A Decade of Stone Age Distributed Computing

Yuval Emek

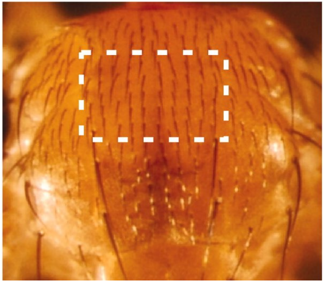
Technion — Israel Institute of Technology

SIROCCO 2023
Special Models of Computation Session
Alcala de Henares, Spain

Based on joint works with Yehuda Afek, Eyal Keren, Noa Kolikant, Oren Louidor, Jara Uitto, Roger Wattenhofer

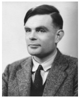
Distributed biological processes

- Social insects, vertebrate colonies, the brain, ...
- **Our focus:** biological processes in cellular networks
  - Abstracted as distributed graph algorithms
- [Afek, Alon, Barad, Hornstein, Barkai, Bar-Joseph 2011]:
  growth of sensory organs in drosophila's nervous system = MIS



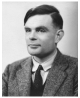
- Cell $\neq$ modern computer
- Aim for networks of "sub-silicon devices"
  - nano-networks, programmable matter, smart paint, smart dust

Sub-silicon devices — communication

- [AABHBBJ]: restricted form of local interactions
- The *beeping* communication scheme
  [Cornejo, Kuhn 2010], [Flury, Wattenhofer 2010]
  - In each (synchronous) round, node beeps or silent
  - Node's signal: 0 beeps or ≥ 1 beeps among neighbors
  - Resembles juxtacrine cell-cell communication
- Special case of *set-broadcast* communication scheme
  [Hella, Järvisalo, Kuusisto, Laurinharju, Lempiäinen, Luosto, Suomela, Virtema 2015]
  - In each (synchronous) round, node broadcasts message
  - Node's signal: set of message types broadcast by neighbors
    - For each message type, distinguish 0 copies from ≥ 1 copies
  - Juxtacrine communication with multiple ligand-receptor types
- What about the node's computational power?
  - [AABHBBJ]: 

Sub-silicon devices — computation

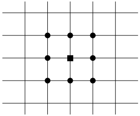
- Computational power of cell?

 $\ll$ 

- [Benenson, Paz-Elizur, Adar, Keinan, Livneh, Shapiro 2001]: cell enzymes “programmed” to implement finite automata
- **Model choice:** abstract  as *finite automaton*
- Distributed computing over finite automata
  - Population protocols
  - Amoebot
  - Cellular automata

Cellular automata

Infinite grid of finite automata

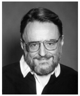


$$q_{t+1}(x, y) \longleftarrow q_t(x, y), \{q_t(x', y') \mid \|(x', y') - (x, y)\|_\infty \leq 1\}$$

Typical question: How initial (finite) configuration evolves?

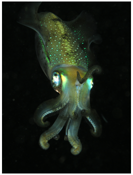
Invented by   (self-replicating systems, crystal growth)

Game of life



Digital physics

Biological processes



Melting pot of model features

- [Emek, Wattenhofer 2013]
  - cellular automata over arbitrary (finite) topologies
  - set-broadcast communication scheme [HJKLLLSV]
  - asynchronous executions
- **Nowadays:** simplified version

Formal definition

- Finite undirected communication graph $G = (V, E)$
- Each node runs *local automaton* over fixed state set Q
  - $q_t(v)$ = state of node $v \in V$ in *step* $t \in \mathbb{Z}_{\geq 0}$
- Signal of node v in step t: $\{q_t(u) \mid u \in N(v)\}$
  - set rather than multiset or vector
- Adversarial asynchronous *schedule* $A_0, A_1, \dots \subseteq V$
  - A_t = subset of nodes activated in step t
- *State transition function* $\delta : Q \times 2^Q \to Q$ (possibly randomized)

$$q_{t+1}(v) = \begin{cases} \delta\left(q_t(v), \{q_t(u) \mid u \in N(v)\}\right), & v \in A_t \\ q_t(v), & v \notin A_t \end{cases}$$

- Function $\omega : Q \to \mathcal{O} \cup \{\perp\}$ determines *output* associated with $q \in Q$
  - $\mathcal{O}$ = task dependent output value set
  - $\perp$ = no output (needed for termination detection)

Uniform algorithms

- Crux of SA model: $|Q|$ and $|\delta|$ are fixed
- Local automaton oblivious to global parameters
  - $n = |V|$
  - $\Delta = \max_{v \in V} \deg(v)$
  - $D = \max_{u,v \in V} \text{dist}(u, v)$
- Sometimes graphs of bounded Δ or D
  - n is always unbounded
- Local automaton of $v \in V$ oblivious to $\deg(v)$
- Truly uniform!

