The BFS Kernel: Applications and Implementations

Peter M. Kogge

Graph Exploration

- Common graph problem: "explore" region around some vertex
- Exploration: follow edges to see what's reachable
- Possible outputs:
  - Identification of reachable vertices
  - "labelling" of vertices
  - Properties of reachable sub-graph
- Options:
  - Constraints on "how far"
  - Constraints on "which edges"

Major Variants of Exploration

- Depth-First: Keep jumping from vertex to vertex until stopping
  - And then back up to last vertex and see if any untried edges
- Breadth-First: Explore in waves
  - Explore all edges from current "Frontier"
  - Mark as all new vertices as "New Frontier"
  - Start over with new frontier when all current one is searched
- This kernel: Breadth-First

Example: Airline Routes

- Consider graph with
  - Vertices: airports (~17,000 in world):
    - Properties: Country, International designation, Control tower, etc
  - Edges as flights between airports (100,000/day):
    - Properties: Airline (5,000 different airlines in world), Flight number, equipment, …
    - Edges are directional
  - Note graph changes dynamically
- Possible explorations
  - What airports are reachable from some specific one
  - What if we constraint # of stops or airlines,
  - …

Example: Six Degrees of Kevin Bacon

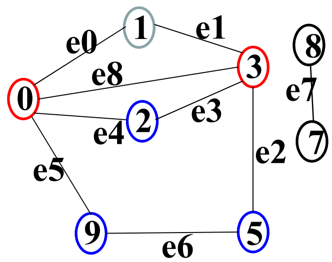
- IMDb Data base
  - Vertices: Multiple "classes"
    - 8.7M+ people
    - 4.8M+ titles of 10 types
  - Edges: (u,v) between people and titles
    - Person u has had one of 34 roles in title v
    - Again directional
- Possible exploration:
  - Can a chain of (u_1,t_1), (u_2,t_1), (u_2,t_2), (u_3,t_2), ... connect any one person u_1 to all other people in database?
  - Kevin Bacon: 6 titles away from everyone else
- See https://oracleofbacon.org/

Other Interesting Applications

- From: https://www.geeksforgeeks.org/applications-of-breadth-first-traversal/
  - Search for neighbors in peer-peer networks
  - Search engine web crawlers
  - Social networks – distance k friends
  - GPS navigation to find "neighboring" locations
  - Patterns for "broadcasting" in networks
- From Wikipedia: https://en.wikipedia.org/wiki/Breadth-first_search
  - Community Detection
  - Maze running
  - Routing of wires in circuits
  - Finding Connected components
  - Copying garbage collection, Cheney's algorithm
  - Shortest path between two nodes u and v
  - Cuthill–McKee mesh numbering
  - Maximum flow in a flow network
  - Serialization/Deserialization of a binary tree
  - Construction of the failure function of the Aho-Corasick pattern matcher.
  - Testing bipartiteness of a graph

Key Kernel: BFS - Breadth First Search

- Given a huge graph
- Start with a root, find all reachable vertices
- Performance metric: **TEPS**: Traversed Edges/sec



Starting at 1: 1, 0, 3, 2, 9, 5

No Flops – just Memory & Networking

Graph500: www.graph500.org

- Several years of reports on performance of BFS implementations on
  - Different size graphs
  - Different hardware configurations
- Standardized graphs for testing
- Standard approach for measuring
  - Generate a graph of certain size
  - Repeat 64 times
    - Select a root
    - Find "level" of each reachable vertex
    - Record execution time
    - TEPS = graph edges / execution time

Graph500 Graphs

- Kronecker graph generator algorithm
  – D. Chakrabarti, Y. Zhan, and C. Faloutsos, R-MAT: A recursive model for graph mining, SIAM Data Mining 2004
- Recursively sub-divides adjacency matrix into 4 partitions A, B, C, D
- Add edges one at a time, choosing partitions probabilistically
  – A = 57%, B = 19%, C = 19%, D = 5%
- # of generated edges = 16*# vertices
  – Average Vertex Degree is 2X this

Graph Sizes

Level	Scale	Size	Vertices (Billion)	TB	Bytes /Vertex
10	26	Toy	0.1	0.02	281.8048
11	29	Mini	0.5	0.14	281.3952
12	32	Small	4.3	1.1	281.472
13	36	Medium	68.7	17.6	281.4752
14	39	Large	549.8	141	281.475
15	42	Huge	4398.0	1,126	281.475
				Average	281.5162

Scale = log2(# vertices)

Notional Sequential Algorithm

- Top-Down search: Keep a "frontier" of new vertices that have been "touched" but not "explored"
  - Explore them and repeat
- Bottom-up search: look at all "untouched vertices" and see if any of their edges lead to a touched vertex
  - If so, mark as touched, and repeat
- Special considerations
  - Vertices that have huge degrees

