Research on Minimum Feedback Arc Set Problem

Houyu Lin

University of British Columbia, Vancouver, V6T 1Z4, Canada

Abstract. The feedback arc set of a directed graph is the subset of the graph that consists of at least one edge from every cycle in the graph. Removing the fewest number of these edges gives the minimum feedback arc set. Three methods that attempt to solve this problem use equivalency and NP reductions, an approximate greedy algorithm, and an integer linear program for an exact solution. The minimum FAS can be applied to real world events such as sporting tournaments and ranked voting.

Keywords: minimum FAS, graph theory, set problem.

1. Introduction

1.1. Graph Theory

Graph: V is a set of non-empty elements called vertices, E is the set of edges between any two vertices in V, and each edge in E can be represented as a pair vertex (a, b) in V. Then the set formed by the elements of V and E is called a graph, denoted as G= (V, E), namely: $G = \{(a, b) \mid a \in V, b \in V, ab \in E\}$. The edges of a directed graph are directional, that is, two connected vertices can only lead from one vertex to the other according to the direction of the edges. Otherwise, it is called undirected, in which edges of an undirected graph are directionless. In undirected graphs: an edge (x, y) has the same result as (y, x), In a directed graph: an edge (x, y) does not represent the same result as (y, x). The x and y are called endpoints. A directed edge is also called an arc, where if x is the tail of an arc and y is the head of an arc, then (x, y) represents the arc, and (y, x) represents y as the tail of an arc and x as the head of an arc. (As shown in figure 1)

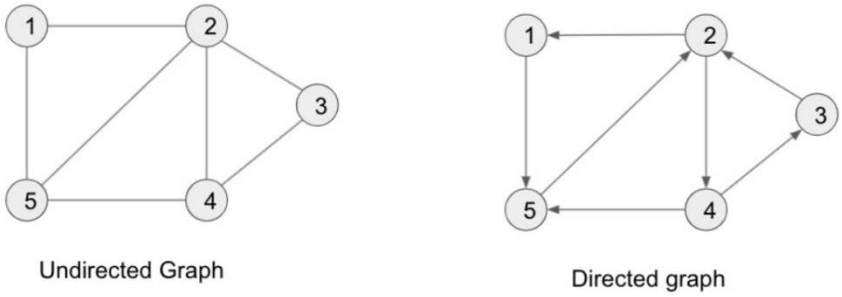


Figure 1. Graph Theory

Cycle: Cycle is an edge path that starts at a node and returns to the node itself.

Acyclic graph: Directed acyclic graphs refer to a directed graph without cycles. For example, a directed graph with cycles has A going from B to C and coming back to A. Changing the direction of the edges from (C, A) to (A, C) produces a directed acyclic graph. (As shown in figure 2)

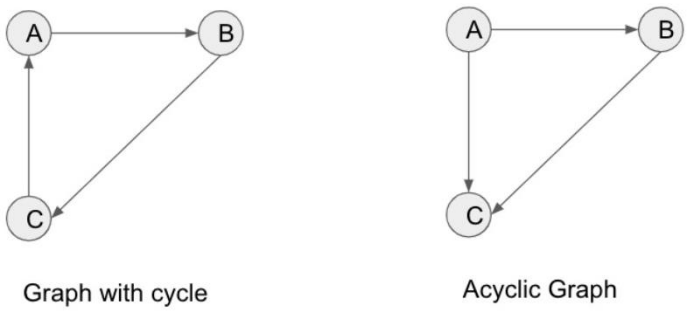


Figure 2. Acyclic graph

Subgraph: If there are two graphs G_1 and G_2, $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$, the following conditions are met: It was a $V_2 \subseteq V_1$ and $E_2 \subseteq E_1$ subset that V_2 was a subset of V_1 and E_2 was a subset of E_1. (As shown in figure 3)

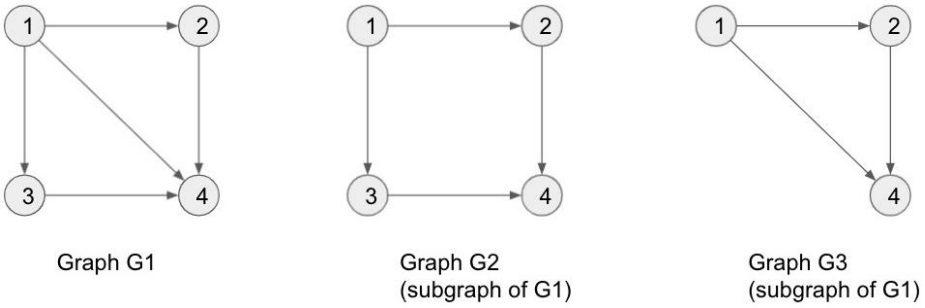


Figure 3. Subgraph

Weighted Graph: Each edge E of the graph corresponds to a real number W(E) (which can be commonly understood as the "length" of the edge, except that the weight of the graph can be negative in the mathematical definition), and we call W(E) the "weight" of E. Such a graph G is called a "weighted graph". (As shown in figure 4)

Figure 4. Weighted Graph

Tournaments: A tournament is a directed graph on n vertices where, for every pair of vertices v_i and v_j we either have an edge from v_i to v_j or we have an edge from v_j to v_i. This gives us a directed graph whose underlying graph is K_n.

1.2. Feedback Arc Set

Feedback Arc Set: The feedback edge set M of the directed graph G = (V, E) is a subset of the edge set E that intersects all the cycles in the graph. Therefore, removing the edges in M will produce an acyclic subgraph of G, and M is the feedback arc set. The optimization problem is the minimum feedback arc set (or minimum feedback edge set) which is to minimize the feedback arc set and produce the maximum acyclic subgraph. The graph also becomes acyclic if the edges in the minimal feedback edge set are reversed rather than removed from the original graph. The minimum feedback arc set is a NP-hard problem. In addition, the minimum weight feedback arc set is also an optimization problem. (As shown in figure 5)

