

## Introduction

The path search problem considers a simple model of communication networks as channel graphs: directed acyclic graphs with a single source and sink. We consider each vertex to represent a switching point, and each edge a single communication line. Under a probabilistic model where each edge may independently be free (available for use) or blocked (already in use) with some constant probability, we seek to efficiently search the graph: examine (on average) as few edges as possible before determining if a path of free edge exists from source to sink. We consider the difficulty of searching various graphs under different search models, and examine the computational complexity of calculating the search cost of arbitrary graphs.

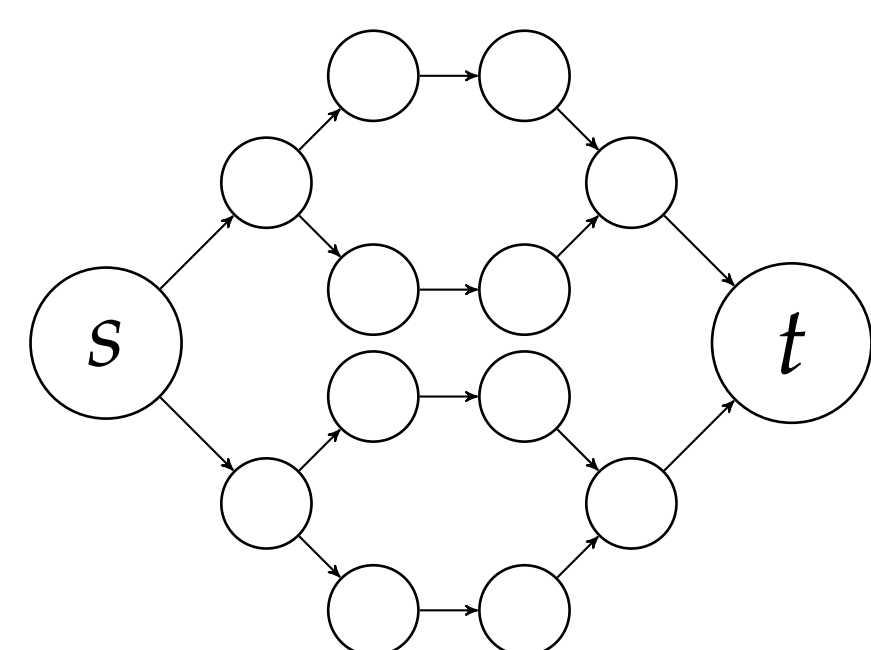


Figure 1: A simple channel graph,  $F_2$ .

In particular, we want to find (for various families of graph) the expected asymptotic number of probes to determine whether the graph is free or blocked. We write this  $E(G, q)$ , where  $q$  is the (independent) probability that any single edge is free, and  $G$  is the graph being searched.

We wish to find infinite families of related graphs whose search complexity grows in various ways with the size of the graph. The  $k$ th graph in our families have  $O(c^k)$  edges, but paths of lengths  $O(k)$ ; we say that a graph family  $G_k$  admits *efficient* search if for any  $0 < q < 1$ ,  $E(G_k, q) \in O(k^d)$ ; in other words, if we expect to probe only a polynomial number of the exponential number of edges in  $G_k$ .

## Local and global search

Our main objects of study are *local* search algorithms. We will see that efficiently searching some graphs requires access to arbitrary edges at arbitrary points in the algorithm. However, we are considering a model of *establishing* communication from  $s$  to  $t$ : how can we

reasonably probe edges adjacent to  $t$  without already having communication?

Thus we consider *local* algorithms, which can only probe edges which are at the end of a fully-probed free path from the source (edges we have demonstrated the ability to communicate with.)

Our main question is simple: Local search is more realistic; is it equally efficient? Does any graph admitting an efficient global search algorithm admit an efficient local search algorithm? We will discover that the answer is *no*; some graphs can be searched globally in linear time, but any algorithm for locally searching them takes exponential time.

## Fully parallel graphs

Most of our analysis revolves around the *fully parallel graphs*, a graph family denoted  $F_k$ . Lin and Pippenger (1996) describe how the fully parallel graphs form a good approximation for various classes of routing networks. They admit a relatively simple analysis (though it is telling that calculating a lower bound on local search is still exceptionally tricky.)

