Parameterized Complexity of Streaming Diameter and Connectivity Problems

Jelle J. Oostveen Erik Jan van Leeuwen

Dept. Information and Computing Sciences, Utrecht University, The Netherlands.
{j.j.oostveen | e.j.vanleeuwen}@uu.nl

A full version of this work is on the arXiv. An extended abstract is under submission.
The first author is partially supported by the NWO grant OCENW.KLEIN.114 (PACAN).

Introduction. In the analysis of a network, its diameter and whether or not it is connected are important measures. Algorithms to compute the diameter or decide connectivity of a network often rely on keeping the entire network in (random access) memory. However, very large networks might not fit in memory. Graph streaming is a paradigm where the network is inspected through a so-called stream, in which its edges appear one by one, and only limited memory is available [Henzinger et al. '98]. To compensate for the limited memory, multiple passes may be made over the stream and computation time is unlimited. The question is to determine the computational complexity of (graph) problems in this model, taking into account trade-offs between the amount of memory and the number of passes.

An ideal would be to use $O(\log n)$ bits of memory (e.g., a constant number of pointers or counters) in a single pass on n-vertex graphs. Unfortunately, any p-pass algorithm for Connectivity needs $\Omega(n/p)$ bits of memory, unconditionally [Henzinger et al. '98]. Single pass algorithms for Connectivity or Diameter need $\Omega(n \log n)$ bits on sparse graphs [Sun Woodruff '15]. Naive streaming algorithms for Connectivity or Diameter store the entire graph, using $O(m \log n)$ bits and a single pass. For Connectivity this can be improved to a 1-pass, $O(n \log n)$ bits algorithm [McGregor '14]. This is far from ideal.

We ask what structure a graph must have in order to improve on these naive algorithms and to circumvent the complexity barriers formed by the lower bounds. We use parameterized complexity to address this question.

Results. As our main result, we show that if a graph has *vertex cover number* k (i.e., it can be made edgeless by deleting k vertices), then Diameter and Connectivity can be solved using $O(2^k k)$ passes, $O(k \log n)$ bits of memory or one pass, $O(2^k + k \log n)$ bits of memory. This algorithm assumes the stream is given in the *AL model*: edges incident on the same vertex appear consecutively in the stream and all edges appear twice in a single pass (once for each endpoint). Underlying these algorithms is a method to execute a breadth-first search in $O(k)$ passes and $O(k \log n)$ bits of memory. Note that for constant values of k, we achieve the desired $O(\log n)$ bits of memory.

We also show (unconditional) hardness results that highlight two important aspects of our algorithm. First, the assumption of the AL model is necessary. In the slightly weaker VA model, where edges incident on the same vertex still appear consecutively but only once (for the second endpoint), we observe that any p-pass algorithm needs $\Omega(n/p)$ bits of memory even when the vertex cover number is 3 or 2 for Diameter and Connectivity respectively.

Second, we need our choice of parameter. For any graph H, an *H-free modulator* of a graph G is a set X of vertices such that $G - X$ does not have any induced subgraph isomorphic to H. For $H = P_2$ (P_2 is the path on two vertices), the minimum size of an H-free modulator is the aforementioned vertex cover number, for which we achieve our positive result. For almost any other choice of H, we prove that any p-pass streaming algorithm for Diameter or Connectivity needs $\Omega(n/p)$ bits in the AL model even when a modulator of a fixed, constant size is known. For example, for Diameter, we show such hardness if the graph is a tree or if the graph is two vertices away from being a path.

For some cases, we can also show one-pass, $\Omega(n \log n)$ bits of memory lower bounds. We also prove a much stronger lower bound for Diameter on bipartite graphs: any p-pass algorithm needs $\Omega(n^2/p)$ bits of memory.