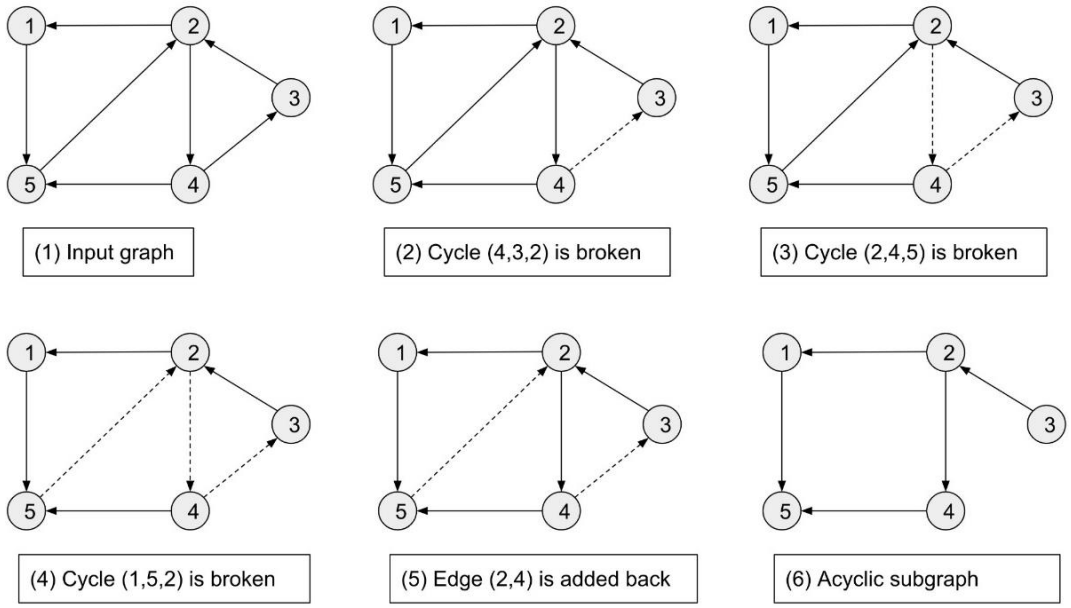


Figure 5. The edges in the feedback arc set are dotted lines in diagram (4). The minimum feedback arc set is the dotted line in diagram (5). The maximum acyclic subgraph is the diagram (6).

Minimum Weight FAS: Given a directed graph with weights, the problem is to find a subset M of the edge set E with the minimum total weight while removing these edges intersects all the cycles in the graph. In the weighted feedback edge set problem, each edge has its corresponding weight. An unweighted graph can be considered a weighted graph with a unit weight for each edge. (As shown in figure 6)

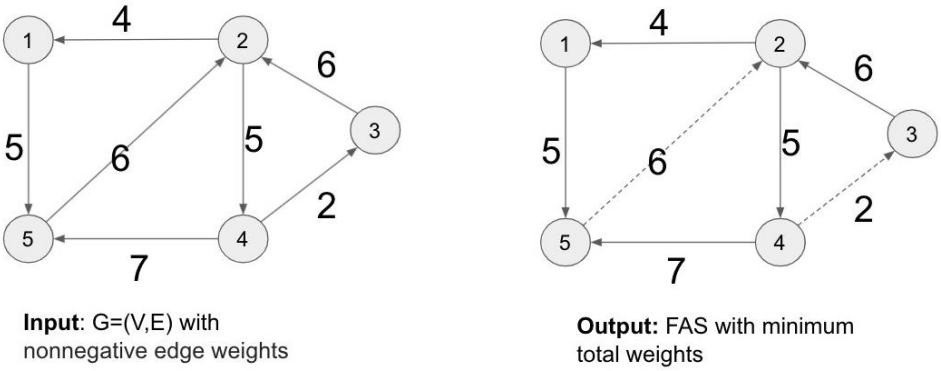


Figure 6. Minimum Weight FAS

2. Concepts

2.1. Linear Ordering

A very useful concept for analyzing minimum feedback arc set problems is using a linear ordering of the vertices of G. This is where we lay out all the vertices of G in a line and, for our purposes, we will represent all edges that go from left to right as forward edges and display them on top, while all edges that go from right to left will be called the backward edges on will be placed graphically on the bottom. An example of such a linear ordering is shown below in figure 7 with blue forward edges and green backward edges. (As shown in figure 7)

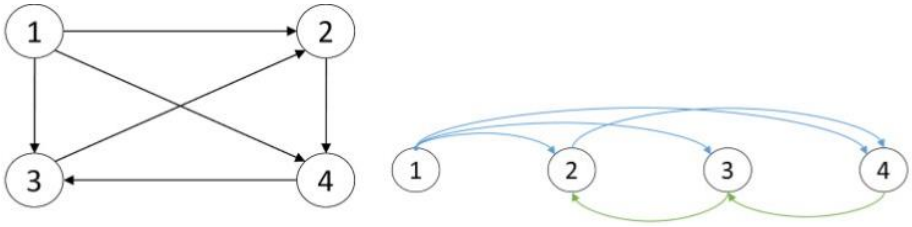


Figure 7. Linear Ordering

It should be somewhat clear that we need at least one edge going forward and one edge going back to create a cycle in such a linear ordering. Thus, removing the set of edges on the bottom of some linear ordering should give us a set of edges that we can remove to make the graph acyclic. Indeed, this is one method to solve the feedback arc set problem. More explicitly, we can try the $n!$ possible linear orderings of the vertices and select the one with the fewest backward arcs or arcs on the bottom, this set will be the minimum arc set. Alternatively, we can check for the smallest set of either backward or forward arcs and eliminate all the orderings of the vertices that are symmetric in reverse order; leaving us with $\frac{n!}{2}$ orderings. While this method may work for graphs on a handful of vertices, it quickly becomes untenable for larger numbers of vertices.

A topological sort is a linear ordering of the vertices of G such that for every edge from v_i to v_j, we have v_i to the left of v_j in the ordering. In other words, it is a special case of a linear ordering of the vertices of G such that every edge is going forward. In our graphical representation of the linear ordering this means every edge is on the top and going forward. We claim that having a topological sort of a graph is equivalent to the graph being acyclic, which is proved by Theorem 1.1.

