IMPROVEMENT OF K-TRUSS DECOMPOSITION ALGORITHM

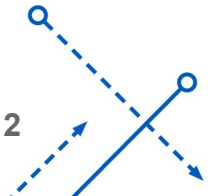
Name: Xingyu Yan

Director: A. Erdem Sariyuce

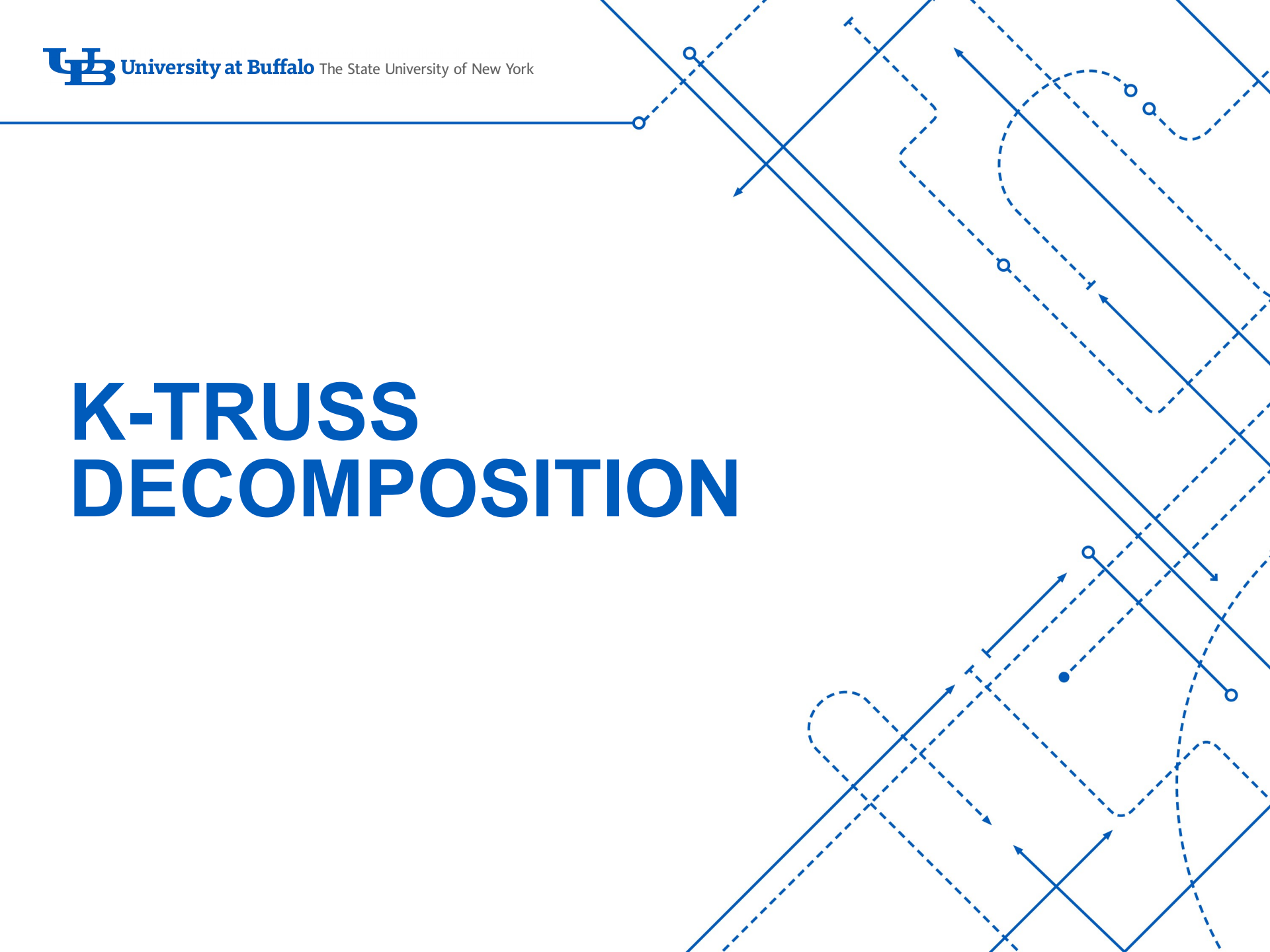
University at Buffalo The State University of New York

