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

Samuel Chase

2018 June

1 Introduction

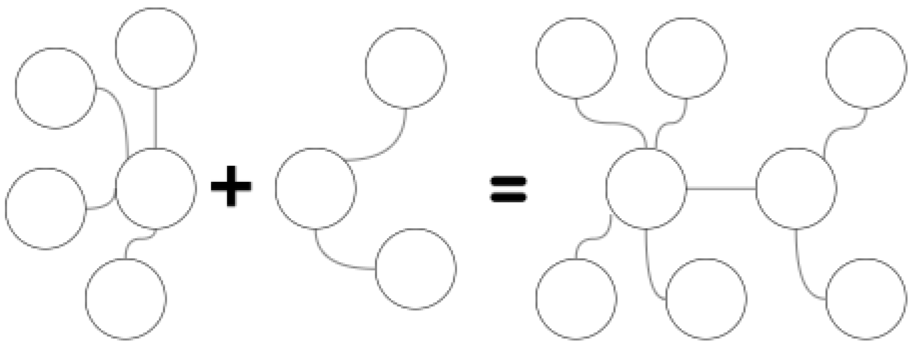
In this project, I sought to develop and prove new algorithms to create spanning trees in general graphs with per-vertex constraints. This means that each vertex in the graph would have some additional value, a degree constraint d. For a spanning tree to be correct every vertex in the spanning tree must have a degree exactly equal to that degree constraint d. This poses an additional constraint on what would otherwise be a trivial spanning tree problem. In this paper two proofs will be discussed and analyzed, which inspired my studies and led to the development of the 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

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

Figure 1:



Double stars also have the interesting property of being able to cover grap hs incredibly quickly. In fact, a graph of n vertices 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), and then add double stars centered around those edges, where the two vertices connected to the edge form the individual single stars. Lovász has proved this to be true, and that $2n/3$ double stars are required. Discussed in this paper will be the key determinants in a maximal double-star cover, as well as examples of such covers on simple graphs.

2.1 The algorithm

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 as the maximal matching of G, having size r. I is defined as the vertices not covered by 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 in the worst case it is still impossible that double stars cover worse than single stars. This leads Lovás to know that $n - \tau \leq n - s = 2r$, or the number of vertices minus independent vertices is less than or rqual to the vertices minus the vertices consained in I, which is equal to the number of vertices in M.

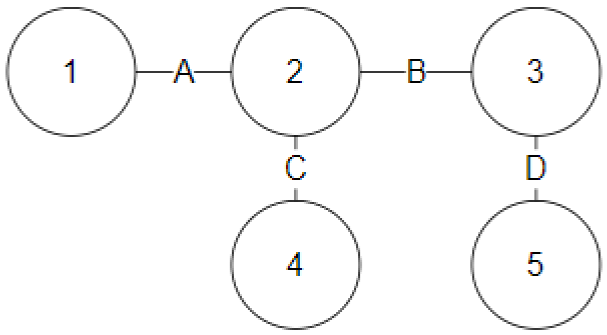
In section 3 Lovász moves on to looking at S_i. S_i is a double star of parts of M. T_i is the tree of all edges containing c_i (from i). We know that the set of all S and T covers G, therefore $f_3(g) \leq r + s = n - r$, or the maximum number of double stars required to cover a graph is less than or equal to the size of m plus the size of i, which 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 Examples

Presented below are examples of graphs of various sizes. It may be noted that some of these graphs themselves are double stars. 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. Similarly, the steps taken will be laid out sequentially, and if different edges are selected the end cover might be very different.

Figure 2:



Above, in figure 2, we see a double star graph. We know that n is equal to 5. Running our greedy algorithm to find a maximal matching, we will 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 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 necessarily have a full cover. Although this is something our algorithm would never provide as a cover (due to a and b being adjacent) this shows a theoretical maximum size for the vertex cover. It is simply impossible to increase this value, knowing that no matter which edges we choose in any order we will cover the graph in at most 3 double stars.

