

Exploring the similarities and differences between distributed computations in biological and computational systems.

BY SAKET NAVLAKHA AND ZIV BAR-JOSEPH

Distributed Information Processing in Biological and Computational Systems

BIOLOGICAL SYSTEMS, RANGING from the molecular to the cellular to the organism level, are distributed and in most cases operate without central control. Such systems must solve information processing problems that are often very similar to problems faced by computational systems, including coordinated decision making,²⁹ leader election,² routing and navigation,⁵² and more.⁴²

Over the last few years our ability to study and model biological systems has improved dramatically. Using advanced sequencing technologies we can now determine the composition of the genomes of hundreds of organisms. For specific cells and tissues



» key insights

- Biological and computational systems are often required to solve similar distributed information processing problems including coordinated decision making, leader election, routing, and navigation.
- The emergence of new computing technologies (wireless, sensor, and mobile computing), coupled with our ever-increasing ability to obtain large quantitative datasets describing biological systems at unprecedented details, opens the door to new joint studies of these two domains.
- Bidirectional studies in which researchers use ideas from one domain to study the other can concurrently lead to improved biologically inspired algorithms and novel computational understandings of how biological systems function.



we can query the set of active genes, their expression levels, and how their responses change in different conditions and over time. We can also determine which molecules within cells interact and how such interactions are wired to form the control diagram regulating cellular activity. Using this data we can build models of information processing within and between cells and answer questions regarding the methods biological systems use to achieve their goals that were out of reach even a few years ago. This data can also help us understand what computational problems are being solved by biological systems and how, which in turn can lead to specific algorithms that may also benefit computational systems.

Theoretical distributed computing has also been transformed by the recent adaptation and pervasiveness of wireless and mobile computing devices. These technologies have introduced new computational problems not typically faced by traditional, wired-based distributed systems.³³ Even in cases where algorithms initially designed for wired networks can be employed, new solutions are required to account for the dynamic nature and new constraints imposed by mobile devices (including energy conservation,⁴ limited transmission range,²⁸ reliance on broadcast communication,³³ and many more).

These two converging technological changes have led to several recent studies in which researchers use ideas from

one domain (either biology or computer science) to study the other. Such bidirectional studies can concurrently lead to biologically inspired algorithms and novel computational understandings of how biological systems function, which in turn can lead to new testable hypotheses (for example, Tero et al.⁵² and Afek et al.²). Several recent reviews have discussed these studies primarily focusing on the type of computational problem⁴² (networks-related, coordination, computer vision) or a specific biological system and their modeling (flocking birds, social insects).^{14,22}

Our goal in this article is to focus on the similarities and differences in the constraints, goals, and algorithms employed in both domains, especially

with regard to distributed information processing. We hope this perspective will allow researchers in both areas to focus on the most promising problems that can, and should, be studied bidirectionally. We first discuss several constraints that affect models of communication between entities (molecules, cells, mobile devices, and so on) in the two domains (Figure 1). Next, we argue that speed (or runtime), a key optimization goal for computational algorithms, is typically less important for biological systems, which focus more on robustness and adaptability. Lastly, we discuss similarities and differences in algorithmic strategies employed by both systems to achieve their goals under these different constraints.

The key for successful studies at the intersection of distributed computing and biology is to identify problems in which similar constraints and goals may apply to both systems. Networks provide one of many popular abstractions that have been immensely useful in understanding large, distributed systems. In biology, networks depict how molecules (metabolites, proteins), cells (bacteria, neurons), or organisms (ants) interact to jointly solve problems and coordinate responses. In computer sci-

ence, they depict how processors, machines, and devices communicate and process information. On the biological side, systems that involve dynamic networks and message passing (either within and between cells or between members of a population) are often well suited for ‘distributed thinking.’ From the computational point of view, mobile and sensor networks are ideal candidates that can benefit from new models and algorithms. In this article, we take a broad perspective of networks at all levels of life to demonstrate how distributed perspectives may apply and guide future investigations.

Communication Models

Most biological systems are distributed and must make decisions and respond to stimuli without a centralized coordinator and under severe constraints (energy conservation, limited communication range, limited messaging language, among others). While computer scientists have explored distributed computing algorithms for decades, the study of a class of severely limited communication models is more recent and has largely emerged to support mobile and sensor networks that are often required to operate for long durations

with limited resources and energy.⁴

Most distributed communication models are based on message passing. Even simple algorithms under such models use messages whose size is logarithmic in the number of participating nodes, which allows messages to include a unique identifier for both sender and receiver (for example, Luby’s famous algorithm under PRAM,³⁶ which can be adjusted to run under the message passing model). While such messages are very small compared to most traffic requirements in communication networks (for example, movie downloads), there are cases where even logarithmic message size may be problematic. For example, in crowded wireless networks, interference may cause larger messages to be dropped or missed,⁵¹ whereas in sensor networks, energy conservation also necessitates smaller messages. Information processing in biology is also often based on message passing. Cells secrete proteins to interact with other (neighboring or distant) cells in order to activate various signaling networks. While the number of proteins that can be secreted, and their levels, can vary greatly, there is recent evidence that most biological communication involves mes-

Figure 1. Similarities and differences in the properties of computational and biological systems.