### Definition

We define  $F_k$  recursively:

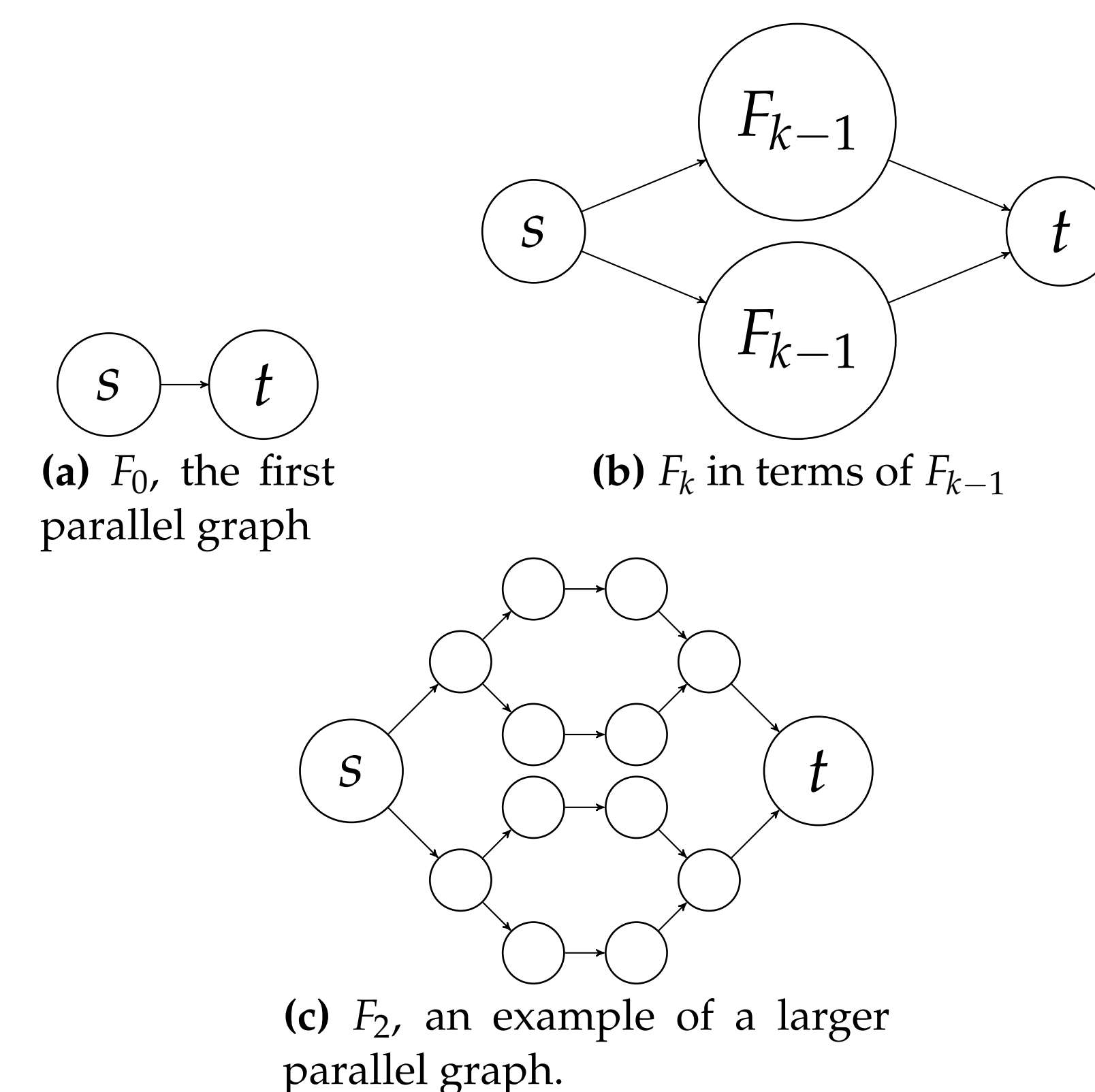


Figure 2: The definition of fully parallel graphs.

## Global search on $F_k$

We know (Lin and Pippenger, 1996) that for all  $q$ , when global search is allowed,

$$E(F_k, q) \in O(k)$$

So globally searching fully parallel graphs is highly efficient. The algorithm is simple: a depth-first search from both ends simultaneously. On average, we expect to examine at most one of the subgraphs  $F_{k-1}$  and a simple analysis shows that the cost is bounded by a radius-1 geometric series.

## Local search on $F_k$

Our main result is the following theorem.