Linear orderings can also help us see the relationship between our standard problem: the minimal number of edges we need to remove to make the graph acyclic, and an equivalent problem: the minimum number of edges we need to reverse to make the graph acyclic. The full equivalency is proved in Theorem 1.2 but it can be seen that for some linear ordering with the minimum number of backward edges, reversing or deleting these edges would make that ordering a topological sort. (As shown in figure 8)

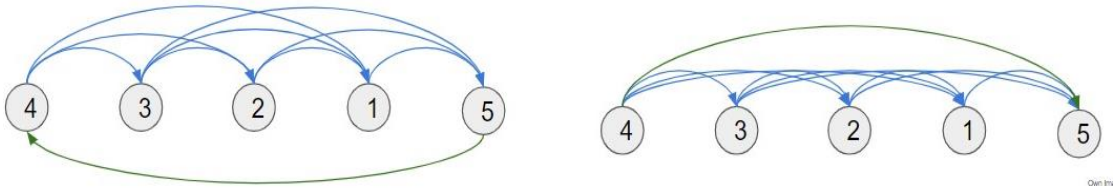


Figure 8. Linear Ordering

In summation linear ordering is a tool that we can use to better analyze directed graphs and their cycles. In a linear ordering we can classify edges as either forward or backward edges and we need at least one of each in order to have a cycle. One way of finding a feedback arc set is to try all the linear orderings and see which one gives us the smallest set of backward edges, and this set can be either removed or reversed to make the graph acyclic. Finally, we see that an ordering with no backwards edges is called a topological sort and represents an acyclic graph.

Theorem 1.1: A graph G has a topological sort if and only if it is acyclic.

Proof:

Case 1: If a graph G has a topological sort, then it is acyclic.

We will use proof by contradiction and assume that there is a cycle in a graph G and there exists at least one topological sort of G. In order for there to be a cycle there must be a path from some vertex v_i that ends back at v_i. Let v_j be the last vertex on this path before v_i. Since there exists a path from v_i to v_j we must have v_i to the left of v_j in the topological sort. Similarly, since we have an edge from v_j to v_i then v_j must be to the right of v_i. This yields a contradiction.

Case 2: If a graph G is acyclic then it has at least one topological sort.

Lemma: Given an acyclic connected graph G, there exists at least one sink.

We take some vertex v_i in G. If v_i is a sink then we are done. If v_i is not a sink then we can construct a path from v_i continuing along some outgoing edge of each vertex. We can also see that this path must end as it has a maximum length $n-1$. Since G is acyclic, we cannot return to the same vertex twice as any such path would contain a cycle. Therefore, the path must end at some v_j that has not been reached and has no outgoing edges left. Thus, it is a sink.

We will use proof by algorithm on an acyclic graph G on *n* vertices to create a topological sort. We will denote S to be the ordered set of vertices.

While $n \geq 0$

Take some vertex v_i

If v_i is a sink remove v_i from G and place it at the beginning of the set S

Else, take construct a path from the vertex v_i that ends at a sink v_j. We then remove v_j and place it at the beginning of the set S.

To prove that this set contains no backward edges we will use induction. In the base case we can see that the last vertex in S v_z cannot have an edge to any other vertices as it is a sink in the original graph. For the induction step we suppose that we have some vertex v_k such that all vertices to the right of v_k have no backwards edges to vertices to the left of v_k. Then v_k can only have forward edges to vertices to the right of v_k since v_k is a sink after these vertices are removed. Thus, there cannot exist any path from v_k to any vertex to the left of v_k.

Theorem 1.2: The minimum feedback arc set by deletion is the equivalent to the minimum feedback arc set by reversal.

Proof:

The minimum feedback arc set by deletion is less than or equal to the minimum feedback arc set by reversal.

Suppose we have a minimum feedback arc set by reversals M. Then the graph G' is acyclic when the set of arcs M is reversed. Because G' is acyclic we can see that since G-M = G'- M we have G-M is acyclic. Thus, the minimum feedback arc set by deletion is less than or equal to $|M|$.

The minimum feedback arc set by reversal is less than or equal to the minimum feedback arc set by deletion.

Suppose we have a minimum feedback arc set by deletion M of a graph G. Then by Theorem 1.1 G - M is an acyclic graph and must have a topological sort S of its vertices. Because the set M is minimum, we can see that adding any edge e_i from M must create a cycle. Therefore, it must be a backward edge if added to S. Thus, if we take M' to be the set of edges in M in reverse, every edge e_i' in M' must be a forward arc in S. Thus S + M' is still an acyclic graph and the set M is also a feedback arc set by reversal. Therefore, the minimum feedback arc set by reversal is less than or equal to $|M|$.

2.2. Vertex Cover Problem

The vertex cover problem involves finding the minimum sized vertex cover. A vertex cover of a graph is the subset of the graph's vertices. Take a graph Q for instance, the vertex cover for T would be a set of vertices. This would hence mean that all the edges in graph T possess at least a single vertex in as an endpoint. From the above statements, it is notable that the application of vertex cover can come in handy while solving a problem in which all the edges in a graph that are related to the problem must be included in the solution. Below are illustrations of vertex cover. (As shown in figure 9)

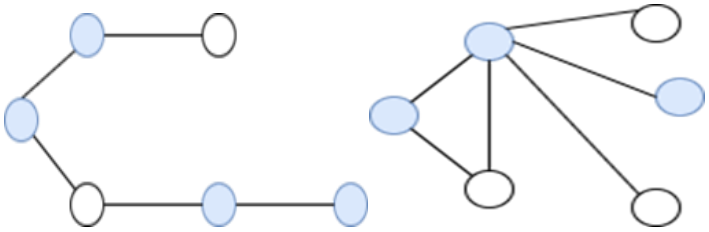


Figure 9. The vertex cover problem

In both graphs, the blue vertices make up the vertex cover of the graphs. In other words, all the blue nodes encompass each edge in the graphs.

Solving the vertex cover problem can be accomplished via binary trees and algorithms. Nonetheless, finding the minimum sized vertex cover can be termed as a NP Complete problem. The NP Complete problems are those that have unknown statuses. Notably, there exist no known polynomial time algorithm for the NP Complete problems unless P = NP. In this case, P represents a group of issues that are solvable in polynomial time using a deterministic Turing machine while the NP represents a set of problems that are solvable in polynomial time using a non-deterministic Turing machine. The NP problems can only be solved through guessing. Thus, approximate polynomial-time algorithms that can solve the problems might exist, though they are yet to be proved.