Top row: Models, goals, and algorithmic strategies often used by conventional distributed systems but rarely in biological information processing. Bottom row: Shared features of dynamic distributed and biological systems. These and other common aspects are the basis for studies that model information processing in biology or develop a biologically motivated algorithm for a distributed computing problem.

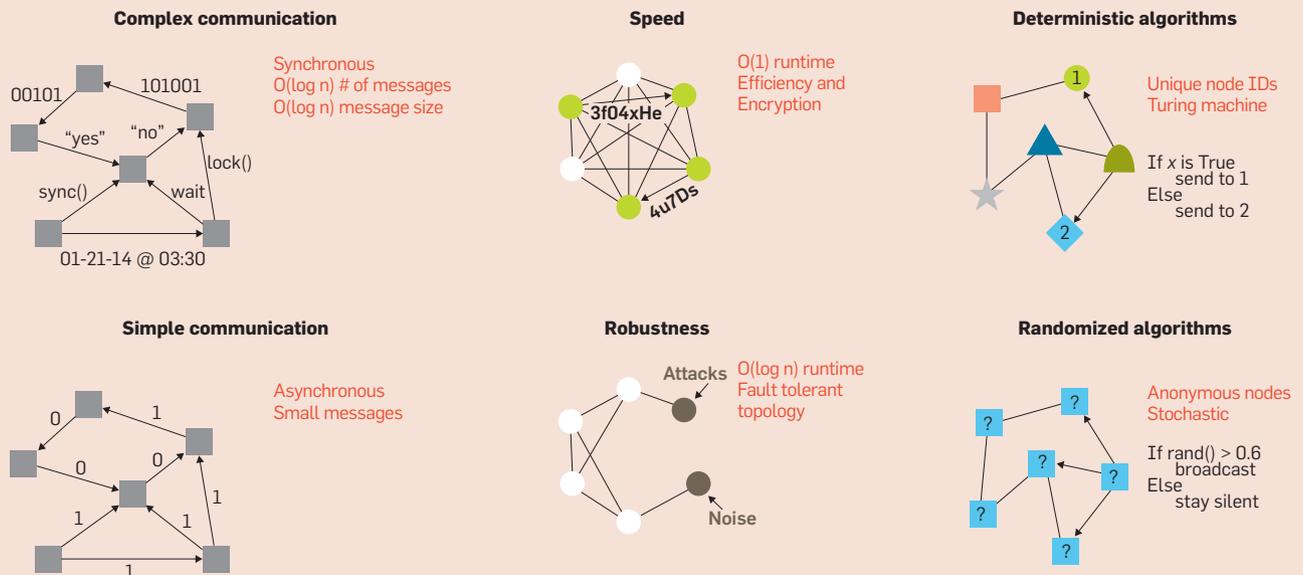
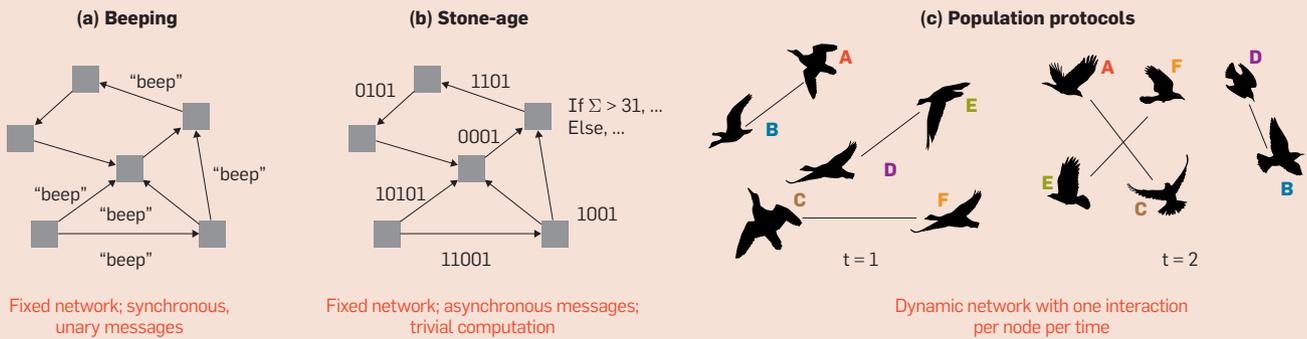


Figure 2. Distributed communication models.

The beeping model assumes an anonymous broadcast network with synchronous communication and allows only unary messages (beeps) to be sent. The stone-age model also assumes an anonymous broadcast network, but allows asynchronous communication; it also allows a richer messaging language than the beeping model but provides less computing power to each node. The population protocol assumes random, asynchronous pairwise interactions between agents also with limited memory and computational power.



sages whose effective information content is very small³¹ (on the order of one bit). For example, Cheong et al.¹⁵ recently demonstrated that cells cannot distinguish between varying levels of a certain protein input, leading to an effective binary message communicated by secreting this protein.

In addition to using a limited communication language (often of constant size), another goal shared by the two domains is reducing overall message complexity. Such reduction leads to more efficient use of available resources and energy (in biology, metabolites and for computational systems, electric power).

Motivated by these similar requirements, recent theoretical work in distributed computing and systems biology has analyzed the ability of extremely weak communication models to solve important computational problems and to explain the activity of key biological processes. In both cases the focus is on limiting message size and complexity, often at the expense of runtime. Here, we provide a few examples of the communication models that have been proposed for such systems and discuss both their use for solving fundamental distributed computing problems and their application to study the activity of complex biological processes.

