Mahidol University International College

ICMA 393/484 Graph Theory & Combinatorics

Minimum Feedback Vertex Set Report

Achira Laovong

March 2024

Introduction

A feedback vertex set (FVS) of a graph is a set of vertices whose removal leaves a graph without a cycle. The minimum FVS problem is an NP-complete problem. The problem stated that "Given an undirected graph $G = (V, E)$, find the smallest set of vertices such that removing them from the graph makes it acyclic (a forest)." In this project, we are dealing with an undirected simple graph.

The minimum-FVS problem is used in the study of deadlock recovery. Each cycle is the deadlock situation that we are trying to solve, and the minimum feedback vertex set is the minimum number of processes that need to be aborted.

In this project, we will be looking at approximation algorithms to find the minimum feedback vertex set and comparing the approximate solution to the optimal solution.

Naive Solution

The naive solution for the Minimum Feedback Vertex Set (MFVS) problem involves generating all subsets of vertices and checking if the removal of those vertices results in an acyclic graph.

The algorithm can be broken down into two main steps:

1. **Cycle Detection**: The algorithm uses Depth First Search (DFS) to determine whether the given graph is acyclic. The worst-case running time for DFS is $O(V + E)$, where V is the number of vertices and E is the number of edges in the graph.

2. **Subset Generation**: The algorithm generates all possible subsets of vertices within the graph G. If the graph $G - S$ is found to be acyclic for a subset $S \subseteq G$, the algorithm returns immediately. The running time for generating all subsets is $O(2^V)$, as there are 2^V possible subsets of V vertices.

Therefore, the overall worst-case running time of this naive algorithm is $O(2^V \cdot (V + E))$. This is because for each subset, the algorithm performs a DFS to check if the graph is acyclic.

Algorithm 1 Naive Feedback Vertex Set (FVS) Algorithm

```
1:  function NAIVE_FVS(G)
2:      if IS_ACYCLIC_GRAPH(G) then
3:          return ∅
4:      end if
5:      for i ← 1 to number of nodes in G do
6:          for all subsets of size i in the power set of nodes in G do
7:              G_copy ← copy of G
8:              Remove nodes in subset from G_copy
9:              if IS_ACYCLIC_GRAPH(G_copy) then
10:                 return subset as a set
11:             end if
12:         end for
13:     end for
14:     return None
15: end function
```

Although a naive algorithm ensures an optimal solution, it is not widely used as the running time is exponential. The approximation algorithm is much preferred as it is faster and the solution itself can be ensured within a certain range from the optimal solution.

Local-Ratio Feedback Vertex Set Algorithm (3-Approximation Algorithm)

The LR-FVS algorithm is an algorithm designed to approximate solutions for the minimum Feedback Vertex Set (FVS) problem. The basic principle behind this algorithm is using the local ratio method, which involves breaking complex problems into simpler subproblems, solving these subproblems, and then combining the solutions of the subproblems to form a solution for the original problem. The algorithm also makes use of the idea that Minimum Feedback Vertex Set cannot have a vertex v where $deg(v) < 2$.

Definition:

Clean Graph: A graph without vertex v where $deg(v) < 2$. Graph without leaf and isolated vertex.

The Local Ratio Method can be applied in LR-FVS as follows:

1. The algorithm calculated the minimum ratio between the cost and degree of a vertex - 1.

2. The algorithm breaks down the problem into two subproblems. One subproblem involves finding the ratio as mentioned in the previous step and calculating the next cost function to be used in the next iteration. Another subproblem is that it uses the old cost function and the new cost function to determine which vertex to exclude from the graph, thus breaking down the problem. After solving, the algorithm combines all the solutions into one.

Suppose the size of exact solution of minimum Feedback Vertex Set is k and solution of LR-FVS is s. The size of the approximated solution provided by LR-FVS is within this $k \leq s \leq 3k$.

