Core-periphery networks

A novel network architecture for parallel and distributed computing, inspired by social networks and complex systems, proposed by Avin, Barkovitch, Lotker, and Peleg [2]. A core-periphery network \( C = (V, E) \) has its node set partitioned into a core \( C \) and a periphery \( P \), and satisfies the following axioms:

- **Core boundary**
- **Clique emulation**
- **Periphery-core convergecast**

Clique emulation

The core can emulate the clique in a constant number of rounds in the CONGEST model. That is, there is a communication protocol running in a constant number of rounds in the CONGEST model such that, assuming that each node \( v \in C \) has a message \( M_v \), on \( O(\log n) \) bits for every \( v \in C \), then, after \( O(1) \) rounds, every \( v \in C \) has received all messages \( M_v \) for all \( v \in C \).

Periphery-core convergecast

There is a communication protocol running in a constant number of rounds in the CONGEST model such that, assuming that each node \( v \in P \) has a message \( M_v \), on \( O(\log n) \) bits for every \( v \in P \), at least one node in the core has received \( M_v \).

Using \( 2 \) rounds to emulate the clique

Consider the Johnson graph \( J(n, 3) \), where \( n = |V| \). There exists an independent set of size \( \frac{n}{3} \) in \( J(n, 3) \). It can be determined by finding \( k \) maximizing \( |I_k| \), where \( I_k = \{(i, j, z) \in V(J(n, 3))| i + j + z = k \mod n\} \). For each triple in \( I_k \), one arbitrary edge can be removed, removing in total about \( \frac{1}{3} \) of the edges.

Using more rounds to emulate the clique

Consider a bipartite graph with \( n \) nodes on one side and \( k \) on the other side, divided in groups of \( m \) nodes. The message of \( a_j \) is routed to \( b_{j'} \) via node \( a_{j'} \) where \( j + j' + k = 0 \) \((\mod a)\) in round \( \frac{k}{2} \).

Tradeoff between edges and rounds

- Let \( n \geq 1 \), and \( k \geq 2 \). There is an \( n \)-node graph with \( \frac{n}{k} \) edges that can emulate the \( n \)-node clique in \( k \) rounds. Also, there is an \( n \)-node graph with \( \frac{k}{4} \) edges that can emulate the \( n \)-node clique in \( 2 \) rounds.
- Let \( n \geq 1 \), \( k \in \{1, \ldots, n - 1\} \), and let \( G \) be an \( n \)-node graph that can emulate the \( n \)-node clique in \( k \) rounds. Then \( G \) has at least \( \frac{n}{k} \) edges.

Other graphs that can emulate the clique

Let \( r \geq 0 \), \( n \geq 1 \), \( a = \sqrt{\left(\frac{3}{2}\right) + \frac{r}{n}} \) where \( r \) is the base of the natural logarithm, and \( p \geq 4 + n^2/n \). For \( G \in \Omega_p \), \( Pr(G) \) can emulate \( K_n \) in \( O(\min\{\frac{n}{\log n}, 3\}) \) rounds \( \geq 1 - \Omega\left(\frac{1}{n^2}\right) \).

Finding a routing schema for \( G \in \Omega_p \):

1. Process each sender sequentially
2. Consider the sender \( i \), the receiver \( j \) and the set of paths \( \{(i, k_j) | (i, k_j) \in E \} \)
3. Create \( r \) sets of \( d \) paths, chosen uniformly at random
4. For each set, choose the path \( (i, k_j) \) where the load of \( (i, k_j) \) is minimum
5. Among the chosen paths, choose the path \( (i, k_j) \) where the load of \( (i, k_j) \) is minimum
6. Increase the load of \( (i, k_j) \) and \( (i, k_j) \)

Ideally, analyze separately senders and receivers assuming that the choices of the other side are adversarial, using \( d = \ln_2 n, r = (c + 3) n^3 \ln n \) for \( c = - \ln(3/2 - 1/n) \) and techniques similar to [4].

Minimum Spanning Tree

Algorithm:

1. Each node sends towards the core its minimum weight outgoing edge.
2. By Axiom 1 each node \( v \in C \) received \( O(\log n) \) edges.
3. Each node \( v \in C \) keeps only one edge for each fragment, the lightest.
4. Group edges by their starting fragment (there are \( O(\sqrt{n}) \) edges per group).
5. Keep only the lightest edge of each group.
6. The remaining edges form components composed by a tree and a \( 2 \)-cycle, every node of the tree should know the id of the root (the node in the 2-cycle with smallest id), that will be the id of the new fragment.
7. Group edges by their ending fragments.
8. Do pointer jumping, each node \( v \in C \) has to send and receive \( O(\sqrt{n}) \) messages.
9. Repeat \( \lceil \log_3 n \rceil \) times.

In order to find the root of a tree it could be necessary to perform \( O(\log n) \) steps of pointer jumping, giving a \( O(\log n) \) algorithm. This process can be amortized on multiple phases, i.e. performing a constant number of pointer jumps at each phase. One step of pointer jumping can be executed in \( O(1) \) using Lenzen routing protocol [3]. The grouping parts can be performed in \( O(1) \) rounds using Lenzen sorting protocol. The result is a deterministic \( O(\log n) \) rounds algorithm.

References