embedding parallel computation in a stochastic mesh network: a morphogenetic approach

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 abstraction of morphogenetic cell sorting in the development of multicellular organisms, explores how we might compute in this novel environment.

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.

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.

Operations: take, push, pass, extend, swell, pull. Each node is a decision tree:

- **take**
  - If the active node is at the end of the linkage, it deactivates after pulling.
- **push**
  - Only used to extract data.
- **pass**
  - Always picks the inactive mutual neighbor having the highest number of its own inactive neighbors.
- **extend**
  - 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.
- **swell**
  - Always picks the inactive mutual neighbor having the highest number of its own inactive neighbors.
- **pull**
  - If the active node is at the end of the linkage, it deactivates after pulling.

**References and related work:**


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!