Beeping: The beeping model¹⁷ (Figure 2a) assumes the only message that can be sent or received is a beep

(a unary signal). The model assumes an anonymous broadcast network in which nodes have no knowledge about the topology of the network or even an upper bound on its size. In each time slot a node can either beep or be silent. At a particular time slot, beeping nodes receive no feedback (they cannot determine if other nodes beeped as well), while silent nodes can only differentiate between two states: none of its neighbors beeping, or at least one neighbor beeping. Such a model is also appropriate for cellular signaling networks as discussed here.

Even with such limits on communication, several important distributed problems can be solved. The first problem solved under the beeping model was interval coloring, a variant of vertex coloring.¹⁷ Given a set of resources, the goal of interval coloring is to assign every node a large contiguous fraction of the resources, such that neighboring nodes have disjoint resources. Using beeping, the problem can be solved in $\mathcal{O}(\log n)$ time with high probability compared to $\mathcal{O}(\sqrt{\log n})$ when using unrestricted message sizes. More recently beeping was used to solve an even harder coordination problem: Maximal Independent Set (MIS). MIS attempts to find a subset of the nodes in the network such that: (1) Every node is either a MIS node or directly connected to one and (2) no two nodes in the set are connected to each other. MIS is a basic procedure in distributed computing and

serves as a building block for several algorithms including routing and clustering. Under mild additional assumptions (nodes can be woken up by neighbors' beeps), MIS can be solved in the beeping model in $\mathcal{O}(\log^2 n)$ time, as opposed to the classic $\mathcal{O}(\log n)$ solution when using larger message sizes (Luby's algorithm¹) or as opposed to relying on knowledge of the topology (Metivier's algorithm³⁹). The beeping model leads to an optimal communication load minimizing overall system requirements.² Follow-up work also showed that MIS could be solved under the beeping model in logarithmic time assuming sender collision detection.⁴⁹

Sensory Organ Precursor (SOP) selection. To illustrate the usefulness of the beeping model to study a biological system consider the SOP selection process, which is a key step in the development of the fruit fly brain. The process involves the selection of a subset of cells that later become sensory bristles on the fly's forehead. Similar to the MIS problem, such selection requires that no two neighboring cells become SOPs (known as lateral inhibition in biology) and that every cell is either a SOP or connected to a SOP. Cells communicate to determine which will become a SOP and while several mathematical models previously studied the two-way signaling involved in lateral inhibition (interactions between a SOP and one of its neighbors), only recently have biologists looked at the bigger picture: How

a subset of cells is selected from the overall population of cells. A variant of the beeping model can be used to explain such behavior (where the beeps in this case are a specific type of a protein called Delta).² This insight, coupled with new microscopy experiments following SOP selection in developing flies, led to the discovery of a novel stochastic feedback process used to determine cell fate and to a new distributed algorithm for MIS using the beeping model as discussed earlier.

Stone-age distributed computing.

While beeping uses a very limited set of messages, it assumes nodes can access internal memory to perform computations that is logarithmic in the size of the network (that is, nodes can count up to $\mathcal{O}(\log n)$). Emek et al.²⁰ proposed a new communication model based on a network of finite state machines (nFSM) (Figure 2b). The nFSM model assumes a richer set of messages coming from a fixed-size language. These messages are asynchronously delivered to a dedicated channel in the receiving node (similar to receptors on interacting cells). However, unlike the beeping model, in nFSM nodes can only count up to a constant number. In other words, when transitioning to a new state, nodes evaluate the set of incoming messages (from all neighbors, though these neighbors are anonymous) according to the one-two-many principle: a node can only count up to some predetermined number and all values beyond this threshold are indistinguishable. As mentioned earlier, such a model corresponds to recent biological findings regarding the limited ability of cells to ‘count’ the levels of incoming proteins.¹⁵ Using the nFSM model, Emek et al. showed MIS can be solved in $\mathcal{O}(\log^2 n)$ time and it can also be used to 3-color an undirected tree in $\mathcal{O}(\log n)$ matching the optimal bound for this problem.

Probabilistic inference by neurons.

One example of the potential usage of the nFSM model comes from networks of spiking neurons in the brain. Neuroscientists have experimentally shown that neurons are often “unreliable,” that is, there is significant trial-to-trial variability in neural output under the same input conditions.¹⁹ This suggests neural populations encode information by sampling from underlying

probability distributions. These distributions represent internal models of the external world that integrate new sensory stimuli with prior knowledge and memories. Recent computational work has remarkably shown that networks of stochastically firing neurons can carry forth probabilistic inference in a manner similar to Markov chain Monte Carlo sampling in distributed systems,¹¹ and such work has also led to several experimentally testable predictions about the firing dynamics of collections of neurons.⁴⁵ Neurons also use a one-two-many-like principle in the sense they count inputs within a time window up to a certain threshold before firing, and firing thresholds can vary depending on the type of neuron. Experimental work has also shown such internal representations are often sparse (only a few neurons are engaged per stimulus), indicating such energy-efficient representations may also be applicable to information processing problems in wireless sensor networks.²⁷