A binary tree solution for the vertex cover problems follows below. (As shown in figure 10-11)

Input: a binary tree

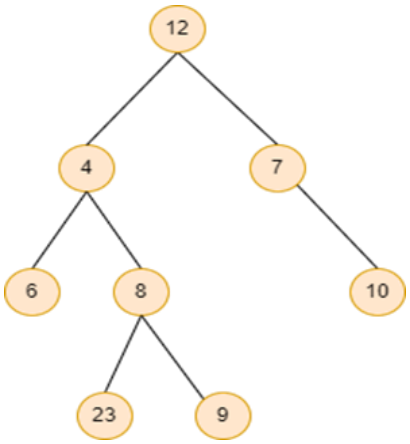


Figure 10. Input

Output: The vertex covers which is 3

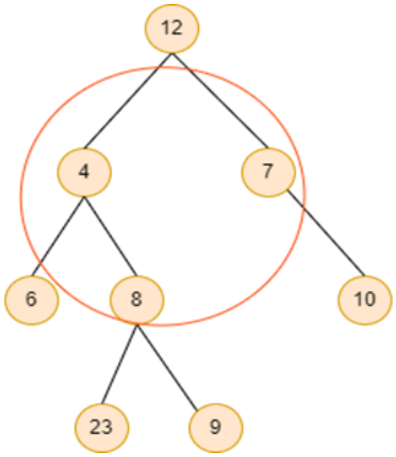


Figure 11. Output

The fore-mentioned algorithms are several. Included below is a pseudocode of among the several algorithms that can guess the vertex cover while applying matching ang greedy algorithms These approximation algorithms often prove to be crucial and faster means of finding the vertex cover.

```
define greedy (B, C)
A = {}
while D not empty:
        select any edge with endpoints (u, c) from D
        add (u, c) to A
        remove all edges incident to u or c from D
return A
```

This algorithm applies to finding the maximal matching in a graph T then adds at least an endpoint of each edge to the covering set of vertices A. The ideal solution is one that contains a vertex from each edge. The best application of the vector cover problem solution would be in the famous travelling salesman problem where a salesman travels to multiple known cities with known distances between them only once before returning to their origin [14].

2.3. Feedback Vertex Set

A feedback vertex set of a graph is a set of vertices that, when removed, leaves said graph non empty and without cycles. In graphs, a cycle refers to trails which are empty where only the first and the last vertices are equal. Each feedback vertex set consists of at least a single vertex a random cycle in the graph. The feedback vertex set number refers to the size of the smallest feedback vertex set.

Take for instance a graph G = (v, e) and a positive integer L, then establish whether a subset where $|X| \leq L$ exists on removing all the vertices of X and the edges next to them from the graph G. In this instance, the graph that remains after removing X is referred to as an induced forest thus finding the minimum feedback vertex set is similar to finding the maximum induced forest. In graphs, a forest is an undirected graph whose any two vertices are joined by at most a single path. Feedback Vertex Sets are mainly applied in operating systems as they are crucial in the study of deadlock recovery [9]. During the wait period in an operating system, all the different cycles work on the deadlocks. To fully resolve the deadlocks, some processes which were lagging get aborted. A minimum feedback vertex set in this situation is the minimum number of processes that are aborted to remedy the situation. Additionally, the feedback vertex set problem is largely applied in Very Large-Scale Integration (VLSI) chip design. Other applications of the feedback vertex set include complexity theory among many others.

3. Equivalent Methods

3.1. The NP-Complete Problem

In the study of computational complexity theory, problems are classified based on the number of resources used, eg. time, space, etc., when solved by an algorithm [12]. One specific instance is the study of decision problems, where a "yes or no" question is asked based on the inputs to the algorithm. Nondeterministic polynomial time, or NP, is a complexity class used to classify and determine runtime of a decision problem. Problems are in NP if the output of the algorithm is "yes" and if there is a certificate that proves this solution can be found in polynomial time. A subset of problems in NP are known as NP-complete problems. To show that a problem is NP-complete, it must be shown that the problem is both in NP and NP-hard.

Theorem*:* The Feedback Arc Set problem is a NP-complete problem.

proof*:* We first claim that the feedback arc set is in NP. In order for this problem to be in NP, we need to transform the optimization problem into a decision problem. In this case, we ask "*can all cycles in a graph G be broken by removing at most k edges?"* and if the answer is yes, we must be able to verify the solution in polynomial time.

For the verification, we construct an algorithm as follows:

1. Select a vertex in the given graph G
2. Traverse all directed edges leaving the vertex

3. Repeat for each vertex

If no cycles are found, then our solution is verified. This algorithm runs in polynomial time. (As shown in figure 12)

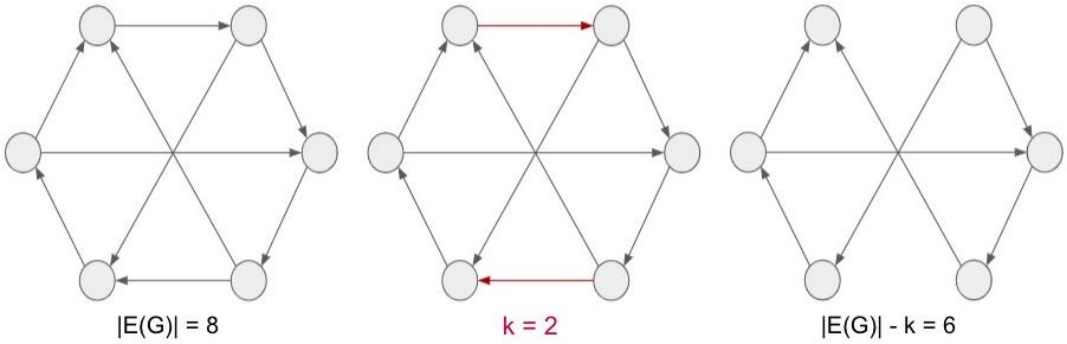


Figure 12. Schematic diagram of proof

The algorithm takes in 2 inputs: a directed graph G and a number k, and asks if after removing k edges, whether the resulting graph is acyclic. In Figure above, we are given a graph with 8 edges and k = 2. When 2 edges are taken out, the subgraph with 6 edges is expected to have had all cycles removed. The proposed acyclic graph is then run through the algorithm and we see that there are indeed no cycles produced from traversing through the vertices. (As shown in figure 13)