Algorithm 2 LR-FVS

function LR-FVS(G, c)
  $G \leftarrow$ CLEAN(G)
  if $V(G) = \emptyset$ **then**
    return $\emptyset$
  end if
  $\alpha \leftarrow \min\{\frac{c_v}{(d_v-1)} : v \in V(G)\}$
  for all $v \in V(G)$ **do**
    $\bar{c}_v \leftarrow \alpha \cdot (d_v - 1)$
    $c'_v \leftarrow c_v - \bar{c}_v$
  end for
  $\bar{F} \leftarrow \{v \in V(G) : c'_v = 0\}$
  $G' \leftarrow G \setminus \bar{F}$
  $F' \leftarrow$ LR-FVS(G', c'_v)
  Set $F \subseteq F' \cup \bar{F}$ to be a minimal FVS in G
  return F
end function

function CLEAN(G)
  while G contains any vertex of degree less than 2 **do**
    Remove the vertex from G
  end while
end function

Although the output of this algorithm does not give out optimal solution, it is significantly faster and it is no longer running in exponential time.

$$T(V) = O(V^2) + T(V-1) = O(V^3)$$

The algorithm worst-case is $O(V^3)$ since it have to copy the graph and supposedly that every iteration removes only single vertex.

Claim 1. *Let $G = (V, E)$ be a clean graph, If $c_v = \alpha \cdot (deg(v) - 1)$, for $\forall v \in V$, is a degree -1 weighted cost function on G, the cost of any minimal FVS in G is at most thrice the cost of the minimum FVS in G.*

Proof. Let G have $n = |V|, m = |E|$. Without loss of generality, let $\alpha = 1$ and that G is connected. The proof can easily be extended for when G is disconnected. There are 2 parts we need to proof. One is to show that the cost of minimum FVS in G is at least $m - n + 1$ and then show that it is at most $3(m - n + 1)$.

Suppose F^* is the minimum FVS in G and that $|F^*| = k$. Since $G \backslash F^*$ is acyclic, the number of edges is at most $n - k - 1$. This means that the number of edges connecting F^* and $G \backslash F^*$ is at least $m - (n - k - 1)$. We can now determine the lower bound of the cost of the minimum FVS in G.

$$c(F^*) = \sum_{v \in F*} (deg(v) - 1) = \sum_{v \in F*} deg(v) - k \geq m - (n - k - 1) - k = m - n + 1$$

We have now shown that the cost of minimum FVS in G is at least $m-n+1$. We need to show that it is at most $3(m-n+1)$.

Let F be the minimum FVS in G and $|F| = k$. Let $\mathcal{T} = \{T_1, T_2, ...T_c\}$ be the connected tree components in $G \backslash F$. For every $T \in \mathcal{T}$, there are at least 2 edges connecting T with F. This means that the number of edges T and F, denoted as x, is at least $2c$. The cost of FVS (F), is

$$c(F) = \sum_{v \in F} (deg(v) - 1) = \sum_{v \in F} deg(v) - k = 2m - \sum_{v \in V \backslash F} deg(v) - k$$

Since $G \backslash F$ is acyclic, the number of edges remained is $n - k - c$. Then $\sum_{v \in V \backslash F} deg(v) = 2(n - k - c)$. We also need to consider adding the number of edges connecting F and G (i.e. x) We can substitute it in to get:

$$c(F) = 2m - (2n - 2k - 2c + x) - k = 2(m - n) + 2k + 2c - x - k \leq 2(m - n) + k$$

□

Proposition 1. *Let G be a connected graph with n vertices and m edges. Then the size (number of vertices) of any minimal FVS is at most $m - n + 1$.*

Proof. The graph $G \backslash F$ have $\mathcal{T}$ components. Since G is connected, we can add some edges from G to $G \backslash F$ to form a single tree T. For every vertex $v \in F$, there must be at least 2 edges connected to one of the component in $\mathcal{T}$ and v. It is important to note that for each vertex v, both edges cannot exist in the tree T as it would form a cycle which means that there are at least $|F|$ edges of G not in T. Thus, the number of edges in G is $m \geq |F| + n - 1$ since T have $n - 1$ edges. Rearranging the equation we can create $|F| \leq m + n - 1$ which completes the proof for the proposition. □

Let's use the Proposition 1 into Claim 1.

Theorem 1. *For any instance (G, c) of minimum feedback vertex set, the cost of the feedback vertex set F returned by the algorithm LR-FVS is $c(F) \leq 3 \cdot c(OPT(G, c))$.*

Proof. Suppose F is the FVS of G and that the base case is an empty graph. Using induction on (G', c'), F' is an FVS in G'. Since $G' = G \backslash \bar{F}$, the cycle not covered in G by F' are covered by $\bar{F}$. This means that $F' \cup \bar{F}$ is the FVS of G. Thus, the minimum FVS does exist in G.