© JCorbettcartoon

easier to manufacture

Model variants

- Schedule is *synchronous* if $A_t = V$ for all $t \in \mathbb{Z}_{\geq 0}$
- *Self-stabilization*: adversarial initial configuration $\langle q_0(v) \rangle_{v \in V}$
  - Alternative: *graceful initialization*
    - Alg designer determines initial state $q_{\text{init}} \in Q$
    - $q_0(v) = q_{\text{init}}$ for all $v \in V$
  - **Convention:** graceful initialization unless self-stated stated
- *Termination detection*:
  if $\omega(q_t(v)) \neq \bot$, then $\omega(q_{t'}(v)) = \omega(q_t(v))$ for all $t' \geq t$
  - **Convention:** require termination detection iff graceful initialization
- *Sender collision detection*: G has no self-loops
  - Self-loop on v implies $v \in N(v)$, thus $q_t(v)$ included in v's signal
  - **Notation:**
    - $\text{SA}^{\circlearrowleft}$ = model with self-loops (w.l.o.g. on all nodes)
    - $\text{SA}^{\updownarrow}$ = model with sender collision detection (no self-loops)

Complexity measures

- Runtime a la [Dolev, Israeli, Moran 1997]
  - *Async round*:
    shortest step interval in which each node activated at least once
    - Inductively partition step axis $\mathbb{Z}_{\geq 0}$ into rounds
  - *Runtime of alg* = number of async rounds until alg terminates
  - Sync schedule: each step forms a round
- *State space* = $|Q|$

From synchronous to asynchronous algorithms

Theorem ([Emek, Wattenhofer 2013])

Suppose that a distributed task $\mathcal{T}$ admits a sync $\mathrm{SA}^{\circlearrowleft}$ *(resp.,* $\mathrm{SA}^{\updownarrow}$*) alg that solves $\mathcal{T}$ with state space S and runtime R w.h.p. Then, $\mathcal{T}$ admits an async* $\mathrm{SA}^{\circlearrowleft}$ *(resp.,* $\mathrm{SA}^{\updownarrow}$*) alg that solves $\mathcal{T}$ with state space $O(S^2)$ and runtime $O(R)$ w.h.p.*

Theorem ([Emek, Keren 2021])

Suppose that a distributed task $\mathcal{T}$ admits a sync self-stab $\mathrm{SA}^{\circlearrowleft}$ *(resp.,* $\mathrm{SA}^{\updownarrow}$*) alg that solves $\mathcal{T}$ with state space S and runtime R w.h.p. Then, $\mathcal{T}$ admits an async self-stab* $\mathrm{SA}^{\circlearrowleft}$ *(resp.,* $\mathrm{SA}^{\updownarrow}$*) alg with that solves $\mathcal{T}$ with state space $O(D \cdot S^2)$ and runtime $R + O(D^3)$ w.h.p.*

- Suffices to develop sync algs

Asynchronous unison

- Essence of self-stab synchronizer: *asynchronous unison (AU)* [Couvreur, Francez, Gouda 1992], [Awerbuch, Kutten, Mansour, Patt-Shamir, Varghese 1993]
- Additive cyclic group $\mathcal{K}$
  - $|\mathcal{K}| \geq \Omega(D)$
- Node $v \in V$ outputs *clock value* $\kappa_v \in \mathcal{K}$
- **Post-stabilization conditions:**
  - *Safety*: $u \in N(v) \implies \kappa_u \in \{\kappa_v - 1, \kappa_v, \kappa_v + 1\}$
  - *Liveness*: κ_v incremented $\geq i$ times during any span of $D + i$ rounds
    - increment $=$ $\mathcal{K}$'s $+1$ operation

Theorem

Consider a self-stab AU alg with state space S and runtime R. If task $\mathcal{T}$ admits a sync self-stab alg with state space $S_{\mathcal{T}}$ and runtime $R_{\mathcal{T}}$ w.h.p., then $\mathcal{T}$ admits an (async) self-stab alg with state space $O(S \cdot S_{\mathcal{T}}^2)$ and runtime $O(R + R_{\mathcal{T}} + D)$ w.h.p.