Contents

- K-truss Decomposition
- Cuckoo Filter
- Modification
- Analysis

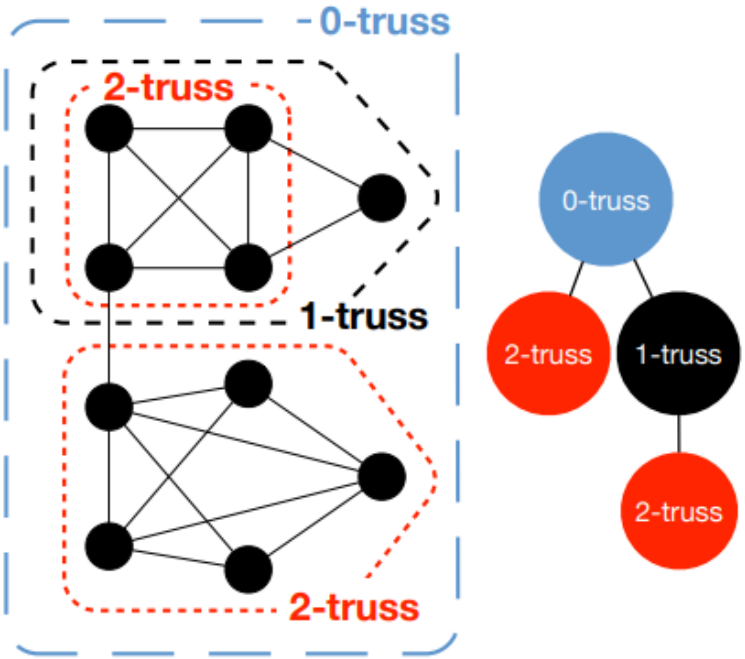


University at Buffalo The State University of New York



K-TRUSS DECOMPOSITION

K-truss Decomposition



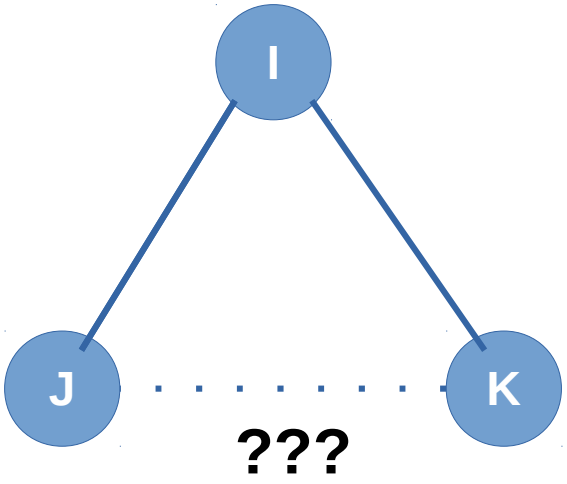
[Cohen, Jonathan]

```
inline int checkConnectedness (Graph& graph, Graph& orderedGraph, Graph& TC, vertex u, vertex v, vector<vertex>* xel = NULL) {

    vertex a = u, b = v;
    if (less_than (b, a, graph))
        swap (a, b);
    for (size_t k = 0; k < orderedGraph[a].size(); k++)
        if (orderedGraph[a][k] == b) {
            TC[a][k]++;
            if (xel == NULL)
                return b;
            else
                return (*xel)[a] + k;
        }
    return -1;
}

// per edge
lol countTriangles (Graph& graph, Graph& orderedGraph, Graph& TC) {

    lol tc = 0;
#ifdef SAVE_TRIS
    FILE* fp = fopen ("tris", "w");
#endif
    for (size_t i = 0; i < orderedGraph.size(); i++) {
        for (size_t j = 0; j < orderedGraph[i].size(); j++) {
            for (size_t k = j + 1; k < orderedGraph[i].size(); k++) {
                vertex a = orderedGraph[i][j];
                vertex b = orderedGraph[i][k];
                vertex c = checkConnectedness (graph, orderedGraph, TC, a, b);
                if (c != -1) {
#ifdef SAVE_TRIS
                    fprintf (fp, "%d %d %d\n", a, b, c);
#endif
                    TC[i][j]++;
                    TC[i][k]++;
                    tc++;
                }
            }
        }
    }
#ifdef SAVE_TRIS
    fclose (fp);
#endif
    return tc;
}
```

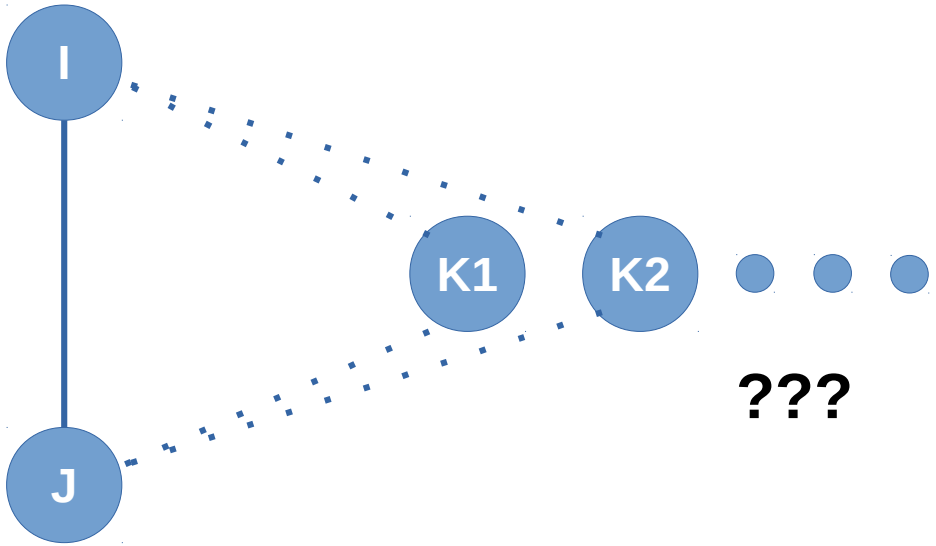

I
J
K
???

```
    vertex monitor = 0;
    while (true) {
        edge e;
        vertex val;
        if (nBucket.PopMin(&e, &val) == -1) // if the bucket is empty
            break;
#ifdef MONITOR
        if (monitor % 10000 == 0)
            printf ("e: %d    val: %d    counter: %d  nEdge: %d\n", e, val, monitor, nEdge);
        monitor++;
#endif
        if (hierarchy) {
            unassigned.clear();
            subcore sc (val);
            skeleton.push_back (sc);
        }

        tc_e = K[e] = val;

        vertex u = el[e].first;
        vertex v = el[e].second;
        vector<vertex> commonNeighbors;
        intersection (graph[u], graph[v], commonNeighbors);
        for (auto w : commonNeighbors) { // decrease the TC of the neighbor edges with greater TC
            edge f = getEdgeId (u, w, xel, el, graph);
            edge g = getEdgeId (v, w, xel, el, graph);
            if (K[f] == -1 && K[g] == -1) {
                if (nBucket.CurrentValue(f) > tc_e)
                    nBucket.DecVal(f);
                if (nBucket.CurrentValue(g) > tc_e)
                    nBucket.DecVal(g);
            }
            else if (hierarchy)
                createSkeleton (e, {f, g}, &nSubcores, K, skeleton, component, unassigned, relations);
        }

        if (hierarchy)
            updateUnassigned (e, component, &cid, relations, unassigned);
    }
```


I
J
K1
K2
???

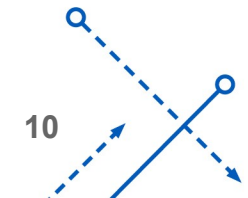
University at Buffalo The State University of New York

CUCKOO FILTER

Objective:

filter unnecessary check query at beginning

reduce the number of check



Cuckoo Filter

Idea

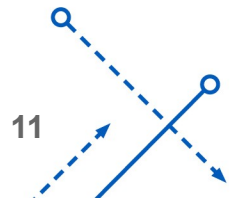
- Two hash function, H1 and H2
- Save to H1, if occupied, save to H2
- If H2 is occupied, continue to find a place

