Finding Spanning Trees in Strongly Connected Graphs with Per-Vertex Degree Constraints

Samuel Chase

2018 June

Written under the supervision of Dr. Theresa Migler-VonDollen

California Polytechnic State University, San Luis Obispo

Department of Computer Science and Software Engineering

Contents

1 Introduction **2**

2 Fast vertex cover with double star graphs
Based on "Covering by trees of small diameter" by Lovász [1] **2**
2.1 Algorithm Overview 3
2.2 Example 3

3 Determining subgraphs of maximum edges on degree constrained graphs
Based on "Another look the the degree constrained subgraph problem" by Yossi Shiloach [2] **4**
3.1 Algorithm Overview 4
3.2 Example 5

4 Spanning trees in connected graphs with per-vertex degree constraints **7**
4.1 Algorithm Overview 7
4.2 Proof of Correctness 8

5 Conclusion **9**

6 References **10**

1 Introduction

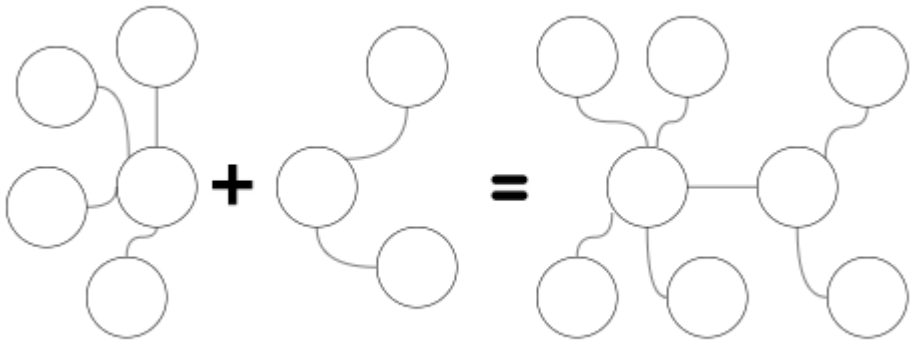
In this project, I sought to develop and prove new algorithms to create spanning trees on general graphs with per-vertex degree constraints. This means that each vertex in the graph would have some additional value, a degree constraint d. For a spanning tree to be *correct*, every vertex v_i in the spanning tree must have a degree exactly equal to a vertex constraint d_i. This poses an additional constraint on what would otherwise be a trivial spanning tree problem. In this paper, two proofs related to my studies will be discussed and analyzed, leading to my algorithm for determining a spanning tree on strongly connected graphs. It is my hope that in the future this algorithm can be modified to apply to the general case.

2 Fast vertex cover with double star graphs

Based on "Covering by trees of small diameter" by Lovász [1]

In his paper "Covering by trees of small diameter," Lovász discusses double star graphs. A graph's diameter is the maximum number of edges to connect any two vertices in the graph. Double star graphs are graphs of diameter less than or equal to 3. A double star can be formed by adding an edge between the central nodes of any two star graphs, graphs featuring a central node with any number of edges connected only to that central node, as seen in Figure 1.

Figure 1:



Double stars also have the interesting property of being able to cover graphs incredibly quickly. In fact, a graph of n vertices can be covered by a greedy algorithm taking at most $2n/3$ steps. All that is required to cover a graph with

double stars is to find a maximal matching of edges, a set of edges E such that no edge can be added without touching a vertex already touched by E. Then, add double stars centered around those edges, where the two vertices connected to the edge are individual single stars. Lovász has proved that at most $2n/3$ double stars are required to cover a graph. This paper will discuss the key determinants in a maximal double-star cover, as well as provide an example of such covers on simple graphs.

2.1 Algorithm Overview

Lovász begins by asserting that τ is the maximal number of independent vertices of G, a maximal set of vertices such that no two share an edge. He also states that M is the maximal matching of G, having size r. I is defined as the set of vertices which are not endpoints of any edges in M, with a size of s. If $n = 2r$ then $I = \emptyset$.

In section 2, it is stated that $f_3(g)$ is the maximum number of double stars required to cover a graph. We also know $f_3(g) \leq f_2(g)$, as even in a worst case scenario it is still impossible that double star graphs cover less efficiently than single stars. This leads Lovász to know that $n - \tau \leq n - s = 2r$, or the number of vertices minus independent vertices is less than or equal to the vertices minus the vertices contained in I, which is equal to the number of vertices in M.

