Anthós

Volume 3 | Issue 1 Article 3

6-2011

Embedding Parallel Computation in a Stochastic Mesh Network: A Morphogenetic Approach

Max Orhai
Portland State University

Recommended Citation
Orhai, Max (2011) "Embedding Parallel Computation in a Stochastic Mesh Network: A Morphogenetic Approach," *Anthós*: Vol. 3: Iss. 1, Article 3.
https://doi.org/10.15760/anthos.2011.22

embedding parallel computation in a stochastic mesh network:

a morphogenetic approach

Max OrHai
Spring 2011
simulation source code available:
http://cs.pdx.edu/orhai/mesh-sort

teuscher.:Lab
Emerging Computing Models and Technologies

Portland State UNIVERSITY

Maseeh College of Engineering and Computer Science
Undergraduate Research and Mentoring Program

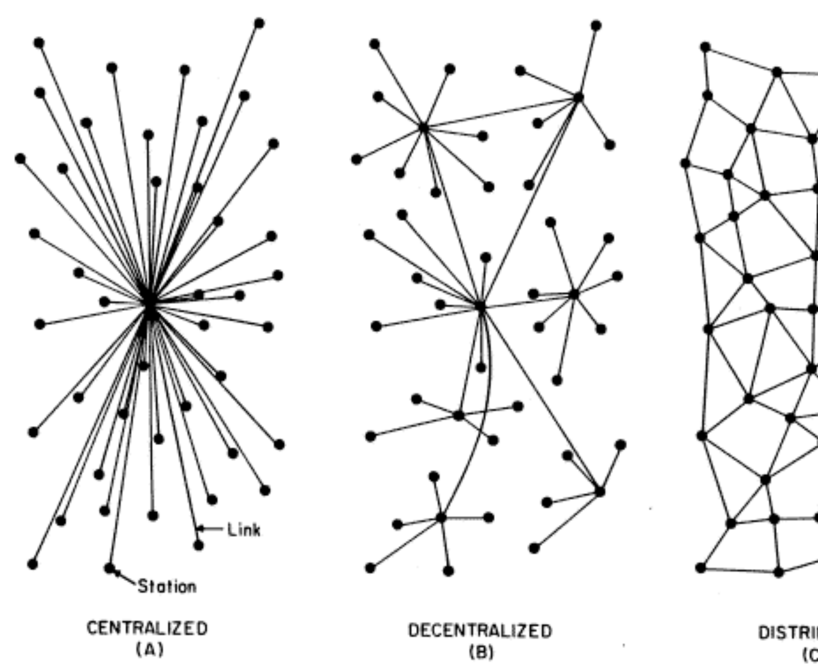


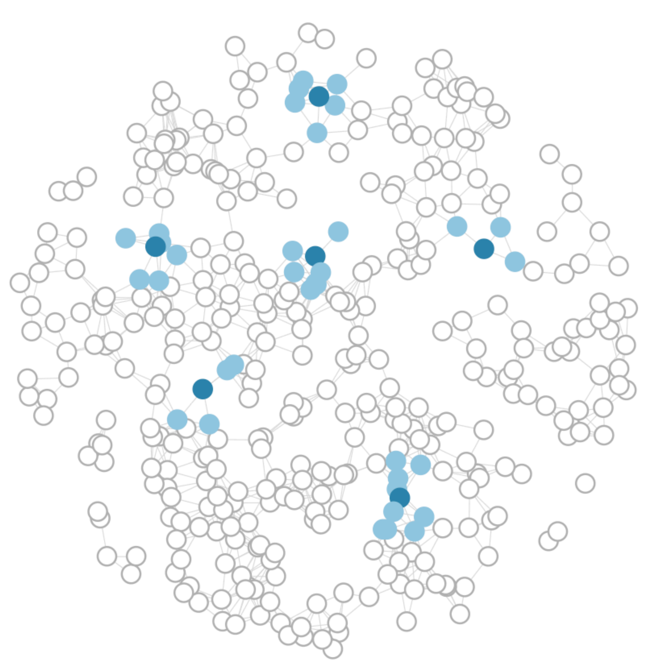
FIG. 1 — Centralized, Decentralized and Distributed Networks

