sets can be used to construct a backbone in a radio network
  - being part of the backbone might be undesirable (e.g., because it causes high energy consumption)
- Vertex coloring can be used to assign radio frequencies to nodes in a radio network
  - some frequencies might be preferred (e.g., because they interfere with local transmitters)

➡ Some selfish nodes might be tempted to deviate from the algorithm, by not respecting the specification of the random distribution D governing the choice of the labels.

Framework

- Nodes *communicate honestly* their state, and *correctly transfer messages*, e.g., to avoid being caught.
- Selfish nodes may privately rationally cheat *about their choices* of the randomly selected labels.

Nodes want to solve the problem quickly because the solution provides some desirable service

Every node has preference for some of the solutions, and may wish to avoid undesirable solutions

The Game

- **Players:** the n nodes of a network $G=(V,E)$
- **Strategy** of node u: a distribution $D(u)$ over the good balls centered at u
- **Payoff** for node u:
  - $\text{pref}_u : \mathcal{B} \rightarrow [0,1]$
  - k = #rounds for the algorithm to terminate at u
  - Payoff:

$$\pi_u = \text{pref}_u(B) / 2^k$$

u aims at being the center of a ball that it prefers

u aims at terminating quickly

where $B = B_G(u,r)$ is the ball around u when the algorithm terminates at u

Question

What form of *equilibria* can be derived for LCL games?

Related Work (general)

	Strategic Games	**Extensive games with perfect information**	**Extensive games with imperfect information**
Finite games	[25] Nash equilibrium Mixed strategies	[28] Subgame-perfect equilibrium Pure strategies	[29] Trembling-hand perfect eq. Behavior strategies
Games with a finite action set		[12] Subgame-perfect equilibrium Pure strategies	[12] Sequential equilibrium Behavior strategies
Games with an infinite action set	[11] [14] Nash equilibrium Mixed strategies	[16] Subgame-perfect equilibrium Pure strategies	[10] Nash equilibrium Behavior strategies

Table 1 *A summary of results about the existence of equilibria*

Classical game theoretical results do not directly apply to LCL games because:

- the usual notion of imperfect information is *solely related to the fact that players play simultaneously*
- in LCL games, imperfect information also refers to the fact that each node is *not aware of the states of far away nodes* in the network.

Related Work (games in networks)

Distributed computing by rational agents:

- Abraham, Dolev, and Halpern [DISC 2013]
- Afek, Ginzberg, Feibish, and Sulamy [PODC 2014]
- Afek, Rafaeli and Sulamy [DISC 2018]

Framework:

- agents strategies define the algorithm itself, including which messages to send, which information to reveal
- the algorithms are “global” (they can take Ω(n) rounds)
- specific tasks are analyzed

Our result

A *trembling-hand perfect* equilibrium is a stronger form of Nash equilibrium.

- In Nash equilibria, players are assumed to play precisely as specified by the equilibrium.
- Trembling-hand perfect equilibria include the possibility of *off-the-equilibrium* play (players may, with small probabilities, choose unintended strategies).

Theorem
For any (greedily constructible) LCL task, the associated game has a symmetric trembling-hand perfect equilibrium.

Implications

For every LCL game, there is a distributed strategy from which the players have no incentive to deviate, in a robust sense (i.e., it supports small deviations).

➥ One can keep control of the system even in the presence of rational selfish players optimizing their own benefit.

Techniques

Lemma 1 Every infinite, continuous, measurable, well-rounded, extensive (symmetric) game with perfect recall and finite action set has a (symmetric) trembling-hand perfect equilibrium.

Lemma 2 LCL games are symmetric, infinite, continuous, measurable, well-rounded, extensive games with perfect recall and finite action set.

Conclusion and Open Problems

- We have proved that natural games occurring in the framework of local distributed network computing have trembling-hand perfect equilibria, a strong form of Nash equilibria.