- Holds also for $SA^{\circlearrowleft}$ and $SA^{\updownarrow}$

AU literature

- Problem introduced in [CFG], [AKMPSV]
- Studied further in [Boulinier, Petit, Villain 2004], [Boulinier, Petit, Villain 2005], [Boulinier, Petit, Villain 2006], [Devismes, Petit 2012], [Devismes, Johnen 2019]
  - Various computational models and graph classes
- **General graphs:**

paper	state space	runtime	comments
[AKMPSV]	∞	$O(D)$	
	$\Theta(n)$	$O(D)$	unique IDs
[BPV]	$\Theta(C_G + C'_G)$	$O(C_G + C'_G)$	SA$^{\circlearrowleft}$

  - C_G = minimum longest cycle length among all cycle bases of G (or 1)
    C'_G = length of longest chordless cycle of G (or 1)
    - $C_G + C'_G$ is incomparable to D
- **Goal:** self-stab AU alg with state space and runtime that depend only on D (linear dependency (existentially) unavoidable)

SA to the rescue

Theorem ([Emek, Keren 2021])

There exists a (deterministic) self-stab $\text{SA}^{\circlearrowleft}$ *AU alg with state space* $O(D)$ *and runtime* $O(D^3)$.

Open question: self-stab AU alg with state space $O(D)$ and runtime $O(D)$?

- Relevant outside SA scope

Graceful initialization

Theorem ([Emek, Wattenhofer 2013])

There exists a sync $\mathsf{SA}^{\updownarrow}$ *alg that solves MIS with state space* $O(1)$ *and runtime* $O(\log^2 n)$ *w.h.p.*

Theorem ([Afek, Emek, Kolikant 2018])

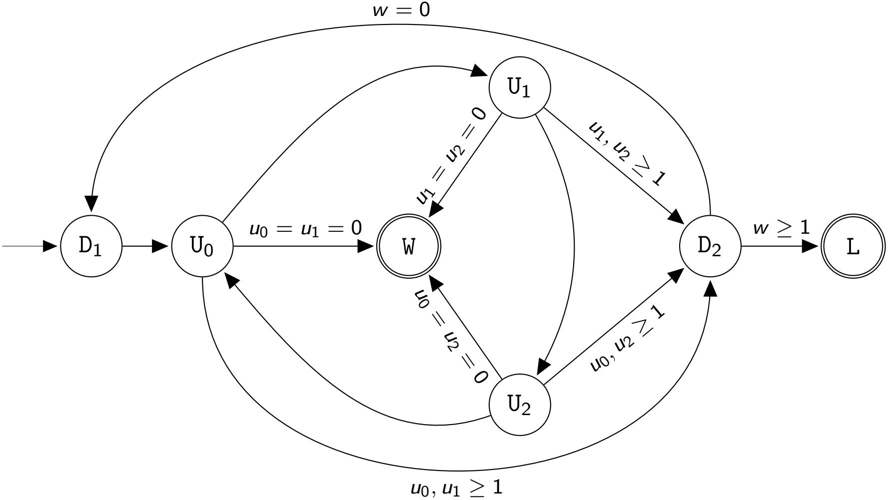
There does not exist a sync $\mathsf{SA}^{\circlearrowleft}$ *alg that solves MIS with state space* $O(1)$ *w.p.* $\Omega(1)$*.*

Theorem ([Kothapalli, Onus, Scheideler, Schindelhauer 2006])

Any anonymous MIS alg with message size $O(1)$ *(including* $\mathsf{SA}^{\updownarrow}$*) requires* $\Omega(\log n)$ *time in expectation.*

Open question: $\mathsf{SA}^{\updownarrow}$ alg that solves MIS with runtime $O(\log n)$?

Local automaton of the SA$^{\updownarrow}$ MIS algorithm



Self-stabilization

Theorem ([Emek, Keren 2021])

There exists a sync self-stab $\mathsf{SA}^{\circlearrowleft}$ *alg that solves MIS with state space* $O(D)$ *and runtime* $O((D + \log n) \cdot \log n)$ *w.h.p.*

- No conflict with impossibility result of [AEK] since
  - D is large in bad graph
  - Requires termination detection
    - Self-stab may be easier!

Self-stabilization — cont.

