The iFUB Algorithm for Diameter Computation

(Based on Crescenzi et al., TCS 514 (2013) 84–95)

By Aditya

Abstract

Computing the diameter of a large undirected unweighted graph naively requires $O(nm)$ time via all-pairs BFS. Crescenzi et al. introduce the *iFUB* (iterative Fringe Upper Bound) algorithm, which in practice runs in $O(m)$ time on real-world graphs by carefully interleaving BFS traversals to refine lower and upper bounds until they meet. This note summarizes the key ideas, algorithmic structure, and root-selection strategies.

1 Problem Statement

Let $G = (V, E)$ be an undirected unweighted graph with $n = |V|$, $m = |E|$. The *distance* $d(u, v)$ is the length of a shortest path between u, v, and the *diameter* is

$$D \;=\; \max_{u,v \in V} d(u, v)\,.$$

A textbook approach runs a BFS from every node in $O(nm)$ time, which is infeasible on million-edge networks.

2 Overview of iFUB

iFUB maintains two values:

$$\text{lb} \;\leq\; D \;\leq\; \text{ub}$$

and iteratively refines them using BFSs from carefully chosen "fringe" vertices:

- Perform a BFS from a *root* u to compute its eccentricity $(u) = \max_v d(u, v)$, which yields

$$\text{lb} \leftarrow (u), \quad \text{ub} \leftarrow 2\,(u).$$

- Record the *levels* (fringes) $F_i(u) = \{v \mid d(u, v) = i\}$ for $0 \leq i \leq (u)$.
- *Bottom–up sweep*: for $i = (u), (u) - 1, \ldots$ until ub-lb converges, pick each $v \in F_i(u)$ in turn, run BFS to get (v), then

$$\text{lb} \leftarrow \max(\text{lb},\ (v)), \quad \text{ub} \leftarrow \min(\text{ub},\ 2\,(v)).$$

  Stop when $\text{lb} = \text{ub}$ (exact diameter) or when $\text{ub} - \text{lb} \leq k$ for a tolerance k.

Pseudocode for exact diameter ($k = 0$):

Algorithm 1 iFUB(G, u) for exact diameter

BFS(G, u) compute levels $F_i(u)$ and (u)
lb $\leftarrow (u)$; ub $\leftarrow 2\,(u)$
for $i = (u)$ **to** 1 **by** -1 **do**
  for each $v \in F_i(u)$ **do**
    BFS(G, v) to compute (v)
    lb $\leftarrow$ max(lb, (v))
    ub $\leftarrow$ min(ub, $2\,(v)$)
    if lb = ub **then**
      return lb
    end if
  end for
end for
return lb

3 Root-Selection Strategies

The choice of the initial root u greatly affects practical performance. Crescenzi et al. evaluate:

1. **Random**: pick u uniformly at random.
2. **Highest-degree**: choose a node of maximum degree.
3. **4-Sweep-rand**: run "4-Sweep" heuristic starting from a random node:
   - BFS from $r_1 \to a_1 \to b_1$, set r_2 to the midpoint of a_1–b_1;
   - BFS from $r_2 \to a_2 \to b_2$, set u as midpoint of a_2–b_2.
4. **4-Sweep-hd**: same as above but start 4-Sweep from the highest-degree node.

Each 4-Sweep uses exactly four BFSes to approximate a "center" of G, yielding a root with small eccentricity and small fringe sizes.

4 Theoretical Analysis

- **Worst-case time**: still $O(nm)$, since in the worst case almost all vertices may be visited as fringe roots.
- **Amortized bound**: if u has eccentricity R, then iFUB performs at most $N_{\geq R/2}(u)$ BFSes, where $N_{\geq h}(u)$ is the number of nodes at distance $\geq h$ from u.
- **Termination correctness**: the "bottom-up" sweep is justified by the observation that once a fringe vertex v yields $(v) \leq 2(i-1)$, no vertex at distance $< i$ can exceed that bound.

5 Negative Examples

The authors exhibit graph families where:

- *4-Sweep* can fail to find a tight lower bound (approximation ratio ≈ 2).
- *iFUB* degenerates to n BFSes (e.g. odd cycles), achieving $\Theta(nm)$ time.

These pathological cases rely on highly regular structures not found in most real-world networks.

6 Conclusion

iFUB combines simple BFS routines with a clever fringe-based bound-refinement to compute exact diameters. With appropriate root-selection (notably 4-Sweep variants), it performs only a handful of BFSes in practice, achieving near-linear $O(m)$ behavior on complex networks.