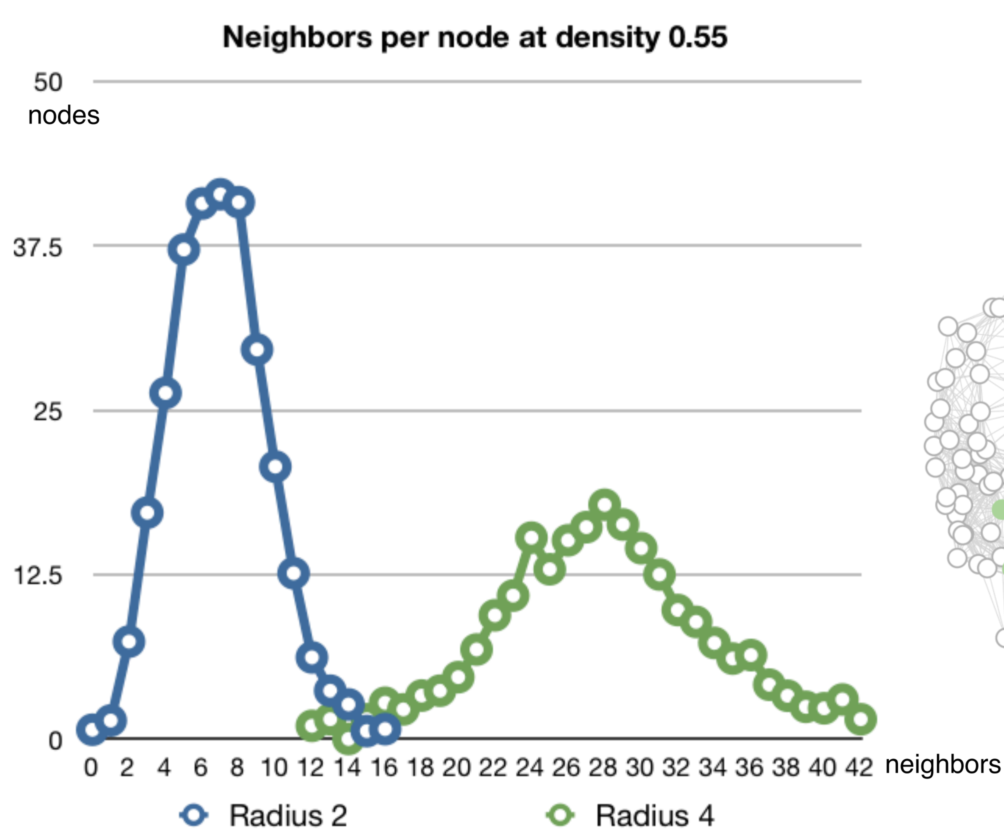
Many basic techniques in computer science have been founded on the assumption that physical computing resources are scarce but orderly, and that the cost of effective direct communication between physically distant parts of a computer system is affordable. In **ubiquitous computing** systems such as sensor networks, or in the design of **nano-scale systems**, these familiar assumptions may not hold.

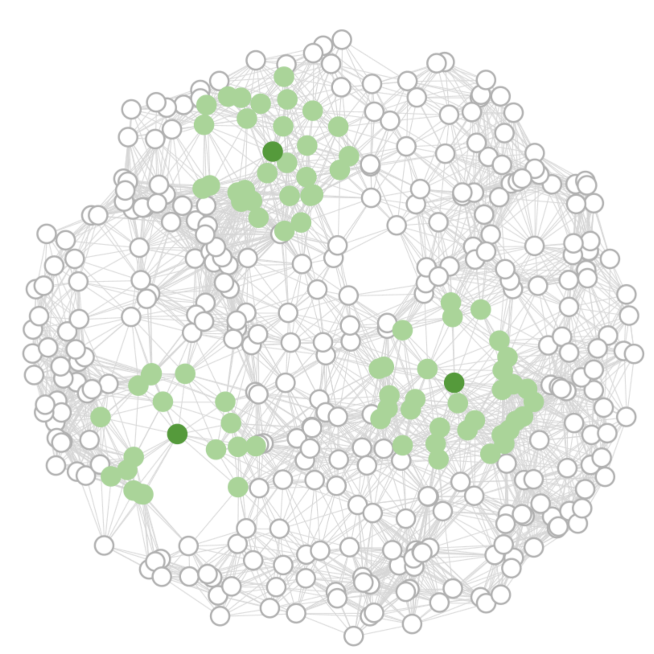
What if we suppose instead that **computing capacity is plentiful**, but that only local communication is possible, and the exact structure of the communication network is not known in advance? This is the domain of spatial programming.

How can we program a locally connected network of randomly placed computing nodes to do a **practical computing task**, while taking advantage of the inherent **parallel processing** capacity of the network?