Advantages

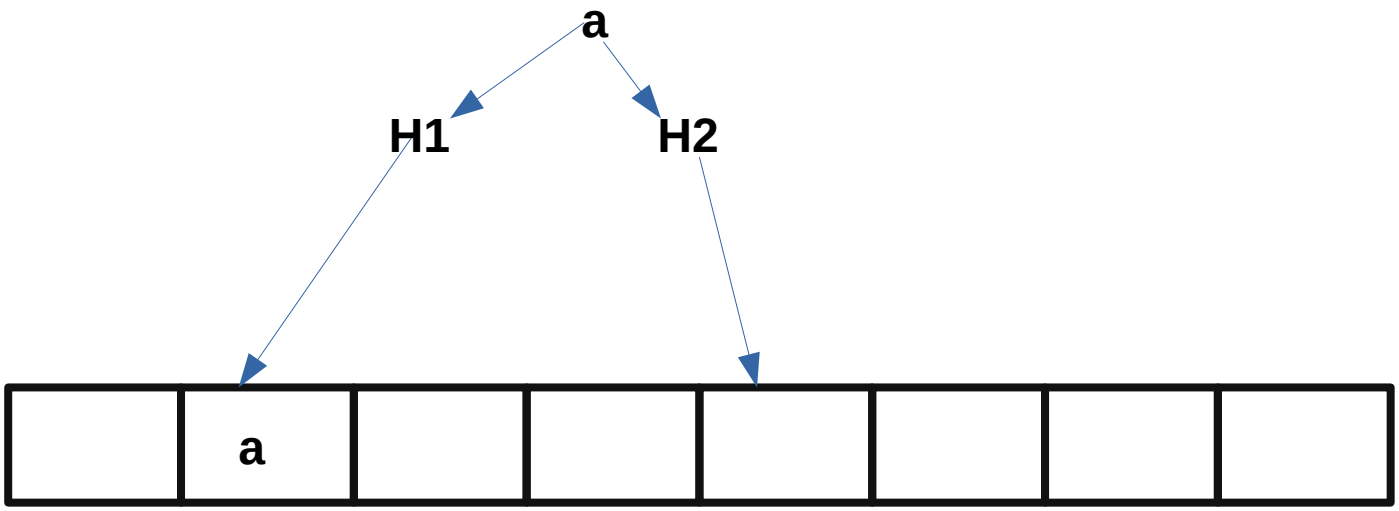
- Query, delete, worst case in O(1) time
- Insertions expected in O(1) time
- High load factor

Disadvantages

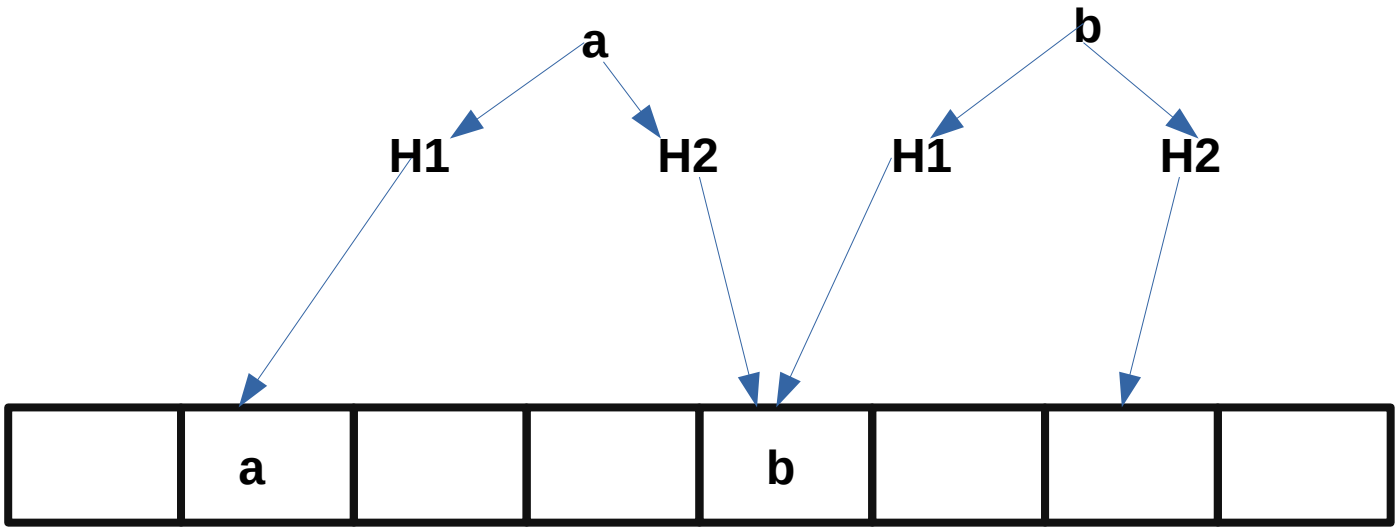
- Possibly return false positive
- Need additional check