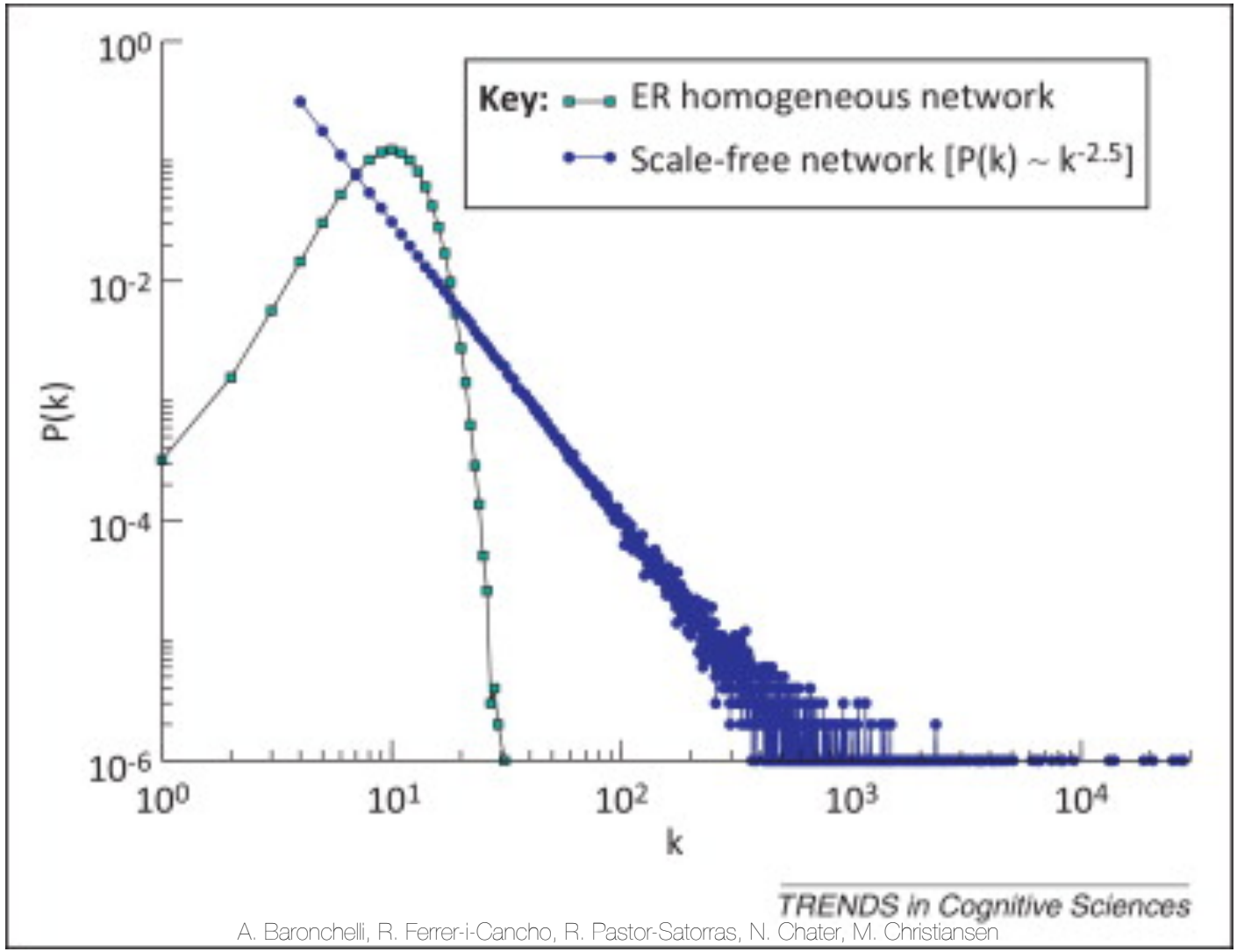
What are the performances of the robust algorithms resulting from these equilibria?

- Note that determining the performances of iterative distributed construction algorithms such as the generic algorithm is non trivial, even if nodes follow the prescribed actions imposed by the algorithm (e.g., Luby's algorithm).