The cost of solution F can be written as $c(F) = c'(F) + \bar{c}(F)$. We can write the bound of the cost as $c'(F) \leq c'(F') + c'(\bar{F})$. From the induction hypothesis, we know that $c'(F') \leq 3 \cdot c'(OPT(G', c'))$. By plugging it into the bound, $c'(F) \leq 3 \cdot c'(OPT(G', c')) + c'(\bar{F})$. Utilizing the claim and the proposition where F is the minimum FVS and $\bar{c}$ is the *degree* -1 cost function, we get $\bar{c}(F) \leq 3 \cdot \bar{c}(OPT(G, \bar{c}))$. Our final bound can be written as

$$c(F) = c'(F) + \bar{c}(F) \leq 3 \cdot c'(OPT(G', c')) + 3 \cdot \bar{c}(OPT(G, \bar{c})$$

It is important to know that

$$c'(OPT(G', c')) \leq c'(OPT(G, c))$$
$$\bar{c}(OPT(G, \bar{c}) \leq \bar{c}(OPT(G, c)$$

$\therefore c(F) \leq 3 \cdot c(OPT(G, c))$. □

Improved Local Ratio Feedback Vertex Set Algorithm (2-Approximation Algorithm)

This proposed algorithm by Balfa et al. is an extended version of the original LR-FVS that ensures that the result FVS is within $k \leq s \leq 2k$.

Definition:

Semi-disjoint cycle: A cycle with one and only one vertex v with $deg(v) > 2$.

Algorithm 3 LR-FVS Modified

```
function LR-FVS-Mod(G, c)
    G ← Clean(G)
    if V(G) = ∅ then
        return ∅
    end if
    if G contain semi-disjoint cycle then
        u_c ← vertex with maximum degree in cycle
        min_cost ← minimum cost of vertices in cycle
        u_min ← vertex in cycle with minimum cost
        for all vertex v in graph do
            if v is in cycle then
                reduce cost of v by min_cost
            end if
        end for
        G'' ← G\(C\u_c)
        F'' ← LR-FVS-Mod(G'', c'')
        return F'' ∪ u_min
    else
        α ← min{c_v/(d_v − 1) : v ∈ V(G)}
        for all v ∈ V(G) do
            c̄_v ← α · (d_v − 1)
            c'_v ← c_v − c̄_v
        end for
        F̄ ← {v ∈ V(G) : c'_v = 0}
        G' ← G \ F̄
        F' ← LR-FVS-Mod(G', c'_v)
        Set F ⊆ F' ∪ F̄ to be a minimal FVS in G
        return F
    end if
end function
```

In this version, the running time is approximately $O(V^3C)$ where V is the number of vertices and C is the number of cycles. This running time is slightly larger compared to the 3-approximation algorithm since we have to check whether the graph contains a semi-disjoint cycle every iteration but it ensures better approximation when there exist semi-disjoint cycle within a graph.

This algorithm guarantees a solution within $2 \cdot OPT(G, c)$, where $OPT(G, c)$ represents the optimal cost. The induction proof of this result is quite long and complicated. We'll provide a brief overview of its structure here.

The proof is divided into two main cases based on whether the graph G contains a semi-disjoint cycle or not.

1. Case: Graph Contains a Semi-disjoint Cycle: In this scenario, the proof begins by considering the cost $c(F)$ of the feedback vertex set F. It decomposes F into two parts: F'', the feedback vertex set obtained after removing the semi-disjoint cycle, and u_{min}, the minimal vertex in the cycle. The cost of the solution F is as follow:

$$c(F) = c(F'') + c(u_{min}) = c(F'') + c_{min} \leq c''(F'') + c_{min} + c_{min} \leq 2 \cdot c''(OPT(G'', c'') + 2 \cdot c_{min} \leq 2 \cdot c(OPT(G, c)).$$

2. Case: No Semi-disjoint Cycle Exists: In this case, the proof follows a similar structure to Theorem 1, as mentioned earlier. It relies on the properties of the graph and the cost function to bound the cost of the feedback vertex set.