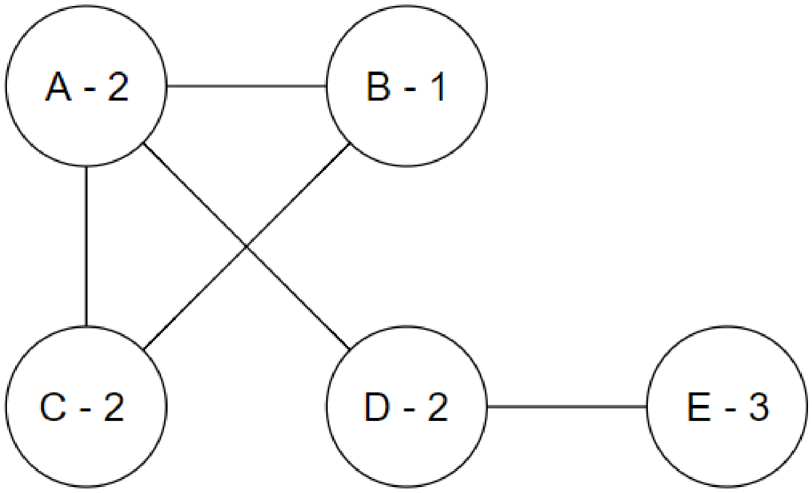
3 Determining subgraphs of maximum edges on degree constrained graphs

Based on *Another look the the degree constrained subgraph problem* by Yossi Shiloach

In this paper Yossi Shiloach details a wonderful algorithm for determining subgraphs of maximum edges on a degree constrained graph $G(V, E)$. He defines degree constraints in a similar way to we do, a list of numbers representing the maximum degree a vertex may have, rather than the required degree as we define degree constraint. However, rather than determining a spanning tree he wishes to find a edge set, E', such that $G'(V, E')$ adheres to degree constraints, maximizing $|E'|$. The method for generating E' very likely will prove useful in generalizing our own algorithm.

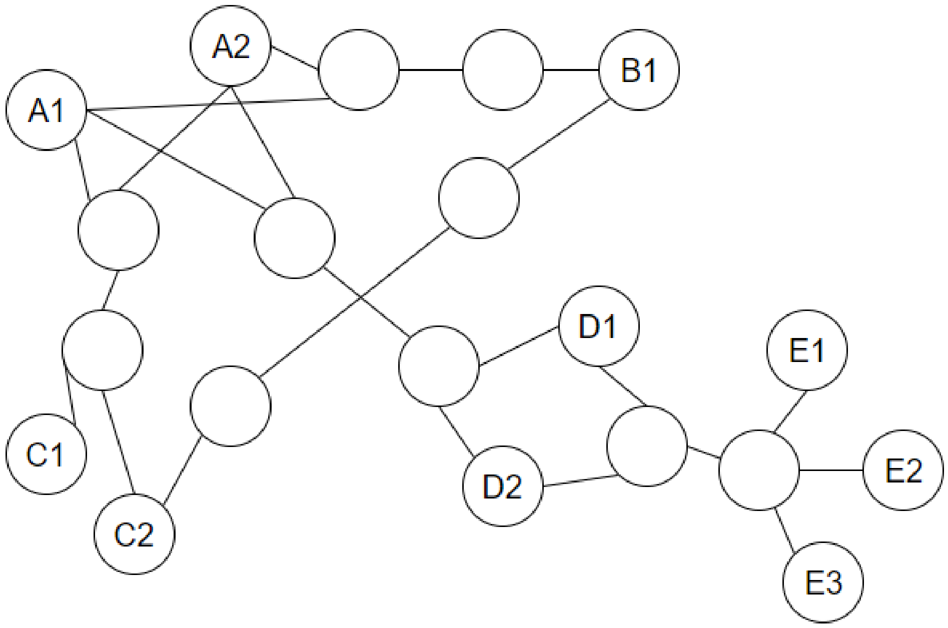
3.1 The algorithm

Figure 3:



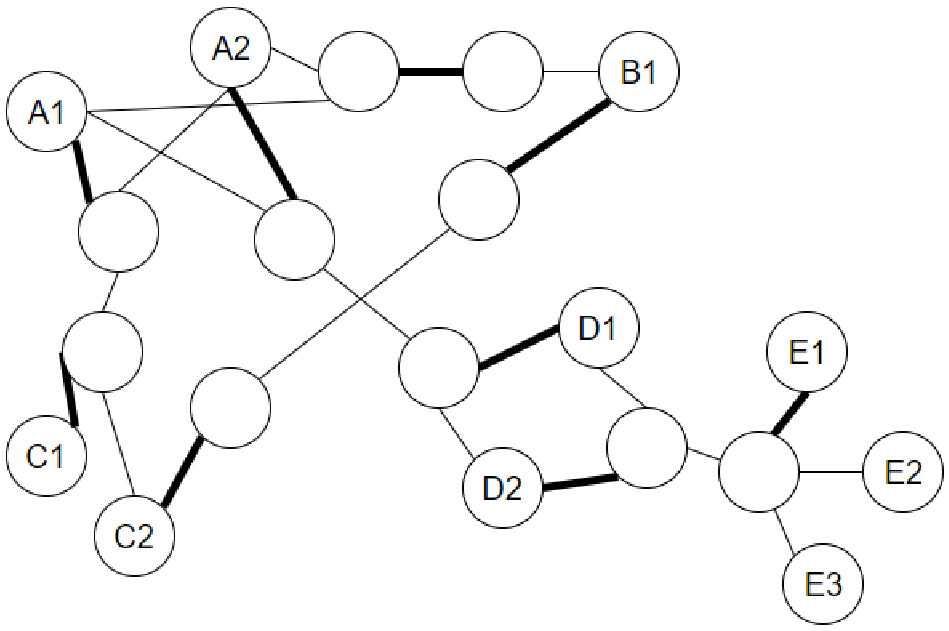
Observed in Figure 3 is a graph, with vertex labels and maximum degrees. In order to determine the maximum subgraph $G(V, E')$ we must create graph H. This is done by taking every vertex, and replicating it into subvertices. A degree constraint of 2 leads to 2 subvertices, 3 leads to 3 subvertices, and so on. Then, connection vertices are created, representing each edge in G. Therefore, there will be a vertex created for A->B and a vertex for B->A. A->B will connect to every subvertex of A, while B->A will connect to every subvertex of B. Then $A- > B$ will be connected to B->A. This will be done for every edge in G, to yield a new graph H, depicted in figure 4.

Figure 4:



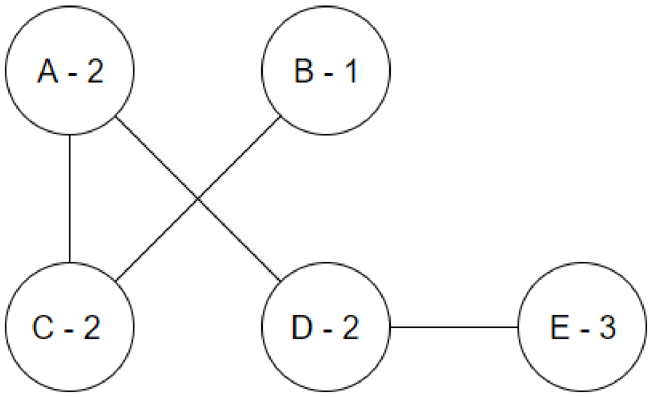
Then a maximum matching must be drawn on H, such as the one depicted in Figure 5. Bolded edges are in the matching.

Figure 5:



Then, one must look at the edges between the unlabeled connector vertices. If the edge between the connectors is in the matching, as in the A - B connector, then that edge from G is not in G'. Otherwise that edge will be in G'. G' is shown below in Figure 6. It can be verified that G' does not violate any degree maximums, and contains a maximum 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 graphs 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

Assume that we wish to create a spanning tree for a graph. Correct algorithms for this problem proven by Kruskal and other mathematicians, however in this case we wish to consider an additional constraint. Assume that in the graph every vertex has some degree constraint d, where for a spanning tree to be correct every vertex in the spanning tree must have a degree exactly equal to that degree constraint d.

An algorithm for solving this problem can be seen below:

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

```
1: Graph $G(V, E)$ with a total degree of 2n-2 where n is the magnitude of $V$
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. 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
```

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

Proof. Allow the algorithm to run until step 3, producing tree T. $G \setminus T$ will have a total degree constraint of $2(|V| - 1) - 2$, accounting for the single edge in T. Therefore, because it has a total degree constraint of $2(|V| - 2)$ and needs to connect $|V| - 2$ vertices to T. Thus the spanning tree will be able to be completed as the number of possible edges is equal to twice the number of vertices, 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, 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 ≥ 2, and then decrease T's total degree constraint by 2. Therefore the change in T cannot be less than zero, and thus T will not be decreased and therefore 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,

$\square$