Preferential Attachment as a Unique Equilibrium

[1]Joint work with Chen Avin, Avi Cohen, Zvi Lotker, and David Peleg

Common Knowledge

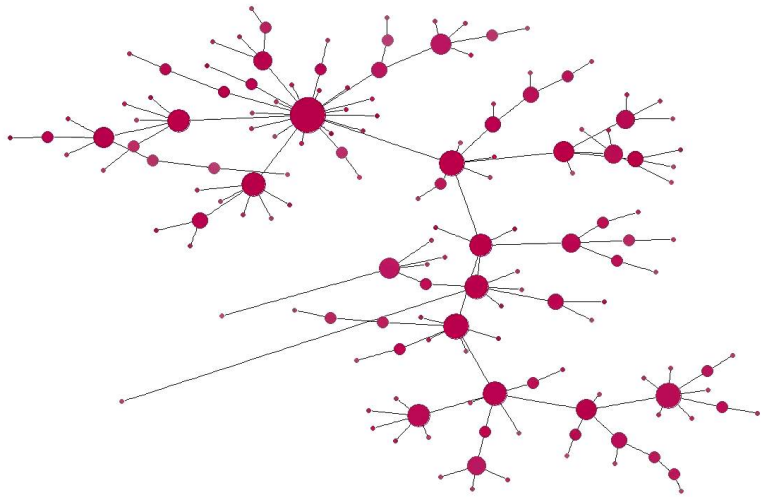


A. Baronchelli, R. Ferrer-i-Cancho, R. Pastor-Satorras, N. Chater, M. Christiansen

Social Network Model

Preferential Attachement (Barabási–Albert)

- Nodes arrive one after the other
- A new node u connects to $k \geq 1$ existing nodes
- $\Pr[u \to v] \approx \deg_G(v)$
- For $k=1$, PA yields a tree



Rationals for Preferential Attachement

Empirical

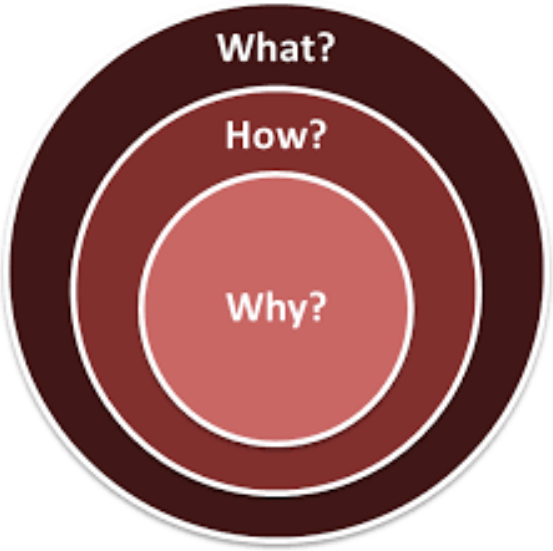
- Rich get richer aphorism (a.k.a. Matthew effect)
- Special case of Price's model

Analytical

- Generate graphs “similar to” real networks
- Has desirable properties (degree sequence, short paths, etc.)

The Golden Circle

- **The what:** Heavy tailed degree distribution
- **The how:** Random graph theory:
  - ➥ preferential attachement (PA)
- **The why:** Game theory



A Hint why Social Networks are PA Graphs

PA is the unique Nash equilibrium of
a natural network formation game

Network Formation Game: Framework

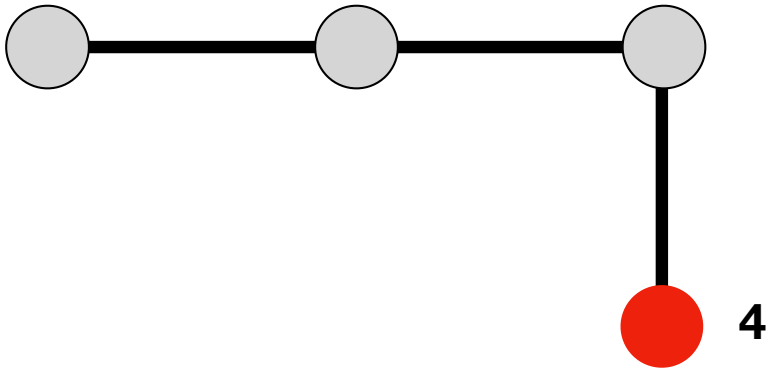
- Society = graph
- Social capital of a node = degree
- Wealth of society = $\alpha \in [0,1]$
- Formation process = new connections are:
  - accepted with prob α
  - rejected with prob $1-\alpha$, and pushed to a neighbor chosen u.a.r. (the latter must accept the connection)