Population protocols. Unlike the previous two models that rely on broadcast, population protocols are based on direct, asynchronous physical interactions between a pair of agents⁶ (Figure 2c). Such interaction models are very common in nature and indeed, the development of population protocols has been motivated by a study of sensor networks attached to a flock of birds and was also recently applied to study distributed sensor networks on zebras.⁹ The model assumes that in each time slot, interactions occur between a pair of agents, which allows them to directly exchange messages and update their states. Unlike standard networks though, these interactions are either random or scheduled by an adversary, subject to a fairness constraint, which provide weak guarantees about the ability of every pair to interact eventually. Several types of problems can be solved distributively under this model including OR computations, majority, summation, and under some additional assumptions regarding the inputs, leader election, and consensus.⁶

Ant foraging. While some ants communicate by leaving pheromone trails, several harvester ant species only interact by direct physical contact. However, these ants are still able to find food

sources, while also recruiting other ants in the search and determining the amount of food available in an environment. A recent study demonstrated that with limited communication, ants solve the foraging problem by implementing a version of the Transmission Control Protocol (TCP), which is used on the Internet to determine available bandwidth when routing packets.⁴⁶ If packet acknowledgments (ACKs) are received quickly, the sender assumes bandwidth is available and boosts transmission; but if ACKs are returned slowly, the sender assumes the network is congested and throttles down transmission. Similarly, the important factor for the ants is the rate of antennal contacts (a binary indicator) between ants currently in the nest and successful ants (with food) returning to the nest. If the rate of contact is high, it implies food in the environment is plentiful, and thus outgoing ants also leave the nest at a faster rate.

Shared memory models. The previous models assume nodes communicate by exchanging messages. Another popular distributed communication method is the use of shared memory.³⁷ In such models, readers read from the shared memory, writers write to the shared memory, and erasers (which may or may not be the writers), remove data from this shared memory. Several papers have discussed the relationships between message passing and shared memory models. A classic result in this area is that any message-passing algorithm can also be solved in the shared memory framework,⁷ and vice versa, though runtime may drastically increase.

Chromatin computation. While most communication in biology is through message passing, in recent years shared memory has also emerged as a novel type of communication between proteins inside cells. Specifically, proteins were shown to modify DNA (by leaving, erasing, or reading specific marks on a set of proteins called histones over which DNA is wrapped). These marks play an important role in regulating the expression and activity of genes. There are many proteins (processors) that interact with these histones, which can lead to downstream effects on expression levels of genes next to marked sites. While the

large-scale study of such modifications is still in its infancy, issues that have been addressed by distributed algorithms—including competition, ordering of shared memory access, and the set of proteins that are able to access a specific site (memory location)—are of great current interest in molecular biology.³⁰ Recent studies have proposed computational models for such processes,¹⁰ and it is widely believed that histone modification and other epigenetic events play vital roles in development.

In addition to the specific communication type (broadcast or physical), message size (unary, fixed, or logarithmic) and buffer size, several other issues can be studied in the context of limited communication protocols for both biology and computational systems. These include various collision detection models, asynchronous vs. synchronous (or weakly synchronous) models,¹ bounded synchrony models,⁴⁴ and wakeup protocols.⁸ In all cases, parallels between biological and computational systems are leading to new insights that benefit both fields.

Speed vs. Robustness

Algorithms using restricted communication models often require longer runtimes compared to methods that utilize larger messages. This is no coincidence. Biological algorithms often need to trade off among speed, accuracy, and robustness, especially in noisy environments.¹⁶ Robustness can be improved by using extremely weak communication models that require few assumptions, as mentioned here. Here, we discuss another issue that affects robustness: network topology.

In this context, robustness denotes the ability of an algorithm to tolerate faults. Many models have been proposed in the distributed computing literature based on common types of failures, including link failures (where a single edge in the network is lost), node failures, and Byzantine failures (where a node is compromised and can participate in adversarial attacks). Many algorithms have been proposed to solve key distributed computing problems under each of these failure models, though they often assume fixed topologies, such as rings or cliques.³⁷ In biological systems, link and node



The key for successful studies at the intersection of distributed computing and biology is to identify problems in which similar constraints and goals may be applied to both systems.

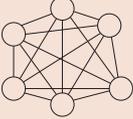
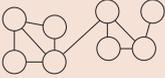
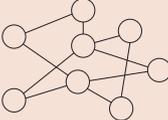


failures are also common; for example, mutation or protein misfolding can result in loss of specific interaction,⁵⁷ and genetic mutations can also result in a complete loss of a gene. Targeted attacks on specific nodes are also common, for example, in host-pathogen interactions, where virus proteins attack host proteins in an attempt to infect host cells.³² Overcoming these types of failures is a key requirement for any self-sustaining biological system.

While both computational and biological systems need to address these similar types of failures, the methods they use to do so differs. In distributed computing, failures have primarily been handled by majority voting methods,³⁷ by using dedicated failure detectors,²⁵ or via cryptography.⁴¹ In contrast, most biological systems rely on various network topological features to handle failures. Consider for example the use of failure detectors. In distributed computing, these are either implemented in hardware or in dedicated additional software. In contrast, biology implements implicit failure detector mechanisms by relying on backup nodes²⁴ or alternative pathways. Several proteins have paralogs, that is, structurally similar proteins that in most cases originated from the same ancestral protein (roughly 40% of yeast and human proteins have at least one paralog²⁶). In several cases, when one protein fails or is altered, its paralog can automatically take its place²⁴ or protect the cell against the mutation.²⁶ Thus, by preserving backup functionality in the protein interaction network, cells can overcome node failures without explicit use of failure detection mechanisms. While node failures occur within cells, a much more common scenario is the need to handle noisy, unreliable, inputs. For example, fluctuating environmental conditions can make it difficult for a bacteria to decide whether to sporulate or germinate.³⁸ In molecular networks, environmental noise can make it difficult to determine what type of regulatory response is needed and how quickly. Noise may also spread through the network and infect communication partners in a similar manner to epidemiological virus propagation models. To handle such Byzantine-like failures biological systems have optimized the topology

Figure 3. The effect of topology on speed and robustness.

