DISTRIBUTED ALGORITHMS AND BIOLOGICAL SYSTEMS

Nancy Lynch, Saket Navlakha
BDA-2014
October, 2014
Austin, Texas
Distributed Algorithms + Biological Systems

- Distributed algorithms researchers have been considering biological systems for the past few years, looking for:
  - Biological problems and behaviors that they can model and study using distributed algorithms methods, and
  - Biological strategies that might be adapted for use in computer networks.

- This has yielded interesting distributed algorithms results.

Q: But what can distributed algorithms contribute to the study of biological systems?

This talk:
- Overview fundamental ideas from the distributed algorithms research area, for biology researchers.
- Consider how these might contribute to biology research.
What are distributed algorithms?

- Abstract models for systems consisting of many interacting components, working toward a common goal.

- Example systems:
  - Wired or wireless network of computers, communicating or managing data.
  - Robot swarm, searching an unknown terrain, cleaning up, gathering resources, …
What are distributed algorithms?

• Abstract models for systems consisting of many interacting components, working toward a common goal.

• Example systems:
  • Wired or wireless network of computers, communicating or managing data.
  • Robot swarm, searching an unknown terrain, cleaning up, gathering resources,…
  • Social insect colony, foraging, feeding, finding new nests, resisting predators,…

• Components generally interact directly with nearby components only, using local communication.
Distributed algorithms research

- Models for distributed system platforms.
- Problems to be solved.
- Algorithms, analysis.
- Lower bounds, impossibility results.
- **Typical problems:** Communication, consensus, data management, resource allocation, synchronization, counting,…

- **Models:**
  - Interacting automata.
  - Local communication: individual message-passing, local broadcast, or shared memory.
  - Metrics: Time, amount of communication, local storage.
Algorithms

• Some based on simple rules, some use complex constructions.
• Designed to minimize costs, according to the cost metrics.
• Often designed to tolerate limited failures.
• Researchers analyze correctness, costs, and fault-tolerance.
• Try to “optimize”, according to the metrics.
Lower bounds and other impossibility results

• Theorems that say that you can’t solve a problem in a particular system model, or you can’t solve it with a certain cost.
• Distributed computing theory includes hundreds of impossibility results.
  •Unlike sequential computing theory.
  •Because distributed platforms are hard to cope with: locality of knowledge and action impose strong limitations.
Formal modeling

• Distributed algorithms can be complicated:
  • Many components act concurrently.
  • May have different speeds, failures.
  • Local knowledge and action.
• In order to reason about them carefully, we need clear mathematical foundations.
Formal modeling

- Model systems using **interacting automata**.
  - Not finite-state automata, but more elaborate automata that may include complex states, timing information, probabilistic behavior, and both discrete and continuous state changes.
  - Support composition and abstraction.
  - Support rigorous analysis for correctness and costs.
Key ideas

• Distinguish among:
  • The problems to be solved,
  • The platforms on which the problems must be solved, and
  • The algorithms that solve the problems on the platforms.

• Define cost metrics, such as time, local storage space, and amount of communication.

• Use the metrics to analyze and compare algorithms and prove lower bounds.

• Q: How could this approach help biology research?
Biology research

• Model a system of insects, or cells, using interacting automata.
• Define formally, separately:
  • The problems the systems solve (distinguishing cells, building structures, foraging, reaching consensus, task allocation, …),
  • The physical capabilities of the systems, and
  • The strategies (algorithms) that are used by the systems.
• Identify cost metrics (time, energy,…)
• Analyze and compare strategies.
• Prove inherent limitations.
• Use the results to:
  • Predict system behavior.
  • Explain why a biological system has evolved to have the structure and behavior it has.
The rest of the talk:

Two standard examples from distributed algorithms:
1. Leader election
2. Maximal independent set

How one might apply distributed algorithm ideas in biology

Two preliminary biology-related examples:
3. Ant foraging
4. Ant task allocation

Saket: More biology-related examples
Example 1: Leader election

- Ring of processes.
- Computers, programs, robots, insects, cells,…
- Communicate with neighbors by sending “messages”, in synchronous rounds.
Example 1: Leader election

- Ring of processes.
- Computers, programs, robots, insects, cells,…
- Communicate with neighbors by sending “messages”, in synchronous rounds.
- **Problem:** Exactly one process should (eventually) announce that it is the leader.
- **Motivation:**
  - A leader in a computer network or robot swarm could take charge of a computation, directing everyone else’s activity.
  - A leader ant could choose a new nest.
Leader election

- Suppose that:
  - Processes start out identical.
  - The behavior of each process at each round is determined by its current state and incoming messages.

