**Topics In Algorithms Project**

**On**

***Analysis of Parameterized Algorithm for Minimum feedback vertex problem***

**By:**

Ammanamanchi Sai Karthik

B150310CS

**Author’s Solution:** <https://github.com/s-gehring/feedback-vertex-set>

**Github:** <https://github.com/gottacodeemall/Minimal-feedback-vertex-problem>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Problem Statement:**

Given an undirected graph G compute a smallest vertex set S such that removing S from G results in a forest, that is, a graph without any cycles.

**Decision Version:**

Given an undirected graph G and a parameter k,does there exist a feedback vertex set by deleting atmost k vertices.

**Brute Force Approach:**

Consider the graph G<v,e>

1.Generate all subsets (power set) on the vertices.

2.For every subset of vertices compute - run the subsets from 0 to 2^k-1

i.For the vertices not in the subset remove the corresponding vertices and its incident edges from the original graph G resulting in G’.

ii.Apply DFS algorithm and find if there are any back edges in the graph.

iii.If we don’t find any back edges(cycles) in the graph we break the loop.

3.Output the minimum vertex set if found else output no.

**Time Complexity:** O(k \* pow(2,k) Exponential run-time

**Parameterized algorithm:**

**Representation of Graph:**

Class GraphData{

Graph graph; contains the adjacency representation of graph & other info

set<Node> necessary\_nodes; contains vertices with self loops & isolated vertices

Mapping mapping; contains the mapping between the real vertex name and incides

}

**Preprocessing:**

1. Remove all vertices having self loops and vertices with degree less than 2 (isolated vertices or edges).

**Algo:**

Traverse through the adjacency list and remove the vertices satisfying the conditions and add them to necessary\_nodes

1. The graph is cut along all its containing bridges, potentially creating new connected components. (We solve the feedback-vertex-set for each connected component separately and union all local solutions).

**Defn:** articulation points - vertices which when removed results in disconnected components.

In DFS tree, a vertex u is articulation point if one of the following two conditions is true.

**1)** u is root of DFS tree and it has at least two children.

**2)** u is not root of DFS tree and it has a child v such that no vertex in subtree rooted with v has a back edge to one of the ancestors (in DFS tree) of u.

**Tarjan’s Algo for articulation points:**

Refer:<https://www.geeksforgeeks.org/articulation-points-or-cut-vertices-in-a-graph/>

For O(V+E) algorithm using DFS

**Removing bridges algo:**

Input: the edges between articulation points(vertices).

1.Remove these edges from the adjacency list.

2.If this results in an isolated node remove it.

//Doubt- why to remove disjoint cycles after removing articulation points.

1. Remove Semi disjoint cycles(cycles which have atmost 1 vertex of degree >2 from the graph.

**Low degree nodes** - no.of edges <=3

**Algo:**

Loop until no semi disjoint cycles exist

1.Use dfs on a vertex with degree>=2 if in the path we find a node with degree less than or greater than 2 break;

After finding a semi disjoint cycle remove the start node from the graph and all its incident edges and continue the loop

**We now apply the FVS algo on individual connected components.**

If (we encounter a component with either only 2 or 3 degree nodes) then

1.contract deg3 vertices

2.Call **solveDeg3** function with reduced node list.

Else

1. If the resulting graph contains vertices with a higher degree, we furthermore try to minimize it via edge contractions.(we take a vertex with degree two and replace it by an edge connecting its neighbors, possibly creating multiedges)

**Algo:**

1.Get all low\_deg\_nodes if the node has degree 2 select both the neighbours in adjacency list and remove the current node.

**Loop**

2. If (the neigbours already have an edge)

1. Connecting them will result in an multi edge formation so we will not connect them but we maintain a set and remember that one them must be added to the solution of FVS(stored in **branching\_pairs**).

Else

Delete the node and add edge.

While deleting a node we will have to be careful not delete a multiedge node.

1. Create a graph consisting only of the multiedges(generate from branching\_pairs) and detect all minimal vertex covers by a recursive procedure.

**Algo:**

set<set<Node>> multi\_edge\_partitions(set<set<Node>> sets of minFVS,set<Node> taken,multiedge graph)

**Branching Algorithm:**

Let graph G<V,E>

Select a vertex v,

**In branch one**

Include the current vertex in the FVS solution set and if the neighbours neighbour has degree 0 include it also.

Recurse on multi\_edge\_partitions(sol\_set,taken+v,G-edgesof(v))

**In other branch**

Include the neighbours of current vertex in FVS solution set and remove the edges of the neighbours.

Recurse on multi\_edge\_partitions(sol\_set,taken+neigh(v),g-edgesof(neigh(v))

**Base Condition**

if(we empty the adj\_list in G)

Add taken to the solution set (sol\_set)

Now we can find the optimal solution set by considering the size of the of best set in sol\_set.