Top-Down

Algorithm 1 Top Down BFS:
V is the set of vertices; E a set of edges

```
procedure TopDown-BFS(G,root)
    Touched ← {root}
    Frontier ← {root}
    Labels ← N-vector of a large integer
    Label[root] ← 0
    Level ← 0
    while Frontier not empty do
        Level += 1
        TopDown − Pass(Frontier, Touched, Level)X
    return
procedure TopDownPass(Touched, Level)
    Next ← {}
    for u in Frontier do
        for all edges (u, v) in E do
            if v not in Touched then
                Touched ← Touched ∪ {v}
                Next ← Next ∪ {v}
                Label[v] ← Level
    Frontier ← Next
    return
```

Notional Complexity: O(M)

Bottom Up

Algorithm 2 BottomUp BFS:
V is the set of vertices; E a set of edges

```
procedure BOTTOMUP-BFS(G,ROOT)
    Touched ← {root}
    Labels ← N-vector of a large integer
    Label[root] ← 0
    Level ← 0
    TouchedFlag ← True
    while TouchedFlag do
        Level += 1
        TouchedFlag ← BackwardPass(Touched, Level)
    return
procedure BOTTOMUPPASS(INOUT TOUCHED, LEVEL)
    TouchedFlag ← False
    for v not in Touched do
        for all edges (u, v) in E do
            if u in Touched then
                TouchedFlag ← True
                Touched ← Touched ∪ v
                Label[v] ← Level
    return TouchedFlag
```

Notional Complexity: O(NM)

Key Observation

- Forward direction requires investigation of ***every edge*** leaving a frontier vertex
  - Each edge can be done in parallel
- Backwards direction can stop investigating edges ***as soon as 1 vertex*** in current frontier is found
  - If search edges sequentially, potentially significant work avoidance
- In any case, can still parallelize over vertices in frontier

Edges Explored per Level

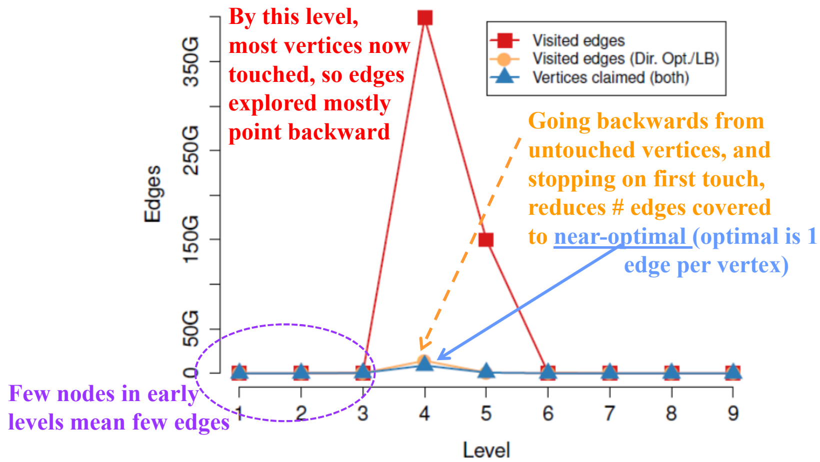


Fig. 5: Graph properties at each exploration level.

Checconi and Petrini, "Traversing Trillions …"

Beamer's Hybrid Algorithm

- Switch between forward & backward steps
  - Use forward iteration as long as In is small
  - Use backward iteration when Vis is large
- Advantage: when
  - # edges from vertices in !Vis
  - are less than # edges from vertices in In
  - then we follow fewer edges overall
- Estimated savings if done optimally: up to 10X reduction in edges
- http://www.scottbeamer.net/pubs/beamer-sc2012.pdf

Hybrid Algorithm

Algorithm 3 Hybrid BFS:
V is the set of vertices; E a set of edges

```
procedure HYBRID-BFS(G,ROOT)
    Touched ← {root}
    Frontier ← {root}
    Labels ← N-vector of a large integer
    Label[root] ← 0
    Level ← 0
    N_f ← 1
    M_f ← outdegree(root)
    M_u ← M − M_f
    while Frontier not empty do
        Level += 1
        Next ← {}
        N_f ← 0
        M_f ← 0
        if (M_f ≥ M_u/α)or(N_f < N/β) then
            for all u in Frontier do
                for all edges (u, v) in E do
                    if v not in Touched then
                        Touched ← Touched ∪ {v}
                        Next ← Next ∪ {v}
                        Label[v] ← Level
                        N_f += 1
                        M_f += outdegree(v)
                        M_u −= indegree(v)
        else
            for all v not in Touched do
                for all edges (u, v) in E do
                    if u in Touched then
                        Next ← Next ∪ v
                        Label[v] ← Level
                        N_f += 1
                        M_f += outdegree(v)
                        M_u −= indegree(v)
            Touched ← Touched ∪ Next
        Frontier ← Next
    return
```

- **N_f = # vertices in current frontier**
- **M_f = # outgoing edges from current frontier**
- **M_u = # incoming edges to current untouched**
- **α = edge reduction factor in bottom-up pass**
- **β = vertex reduction factor when going from bottom-up to top-down**

Switch from top-down to bottom-up when:

$$M_f < M_u/\alpha$$

Switch back from bottom-up to top-down when

$$N_f < N/\beta$$