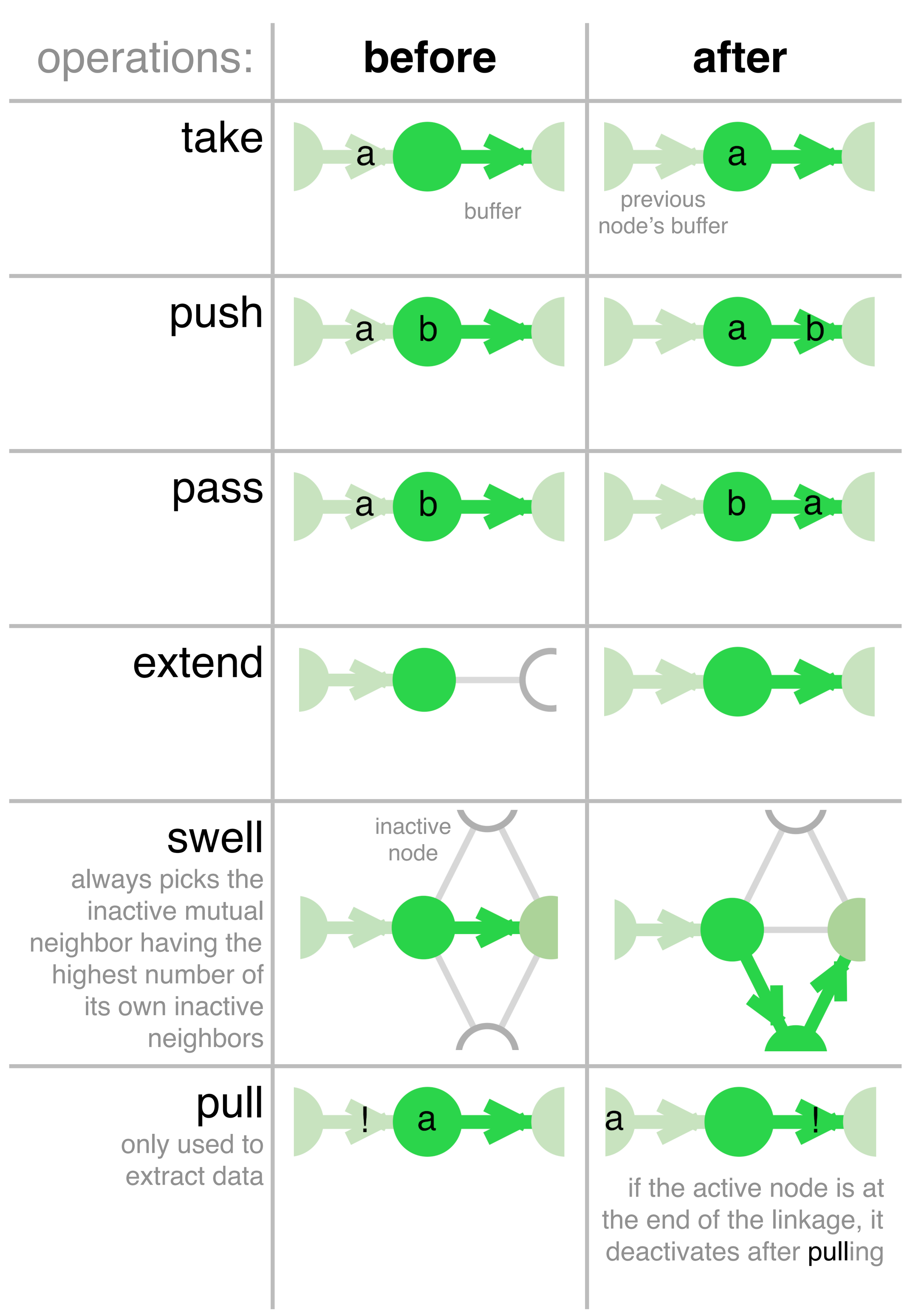
We believe that the **organization and dynamics of biological processes** may offer possibilities for the design of both hardware and software under these new conditions. This algorithm, a variation on **insertion sort** that is also a simplified abstract model of **morphogenetic cell sorting** in the development of multicellular organisms, explores how we might compute in this novel environment.