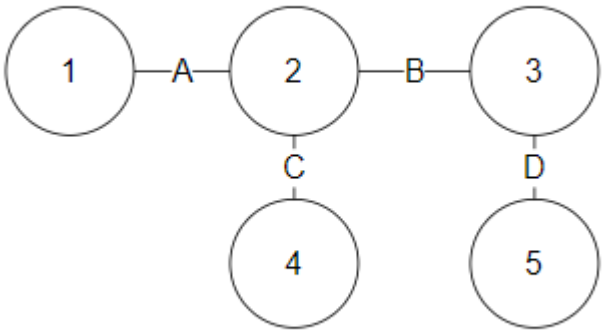
In section 3, Lovász moves on to look at S_i. For each edge e_i in M, there exists S_i, a double star centered around e_i. S_i contains the two vertices at the endpoints of e_i, and all vertices adjacent to those two vertices. T_i is the tree of all edges containing c_i (from I). We know that the set of all S and T covers G. This means $f_3(g) \leq r + s = n - r$, or the maximum number of double stars required to cover a graph is less than or equal to the size of M plus the size of I. This is equal to the size of G minus the size of M.

Combining what is learned in section 4 we are able to prove Lovász's theorem, that $f_3(g) \leq 2n/3$.

2.2 Example

Presented in Figure 2 is an example graph. It may be noted that this graph itself is a double star. This reveals some of the inefficiency with this algorithm - even when a small cover (potentially even one of size 1) is available, a larger cover may be selected simply because of the nature of the greedy algorithm. It is possible for this algorithm to return multiple valid covers, depending on the sequence of selected edges.

Figure 2:



In Figure 2 we know that n is equal to 5. Running our greedy algorithm to find a maximal matching, assume we select edges c and d. This means $M = \{c, d\}$, and therefore $r = 2$. I will be all vertices not touched by M, therefore $I = \{1\}$ and $s = 1$. Knowing that $f_3(g) \leq n - r$, we can say that the maximum number of double stars necessary to cover this graph is 3. An example of this maximum cover is a cover containing double stars centered around a, b, and c. Assuming we make the odd choice of selecting a and c to be the centers of our first two double stars, we notice that if we add either b or d we will have a full cover. Although this is something our algorithm would never provide as a M (as our algorithm only seeks a maximal matching, therefore both a and c will never be selected), this does show a theoretical maximum size for the vertex cover. It is simply impossible to increase the maximum size, knowing that no matter which edges we choose we will cover the graph in at most 3 double stars.

3 Determining subgraphs of maximum edges on degree constrained graphs

Based on "Another look the the degree constrained subgraph problem" by Yossi Shiloach [2]

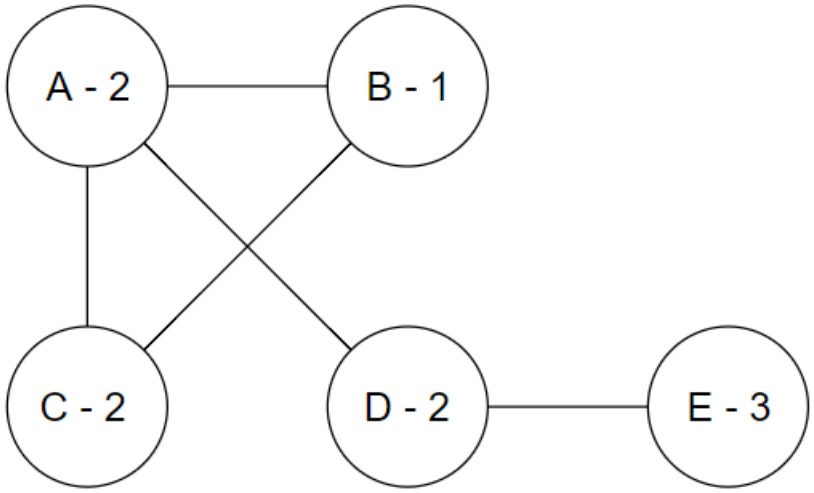
3.1 Algorithm Overview

In this paper, Yossi Shiloach details an algorithm for determining subgraphs of maximum edges on a degree constrained graph $G(V, E)$. He defines degree constraints as a list of numbers representing the maximum degree of a vertex. However, rather than determining a spanning tree, he wishes to find an edge set,

E', such that $G'(V, E')$ adheres to degree constraints, while maximizing $|E'|$. The method for generating E' will very likely prove useful in generalizing our own algorithm.