- [Giakkoupis, Ziccardi 2023]:
  sync self-stab SA$^{\updownarrow}$ MIS alg called *"the 2-state process"*
  - $Q = \{\texttt{IN}, \texttt{OUT}\}$
  - Node $v \in V$ is *satisfied* if either
    - $q_t(v) = \texttt{IN}$ and $\{q_t(u) \mid u \in N(v)\} = \{\texttt{OUT}\}$
    - $q_t(v) = \texttt{OUT}$ and $\texttt{IN} \in \{q_t(u) \mid u \in N(v)\}$
  - If v is satisfied, then $q_{t+1}(v) \leftarrow q_t(v)$
  - If v is unsatisfied, then $q_{t+1}(v) \in_r \{\texttt{IN}, \texttt{OUT}\}$

Theorem ([Giakkoupis, Ziccardi 2023])

The runtime of the 2-state process on $G_{n,p}$ is $\log^{O(1)} n$ w.h.p. if $p \leq \log^{O(1)}(n)/\sqrt{n}$ or $p \geq 1/\log^{O(1)}(n)$.

- Conjecture that runtime upper bound holds for any graph

Fresh from the oven

Theorem ([Emek, Louidor 2023])

There exists a graph on which the (stabilization) runtime of the 2-state process is $n^{\Omega(1)}$ *w.p.* $\Omega(1)$.

- Discussions with Giakkoupis and Haeupler:
  - Holds also for *"the 3-state process"*
    - $\mathrm{SA}^{\circlearrowleft}$ variant of the 2-state process
  - Holds even if $\langle q_0(v) \rangle_{v \in V}$ is chosen u.a.r.

Open question: self-stab SA alg for MIS with state space $O(1)$ and runtime $\log^{O(1)} n$?

Dynamic topology changes

Theorem ([Emek, Uitto 2020])

There exists a sync $\mathrm{SA}^{\updownarrow}$ *alg that solves MIS on dynamic graphs with state space* $O(1)$ *and runtime* $O((C+1) \cdot \log^2 n)$ *w.h.p., where* C *is the number of dynamic topology changes.*

- $O(\log^2 n)$ time amortized over topology changes
- Effect of topology changes is confined:
  "local runtime" of distant nodes is $O(\log^2 n)$ w.h.p.

Theorem ([Emek, Uitto 2020])

There does not exist a sync $\mathrm{SA}^{\updownarrow}$ *alg that solves MIS on dynamic graphs with state space* $O(1)$ *and runtime better than* $\Omega(C+1)$ *w.h.p., where* C *is the number of dynamic topology changes.*

Bounded number of candidates

k-leader selection:
elect a leader out of $\leq k$ (and ≥ 1) initially marked candidates

Theorem ([Afek, Emek, Kolikant 2018])

For every k, there exists a sync $\text{SA}^{\circlearrowleft}$ alg that solves k-leader selection with state space $k^{O(1)}$ and runtime $\tilde{O}(D)$ w.h.p.

Theorem ([Afek, Emek, Kolikant 2018])

There does not exist a sync $\text{SA}^{\updownarrow}$ alg that solves k-leader selection with state space independent of k w.p. $\Omega(1)$.

Self-stabilizing leader election

Theorem ([Emek, Keren 2021])

There exists a sync self-stab $\mathrm{SA}^{\circlearrowleft}$ *alg that solves leader election with state space* $O(D)$ *and runtime* $O(D \log n)$ *w.h.p.*

- No conflict with impossibility result of [AEK] since
  - D is large in bad graph
  - Requires termination detection

Open question: self-stab SA alg for leader election with state space $O(1)$?

Simulating Turing machines

- [EW]: SA$^{\circlearrowleft}$ on path can simulate randomized $O(n)$-space TM
  - TM input encoded in $\langle q_0(v)\rangle_{v \in V}$ "along path"
- How TM input encoded in arbitrary graphs?
  - No natural order among nodes
- Solution: designated *I/O node*:
  - Gets TM input bit-by-bit
  - Returns TM output bit-by-bit

Theorem ([Afek, Emek, Kolikant 2018b])

Any computational task that can be solved by a randomized $O(n)$-space TM with runtime R w.h.p. can be solved by a sync SA$^{\circlearrowleft}$ *alg on bounded degree graphs with state space $O_{\Delta}(1)$ and runtime $O_{\Delta}(R + D)$ w.h.p.*

Wrapping up

- **Take home message 1:** biology through the lens of distributed computing
  - BDA 2023 (co-located with PODC 2023)
- **Take home message 2:** even with (very) weak individual nodes, entire network can be quite powerful
- **Take home message 3:** hard to justify graceful initialization
  - Self-stabilization rocks
    - May be easier (no termination detection requirement)
- Other SA flavors
  - Message passing (async) SA
  - Generalization of set-broadcast to one-two-many counting
  - SA-inspired mobile agents (ANTS problem)
- Additional distributed tasks
  - 3-coloring trees, (α, β)-ruling set, (existentially) small k-dominating set, optimization problems, ...

MUCHAS GRACIAS