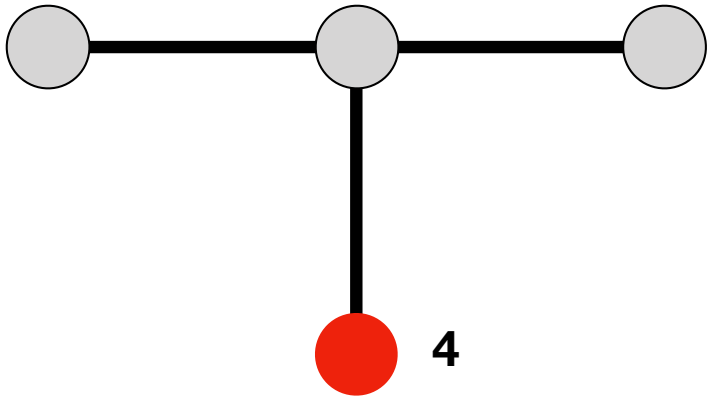
Network Formation Game: Strategy & Utility

- Nodes arrive one after the other
- A new node u arriving at time t connects to one of the existing nodes: it chooses a node v, with probability $\Pr[u \text{ chooses } v]$, and connects to v or one of its neighbors.
- $\Pr[u \text{ chooses } v] = \pi_u(v)$ where π_u is distributed over degree sequences — this is the **strategy** of node u.
- Connections accepted according to probabilities $(\boldsymbol{\alpha}_t)_{t \geq 1}$
- **Utility**(v) at time t = $\mathbb{E}[\deg(v) \text{ at time } t]$

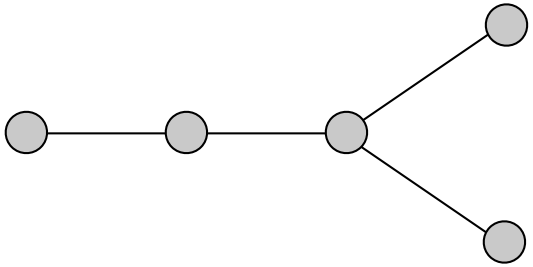
Example

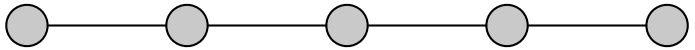
Step 4:

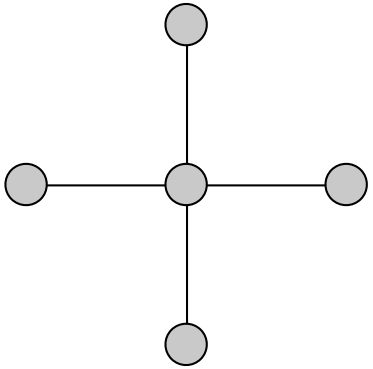




Step 5:







Universal Nash Equilibrium

Remark There is a game for

- every stopping times $\tau \geq 1$, and
- every wealth sequences $(\boldsymbol{\alpha}_t)_{t\geq 1}$.

Definition A strategy profile $(\pi_t)_{t\geq 1}$ is a **universal** NE if it is a NE for **all** stopping times $\tau \geq 1$, and **all** wealth sequences $(\boldsymbol{\alpha}_t)_{t\geq 1}$

Universal NE Exist

Definition Pr[u chooses v] = $\pi_{PA}(v) = \deg(v) / \sum_z \deg(z) = \deg(v) / 2m$

Theorem PA is a universal NE

Lemma Pr[u connects to v | T] = $\pi_{PA}(v)$

Proof: Pr[u connects to v | T] = $\alpha\, \pi_{PA}(v) + \sum_{w \in N(v)} \pi_{PA}(w)(1-\alpha)/\deg(w)$

$= \alpha \deg(v) / \sum_z \deg(z) + \sum_{w \in N(v)} \left((1-\alpha) / \sum_z \deg(z)\right)$

$= \deg(v) / \sum_z \deg(z) = \pi_{PA}(v)$ □

PA is a universal NE

Proof. Assume PA is used.