- **Theorem:** In this case, it’s impossible for any distributed algorithm to elect a leader.

- **Proof:** By contradiction.
  - Suppose we have an algorithm that works.
  - All processes start out identical.
  - At round 1, they all do the same thing (send the same messages, make the same state changes), so they are again identical.
  - Same for round 2, etc.
  - Since the algorithm solves the problem, some process must eventually announce that it is the leader.
  - But then everyone does, contradiction.
If processes aren’t identical:

- E.g., they have unique ID numbers.

**Algorithm:**
- Send a message containing your ID clockwise.
- When you receive an ID, compare it with your own ID.
- If the incoming ID is:
  - Bigger, pass it on.
  - Smaller, discard it
  - Equal, announce that you are the leader.

- Elects the process with the largest identifier.
- Takes $O(n)$ communication rounds, $O(n^{1/2})$ messages.
If they are identical, but can make random choices:

- No unique IDs.
- Assume they know $n$, the total number of processes.

**Algorithm:**
- Toss an unfair coin, with probability $1/n$ of heads.
- If you toss heads, become a “leader candidate”.
- It’s “pretty likely” that there is exactly one candidate.
- The processes can verify this by passing messages around and seeing what they receive.
- If they did not succeed, try again…

- Expected number of attempts is constant.
Example 2: Maximal Independent Set

• Assume a general graph network, with processes at the nodes:

• Problem: Select some of the processes, so that they form a Maximal Independent Set.

• Independent: No two neighbors are both in the set.

• Maximal: We can’t add any more nodes without violating independence.

• Motivation:
  • Communication networks: Selected processes can take charge of communication, convey information to all the other processes.
  • Developmental biology: Distinguish cells in fruit fly’s nervous system to become “Sensory Organ Precursor” cells.
Maximal Independent Set

- **Problem:** Select some of the processes, so that they form a Maximal Independent Set.
- **Independent:** No two neighbors are both in the set.
- **Maximal:** We can’t add any more nodes without violating independence.
Maximal Independent Set

- **Problem**: Select some of the processes, so that they form a Maximal Independent Set.
- **Independent**: No two neighbors are both in the set.
- **Maximal**: We can’t add any more nodes without violating independence.
Distributed MIS problem

• Assume processes know a “good” upper bound on n.
• No IDs.

• Problem: Processes should cooperate in a distributed (message-passing) algorithm to compute an MIS of the graph.
• Processes in the MIS should output in and the others should output out.

• Unsolvable by deterministic algorithms, in some graphs.
• Probabilistic algorithm:
Probabilistic MIS algorithm

- **Algorithm idea:**
  - Each process chooses a random ID from 1 to N.
  - N should be large enough so it’s likely that all IDs are distinct.
  - Neighbors exchange IDs.
  - If a process’s ID is greater than all its neighbors’ IDs, then the process declares itself **in**, and notifies its neighbors.
  - Anyone who hears a neighbor is **in** declares itself **out**, and notifies its neighbors.
  - Processes construct a reduced graph, omitting those who have already decided.
  - Repeat with the reduced graph, until no nodes remain.
Example

- All nodes start out identical.
Example

- Everyone chooses an ID.
Example

• Processes that chose 16 and 13 are in.
• Processes that chose 11, 5, 2, and 10 are out.
- Undecided (gray) processes choose new IDs.
Processes that chose 12 and 18 are in.
Process that chose 7 is out.
• Undecided (gray) process chooses a new ID.
Example

- It’s in.
Properties of the algorithm

- If it ever finishes, it produces a Maximal Independent Set.
- It eventually finishes (with probability 1).
- The expected number of rounds until it finishes is $O(\log n)$.
More examples

• **Building spanning trees** that minimize various network cost measures.

• **Motivation:**
  - Communication networks: Use the tree for sending messages from the leader to everyone else.
  - Slime molds: Build a system of tubes that can transport nutrients from several food sources.

• **Building other network structures:**
  - Routes
  - Clusters with leaders.
  - …
More examples

• Reaching consensus, in the presence of faulty components (stopping, Byzantine).

• Motivation:
  • Agree on aircraft altimeter readings.
  • Agree on processing of data transactions.
  • Ants: Agree on a new nest location.

• Communication
• Resource allocation
• Task allocation
• Synchronization
• Data management
• Failure detection
• …
Recent Work: Dynamic Networks

- Most of distributed computing theory deals with fixed, wired networks.
- Now researchers are also studying dynamic networks, which change while they are operating.
  - E.g. wireless networks, robot swarms.
  - Participants may join, leave, fail, and recover.
  - May move around (mobile systems).
Computing in Dynamic Graph Networks

- Network is a graph that changes arbitrarily from round to round (but it’s always connected).
- At each round, each process sends a message, which is received by all of its neighbors at that round.