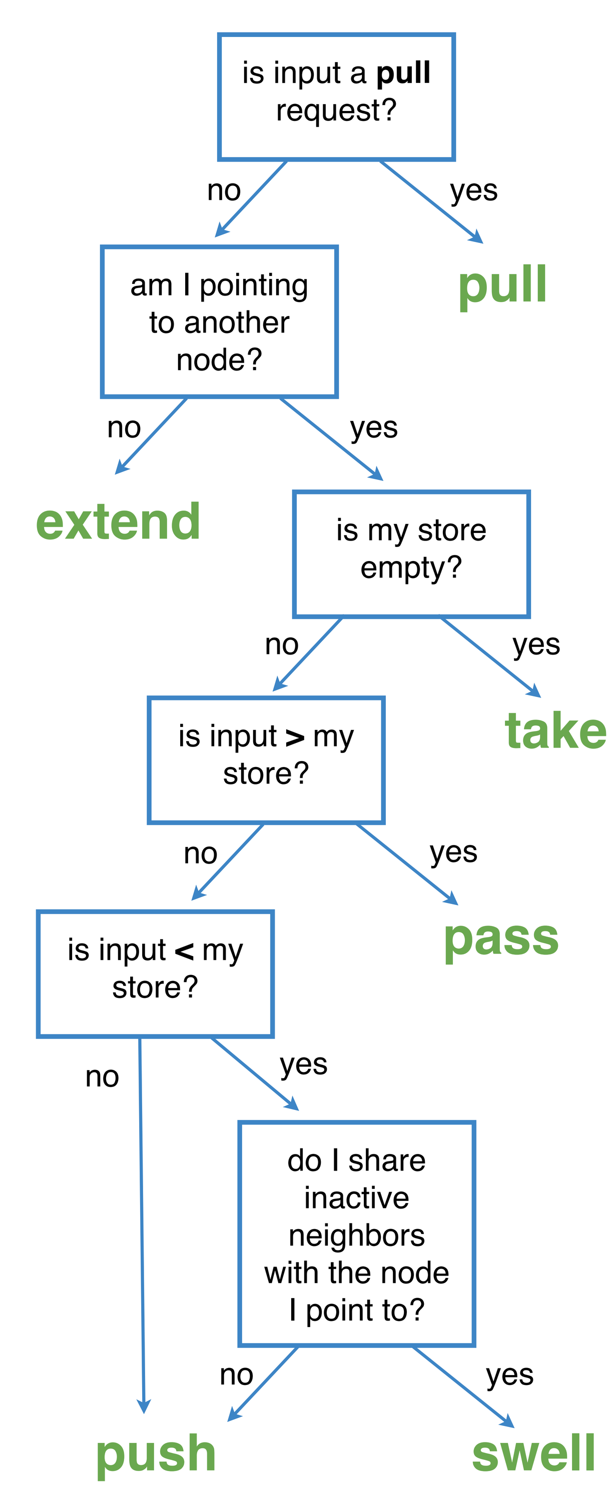
Each node has a **spatial neighborhood** which determines which other nodes it may communicate with. The size of these neighborhoods determines the amount of connectivity in the network. Nodes are **tiny computers** all running the same simple program, with just enough memory to hold two numbers.



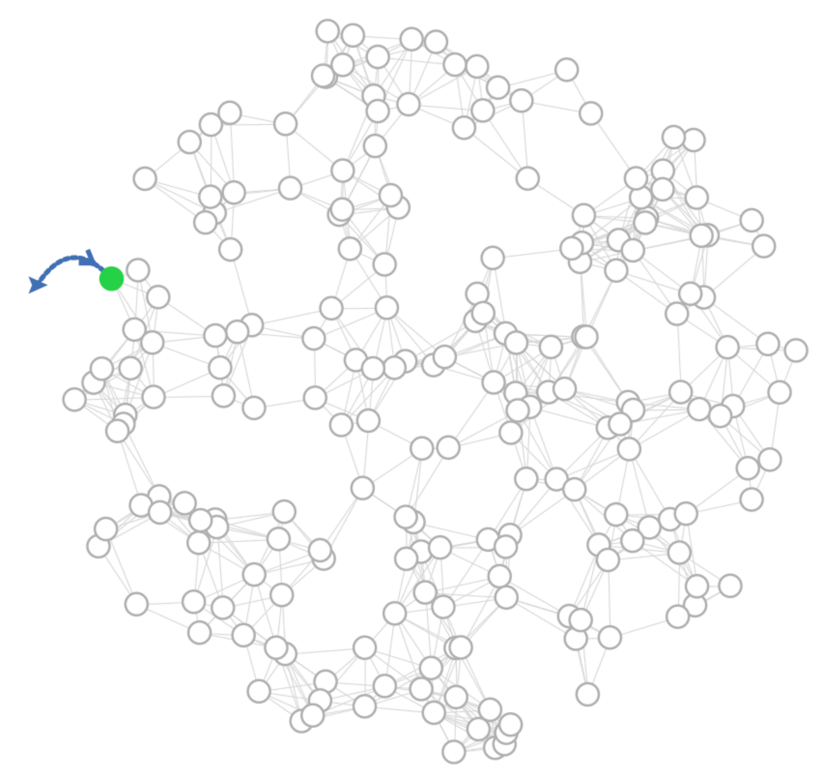
decision tree:

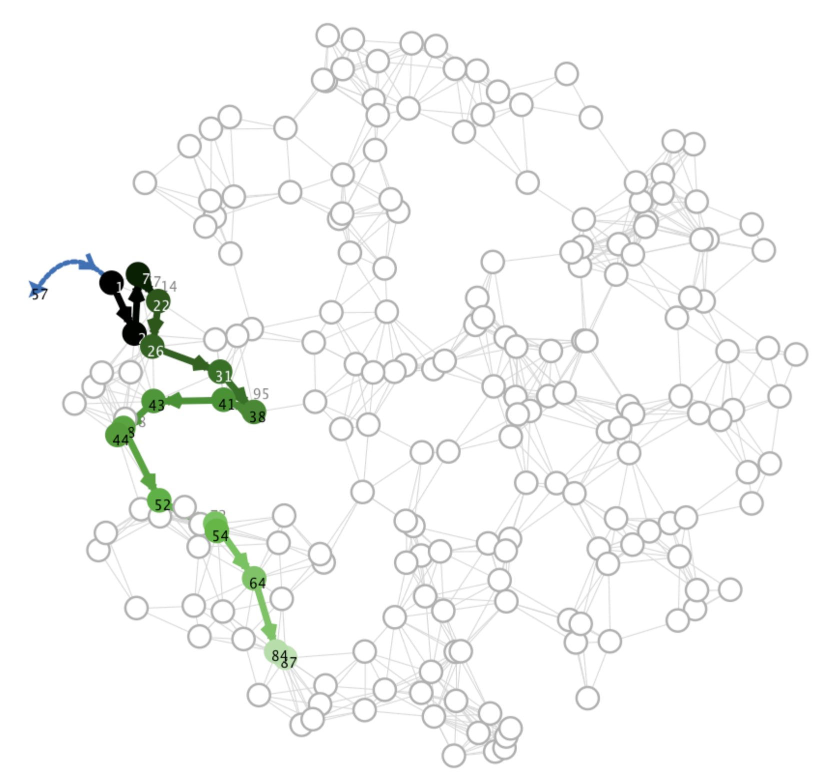
This procedure is run by an active node only when its buffer is empty and an input (**a**) is received from the previous node in the linear linkage. An active node with a full buffer informs the neighboring node to which it points of the contents of its buffer, which will constitute the input for that node.

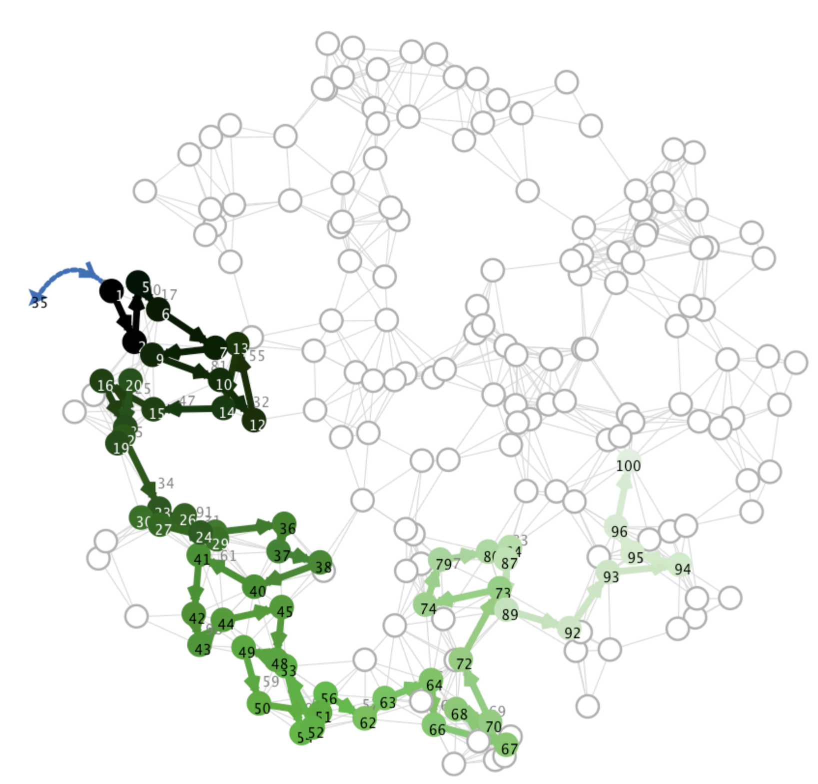
The linkage of active nodes which grows through the network is an **asynchronous reactive system** that depends on a stream of external input consisting of either numbers or pull requests. Without such an input stream, the system will eventually pause in a metastable state where all numbers are sorted and stored in a linkage of active nodes with empty buffers, unless the linkage growth process gets stuck.