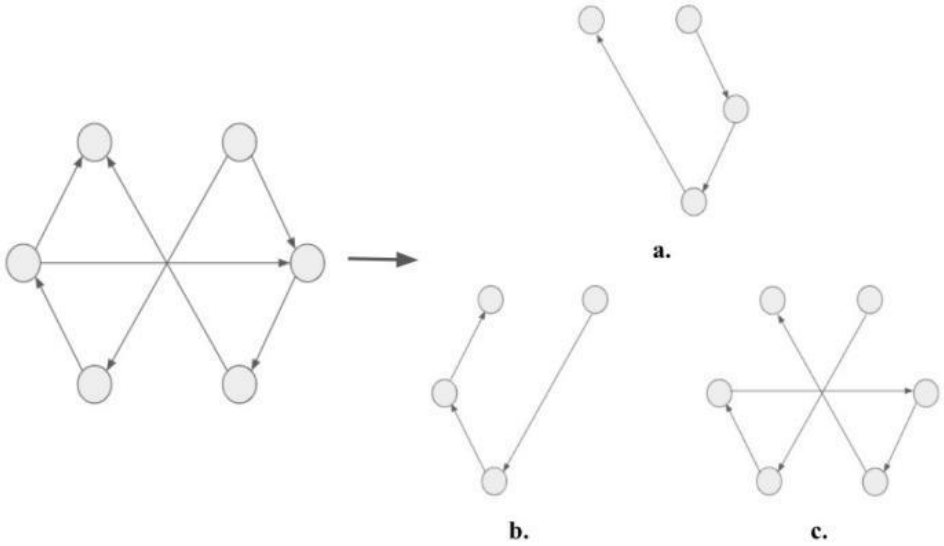
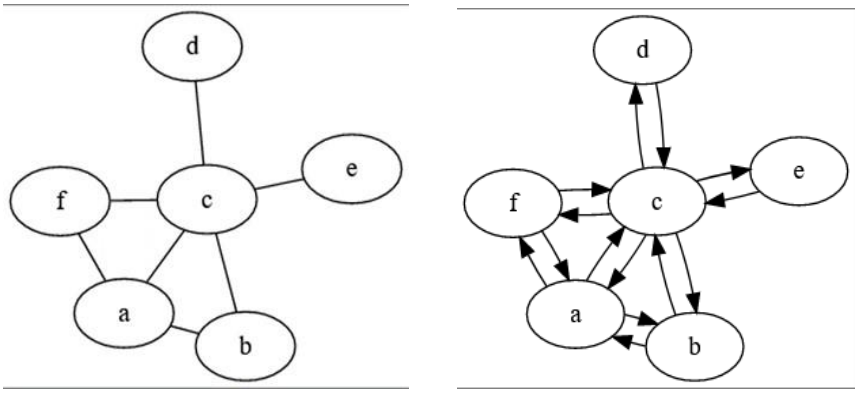


Figure 13. Schematic diagram of proof

The next part of our proof focuses on the NP-hardness of the feedback arc set. This result was proved by Karp and Eugene Lawer [8] in 1972 by showing that the inputs of another NP-hard problem, the Minimum Vertex Cover, can be reduced into equivalent inputs of the Feedback Arc Set decision problem. This transformation, or reduction, is one way to prove that a problem is NP-hard.

More specifically, the minimum vertex cover can be reduced to the minimum feedback vertex set, which is equivalent to our minimum feedback arc set. (As shown in figure 14)

1) Minimum Vertex Cover Problem reduction to Minimum Feedback Vertex Set Problem.



Note: image from https://qph.fs.quoracdn.net/main-qimg-adb9cc89837c7ba89957ef8f7275961f

Figure 14. Minimum Vertex Cover Problem reduction to Minimum Feedback Vertex Set Problem

Consider the following from Karp's work on NP Completeness [5].

Node Cover α Feedback Node Set

$V = N'$

$E = \{<u, v> \mid \{u, v\} \epsilon A'\}$

$K = \mathbb{Q}$

The vertex cover problem involves undirected graphs while the feedback vertex set problems involve directed graphs [5, 7]. In the reduction, an edge (u, v) in the node cover problem transforms to a pair of edges (u, v) and (v, u). For the case of an existing vertex cover of size K in (N′, A′), removal of similar nodes from V would see to it that all edges in (V, E) also are removed so that a feedback vertex set of size k can be created. With every edge (u, v) thus covered, it would mean removal of u or of v. Thus, the remaining graph would lack edges as well as cycles. In the case of their existing feedback vertex set of size K in (V, E), removal of either u or v in all cycles $u \rightarrow v \rightarrow u$ would have to be fulfilled for the purpose of breaking the cycle. This would end up removing similar vertices which would thus derive the vertex cover of size K.

2) Showing that Minimum Feedback vertex set is equal to Minimum Feedback Arc Set.

A feedback arc set in a directed graph refers to a subset of the edges of the graph and consists of at least a single edge in every cycle in the graph [9]. Removal of the aforementioned edges from the graph results in breakage of the cycles, thus forming a direct acyclic graph, and is an acyclic subgraph of the original graph. The minimum feedback arc set is that which has the fewest possible edges. Removal of the minimum feedback arc sets results in a maximum acyclic subgraph.

Previously in this report, the feedback vertex set was discussed. The feedback vertex set and the feedback arc set are similar. Reduction of either would result in deletion of cycles, similar to the feedback arc set, when a set of these vertex subsets is removed from a graph. Take a directed line graph for instance, the graph would have a vertex for every edge of the graph as well as an edge for every two edge paths in the graph.

To achieve the minimum feedback vertex set of a given graph G, the minimum feedback arc set of the graph can also be reached. This can be made possible by dividing all the vertices of the graph G into two where one would thus be for the incoming edges while the other one would be for the outgoing edge. This splitting of the vertex would then allow for precise algorithms to solve for both feedback arc set, and feedback vertex set, as well as be converted to each other while utilizing appropriate translations of complexity bounds. The minimum feedback vertex set can thus be achieved using the minimum feedback arc set as they are similar. As discussed previously, an edge in the node cover problem can be reduced to a feedback vertex set. Thus, a minimum vertex cover is equivalent to a minimum feedback arc set.

4. An Approximation Method

For a quick and simple approximation method of the feedback arc set problem there exists a greedy algorithm developed by Eades, Lin and Smyth [6]. This algorithm works by attempting to find a sequence of vertices that represents the optimal ordering of the vertices of G such that the set of backward arcs or back edges is as close to the minimum feedback vertex set as possible. This is done by removing the vertices one by one from the graph, and placing them into one of two sets based on their relative out degree - in degree, prioritizing vertices that are either sinks or sources. This aims to give a decent ordering given that, in order to minimize the number of backward arcs, vertices with a relatively larger outdegree should be placed further to the left, while vertices with a relatively larger indegree should be placed further to the right. The explicit formulation of this procedure is as follows:

Given simple directed graph G, we create two ordered sets s_1 and s_2 that we will place the vertices in. We will define S to be the concatenation of these sets of vertices and let R(S) denote the set of backward arcs given by an ordering S. We will also define the outdegree, or number of edges directed out from a vertex v_i, as $d^+(v_i)$ while the indegree, or number of edges directed into a vertex v_i, will be represented by $d^-(v_i)$

While $G \neq \emptyset$

If G contains a sink $v*$ we remove $v*$ from G and add it at the front of the set s_2

If G contains a source $v*$ we remove $v*$ from G and add it to the end of the set s_1

Else, take the vertex $v*$ for which $d^+(v*) - d^-(v*)$ is largest, remove v_i from G and it to the end of the set s_1.