Cliques are highly efficient in terms of routing distance and can tolerate any single node failure, but are quickly overcome by cascading failures or noise. Weakly connected networks are slightly less efficient and cannot overcome targeted attacks on bottlenecks, but they typically can isolate cascading failures to localized modules. Sparsely connected networks have longer routing time but can better overcome both types of failures.

		Speed	Robust (single-point attack)	Robust (cascading failure)
Densely connected (cliques)		✓	✓	✗
Weakly-connected modules		✓	✗ (bottlenecks)	✓
Sparsely connected		✗	✓	✓

of the networks they utilize (Figure 3).⁴³ For example, dense topologies with clique-like structures are often used in instances where little-to-no noise is expected, whereas sparser topologies are preferred when networks are expected to face more noise.⁴⁰ Of course, sparser topologies are also less efficient (in terms of routing distance, for example) which means execution times will be longer for such topologies. Weakly linked modules, on the other hand, can isolate occasional noise into nearly independent modules that each perform efficiently.⁵⁶

Along with robustness, biological algorithms are also designed to be adaptive and this is reflected in their underlying network topology. For example, activity-dependent plasticity of synapses is a well-known phenomenon by which neural networks are shaped by environmental stimuli. These signals are processed in a streaming fashion, and input patterns can change the topology of the networks designed to process them. Foraging slime molds have also been shown to adaptively adjust their networks of tubular junctions based on the distribution and availability of food sources in the area, which is typically unknown a priori.⁵² Such adaptive behavior often comes at the expense of runtime and resources. Indeed, while most body organs are

well-formed in young babies (though they still grow in size, their functionality does not change), the human brain overproduces synapses by 50%–60% during development and only converges to a more stable neural circuit in late adolescence. Slime molds also forage using breadth-first search using real cellular material, which is later pruned when optimal paths are found. In both cases, these systems sacrifice speed and resources for robustness of the resulting networks. These issues (of dynamic network structure and participation) are also key in the design of mobile ad hoc networks³³ and ideas from the biological systems mentioned here may prove useful for problems that communication networks face.

Strategies Used By Distributed Algorithms

We have described the operating constraints (communication models) and the goals (speed and robustness) applicable to biological and computational systems. In this section, we focus on differences in the algorithmic strategies used to achieve these goals and how such strategies are utilized in both domains.

One of the most widely used strategy in biological systems is stochastic decision-making. Randomness is used by biological processes to break

symmetry,² to overcome noise, and to ensure the survival of a population under changing environmental conditions.³⁴ Given the similarities mentioned earlier, it is not surprising that stochasticity has also long been employed within distributed algorithms for very similar purposes. For example, for leader election it was shown that if all processes are initially identical it is impossible to elect a leader without using randomized algorithms (in which case, it can be solved only with high probability⁵). Similarly, no deterministic algorithm can solve consensus even if only one node fails, while randomized algorithms can handle such failures and foil adversaries.³⁷ Another widely used strategy is feedback, both positive and negative. The use of feedback requires networks that contain several (often quite short) loops. For example, feedback inhibition is an important mechanism used to control the amount or concentration of a substance produced by a biochemical pathway to the appropriate level regardless of its current state or environmental availability. Homeostatic plasticity is another feedback mechanism that regulates activity levels of neurons to a certain range as a means to control circuit activity and prevent runaway excitation (seizures). Feedback is also central to several computational routing and initialization algorithms (using various types of backward acknowledgments). While useful, feedback can also amplify errors and noise in the system. This occurs in many message-passing schemes when incorrect or malicious information is distributed and amplified through the network over time. Biological systems can deal with this by adjusting network topology as mentioned earlier.

One common difference is in using unique identifiers for nodes. As mentioned in the communication models section, message sizes are usually one bit or of constant size indicating that unlike many traditional distributed algorithms, biological processes do not use such an identifier to label the sender and receiver. While targeted interactions do exist in biology (for example, synapses between two neural cells or domains mediating protein interactions), a node receiving two

messages usually cannot tell if these messages came from the same node or two different nodes.

Finally, while several distributed computing algorithms use majority voting³⁷ to solve coordination problems, biological systems often employ weighted voting schemes. This allows some nodes to have a greater influence on a population based on their own belief. For example, in bacterial swarms, a subgroup may find an undesirable path when foraging and lead the entire population astray. To overcome this, it was found that bacterium can dynamically adjust their decisions based on their own confidence and messages received from other cells.⁵⁰ While many of the rules used by these and other weighted-voting systems are yet to be worked out, they will likely be applicable in similar computing scenarios, for example, when programming distributed robot swarms for search-and-rescue operations.