Problems:
- Global message broadcast,
- Determining the minimum input.
- Counting the total number of nodes.
- Consensus
- Clock synchronization
Robot Coordination Algorithms

- A swarm of cooperating robots, engaged in:
  - Search and rescue
  - Exploration
- Robots communicate, learn about their environment, perform coordinated activities.

Problems:
- Keep the swarm connected for communication.
- Achieve “flocking” behavior.
- Map an unknown environment.
- Determine global coordinates, working from local sensor readings.
Biological Systems as Distributed Algorithms

- Biological systems consist of many components, interacting with nearby components to achieve common goals.
- Colonies of bacteria, bugs, birds, fish, …
- Cells within a developing organism.
- Brain networks.
Biological Systems as Distributed Algorithms

- They are special kinds of distributed algorithms:
  - Use simple chemical “messages”.
  - Components have simple “state”, follow simple rules.
  - Flexible, robust, adaptive.
Problems

• Leader election: Ants choose a queen.
• Maximal Independent Set: In fruit fly development, some cells become sensory organs.
• Building communication structures:
  • Slime molds build tubes to connect to food.
  • Brain cells form circuits to establish memories.
More problems

- Consensus: Bees agree on location of a new hive.
- Reliable local communication: Cells use chemical signals.
- Robot swarm coordination: Birds, fish, bacteria travel in flocks / schools / colonies.
Biological Systems as Distributed Algorithms

- So, we can study biological systems as distributed algorithms.
- Define models, problem statements.
- Devise algorithms.
- Prove impossibility results.
- Goals:
  - Use distributed algorithms to understand biological system behavior.
  - Use biological systems to inspire new distributed algorithms.
Biology-related examples:

3. Ant foraging
4. Ant task allocation
Example 3: Ant Foraging

- [Lenzen, Lynch, Newport, Radeva, PODC 2014]
- $n$ ants exploring a 2-dimensional grid for food, hidden within distance $D$ of the nest.
- No communication.
- Ants can return to the nest at any time.
- In terms of $D$ and $n$, we get an upper bound on the expected time for some ant to find the food.
- We use an algorithm similar to [Feinerman, Korman].

- But we assume strict bounds on:
  - The size of an ant’s memory.
  - The fineness of probabilities used in an ant’s random choices.
Ant Foraging

- **Algorithm** (for each ant):
  - Multi-phase
  - In successive phases, search to successively greater distances.
  - Distances are determined by random choices, using smaller probabilities at later phases.
- **Expected time for some ant to find the food is (roughly) $O(D^{1/2}/n + D)$**.
- Even in the “nonuniform” case, where ants don’t have a good estimate of $D$.
- **Analysis methods**: Basic conditional probability analysis, Chernoff bounds.
Ant Foraging

- Assuming slightly smaller bounds on ant memory size and fineness of probabilities, we get:
  - **Lower bound:** There is a food placement such that, with high probability, the time for the first ant to find the food is (roughly) $\Omega(D^{1/2})$.

**Proof methods:**
- Model movement of an ant through its state space as a Markov chain.
- Enhance this chain with information about the ant’s movement in the grid.
- Use properties of the enhanced Markov chain to analyze probabilities of reaching various locations in the grid within certain amounts of time.
Example 4: Ant Task Allocation

- [Cornejo, Dornhaus, Lynch, Nagpal 2014]
- \(n\) ants allocate themselves among a fixed set of tasks (foraging for food, feeding larvae, cleaning the nest…)
- No communication.
- Obtain information from the environment about tasks’ current energy requirements.
- Try to minimize sum of squares of energy deficits for all tasks.
- Special case we’ve studied:
  - All ants are identical.
  - Synchronous rounds of decision-making.
  - Simple binary info (deficit/surplus for each task).
Ant Task Allocation

• **Algorithm:**
  - Probabilistic decisions, based on deficit/surplus info.
  - **States:** Resting, Reserve1, Reserve2, TempWorker, CoreWorker.
  - Worker ants work on tasks; others are idle.
    - Temps are more biased towards becoming idle.
    - Reserves are biased towards the task they last worked on.
  - Detailed state transition rules, task assignment rules.

• **Theorem:** In a period of length $\Theta(\log n)$ with unchanging demand, with high probability, the ants converge to a fixed allocation that satisfies all demands.

• **Proof:** Analyzes how oscillations get dampened.
Limited ant memory size...

This is my brother Dave. He's real smart - brain the size of a pea!