We find the optimal set of vertices from the multiedge graph and we will now run the **iterative compression algorithm** on the graph without the nodes which have been included in the fvs solution set.

1. If no multiedges are found run **iterative compression algorithm.**

**Iterative Compression Algorithm:**

**Complexity: O(5^k \* k \* n^2)**

**Compression step:**

Algorithm-1 Feedback(G, V1, V2,k)

**Input:** G = (V , E) is a graph with a forest bipartition (V1, V2), k is an integer.

**Output**: An FVS F of G such that |F |k and F ⊆ V1; or report “No” (i.e.,

no such an FVS exists).

1. if (k < 0) or (k =0 and G is not a forest) then return “No”;

2. if (k 0) and G is a forest then return ∅;

3. if a vertex w in V1 has at least two neighbors in V2 then

3.1. if two of the neighbors of w in V2 belong to the same tree in G[V2] then

F1 = Feedback(G − w, V1 \ {w}, V2,k −1);

if F1 = “No” then return “No” else return F1 ∪ {w};

3.2. Else

F1 = Feedback(G − w, V1 \ {w}, V2,k −1);

F2 = Feedback(G, V1 \ {w}, V2 ∪ {w},k);

if F1 = “No” then return F1 ∪ {w} else return F2;

4. else pick any vertex w that has degree 1 in G[V1];

4.1. if w has degree 1 in the original graph G then

return Feedback(G − w, V1 \ {w}, V2,k);

4.2. else return Feedback(G, V1 \ {w}, V2 ∪ {w},k).

**Iteration Step:**

We first apply a **2-approximation algorithm** for FVS which runs in O(n^2) time.

If there is no 2k size FVS

Return k size FVS is not possible

Else

Returns a **F1**, a 2k size FVS.

Consider any subset **V1** of k vertices from the 2k size FVS, now V0 = V1 ∪( V \ F1).

Now G[V0](induced subgraph by vertices V0) forms an FVS of size k.

We now add a vertex from V-V0 and this a k+1 FVS as the induced subgraph on the additional vertex is an isolated node(the node itself).

1.We then apply the compression step on these k+1 vertices

If we cannot find a k vertex FVS

We simply cannot find a k vertex FVS for the complete graph

Return FALSE

Else

Include a new vertex from V-V0 to the k vertex FVS solution from compression step and run the compression algorithm again on this k+1 FVS set

Repeat Until all vertices are included

**Time complexity:**

As we already have k vertices in the initial FVS we need to iterate (|F1|-k) times running compression algorithm.

Since |F1|-k <= k

**O(k) \* O(5^k \* n^2) ~ O(5^k \* k \* n^2) ~ O\*(5^k)**

**Note**: We can also choose k+1 size set and now there exist a k size FVS and we can use iterative steps. The only difference is we multiply there with (n-k) here(by starting with approximation algorithm) the multiplication is with |F1|-k.

**Reference:**

J. Chen, F. V. Fomin, Y. Liu, S. Lu, and Y. Villanger. Improved algorithms for feedback vertex set problems. Journal of Computer and System Sciences, 74(7):1188- 1198, 2008.

**solvDeg3 - Degree 3 case - polynomial time solvable**

Convert the problem to disjoint FVS problem and solve it polynomial time.

**Corollary 3.5:** There is an O(n^2 log^6 n)-time algorithm that on an instance (G; V1, V2; k) of disjointfvs where all vertices in V1 have degree bounded by 3, either constructs a V1-FVS of size bounded by k.

**Paper:**Y. Cao, J. Chen, and Y. Liu. On feedback vertex set, new measure and new struc- tures. CoRR, abs/1004.1672, 2010.