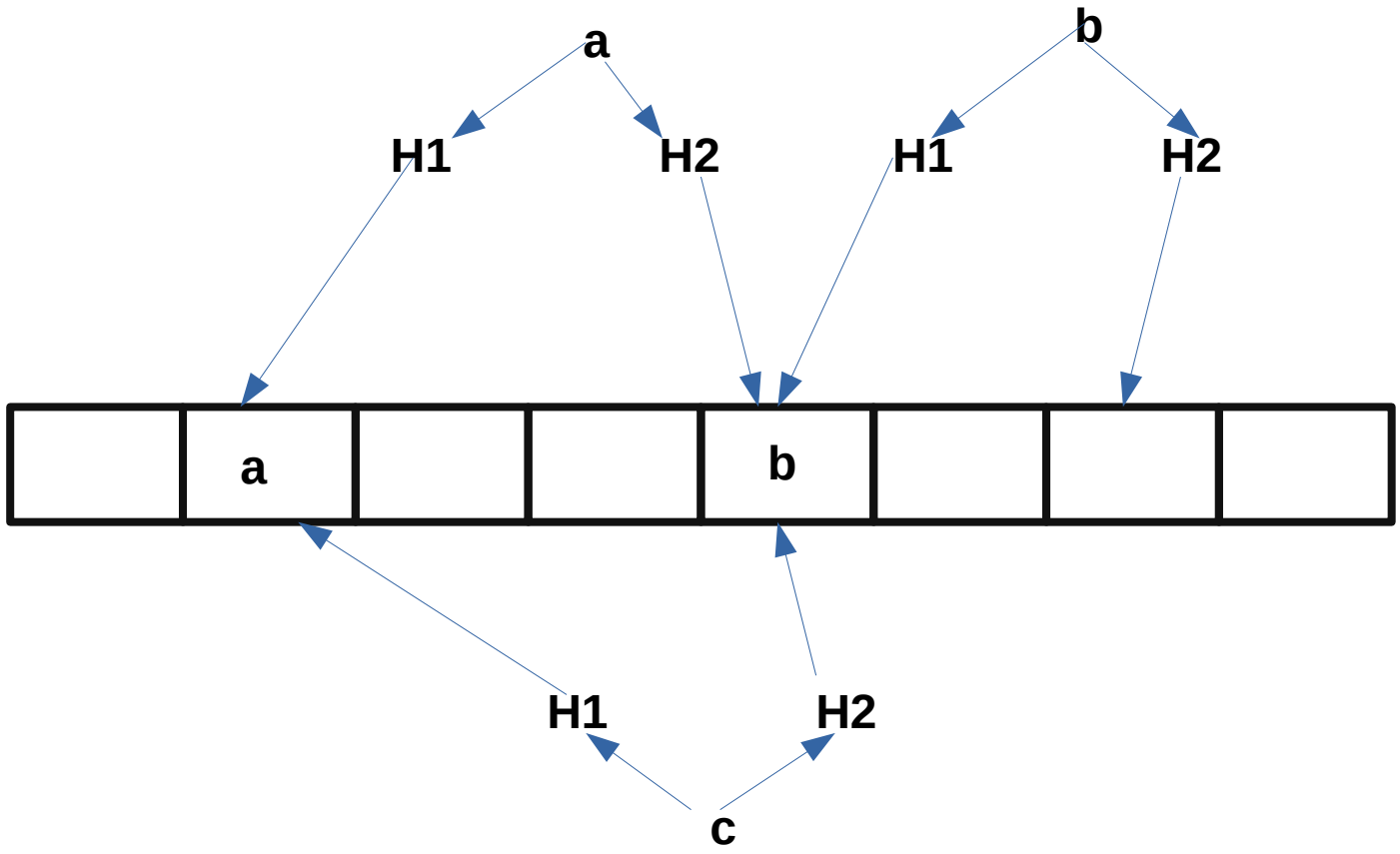
Cuckoo Filter



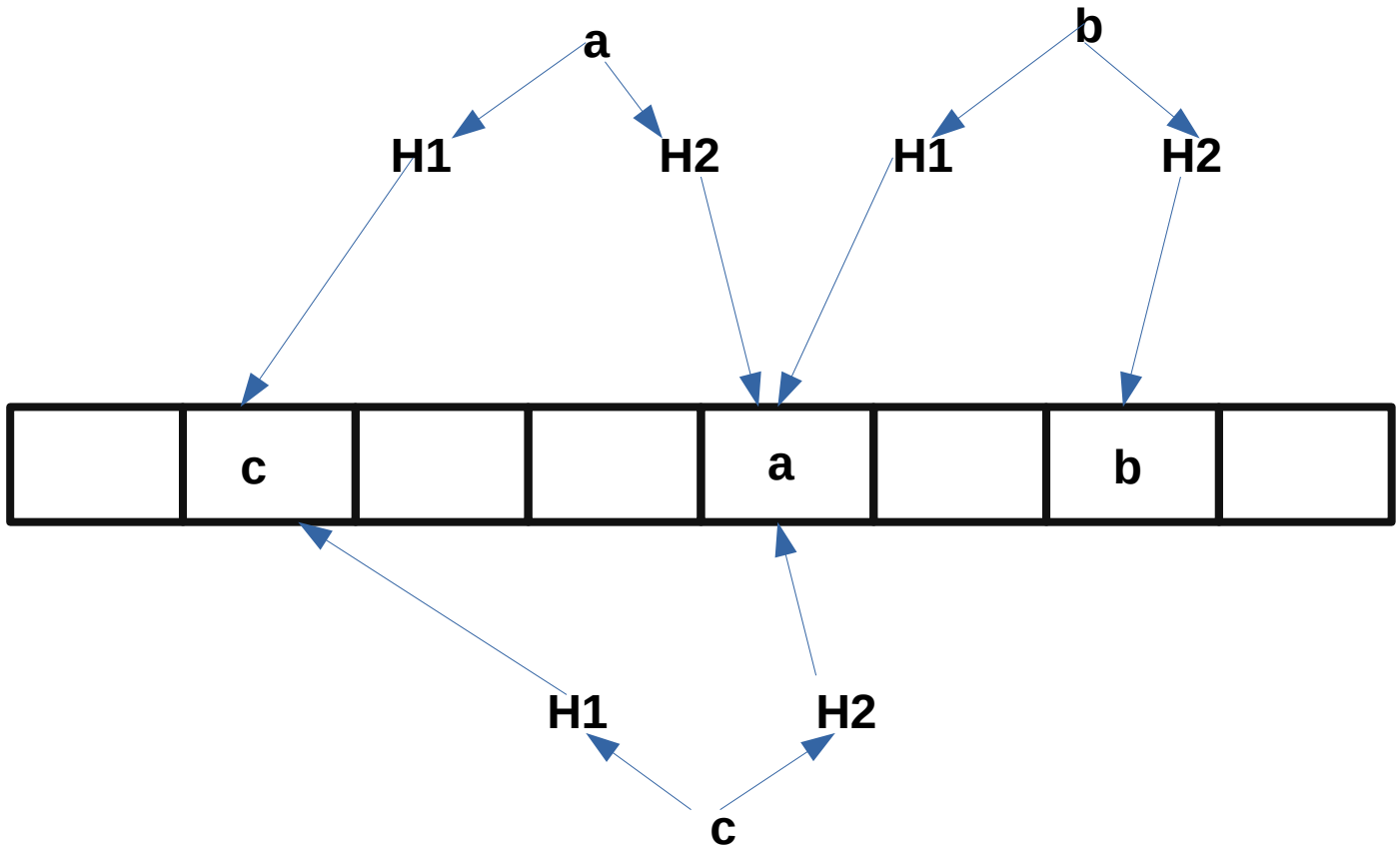
Cuckoo Filter



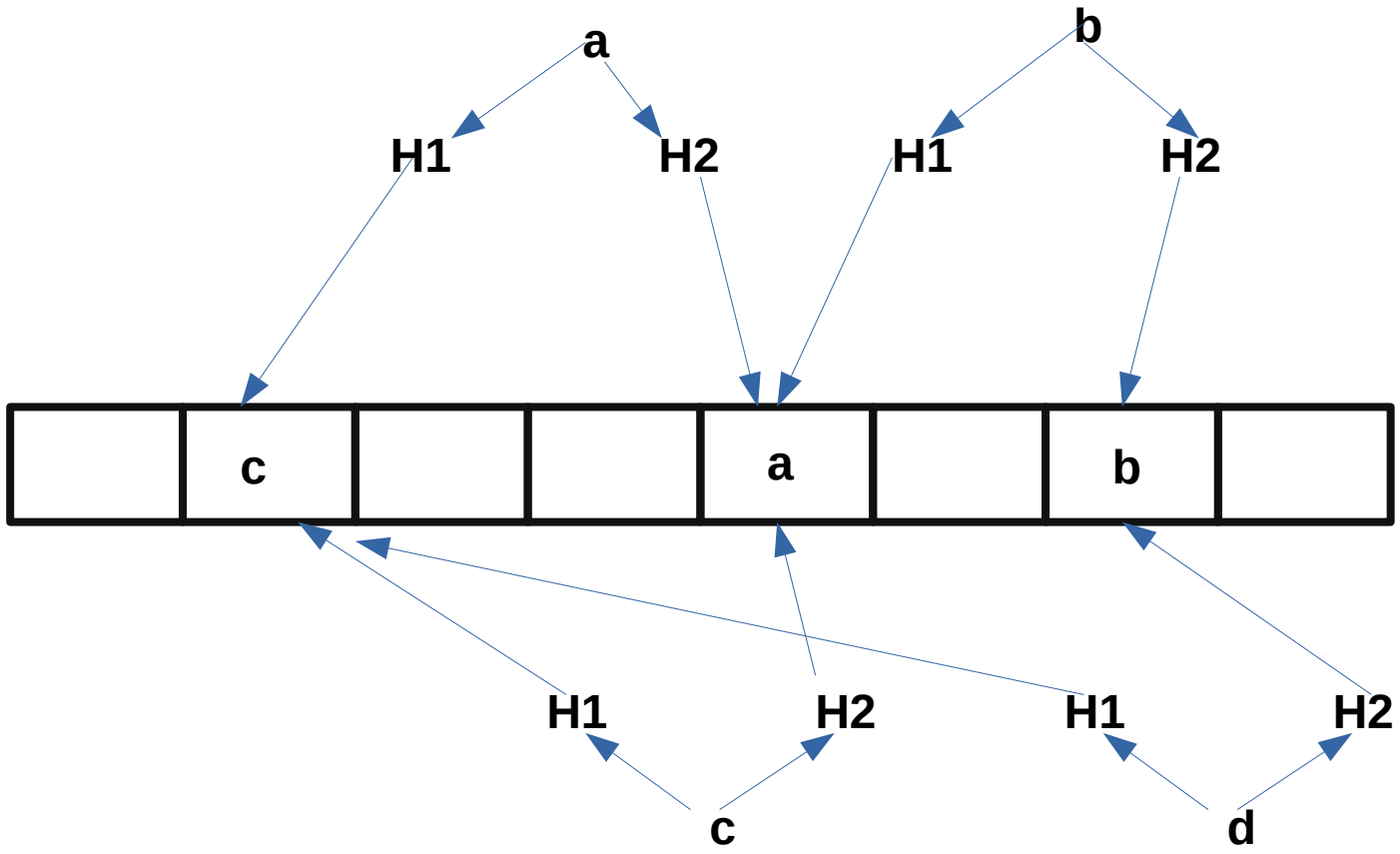
Cuckoo Filter



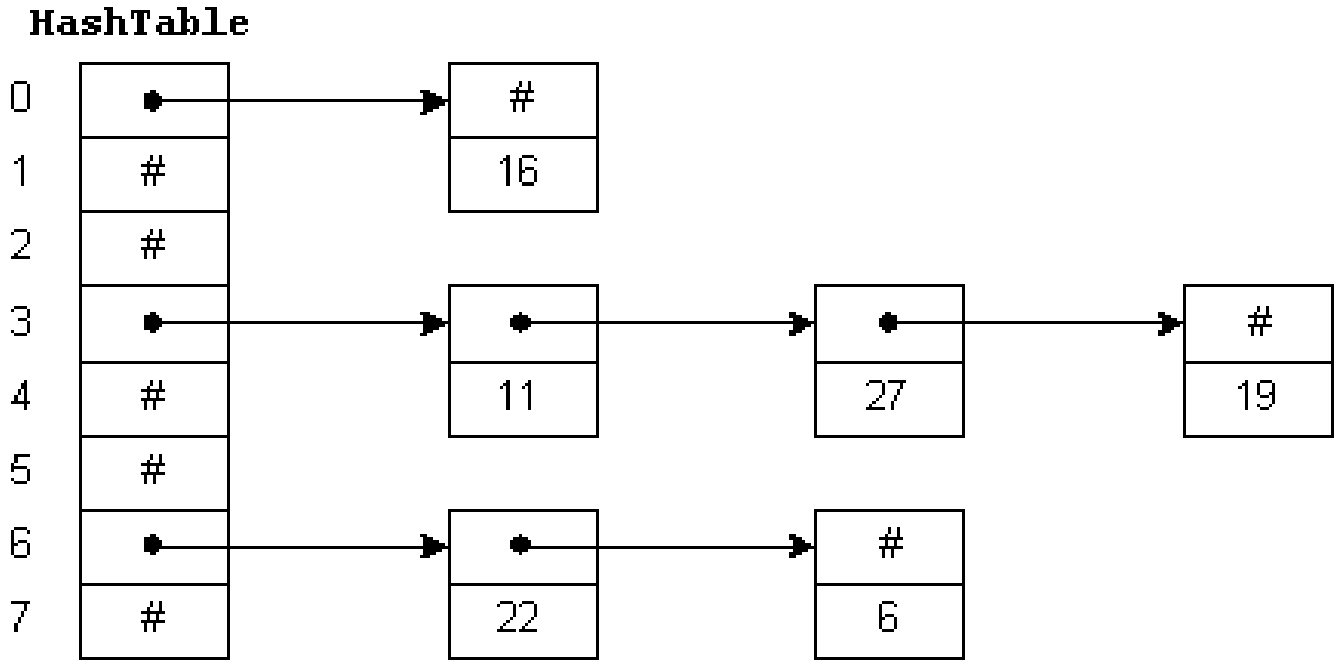
Cuckoo Filter



Cuckoo Filter



Hash Table



University at Buffalo The State University of New York

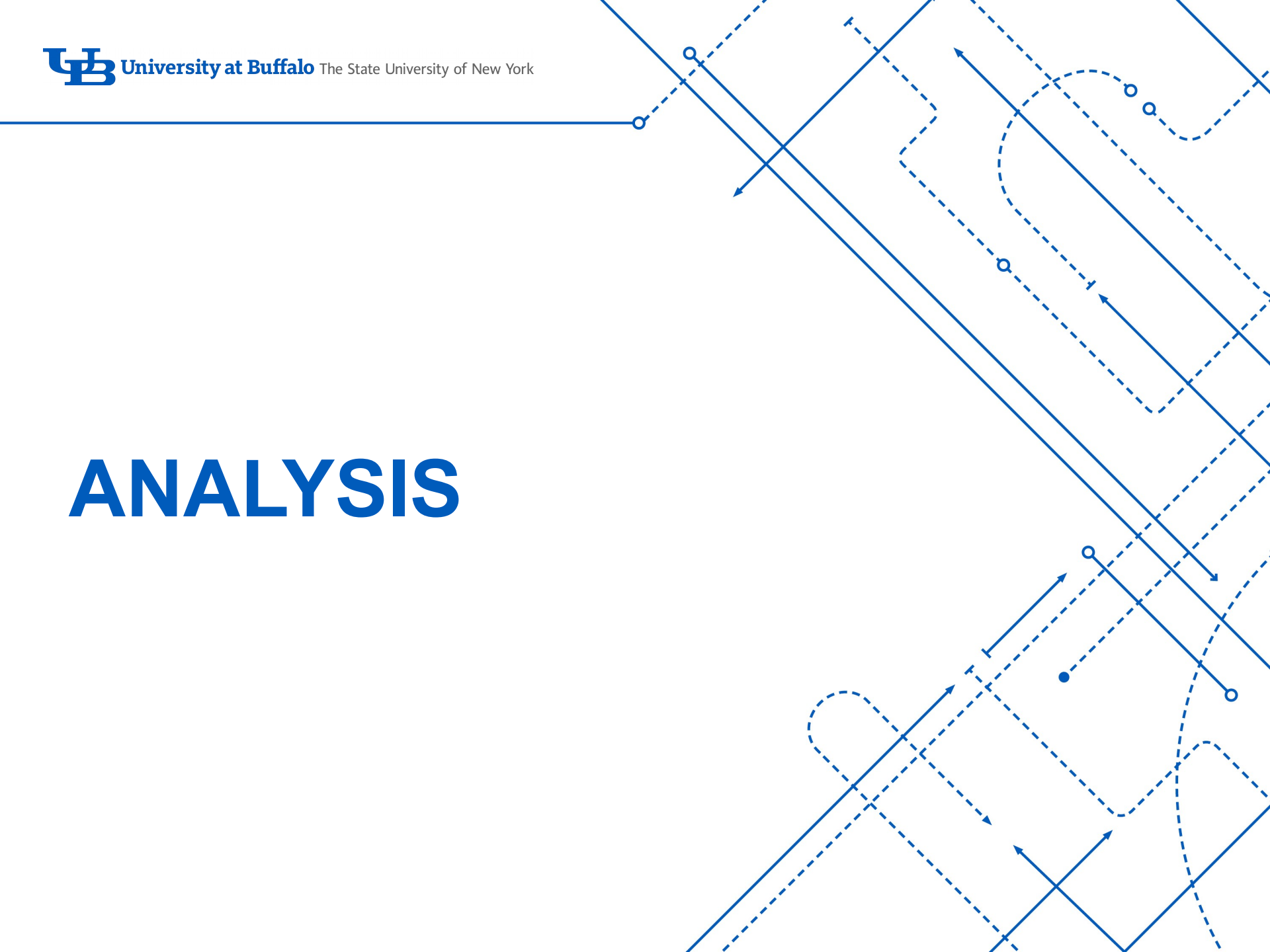
MODIFICATION

```
for (size_t i = 0; i < orderedGraph.size(); i++) {
    for (size_t j = 0; j < orderedGraph[i].size(); j++) {
        for (size_t k = j + 1; k < orderedGraph[i].size(); k++) {
            all++;
            vertex a = orderedGraph[i][j];