For a formal proof, please refer to [3].

Greedy Algorithm (My Attempt)

After comprehensively studying all the approaches, I wanted to try to solve the minimum-FVS problem by combining the insight gained from other attempts by other people. The goal of this greedy algorithm is to find the Minimum-FVS by removing a vertex such that it appears the most in all cycles. The algorithm makes use of idea to quickly remove the vertex such that it deletes the most cycles. The algorithm also makes use of the idea that minimum-FVS cannot have a vertex v where $deg(v) < 2$.

Algorithm 4 Greedy Algorithm for Minimum Feedback Vertex Set (MFVS)

```
function MostWeightedVertices(cycles)
    occurence ← {}
    for cycle in cycles do
        for vertex in cycle do
            if vertex in occurence then
                occurence[vertex] += 1
            else
                occurence[vertex] ← 1
            end if
        end for
    end for
    return argmax(occurence)
end function

function GreedyFVS(G)
    if V(G) = ∅ then
        return ∅
    end if
    G ← G.copy()
    F ← {}
    Clean(G)
    while not IsAcyclicGraph(G) do
        cycles ← cycle_basis(G)
        v ← MostWeightedVertices(cycles)
        G.remove_node(v)
        F.add(v)
        Clean(G)
    end while
    return F
end function
```

Running time Analysis:

Let V be the number of vertices, E be the number of edges and C be the number of cycles.

At each iteration, the algorithm searches for all cycles in a graph $O(V + E)$. Next, it find the most occurring vertex in all cycles $O(VC)$. It then clean the graph $O(V^2)$. At most, there can be $V - 2$ iteration as the graph we are analyzing is an undirected simple graph. Thus, the running time of the whole algorithm is $O(V^3 + V^2 + VE + V^2C)$ which is approximately $O(V^3)$.

Note: This greedy algorithm does not ensure optimal solution but it does give a good-enough solution.

Example of the result of each Algorithm

In this section, each algorithm will be used on a simple cyclic graph of 15 vertices and will give the output of the following graph.

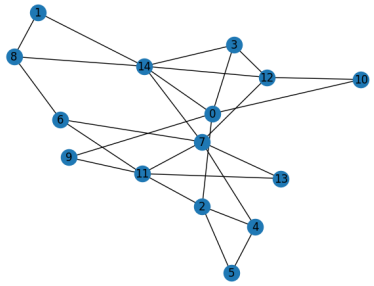

Figure 1: Simple cyclic graph with 15 vertices

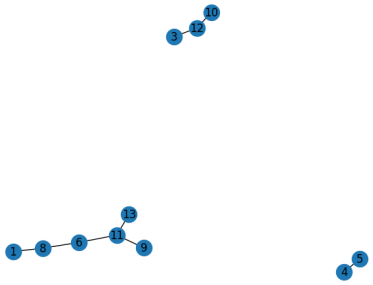

Figure 2: Resulting acyclic graph after running naive algorithm

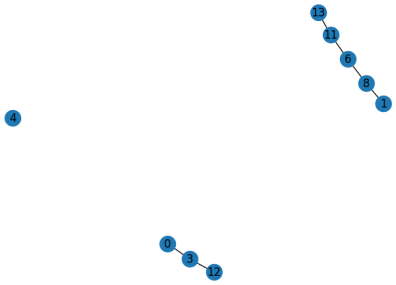

Figure 3: Resulting acyclic graph after running LR-FVS algorithm

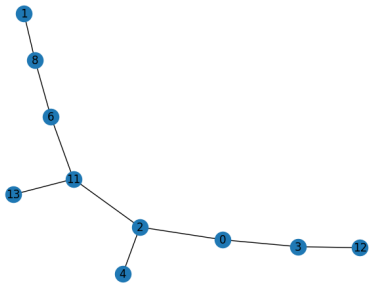

Figure 4: Resulting acyclic graph after running LR-FVS-mod algorithm

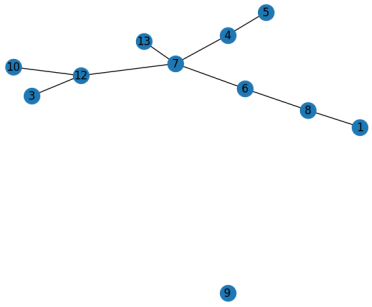

Figure 5: Resulting acyclic graph after running greedy algorithm

The resulting feedback vertex set of each algorithms:

Naive Algorithm: $\{0, 2, 7, 14\}$

LR-FVS Algorithm: $\{2, 5, 7, 9, 10, 14\}$

LR-FVS-mod Algorithm: $\{5, 7, 9, 10, 14\}$

Greedy Algorithm: $\{0, 2, 11, 14\}$

Comparing Algorithms

In this section, we'll compare various algorithms against the optimal solutions obtained from the Naive algorithm. The evaluation measures include the ratio of approximate solution quality to the optimal solution over multiple iterations on random graphs with n vertices. Additionally, we'll analyze the average running time for n vertices on randomly generated graphs. This test will only test for max vertex of $n \leq 20$ due to the fact that optimal solution provided by Naive algorithm would takes 50-100 hours for $n = 30$ as the running time is exponential.

Test 1: 10 vertices, 100,000 random graphs

After conducting 100,000 random graphs with 10 vertices, the results reveal significant differences in algorithm performance.

Algorithm	Average Time(s)	Average Accuracy
Greedy	0.00125	1.40087
LR	0.00123	2.63360
LR-mod	0.00421	2.70026

The Greedy Algorithm consistently demonstrates superior accuracy compared to both LR and LR-mod algorithms. Interestingly, while the running times for the Greedy and LR algorithms remain similar, the LR-mod algorithm exhibits a notably slower execution time, approximately four times longer than the other two algorithms. This test took approximately 45 minutes to conduct.

Test 2: 15 vertices, 1,000 random graphs

Upon conducting on 1,000 random graphs with 15 vertices, the results are as follow:

Algorithm	Average Time(s)	Average Accuracy
Greedy	0.00180	1.18706
LR	0.00190	1.57094
LR-mod	0.00755	1.55641

The result shows that on a larger graph, the accuracy of all algorithms improved significantly, notably the LR and LR-mod algorithms. The differences between the average running times of Greedy and LR remain the same, and the LR-mod algorithm execution time is still approximately four times longer than the other two algorithms. This test took around 36 minutes to conduct.

Test 3: 20 vertices, 100 random graphs

For the following test, it was conducted on 100 randomly generated graph with 20 vertices. The results are as follow:

Algorithm	Average Time(s)	Average Accuracy
Greedy	0.00262	1.12092
LR	0.00288	1.36424
LR-mod	0.02461	1.36905

The average accuracy improves compared to the previous test for all algorithms. The running time for both Greedy and LR algorithms are similar while the LR-mod is now approximately ten times longer than the other two algorithms. This test took around 5 hours and 47 minutes to conduct.

There was an attempt to conduct on 30 vertices in which took around 12 hours with no result. With the current implementation and current CPU (i7-13620H 2.40 GHz), it is expected to take 50-100 hours per graph of 30 vertices to complete.

The conclusion for the three tests is that as the number of vertices increases, the average accuracy of each algorithm also increases. It is also important to note that the LR-mod algorithm's accuracy performance is similar to that of the LR algorithm. The result shown is unexpected since it is expected that the LR-mod algorithm should perform better compared to the LR algorithm in terms of accuracy. This uncertainty might be due to the fact that the LR-mod algorithm is implemented incorrectly. Another important thing that might effect the solution accuracy for both LR and LR-mod is when given a clean graph where all degree within the graph are the same, this will removes every vertex due to how the pseudo code from the reference was written. The original paper does not display how to deal with such case.

References

1. Bafna, V., Berman, P., & Fujito, T. (1999). A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, *12*(3), 289-297.

2. GeeksforGeeks. (2022, September 15). Print all the cycles in an undirected graph. GeeksforGeeks. `https://www.geeksforgeeks.org/print-all-the-cycles-in-an-undirected-graph/`

3. Ravi, R. (2005, October 10). Minimum Feedback Vertex Set. Carnegie Mellon University. `https://www.cs.cmu.edu/afs/cs/academic/class/15854-f05/www/scribe/lec9.pdf`

4. Wikimedia Foundation. (2023, December 11). Feedback vertex set. Wikipedia. `https://en.wikipedia.org/wiki/Feedback_vertex_set`