We then reconstruct G with the the vertex order given by S where $S \leftarrow \{s_1, s_2\}$.

As shown in figure 15, Here is a demonstration of the above algorithm on a graph G_6

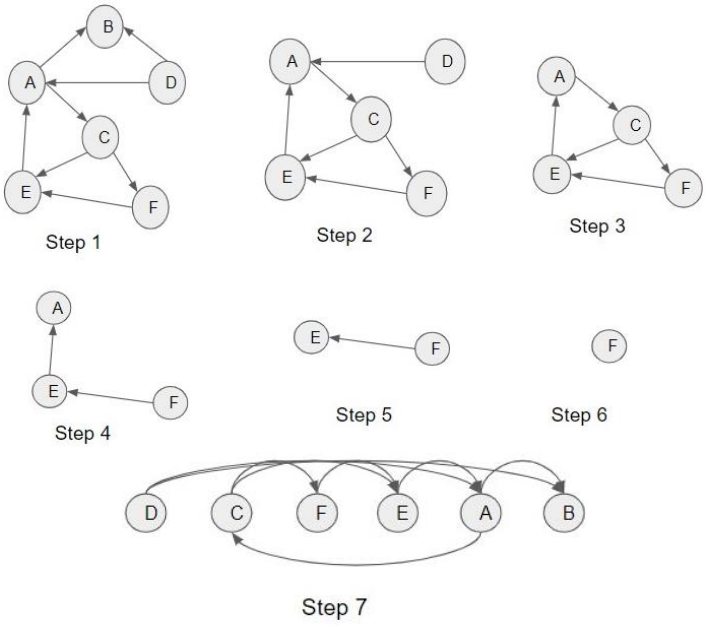


Figure 15. A demonstration of the above algorithm on a graph G_6

Step 1: We remove the sink B	s1{} s2{B}
Step 2: We remove the source D	s1{D} s2{B}
Step 3: We remove C with the largest $d^+(v*) - d^-(v*)$	s1{D, C} s2{B}
Step 4: We remove the sink A	s1{D, C} s2{A, B}
Step 5: We remove the sink E	s1{D, C} s2{E, A, B}
Step 6: We remove the sink F	s1{D, C} s 2{F, E, A, B}
Step 7: We concatenate the two sets to give us	S{D, C, F, E, A, B}

The authors of [6] show that the vertex set this algorithm gives us is bounded by $R(S) \leq \frac{m}{2} - \frac{n}{6}$. They also give evidence that this algorithm runs faster than many other proposed algorithms being strictly linear. For sparse graphs this method tends to give better outcomes than similar algorithms, while coming up short for more dense graphs. Overall, this method's quick computation makes it effective in application for problems with a very large number of vertices, while its maximum bound ensures that the set given will be strictly better than an average ordering.

5. Exact Methods

5.1. An Integer Linear Program [9]

Overall, the feedback arc set is a NP-hard problem. However, when the maximum weight of the specified feedback arc set is k, the problem becomes an NP-complete problem. To deal with this kind of problem, it is necessary to introduce a concept called reducibility. Precisely, NP problem A can be reducible to NP problem B that refers to the method of solution of B to solve with A. In other words, problem A can transform into problem B by a polynomial time algorithm and a NP-complete problem is reducible to any NP problem.

There are many heuristic methods to solve the problem. A heuristic method is aimed at model solving. It is a method to approach the optimal solution step by step. This method makes repeated judgment and practical correction on the obtained solution until it is satisfactory. The model of the heuristic method is quite simple, and there are a small number of schemes which need to be considered so that it is easy to find the final answer. Although it is hard for this method to get the optimal solution, as long as it is handled properly, it can still obtain the approximate optimal solution satisfactory to the decision-maker. In practice, the difference between the solution found by a heuristic method and the optimal solution can be as large as O(n).

First, this method is a greedy algorithm. It is aimed to remove an edge which can eliminate cycles without duplicate nodes and edges. The complexity of this method is O (1 + log d), where D is the maximum cardinality in the subset. The disadvantage of this problem is that even in sparse graphs, there may be exponential cycles, and the commonness of cumulative errors of greedy algorithms is difficult to avoid.

For another heuristic methods, sorting heuristics, we first need to give a node G with a random order π*, then we can classify all the edges, dividing them into forward and backward edges. Next, we pick the backward edge as a feedback arc set. Sorting heuristic transforms the minimum feedback arc set problem into a sorting problem to find the least backward edges. However, some research shows that the algorithm performs poorly in some cases. Although the heuristic algorithm can solve the above problems, it cannot get the optimal solution or even the approximate optimal solution, so we need a more accurate algorithm.

When the number of edges is specified, the problem is NP-complete, so we have three exact algorithms that can be used, namely dynamic programming, branch and bound, and integer programming. Here we focus on integer programming:

$$\min_{y} \sum_{j=1}^{n} \left(\sum_{k=1}^{j-1} c_{k,j} y_{k,j} + \sum_{l}^{n} c_{l,j} (1 - y_{j,l}) \right) \tag{1}$$

$$y_{i,j} + y_{j,k} - y_{i,k} \leq 1, \qquad 1 \leq i < j < k \leq n \tag{2}$$

$$-y_{i,j} - y_{j,k} + y_{i,k} \leq 0, \qquad 1 \leq i < j < k \leq n \tag{3}$$

$$y_{i,j} = \{0, 1\} \qquad 1 \leq i < j < k \leq n \tag{4}$$

Note: image from [1]

The way we get this LP problem is by trying to find this optimal ordering. c_{ij} = 1 if there is an edge in the graph from i to j. and it is equal to 0 if there are no edges from i to j. y_{ij}= 0 if i is in front of j in the ordering. y_{ij} = 1 if i is after j. So, the minimum z for all c_{ij}*y_{ij} is

used to minimize the number of edges going forward in the ordering of the graph. As for the constraints, it ensures whatever the selection is a proper ordering of the graph According to (2), it basically says if i is before j and j is before k, then i must be before k. Similarly, for (3), it means if i come before k then either i must come before j or j must come before k. In the above constraints any y should satisfy the triangle inequalities which states that the sum of two sides of the triangle is greater than the third side.