            vertex b = orderedGraph[i][k];
            if(filters[a]->Contain(b) == 0){
                into++;
                vertex c = checkConnectedness (graph, orderedGraph, TC, a, b);
                if (c != -1) {
def SAVE_TRIS
                    fprintf (fp, "%d %d %d\n", a, b, c);
dif
                    TC[i][j]++;
                    TC[i][k]++;
                    tc++;
                    valid++;
                }
            }
        }
    }
}
```

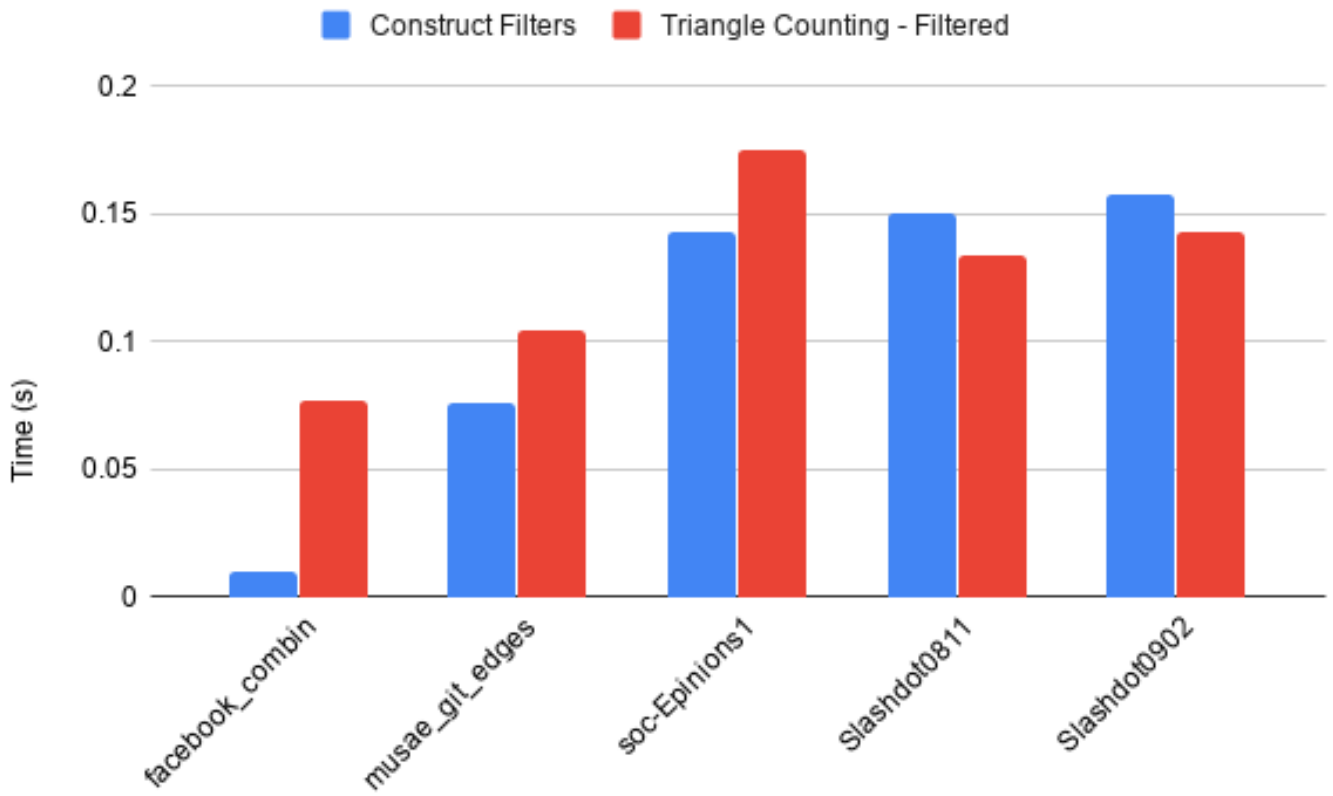
```
vertex u = el[e].first;
vertex v = el[e].second;
vector<vertex> commonNeighbors;
intersectionFiltered (graph[u], graph[v], u, v, commonNeighbors, filters);
for (auto w : commonNeighbors) { // decrease the TC of the neighbor edges with greater TC
    edge f = getEdgeId (u, w, xel, el, graph);
    edge g = getEdgeId (v, w, xel, el, graph);
    if (K[f] == -1 && K[g] == -1) {
        if (nBucket.CurrentValue(f) > tc_e)
            nBucket.DecVal(f);
        if (nBucket.CurrentValue(g) > tc_e)
            nBucket.DecVal(g);
    }
    else if (hierarchy)
        createSkeleton (e, {f, g}, &nSubcores, K, skeleton, component, unassigned, relations);
}
```