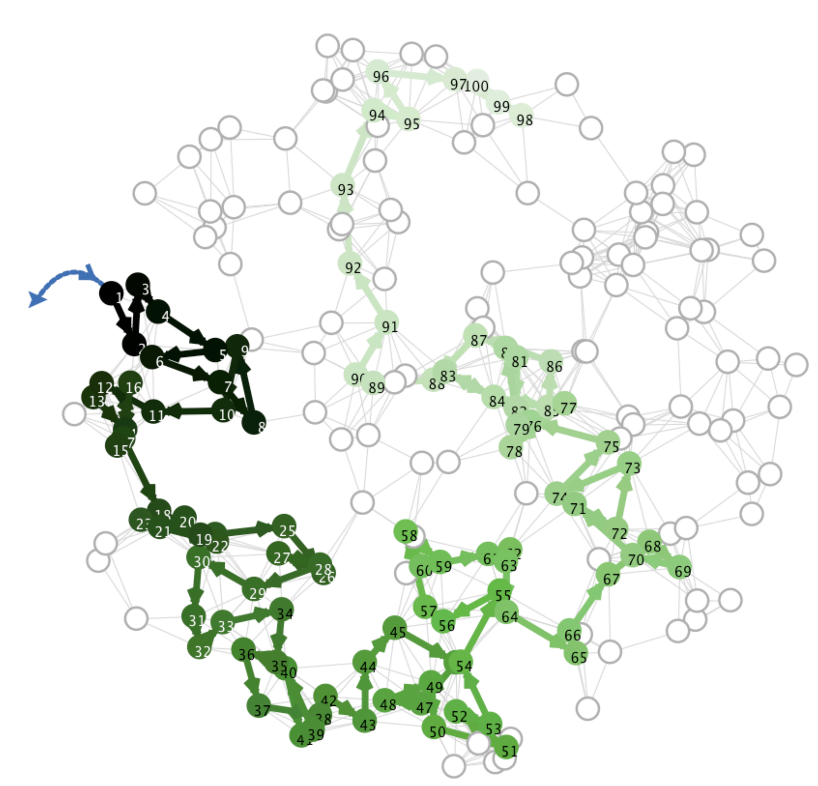


The sequence of images below shows the algorithm in action. As the integers from 1 to 100 in random order are injected into the system at an arbitrarily chosen node, a **dynamic linear linkage** active data structure is grown which sorts the numbers in parallel as a length-conserving and dead-end-avoiding path is found.







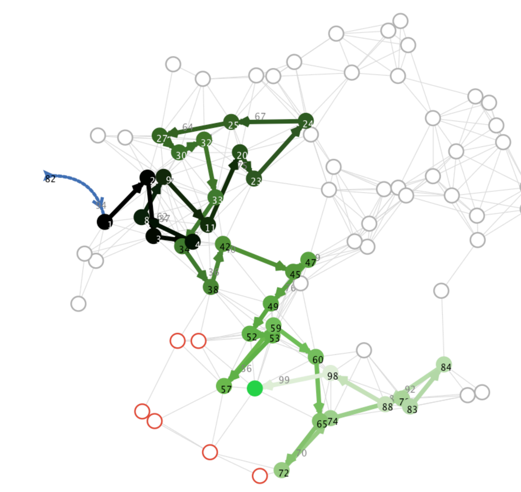


(Lower numbers are shown in darker green.)

When the sort is completed, data can be pulled back out of the system through the linkage.

What can go wrong?

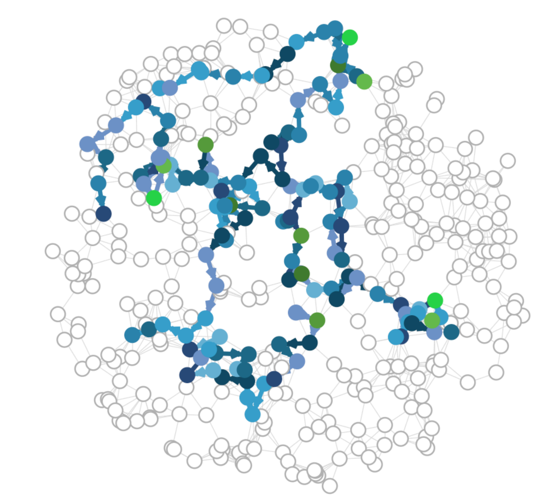
In too-sparse or locally over-crowded networks it is possible for the linkage to grow into a cul-de-sac. Then the buffers of the nodes in the linkage fill up with partially-sorted numbers, which may later be released with the addition of new nodes to the network.