**Real Time Difference in running time between *brute force algorithm* and *parameterized algorithm***

**Machine Specifications:**

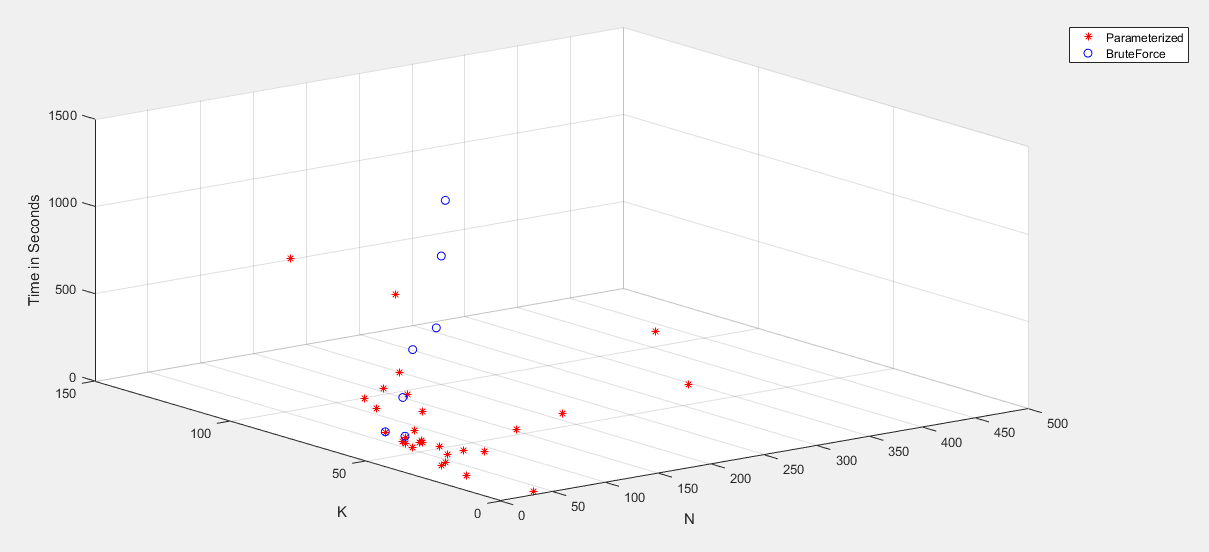
Athena Server Nitc

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **N** | **K** | **Time taken by Parameterized**  **algorithm(sec)** | **Time taken by Brute Force**  **Algorithm(sec)** |
| 1 | 112 | 102 |  |  |
| 2 | 49 | 48 | 63.4405 | TLE |
| 3 | 2888 |  |  |  |
| 4 | 212 | 60 | 2.41192 | TLE |
| 5 | 3133 |  |  |  |
| 6 | 1174 |  |  |  |
| 7 | 1354 |  |  |  |
| 8 | 2383 |  |  |  |
| 9 | 2736 |  |  |  |
| 10 | 2737 |  |  |  |
| 11 | 2746 |  |  |  |
| 12 | 300 |  |  |  |
| 13 | 3012 |  |  |  |
| 14 | 3374 |  |  |  |
| 15 | 89 | 72 | 310.465 | TLE |
| 16 | 56 |  |  |  |
| 17 | 80 | 45 | 0.030928 | TLE |
| 18 | 90 | 67 | 1.55733 | TLE |
| 19 | 100 | 78 | 718.485 | TLE |
| 20 | 150 |  |  |  |
| 21 | 160 |  |  |  |
| 22 | 180 |  |  |  |
| 23 | 200 |  |  |  |
| 24 | 220 |  |  |  |
| 25 | 230 |  |  |  |
| 26 | 260 |  |  |  |
| 27 | 280 |  |  |  |
| 28 | 290 |  |  |  |
| 29 | 300 |  |  |  |
| 30 | 2352 |  |  |  |
| 31 | 2078 |  |  |  |
| 32 | 1816 |  |  |  |
| 33 | 1596 |  |  |  |
| 34 | 1405 |  |  |  |
| 35 | 1079 |  |  |  |
| 36 | 620 |  |  |  |
| 37 | 347 | 66 | 0.207869 | TLE |
| 38 | 12094 |  |  |  |
| 39 | 12882 |  |  |  |
| 40 | 1036 |  |  |  |
| 41 | 146 |  |  |  |
| 42 | 1351 |  |  |  |
| 43 | 6324 |  |  |  |
| 44 | 14229 |  |  |  |
| 45 | 3377 |  |  |  |
| 46 | 16816 |  |  |  |
| 47 | 2431 |  |  |  |
| 48 | 2029 |  |  |  |
| 49 | 5848 |  |  |  |
| 50 | 5534 |  |  |  |
| 51 | 433 | 112 | 0.4516 | TLE |
| 52 | 4749 |  |  |  |
| 53 | 445 |  |  |  |
| 54 | 454 |  |  |  |
| 55 | 4969 |  |  |  |
| 56 | 65 |  |  |  |
| 57 | 112 |  |  |  |
| 58 | 10790 |  |  |  |
| 59 | 10859 |  |  |  |
| 60 | 10886 |  |  |  |
| 61 | 11011 |  |  |  |
| 62 | 11051 |  |  |  |
| 63 | 11174 |  |  |  |
| 64 | 6474 |  |  |  |
| 65 | 3015 |  |  |  |
| 66 | 4450 |  |  |  |
| 67 | 292 |  |  |  |
| 68 | 85 |  |  |  |
| 69 | 274 |  |  |  |
| 70 | 1612 |  |  |  |
| 71 | 366 |  |  |  |
| 72 | 58 | 58 | 0.059188 | 44.074 |
| 73 | 70 | 70 | 0.095044 | 1.92164 |
| 74 | 70 | 70 | 0.095156 | 2.48454 |
| 75 | 70 | 70 | 0.095438 | 2.26651 |
| 76 | 406 |  |  |  |
| 77 | 76 |  |  |  |
| 78 | 118 |  |  |  |
| 79 | 182 |  |  |  |
| 80 | 366 |  |  |  |
| 81 | 144 |  |  |  |
| 82 | 292 |  |  |  |
| 83 | 61 |  |  |  |
| 84 | 62 | 44 | 0.120526 | TLE |
| 85 | 39 | 39 | 0.492251 | 771.618 |
| 86 | 62 | 53 | 205.363 | TLE |
| 87 | 162 |  |  |  |
| 88 | 198 |  |  |  |
| 89 | 209 |  |  |  |
| 90 | 221 |  |  |  |
| 91 | 245 |  |  |  |
| 92 | 307 |  |  |  |
| 93 | 310 |  |  |  |
| 94 | 346 |  |  |  |
| 95 | 361 |  |  |  |
| 96 | 419 |  |  |  |
| 97 | 492 |  |  |  |
| 98 | 511 |  |  |  |
| 99 | 59 |  |  |  |
| 100 | 607 |  |  |  |
| 101 | 758 |  |  |  |
| 102 | 869 |  |  |  |
| 103 | 265 |  |  |  |
| 104 | 577 |  |  |  |
| 105 | 1961 |  |  |  |
| 106 | 105 |  |  |  |
| 107 | 10922 |  |  |  |
| 108 | 19362 |  |  |  |
| 109 | 66 |  |  |  |
| 110 | 258 |  |  |  |
| 111 | 36 | 36 | 0.083587 | TLE |
| 112 | 153 | 54 | 0.014055 | TLE |
| 113 | 149 | 136 | 606.784 | TLE |
| 114 | 55 | 54 | 1.5483 | 561.528 |
| 115 | 73 | 51 | 0.022133 | TLE |
| 116 | 110 | 89 | 3.39238 | TLE |
| 117 | 125 | 99 | 0.533274 | TLE |
| 118 | 754 |  |  |  |
| 119 | 32 | 25 | 0.023372 | 162.08 |
| 120 | 90 | 41 | 0.004812 | TLE |
| 121 | 45 | 38 | 0.045111 | TLE |
| 122 | 145 | 100 | 33.3962 | TLE |
| 123 | 71 | 63 | 0.34487 | TLE |
| 124 | 74 | 58 | 0.01798 | TLE |
| 125 | 69 | 57 | 0.104729 | TLE |
| 126 | 158 | 96 | 1.71833 | TLE |
| 127 | 61 | 60 | 0.093412 | 253.697 |
| 128 | 1157 |  |  |  |
| 129 | 2395 |  |  |  |
| 130 | 4960 |  |  |  |