There is an alternative way known as the minimum set cover where m is the number of arcs and w_j is a non-negative weight. If j is in the feedback set, y_j = 1, otherwise it is 0. If j is in the cycle i, then a_{ij} is equal to 1, otherwise it is 0, and l denotes the number of simple cycles.

$$\min_{y} \sum_{j=1}^{m} w_j y_j$$

$$s.t. \sum_{j=1}^{m} a_{ij} y_i \geq 1 \; for \; each \; i = 1, 2, \ldots, l$$

$$y = \{0, 1\}$$

Note: image from [1]

Before solving, the amount of calculation can be reduced by preprocessing and reducing matrix rows and columns. The disadvantage of this formula is that when the number of cycles is large, the calculation will become complex.

6. Computational Results

Consider an example that we have seen previously. Note that this sorting is not necessarily optimal. (As shown in figure 16-20)

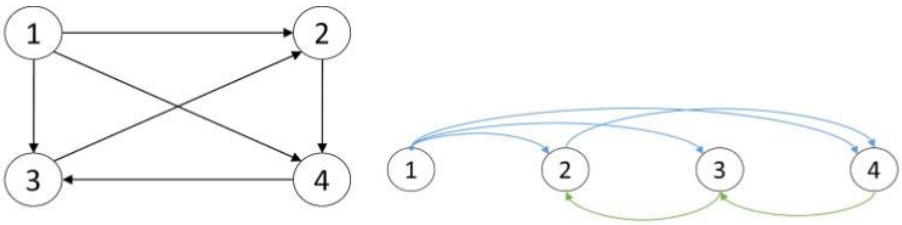


Figure 16. The schematic diagram

Using the integer linear programming formulation as a minimum set cover, we see:
$m = 6$ since there are 6 arcs
$w_j = 1 \; for \; j \in \{1, 2, \ldots, 6\}$ since we have an unweighted graph
$l = 1 \; so \; i = \{1\}$ as there is only 1 simple cycle between 2→4→3
Then the ILP can be written as follows:

$$minimize \sum_{j=1}^{6} y_j$$

$$s.t. \sum_{j=1}^{6} a_{1j} y_1 \geq 1$$

$$y \in 0, 1$$

We label the arcs such that

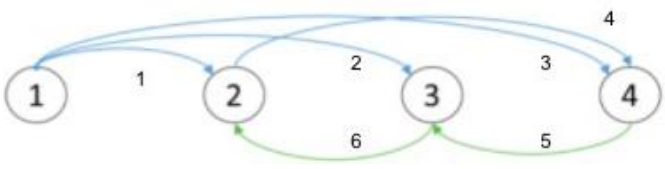


Figure 17. The schematic diagram

Then we see the arcs involved in the cycle are 4, 5, and 6 so $a_{11} = a_{12} = a_{13} = 0$ and $a_{14} = a_{15} = a_{16} = 1$.

These inputs can then be coded such that:

```
var y1 >= 0, <= 1;
var y2 >= 0, <= 1;
var y3 >=0, <=1;
var y4 >=0, <=1;
var y5 >=0, <=1;
var y6 >=0, <=1;

minimize z:     y1  +y2 +y3 +y4 +y5 +y6;

subject to c11:   y4 +   y5 +   y6 >=  1;

end;
```

Figure 18. The inputs

With the output given as

Variable	Value	Value bounds	Status
y1	0	[0, 1]	At lower bound
y2	0	[0, 1]	At lower bound
y3	0	[0, 1]	At lower bound
y4	0	[0, 1]	At lower bound
y5	0	[0, 1]	At lower bound
y6	1	[0, 1]	Basic

Figure 19. The output

Note: program used from: https://online-optimizer.appspot.com/?model=builtin:default.mod

This shows $y_6 = 1$ with all other $y = 0$. In other words, arc 6 is in the feedback arc set, and its removal will give us a directed acyclic graph. We can also see that this is optimal by performing a linear ordering of the graph. If we order the above example by switching the vertices 3 and 4

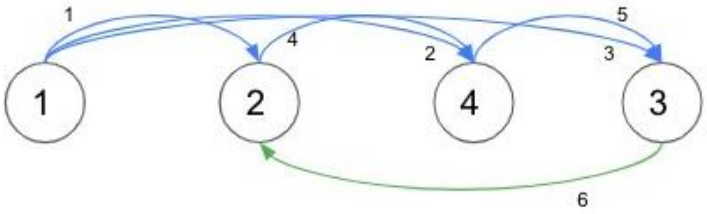


Figure 20. The schematic diagram

then only removing arc 6 gives us a topological sort and also removes the minimum number of backwards edges which gives the same result as our program.

7. Applications

7.1. Sporting Events

In application, the most common uses for the minimum feedback arc set are as a ranking system. In the example of a round robin for a sports tournament, we can create a tournament graph where directed edges from team i to team j represent team i beating tema j in a game. For an example Graph we will consider a tournament between Red Blue Green and Purple with the following

results. Red beats Green, Blue and Purple. Then Green beats Blue and Purple, with the final game having purple beat Blue. (As shown in figure 21)

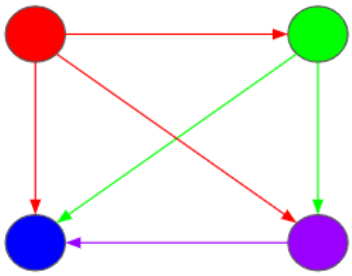


Figure 21. A tournament graph

In this tournament the clear winner is Red as they won 3 games, followed by Green who won two in second place with Purple in third and Blue in last place. We can then create a linear ordering of these vertices ranked from best to worst and display their edges. (As shown in figure 22)

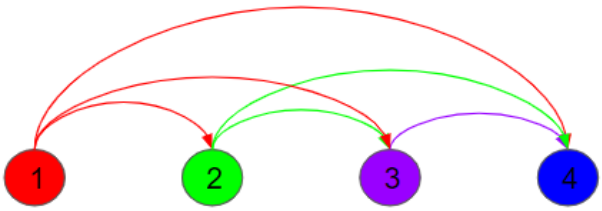


Figure 22. A tournament graph

This type of graph is the ideal state of a ranking system as every higher ranked team beats the lower ranked team in every match up, however this is not always the case. We will define an upset as a higher ranked team losing to a lower ranked team. We can see in the above graph that such an edge would be a backwards edge and create a cycle. When teams are relatively evenly matched, we often have many cycles and it can become hard to determine who should be ranked over whom. This is where the feedback arc set comes in, as the topological sort we create when we find the maximum acyclic graph provides us with a ranking of the teams with the fewest number of games being considered upsets. This ranking system is disadvantaged in perhaps being difficult to understand for sports fans, however an unbiased ranking system could be a useful tool for analysts or sports gamblers.