Assume that there exists a sequence $(\boldsymbol{\alpha}_t)_{t\geq 1}$ and some player v_t for $t\geq 4$ who could increase her utility by deviating from PA to $\pi'_t \neq PA$.

$deg_s(t)$ = degree of player v_t at time $s \geq t$.

$deg_t(t) = 1$, and, for $s>t$, by the lemma, independently from π'_t:

- $deg_s(t) = deg_{s-1}(t) + 1$ with probability $deg_{s-1}(t)/2(s-2)$
- $deg_s(t) = deg_{s-1}(t)$ with probability $1 - deg_{s-1}(t)/2(s-2)$ □

Main Result

Theorem PA is the unique universal NE

Lemma Let $\Pi=(\pi_t)_{t\geq 1}$ be a strategy profile that is not PA. There exists a wealth sequence $(\boldsymbol{\alpha}_t)_{t\geq 1}$ such that Π is not a NE for $(\boldsymbol{\alpha}_t)_{t\geq 1}$.

Remark The result holds for only two different values $\boldsymbol{\alpha}_t \neq \boldsymbol{\alpha}_{t'}$.

Time-Invariant Games

- The wealth remains constant over time
- **Definition** $\alpha_t = \alpha \in [0,1]$ for every $t \geq 1$.
- **Theorem** If a strategy profile $\Pi = (\pi_t)_{t \geq 1}$ is a universal Nash equilibrium for the time-invariant game, then
  - each player plays PA on every graph that is not a star, and
  - if player t plays PA on the star S_{t-1} then all subsequent players $t' > t$ play PA on all graphs.

Static Games

Definition Systematically connect to the host, i.e., $\alpha_t = 1$ for every $t \geq 1$.

- A strategy π_t is degree-k consistent if, for every degree-k node, the probability of selecting that node is independent of the degree sequence.
- A strategy π_t is degree consistent if it is degree-k consistent for every $k \geq 0$.
- A strategy profile $\Pi = (\pi_t)_{t \geq 1}$ is degree consistent if π_t is degree consistent for every $t \geq 1$.

Remark : PA is a degree consistent strategy.

Universal NE for Static Games

Theorem Let $\Pi=(\pi_t)_{t\geq 1}$ be a universal Nash equilibrium for the static game. If the strategy $\pi_{t'}$ is degree consistent for every $t'\in\{1,2,...,t-1\}$, and $\pi_{t'}(k)>0$ for every $k\in\{1,...,t-1\}$, then π_t is a degree consistent strategy.

In particular, if every player $t'\in\{1, 2, ..., t-1\}$ played PA, then π_t is a degree consistent strategy.

Conclusion

- What if the recommendation proceeds recursively? (By same arguments PA remains a universal Nash equilibrium in this case too).
- What if each new node connects to $m > 1$ existing nodes?
- In addition to node-events, considering edge-events
- What if the players have more knowledge about the actual structure than just its degree sequence?

General Conclusion

Distributed Computing is Ubiquitous

- **All scales:** multi-core, distributed data-bases, cloud computing, etc.
- **At the core of applications:** blockchain, peer-to-peer (P2P), Internet of things (IoT), etc.
- **Interest goes beyond human artefacts:** understanding biological phenomena (cells, insects), global social behavior, etc.

Coping with uncertainty!

- **Temporal uncertainty:** Asynchrony, failures, etc.
- **Spatial uncertainty:** Locality, congestion, etc.

Topics left for future ADFOCS

- **Mobile computing:** physical robots, software agents, etc.
- **Fault-tolerance:** very many aspects
- **Limited computational power:** finite automata, oblivious computing, energy issue, etc.
- **Anonymity**, **security**, **verification**, **certification**, …
- Etc.

Thank you!