To avoid this problem, the network size should be larger than the input data, with sufficient connectivity.

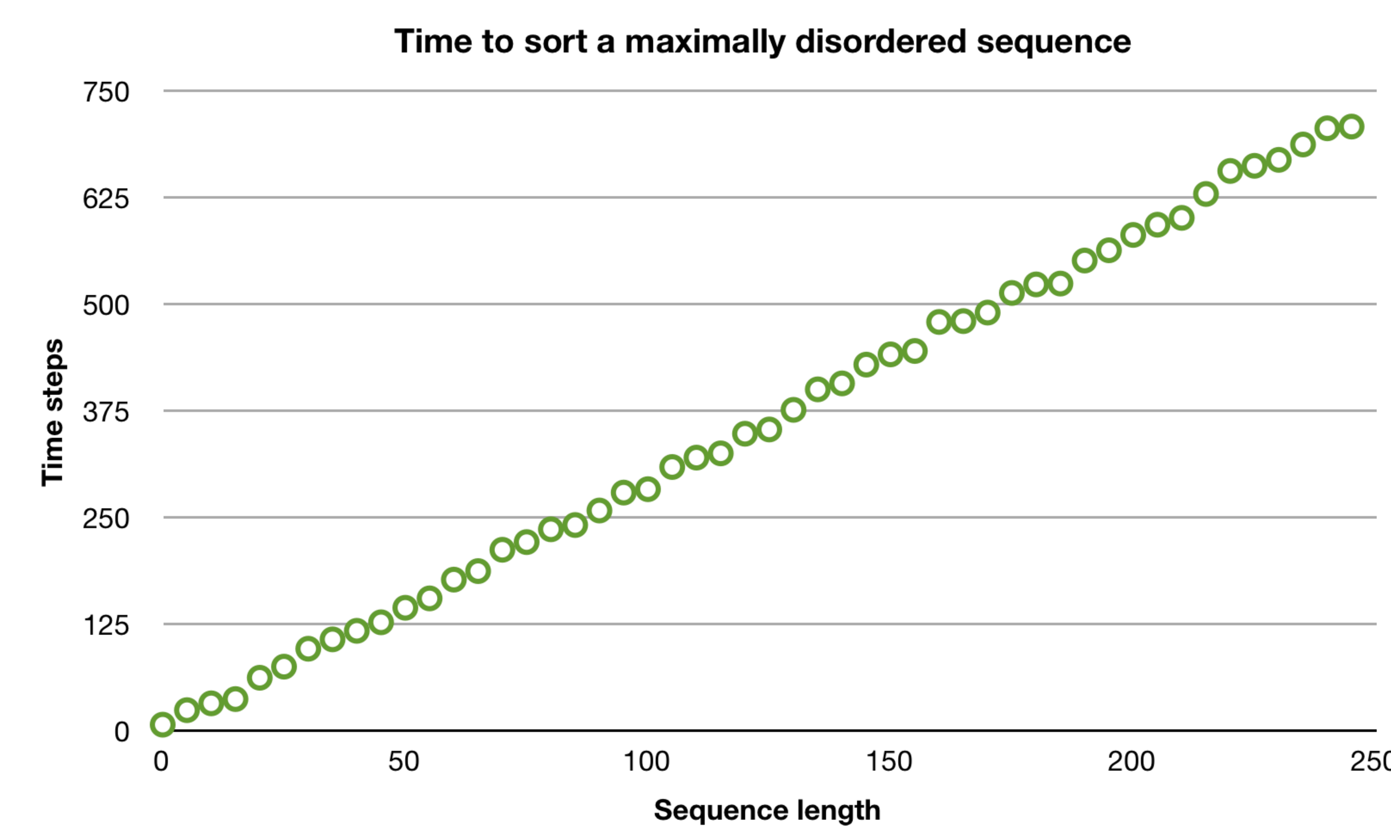
Potential future work:

What other kinds of programs can be realized as parallel dataflow graphs in physically realistic space?