**Theorem 1.** For all  $q > \frac{1}{2}$ , there exist  $c > 1$  such that for any local search algorithm,

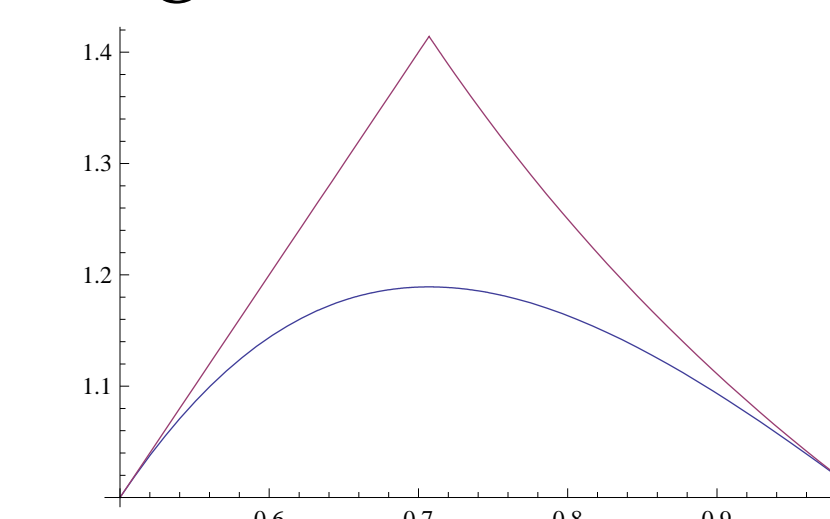
$$E(F_k, q) \in \Omega(c^k)$$

Where in the global case, we are able to probe both ends of the graph—where it is small—and quickly rule out large portions, in the local case we cannot. We need the ability to reach the small end end of the graph, and doing so becomes extremely expensive, in a quantifiable way.

There still exists a significant gap between our lower bound and depth-first search, the best known algorithm for searching  $F_k$  (which we conjecture to be optimal.) DFS, like our bound, is exponential; however, it has a higher base. Our best lower bound on  $c$  is

$$c \geq (2q)^{-\log q},$$

which we plot along with the base of DFS here.



The details of the proof of that lower bound strongly suggest that DFS is in fact optimal, though proving this has so far escaped us.

## Complexity results

While we mainly consider *what* the optimal algorithm for a given graph is, it is interesting to take a viewpoint

rooted in computational complexity, and examine how hard it is to *find* that algorithm for arbitrary graphs.

**Theorem 2.** Calculating the cost of the optimal (local) algorithm for any graph  $G$  is #P-hard. (#P is a complexity class of counting the solutions to problems in NP, the well-known class of intractable problems. It contains NP as a subclass, and is strongly suspected to be strictly larger.)

Thus, in fact, we cannot—in a complexity-theoretic sense—ignore the cost of *making decisions* as we search a graph.

## Conclusions

We have demonstrated a large gap in the efficiency of locally and globally searching  $F_k$ ; however, there exist (many) other graph families which do *not* suffer from this problem, a simple example being the trivial path graphs  $P_k$ .

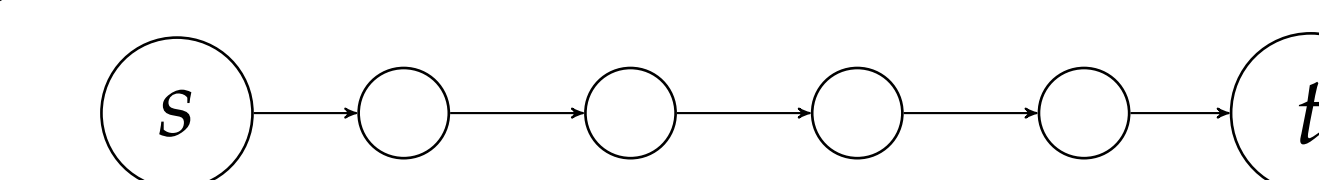


Figure 3: The path graph  $P_5$ .

In short, this fact makes *local* path search interesting: a logical, arguably more physically realistic restriction of the problem often—but not always—results in solutions becoming intractable. Characterizing the reasons why some graphs can be locally searched efficiently and others cannot would be quite useful, especially since calculating explicit lower bounds for a specific family, like most cases of algorithm lower bounds, can be extremely difficult.

## References

Lin, Geng, and Nicholas Pippenger. 1996. Routing algorithms for switching networks with probabilistic traffic. *Networks* 28(1):21–29.