University at Buffalo The State University of New York

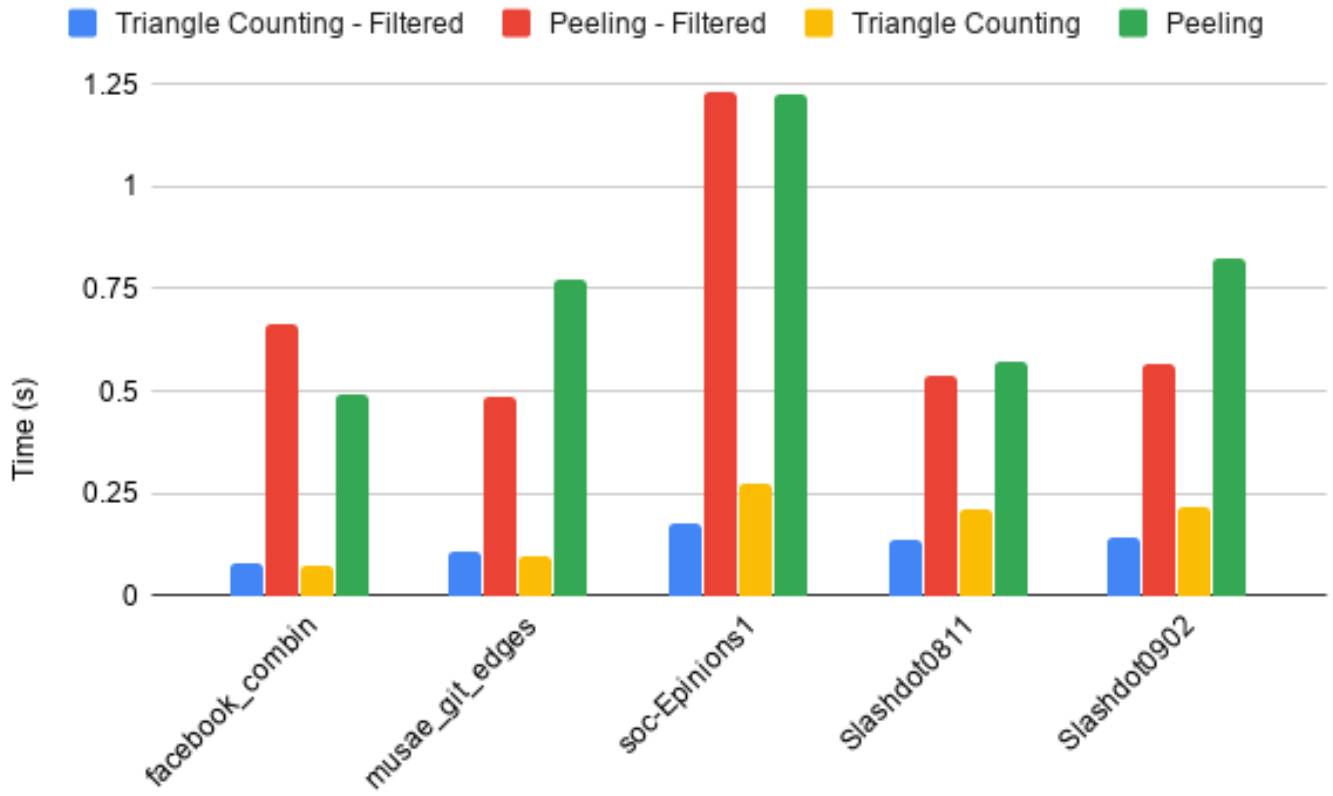
ANALYSIS



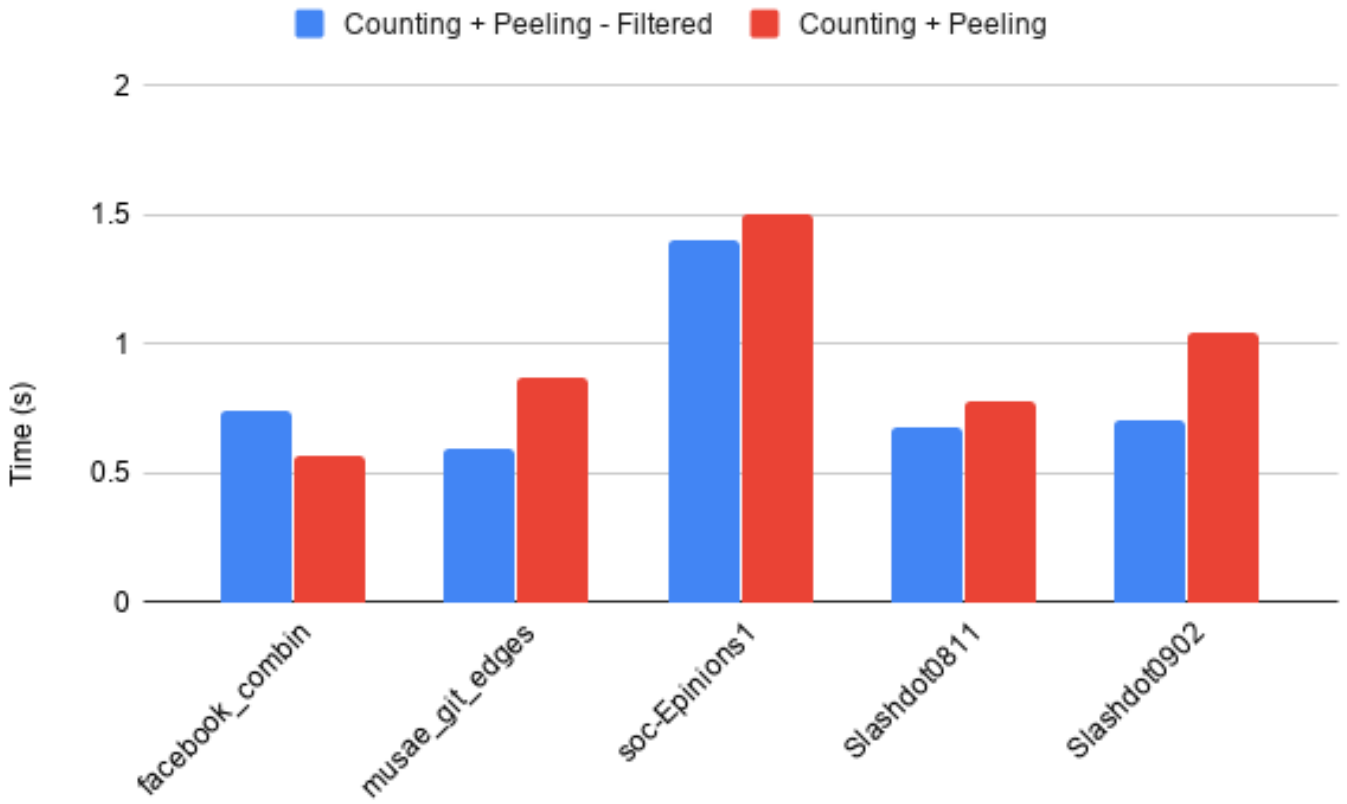
Construct Cuckoo Filter



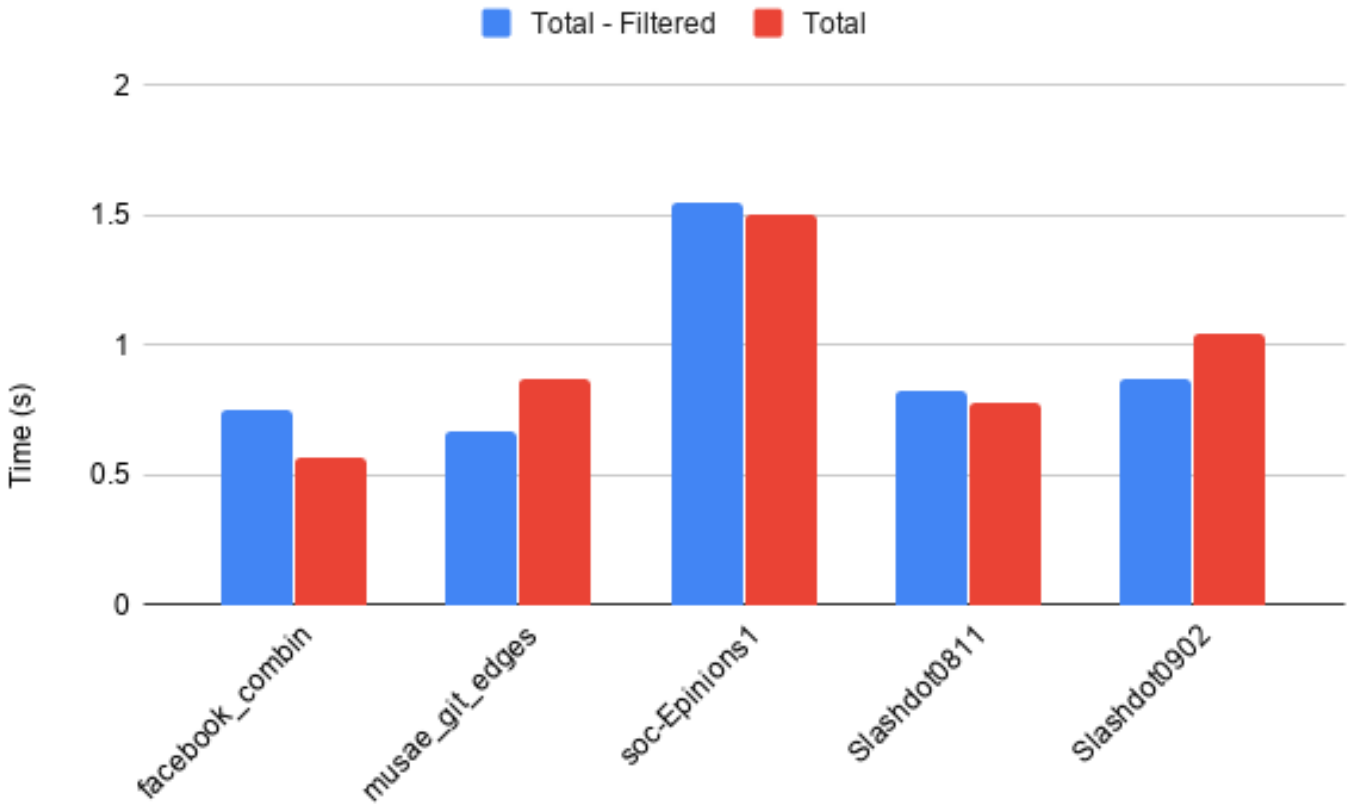
Counting & Peeling Time



Counting & Peeling Time

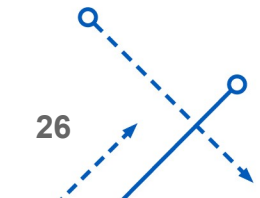


Total Time


Total - Filtered
Total
2
1.5
1
0.5
0
Time (s)
facebook_combin
musae_git_edges
soc-Epinions1
Slashdot0811
Slashdot0902

Conclusion

- Works for certain data set
- Need to reduce the overhead of constructing filter



University at Buffalo The State University of New York

THANKS!