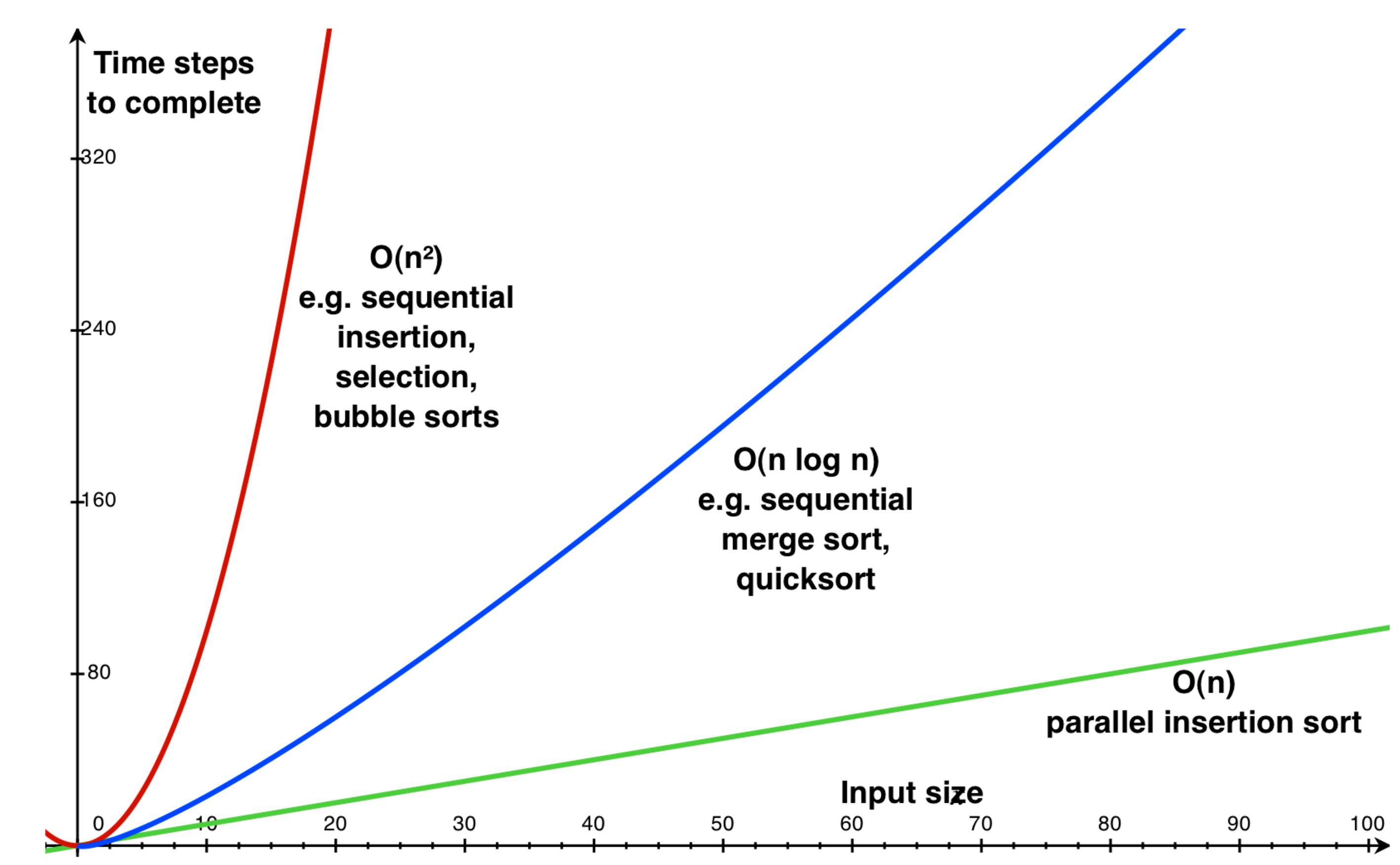


How can these structures be made to grow, adapt, and heal themselves if disrupted?

How might the system be affected by the unforeseen constraints of an implementation technology?



The chart above shows the **performance** of the parallel insertion sort. In a sequential computer, this algorithm takes time proportional to the square of the input size. By **distributing the work spatially**, we are able to complete the task in **linear time**!



References and related work:

Jacob Beal and Gerald Sussman. **Biologically-Inspired Robust Spatial Programming.** MIT Computer Science and Artificial Intelligence Laboratory Memo 2005-001.

Neil Gershenfeld, David Dalrymple, Kailiang Chen, Ara Knaian, Forrest Green, Erik D. Demaine, Scott Greenwald, and Peter Schmidt-Nielsen. **Reconfigurable Asynchronous Logic Automata.** Proceedings of the ACM Conference on Principles of Programming Languages, 2010.

Robert Rosen. **Morphogenesis in Networks.** Chapter in Essays on Life Itself, Columbia University Press, 2000.