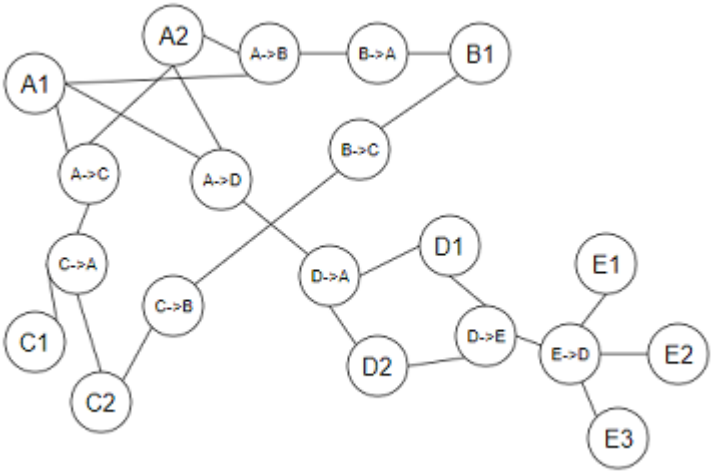
3.2 Example

Figure 3:



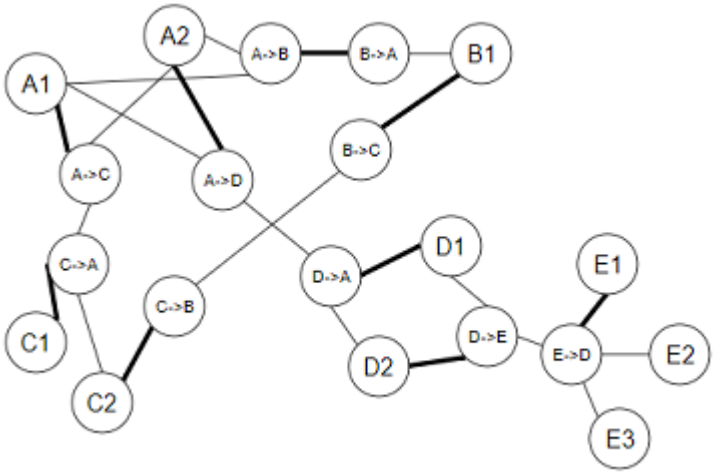
Observed in Figure 3 is a graph, with vertex labels and maximum degrees. In order to determine the maximum subgraph $G'(V, E')$ we must create a graph H. This is done by taking every vertex, and replicating it into subvertices. A degree constraint of two produces two subvertices, a constraint of three produces three subvertices, and so on. Then, connection vertices are created, representing each edge in G. Therefore, as depicted in Figure 4, there will be a vertex created for $A \rightarrow B$ and a vertex for $B \rightarrow A$. $A \rightarrow B$ will connect to every subvertex of A, while $B \rightarrow A$ will connect to every subvertex of B. Then $A \rightarrow B$ will be connected to $B \rightarrow A$. This will be done for every edge in G, to yield a new graph H.

Figure 4:



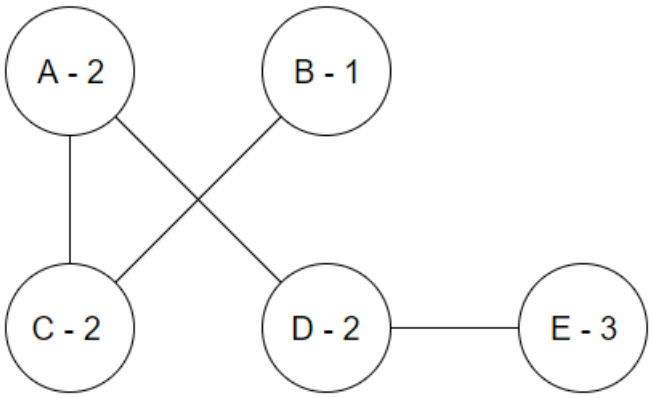
Following this, a maximum matching must be drawn on H, such as the one depicted in Figure 5. Bolded edges are in the matching.