7.2. Ranked Voting

Ranked choice voting, or preferential voting, is a system where voters rank their choices in a sequential order [13]. People pick their first choice, then second, third, and so on. The Kemeny-Young method in ranked voting is NP-hard [3], and uses the weighted feedback arc set tournament and pairwise comparisons to determine the winner. The paper by Karpinski and Schudy [4] formulates this scenario into a minimum weight feedback arc set problem by setting the vertices as the choices, letting the direction of the edges point towards the winner of the pairwise competition, and having the edge weight as the number of upsets if the loser of the pair were given a higher ranking. The vertices of this graph can then be linearly ordered and the goal is to find the minimum total weight of the backward arcs. In other words, the Kemeny-Young method is said to minimize the number of voters who prefer the opposite direction between each pair [10]. (As shown in figure 23)

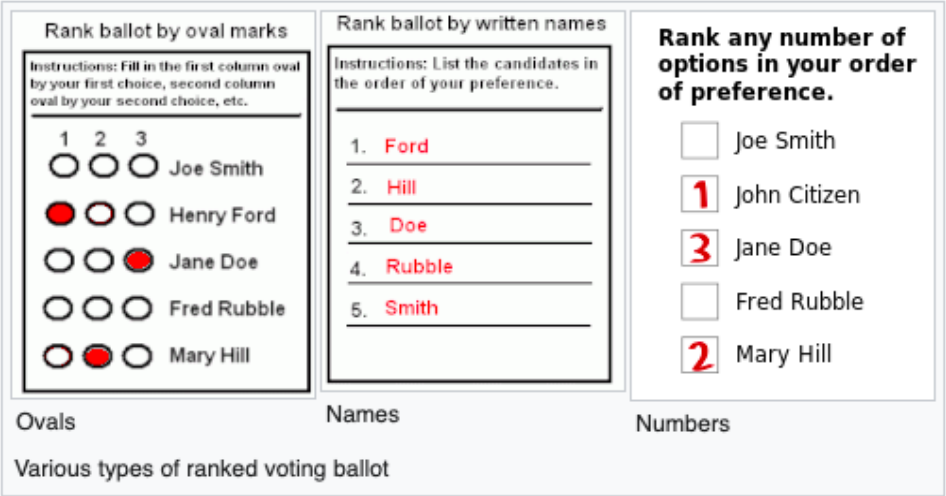


Figure 23. Ranked Voting

Note: image from [13]

8. Conclusion

In this paper, the concepts, thoughts and computing methods of minimum feedback arc set are introduced. As a typical optimization problem, it has corresponding applications in so many fields, such as circuit analysis, competition, and graph drawing. For its algorithms, it is bound to mention heuristic methods is better than exact algorithms in most cases due to computational difficulty.

References

[1] Ali Baharev, Hermann Schichl, Arnold Neumaier, Tobias Achterberg, An Exact Method for the Minimum Feedback Arc Set Problem, *ACM J. Exp.*

Algorithmics, vol. 26 (2021), no. 1.4, pp. 1 - 28, DOI 10.1145/3446429.

[2] Benno Schwikowski, Ewald Spunkmeyer, On Enumerating All Minimal Solutions of Feedback Problems, *Discrete Applied Mathematics*, vol. 117 (2002), no. 1-3, pp. 253 - 265, DOI 10.1016/S0166 - 218X (00) 00339 - 5.

[3] John G. Kemeny, Mathematics Without Numbers, *Daedalus*, vol. 88 (1959), no. 4, pp. 577 - 591.

[4] Marek Karpinski, Warren Schudy, Faster Algorithms for Feedback Arc Set Tournament, Kemeny Rank Aggregation and Betweenness Tournament, *ISAAC*, vol. 6506 (2010), pp. 3 - 14, DOI 10.1007/978 - 3 - 642 - 17517 - 6_3.

[5] Mark Gritter. How would a reduction from Vertex-cover to Feedback Vertex Set work. https://www.quora.com/How-would-a-reduction-from-Vertex-cover-to-Feedback- Vertex-Set-work. (updated 2020).

[6] Peter Eades, Xuemin Lin, W. F. Smyth, A Fast and Effective Heuristic for the Feedback Arc Set Problem, *Information Processing Letters.* vol. 47 (1993), no. 6, pp. 319 - 323, DOI 10.1016/0020-0190 (93) 90079 - O.

[7] R. Ravi. Approximation Algorithms. https://www.cs.cmu.edu/afs/cs/academic/class/15854-f05/www/scribe/lec9.pdf. (updated October 10, 2005).

[8] Richard M. Karp, Reducibility Among Combinatorial Problems, *The Journal of Symbolic* Logic, vol. 40 (1975), no. 4, pp. 85 - 103, DOI 10.2307/2271828.

[9] Wikipedia. Feedback Vertex Set https://en.wikipedia.org/wiki/Feedback_vertex_set (updated 13 July 2021).

[10] Wikipedia. Feedback Arc Set. https://en.wikipedia.org/wiki/Feedback_arc_set (updated November 17, 2021).

[11] Wikipedia. Graph (Discrete Mathematics). https://en.wikipedia.org/wiki/Graph_(discrete_mathematics) (updated October 10, 2021).

[12] Wikipedia. NP (Complexity). https://en.wikipedia.org/wiki/NP_(complexity) (updated September 29, 2021).

[13] Wikipedia. Ranked Voting. https://en.wikipedia.org/wiki/Ranked_voting (updated November 18, 2021).

[14] Wikipedia. Vertex Cover. https://en.wikipedia.org/wiki/Vertex_cover (updated 11 November 2021).