Discussion

A resurgence of interest in studying how distributed biological systems process information and solve computational problems has occurred during the last few years. This revival has largely been triggered by two phenomena: first, our ability to experimentally probe the inner workings of molecular and cellular systems using a variety of new technological devices; and second, the emergence of new distributed computing models inspired by the pervasiveness of mobile, wireless, and sensor devices. Together, the new data and communication

models present a unique opportunity to jointly model, analyze, and learn from biological systems.

To effectively perform such bi-directional studies it is important that researchers realize both the similarities and differences between the models, goals, and algorithms used by the two domains. Most distributed computing models allow for large messages and high communication loads. Algorithms to address problems under such models often focus on speed (rounds), assume fully connected networks, and in many cases are deterministic, breaking symmetry by relying on unique identifiers. In contrast, information processing in biological systems uses much smaller messages (binary or constant), often emphasizes robustness over speed, uses incomplete (sometimes even sparse) networks, and is often stochastic. These constraints, goals, and methods are appropriate to some, but not all (or even most) computational problems. The accompanying table presents several relevant problems that have been addressed using insights from biological systems. In some cases, additional experiments were also required to understand how the problems were solved biologically and to derive algorithms for the corresponding computational problems.

While we discussed some reoccurring algorithmic strategies used within both types of systems (for example, stochasticity and feedback), there is much more to learn in this regard. From the distributed computing side, new models are needed to address the dy-

namic aspects of communication (for example, nodes joining and leaving the network, and edges added and being subtracted), which are also relevant in mobile computing scenarios. Further, while the biological systems we discussed all operate without a single centralized controller, there is in fact a continuum in the term “distributed.” For example, hierarchical distributed models, where higher layers “control” lower layers with possible feedback, represent a more structured type of control system than traditional distributed systems without such a hierarchy. Gene regulatory networks⁵⁵ and neuronal networks (layered columns) both share such a hierarchical structure, and this structure has been well-conserved across many different species, suggesting their importance to computation. Such models, however, have received less attention in the distributed computing literature.³ There are also many abstractions used for analyzing biological systems, ranging from discrete mathematics and graph theory to nonlinear dynamics and differential equations,⁵³ and such distributed computing models can likely benefit both modes of analysis.

Finally, as a cautionary note, biological algorithms designed by millions of years of natural selection do not guarantee optimality⁴⁷ and thus should not be couched in unrealistic terms. Indeed, some results discussed in this review (for example, network motifs,⁴⁰ and gene backup relationships²⁴) are based on high-throughput biological datasets that may be noisy and sometimes even contradictory.¹²

Examples of biological systems and their computational analogs.

Biological System	Computational Problem	Communication	Topology	Stochastic?	Alg.?	Refs.
Slime mold	Network routing	Stone-Age	Incomplete	Yes	Yes	35,48,52 48
Fly brain	Max. independent set	Beeping	Sparse	Yes	Yes	1,2
Harvester ants	TCP congestion control	Population	Random	No	Yes	46
Ants, swarms	Distributed search	Population	Random	Yes	Yes	23,50
Plants	Consensus	Stone-age	Incomplete	No	Yes	21
Fish schools	Consensus	Population	Random	Yes	No	29
Cell cycle switch	Approximate majority	Population	Random	Yes	Yes	13
Spiking neurons	Probabilistic inference	Stone-Age	Incomplete	No	Yes	11,45
Dendritic branching	Distributed MSTs	Stone-Age	Incomplete	No	Yes	18
Gannet colonies	Space partitioning	Population	Random	No	Yes	54
Protein interactions	Network design	Population	Random	Yes	Yes	43

Thus, care should be taken by analyzing multiple datasets to determine whether generalizations hold. There is, however, no doubt that evolution fine-tunes biological processes to optimize function. The emergent systems share many properties with those coveted in man-made distributed systems, and they also make interesting trade-offs between common optimization criteria for example, between efficiency and robustness, or runtime and message complexity) that can be learned from, especially to improve fault tolerance and adaptability. On the biological side, as technology continues to improve and sheds light on molecular and cellular decision-making, we believe computational perspectives will be essential to understand how local, distributed rules give rise to robust, global systems.

Acknowledgments

This work was supported in part by the National Institutes of Health award no. F32-MH099784 to S.N.; and by grants from the McDonnell Foundation program on Studying Complex Systems and from the U.S. National Science Foundation award nos. DBI-0965316 and DBI-1356505 to Z.B.-J. 