Figure 5:



Then, one must look at the edges between the unlabeled connector vertices. An edge will be in E', unless the edge between the connectors is in the matching, as in the $A \leftrightarrow B$ connector. G' is shown below in Figure 6. It can be verified that G' does not violate any degree constraints and contains a maximum valid edge set E'.

Figure 6:



One might notice that on certain graphs there are more than one maximum matching, which may yield different G's. This is expected, as in many cases there are multiple G's, both with maximum $|E'|$. In these cases all possible G's returned by the algorithm are valid.

4 Spanning trees in connected graphs with per-vertex degree constraints

4.1 Algorithm Overview

Assume that we wish to create a spanning tree for a graph. Correct algorithms for solving this problem have been proven by Kruskal and other mathematicians, however, in this case, we wish to consider per-vertex degree constraints. Assume that in the graph every vertex, v_i, has some degree constraint d_i, where for a spanning tree to be valid v_i must be equal to d_i for all $v_i \in V$.

An algorithm for solving this problem can be seen below:

Algorithm 1 Creating a spanning tree on a strongly connected graph with per-vertex degree constraints

1: Graph $G(V, E)$ with a total degree of $2(|V| - 1)$
2: T $= \emptyset$
3: T += ($v1$,$v2$) where $v1$ is the vertex of largest degree in V, and $v2$ is the vertex of second largest degree in V. Decrement the degree of $v1$ and $v2$
4: **while** There exits a vertex in V and T with a degree > 0 **do**
5: T+= (t,v) where t is the vertex of largest degree in t, and v is the largest vertex in $V \backslash T$
6: **end while**

4.2 Proof of Correctness

Theorem 1. *Given a strongly connected graph $G(E, V)$ with total vertex degree constraint of $2(|V| - 1)$ algorithm 1 will return a correct spanning tree.*

Proof. Allow the algorithm to run through step 3, producing tree T. G will have a total degree constraint of $2(|V| - 1) - 2$, accounting for the single edge in T. Therefore, G has a total degree constraint of $2(|V| - 2)$, and needs to connect $|V| - 2$ vertices to T. Thus it will be possible to complete the spanning tree, as the total degree constraint of G is equal to twice the number of vertices not in T, and T will either have a total degree constraint ≥ 1, or be G.

Now, assume T has grown to size k vertices correctly, where T has a total degree constraint large enough to complete the spanning of the graph. T will now select the next untouched vertex of maximum degree. There are 3 cases for this vertex:

Case 1) The next vertex selected has a degree = 1

In this case, if the largest vertex remaining has a degree constraint of 1, then all vertices not in T, called U, must have a degree constraint of 1. Thus, T must have a total remaining vertex constraint of $|V \setminus U|$, which is equal to $|U|$. Therefore, as there are $|U|$ vertices not in T and T has a total degree constraint of $|U|$, we know the degree constraint will not be violated, as each connection from T to U lowers the total degree constraint by one. Thus, T will be able to complete the spanning tree.

Case 2) The next vertex selected has a degree > 1

Adding this vertex to T will increase T's total degree constraint by some value > 1, and then decrease T's total degree constraint by 2. Therefore the change in T cannot be less than zero, and T will not be decreased, so

T will be able to complete the spanning tree.

Case 3) No new vertex can be selected

If no new vertex can be selected this means every vertex in G is in T. Thus the spanning tree will be completed.

Therefore it is clear that T will not violate degree constraints at every step of the algorithm, and the algorithm will proceed to completion.

□

5 Conclusion

The goal of this project was to determine an algorithm to produce spanning trees on graphs with per-vertex degree constraints. I was able to produce an algorithm and proof which work on strongly connected graphs. I hope that perhaps future researchers might modify my algorithm, and potentially utilize the techniques of Yossi Shiloach, to solve the general case.

References

[1] Lovász, L. (1968). On covering of graphs (P. Erdos & G. Katona, Eds.). Theory of Graphs, 234-235.

[2] Shiloach, Y. (1981). Another look at the degree constrained subgraph problem. Inf. Process. Lett., 12, 89-90.