**By constraining the running time to 20 minutes**

**Parameterized Algo** -- 32 testcases.

**Brute Force Algo** -- 7 testcases



**Custom Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case Description** | **Minimum Vertex Cover** | **Size** | **Time taken by Brute Force**  **Algorithm(sec)** | **Time taken by Parameterized**  **algorithm(sec)** |
| 5 vertex 8 edges | { 2 1 } | 2 | 0.000395 | 0.000246 |
| 10 vertex 30 edges | { 8 3 5 1 4 } | 5 | 0.008744 | 0.001738 |
| 15 vertex 60 edges | { 14 7 15 6 4 10 5 1 8 2 } | 10 | 0.453158 | 0.018507 |
| 20 vertex 120 edges | { 13 16 9 8 2 18 1 15 20 4 7 5 14 6 } | 14 | 27.3785 | 0.374193 |
| 21 vertex 150 edges | { 16 15 14 12 8 7 9 2 1 20 3 21 4 5 6 10 } | 16 | 94.6218 | 1.50788 |
| 22 vertex 200 edges | { 16 14 19 11 10 9 8 7 2 22 1 6 18 3 4 12 5 20 } | 18 | 347.075  (5 min) | 7.34601 |
| 23 vertex 250 edges | { 15 14 11 9 10 8 17 7 19 20 2 22 1 12 23 3 21 4 13 5 } | 20 | 1275.12  (21 min) | 77.8364 |
| 25 vertex 200 edges | { 20 17 16 15 14 13 12 11 8 7 24 2 1 25 3 4 18 5 9 6 } | 20 | 3044.25  (50 min) | 13.3523 |

**Graphical Representation of Time complexity**