References

1. Afek, Y. et al. Beeping a maximal independent set. *Distributed Computing; Lecture Notes in Computer Science*. D. Peleg, Ed. (2011). Springer-Verlag, Berlin Heidelberg, 32–50.
2. Afek, Y. et al. A biological solution to a fundamental distributed computing problem. *Science* 331, 6014 (2011), 183–185.
3. Aida, K., Natsume, W. and Futakata, Y. Distributed computing with hierarchical master-worker paradigm for parallel branch and bound algorithm. In *Proceedings of the 31st International Symposium on Cluster Computing and the Grid*. IEEE Computer Society, Washington, DC, 2003.
4. Anastasi, G., Conti, M., Di Francesco, M. and Passarella, A. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Netw.* 7, 3 (May 2009), 537–568.
5. Angluin, D. Local and global properties in networks of processors (extended abstract). In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing*. ACM, New York, NY, 1980, 82–93.
6. Aspnes, J. and Ruppert, E. An introduction to population protocols. *Middleware for Network Eccentric and Mobile Applications*. B. Garbinato, H. Miranda, and L. Rodrigues, Eds. Springer-Verlag, Berlin, Heidelberg, 2009, 97–120.
7. Attiya, H., Bar-Noy, A. and Dolev, D. Sharing memory robustly in message-passing systems. *JACM* 42, 1 (Jan. 1995), 124–142.
8. Babaoglu, O., Binci, T., Jelasity, M. and Montresor, A. Firefly-inspired heartbeat synchronization in overlay networks. In *Proceeding of the First International Conference Self-Adaptive and Self-Organizing Systems* (2007), 77–86.
9. Beauquier, J., Blanchard, P., Burman, J. and Delaët, S. Tight complexity analysis of population protocols with cover times—the Zebra-net example. *Theor. Comput. Sci.* 512 (Nov. 2013), 15–27.
10. Bryant, B. Chromatin computation. *PLoS ONE* 7, 5 (2012), e35703.
11. Buesing, L., Bill, J., Nessler, B. and Maass, W. Neural

dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons. *PLoS Comput. Biol.* 7, 11 (Nov. 2011), e1002211.

12. Bushman, F. et al. Host cell factors in HIV replication: meta-analysis of genome-wide studies. *PLoS Pathog.* 5, 5 (May 2009), e1000437.
13. Cardelli, L. and Csikasz-Nagy, A. The cell cycle switch computes approximate majority. *Sci Rep.* 2:656 (2012).
14. Chazelle, B. Natural algorithms and influence systems. *Commun. ACM* 55, 12 (Dec. 2012), 101–110.
15. Cheong, R., Rhee, A., Wang, C.J., Nemenman, I. and Levchenko, A. Information transduction capacity of noisy biochemical signaling networks. *Science* 334, 6054 (Oct. 2011), 354–358.
16. Chittka, L., Skorupski, P., and Raine, N.E. Speed-accuracy tradeoffs in animal decision-making. *Trends Ecol. Evol.* 24, 7 (July 2009), 400–407.
17. Cornejo, A. and Kuhn, F. Deploying wireless networks with beeps. In *Proceedings of the 24th International Conference on Distributed Computing* (2010). Springer-Verlag, Berlin, Heidelberg, 148–162.
18. Cuntz, H., Forstner, F., Borst, A. and Hausser, M. One rule to grow them all: A general theory of neuronal branching and its practical application. *PLoS Comput. Biol.* 6, 8 (2010).
19. Destexhe, A. and Contreras, D. Neuronal computations with stochastic network states. *Science* 314, 5796 (Oct. 2006), 85–90.
20. Emek, Y. and Wattenhofer, R. Stone age distributed computing. In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing* (2013). ACM, New York, NY, 137–146.
21. Falik, O., Mordoch, Y., Quansah, L., Fait, A. and Novoplansky, A. Rumor has it ... : Relay communication of stress cues in plants. *PLoS ONE* 6, 11 (2011), e23625.
22. Feinerman, O. and Korman, A. Theoretical distributed computing meets biology: A review. *Distributed Computing and Internet Technology*. C. Hota and P. Srimani, Eds. *Lecture Notes in Computer Science* 7753 (2013), 1–18. Springer-Verlag, Berlin Heidelberg, 1–18.
23. Feinerman, O., Korman, A., Lotker, Z. and Sereni, J.-S. Collaborative search on the plane without communication. In *Proceedings of the ACM Symposium on Principles of Distributed Computing* (2012). ACM, New York, NY, 77–86.
24. Gitter, A. et al. Backup in gene regulatory networks explains differences between binding and knockout results. *Mol. Syst. Biol.* 5, 276 (2009).
25. Gupta, I., Chandra, T.D., and Goldszmidt, G.S. On scalable and efficient distributed failure detectors. In *Proceedings of the 20th Annual ACM Symposium on Principles of Distributed Computing* (2001). ACM, New York, NY, 170–179.
26. Hsiao, T.L. and Vitkup, D. Role of duplicate genes in robustness against deleterious human mutations. *PLoS Genet.* 4, 3 (Mar. 2008), e1000014.
27. Hu, T., Genkin, A. and Chklovskii, D.B. A network of spiking neurons for computing sparse representations in an energy-efficient way. *Neural Comput.* 24, 11 (Nov. 2012), 2852–2872.
28. Hu, Z. and Li, B. Fundamental performance limits of wireless sensor networks. *Ad Hoc and Sensor Networks* (2004), 81–101.
29. Ioannou, C.C., Guttal, V. and Couzin, I.D. Predatory fish select for coordinated collective motion in virtual prey. *Science* 337, 6099 (Sept. 2012), 1212–1215.
30. Jakovcevski, M. and Akbarian, S. Epigenetic mechanisms in neurological disease. *Nat. Med.* 18, 8 (Aug. 2012), 1194–1204.
31. Jongeneel, C.V. et al. An atlas of human gene expression from massively parallel signature sequencing (MPSS). *Genome Res* 15, 7 (July 2005), 1007–1014.
32. Kitano, H. and Oda, K. Robustness trade-offs and host-microbial symbiosis in the immune system. *Mol. Syst. Biol.* 2:2006.0022 (2006).
33. Kuhn, F., Lynch, N., and Oshman, R. Distributed computation in dynamic networks. In *Proceedings of the 42nd ACM Symposium on Theory of Computing* (2010). ACM, New York, NY, 513–522.
34. Levy, S., Kafri, M. Carmi, M. and Barkai, N. The competitive advantage of a dual-transporter system. *Science* 334, 6061 (Dec. 2011), 1408–1412.
35. Li, K., Thomas, K., Torres, C., Rossi, L. and Shen, C.-C. Slime mold inspired path formation protocol for wireless sensor networks. In *Proceedings of the 7th International Conference on Swarm Intelligence* (2010). Springer-Verlag, Berlin, Heidelberg, 299–311.
36. Luby, M. A simple parallel algorithm for the maximal independent set problem. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*

(1985). ACM, New York, NY, 1–10.

37. Lynch, N.A. *Distributed Algorithms*. Morgan Kaufmann, San Francisco, CA, 1996.
38. Mehta, P. and Schwab, D.J. Energetic costs of cellular computation. In *Proceedings of the Natl. Acad. Sci.* 109, 44 (Oct. 2012), 17978–17982.
39. Métivier, Y., Robson, J., Saheb-Djahromi, N. and Zemmari, A. An optimal bit complexity randomized distributed mis algorithm. *Distributed Computing* 23, 5-6 (2011), 331–340.
40. Milo, R. et al. Network motifs: Simple building blocks of complex networks. *Science* 298, 5594 (Oct. 2002), 824–827.
41. Mittal, P. and Borisov, N. Information leaks in structured peer-to-peer anonymous communication systems. In *Proceedings of the Conf. on Computer and Communications Security* (2008). ACM, New York, NY, 267–278.
42. Navlakha, S. and Bar-Joseph, Z. Algorithms in nature: The convergence of systems biology and computational thinking. *Mol. Syst. Biol.* 7:546 (2011).
43. Navlakha, S., He, X., Faloutsos, C. and Bar-Joseph, Z. Topological properties of robust biological and computational networks. *J.R.Soc Interface* 11, 96 (2014).
44. Nusser-Stein, S. et al. Cell-cycle regulation of NOTCH signaling during *C. elegans* vulval development. *Mol. Syst. Biol.* 8:618 (2012).
45. Pecevski, D., Buesing, L. and Maass, W. Probabilistic inference in general graphical models through sampling in stochastic networks of spiking neurons. *PLoS Comput. Biol.* 7, 12 (Dec. 2011), e1002294.
46. Prabhakar, B., Dektar, K.N., and Gordon, D.M. The regulation of ant colony foraging activity without spatial information. *PLoS Comput. Biol.* 8, 8 (2012), e1002670.
47. Price, M.N. et al. Indirect and suboptimal control of gene expression is widespread in bacteria. *Mol. Syst. Biol.* 9:600, (2013).
48. Reid, C.R., Latty, T., Dussutour, A., and Beekman, M. Slime mold uses an externalized spatial "memory" to navigate in complex environments. In *Proceedings of the Natl. Acad. Sci.* 109, 43 (Oct. 2012), 17490–17494.
49. Scott, A., Jeavons, P., and Xu, L. Feedback from nature: An optimal distributed algorithm for maximal independent set selection. In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing*. ACM, New York, NY, 147–156.
50. Shklarsh, A., Ariel, G., Schneidman, E. and Ben-Jacob, E. Smart swarms of bacteria-inspired agents with performance adaptable interactions. *PLoS Comput. Biol.* 7, 9 (Sept. 2011), e1002177.
51. Tang, J., Xue, G., Chandler, C., and Zhang, W. Interference-aware routing in multihop wireless networks using directional antennas. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies* (2005). IEEE, 751–760.
52. Tero, A. et al. Rules for biologically inspired adaptive network design. *Science* 327, 5964 (2010), 439–442.
53. Tyson, J.J., Chen, K.C., and Novak, B. Sniffers, buzzers, toggles and blinkers: dynamics of regulatory and signaling pathways in the cell. *Curr. Opin. Cell Biol.* 15, 2 (Apr. 2003), 221–231.
54. Wakefield, E.D. et al. Space partitioning without territoriality in gannets. *Science* 341, 6141 (July 2013), 68–70.
55. Yu, H. and Gerstein, M. Genomic analysis of the hierarchical structure of regulatory networks. In *Proceedings of the Natl. Acad. Sci.* 103, 40 (Oct. 2006), 14724–14731.
56. Yu, H., Kim, P.M., Sprecher, E., Trifonov, V. and Gerstein, M. The importance of bottlenecks in protein networks: Correlation with gene essentiality and expression dynamics. *PLoS Comput. Biol.* 3, 4 (Apr. 2007), e59.
57. Zhong, Q. et al. Edgetic perturbation models of human inherited disorders. *Mol. Syst. Biol.* 5, 321 (2009).

Saket Navlakha (navlakha@salk.edu) is an assistant professor in the Center for Integrative Biology at the Salk Institute for Biological Studies, La Jolla, CA. This work occurred when he was a postdoc at Carnegie Mellon University, Pittsburgh, PA.

Ziv Bar-Joseph (zivbj@cs.cmu.edu) is an associate professor in the Machine Learning Department & Lane Center for Computational Biology, School of Computer Science, at Carnegie Mellon University, Pittsburgh, PA.

Copyright held by owners/authors. Publication rights licensed to ACM. \$15.00.