UNIVERSITÀ di VERONA | Department of ECONOMICS

Working Paper Series
Department of Economics
University of Verona

A constrained minimum spanning tree problem

Alberto Peretti

WP Number: 8 December 2018

ISSN: 2036-2919 (paper), 2036-4679 (online)

A constrained minimum spanning tree problem

Alberto Peretti
Department of Economics, University of Verona
Verona, Italy
e-mail: alberto.peretti@univr.it

December, 2018

Abstract

In the classical general framework of the minimum spanning tree problem for a weighted graph we consider the case in which a predetermined vertex has a certain fixed degree. In other words, given a weighted graph G, one of its vertices v_0 and a positive integer k, we consider the problem of finding the minimum spanning tree of G in which the vertex v_0 has degree k, that is the number of edges coming out of v_0.

We recall that among the various methods for the solution of the unconstrained problem an efficient way to find the minimum spanning tree is based on the simple procedure of choosing one after the other an edge of minimum weight that has not be chosen yet and does not create cycles if added to the previously chosen edges. This technique is known as the "greedy algorithm". There are problems for which the greedy algorithm works and problems for which it does not.

We prove that for the solution of the one degree constrained minimum spanning tree problem the classical greedy algorithm finds a right solution.

Keywords. Graph theory, Trees, Minimum spanning tree problem, Constrained minimum spanning trees.

AMS Classification. 05C05, 05C85

JEL Classification. C650, C690

1 Introduction

Among the combinatorial optimization problems the well known MST (Minimum Spanning Tree) problem consists in finding a spanning tree of minimum cost over a connected weighted graph. It is a classical problem for which many efficient algorithms have been proposed. The problem can be solved in polynomial time in the number of nodes of the given graph.

In the literature many variations of the original problem have been considered during the years. For example, a very general formulation can be the following: given a connected weighted graph G, find a minimum spanning tree for G that satisfies a property P, that is a property of trees, usually assumed to be valuable in polynomial time. The new problem may be indicated as an MST(P) problem. In [1] the case when P is an isomorphism property is considered, that means the tree is required to be isomorphic to a certain model of trees.

Another large class of specification of the original MST problem is the Degree-constrained MST problem. Here the required tree has some constraint on the degree of its nodes. For example the constraint can be to ask for a tree in which all the nodes have degree not greater than a fixed positive integer k. This is known to be a difficult problem.

We aim here to consider a particular case in the Degree-constrained MST problem: in the graph we fix the attention on a particular node and we ask in the optimal tree this node has a predefined fixed degree.

2 Notations and problem definition

2.1 Notations

Let us introduce some of the basic vocabulary and notions of graph theory.

An undirected graph (graph for short in the following) G is a pair $G = (V, E)$ where V is a finite set of vertices and E is a finite set of edges, each one connecting two vertices. We may see the set E as a subset of non ordered pairs of vertices: if $e \in E$, then $e_{ij} = (v_i, v_j)$, $v_i, v_j \in V$. The two vertices are called the neighbors of the edge. The pairs are non ordered, as we are considering undirected graph.

The degree of a vertex v is the number of edges having v as a neighbor. Equivalently, if we define the neighbors of v as the set $N(v) = \{w \in V : (v, w) \in E\}$, the degree of v is $|N(v)|$.

A path $p(u, v)$ in a graph $G = (V, E)$ from vertex $u \in V$ to vertex $v \in V$ is a sequence $p(u, v) = (u, v_1, v_2, \ldots, v_k, v)$ of vertices of V such that $(u, v_1), (v_k, v) \in E$, and $(v_i, v_{i+1}) \in E$ for $i = 1, \ldots, k-1$.

A path $p(u, v)$ in G with $u = v$ is called a cycle in G. The length $|p(u, v)|$ of a path $p(u, v) = (u, v_1, v_2, \ldots, v_k, v)$ is defined by $k + 1$, namely the number of edges between u and v in $p(u, v)$.

A graph is connected if for any two vertices u and v in V there is a path $p(u, v)$ going from u to v. A tree is a connected graph with no cycles.

We call a forest a set of trees whose vertices are disjoint.

If $G = (V, E)$ is a connected graph, a spanning tree for G is a tree (V, E') where $E' \subseteq E$.

We will be dealing with weighted graphs, in the sense a positive cost function is defined on the set of edges in G. We have then $c : E \to \mathbb{R}_+$. If $(u, v) \in E$, the value $c(u, v)$ is the cost of the edge (u, v).

If T is a tree in G the cost of T is the sum of the costs of the single edges in T.

2.2 Problem definition

Let $G = (V, E)$ be a connected weighted graph, where $V = \{v_1, v_2, \ldots, v_n\}$ is the vertex set of G and $E = \{e_1, e_2, \ldots, e_m\}$ is the edge set of G. Let w_i represent the weight or cost of edge e_i, for $i = 1, \ldots, m$, where the weight is restricted to be a nonnegative real number.

The Minimum Spanning Tree problem consists in finding a spanning tree of G with minimum cost. Formally, we may describe any subgraph of G by means of a vector $x = (x_1, x_2, \ldots, x_m)$ where each component x_i is defined as

$$x_i = \begin{cases} 1 & \text{if edge } e_i \text{ is in the subgraph} \\ 0 & \text{otherwise.} \end{cases}$$

Remembering that a subgraph S of G is a spanning tree of G if S contains all the vertices of G, is connected and contains no cycles, let's call T the set of all the spanning trees of G. The MST problem is the problem

$$\min\Big\{z(x) = \sum_{i=1}^{m} w_i x_i \ : \ x \in T\Big\}.$$

Now, suppose we want to set constraints on the degree of vertices in G. Of course we may think of equality/inequality constraints. If b_j is the constraint on the degree $d(v_j)$ of vertex v_j, then the general Degree-constrained Minimum Spanning Tree problem is the problem

$$\min\Big\{z(x) = \sum_{i=1}^{m} w_i x_i \ : \ d(v_j) \leq b_j, v_j \in V, x \in T\Big\}.$$

In this framework, if k is a positive integer, we have a k-bounded MST problem if $b_j = k$ for each j.

We want to consider here a simpler case of a Degree-constrained Minimum Spanning Tree problem, given by an equality constraint on one vertex of G, let's say v_1. The problem is then

$$\min\Big\{z(x) = \sum_{i=1}^{m} w_i x_i \ : \ d(v_1) = k, v_1 \in V, x \in T\Big\}.$$

3 Elementary operations in a graph

Let $G = (V, E)$ be a connected weighted graph and let $T = (V, E')$ be a spanning tree of G.

Definition 1. *If a is an edge belonging to $e \setminus E'$ and b is an edge in the cycle created by adding a to the tree T, we say we are doing an elementary operation on T in adding a and taking off b.*

Remark. By performing an elementary operation on a spanning tree of G we end up with another spanning tree of G. We call the value $c(a) - c(b)$ the cost of the elementary operation. It is possible to characterize minimum spanning trees of a graph by means of elementary operations. In fact the following theorem holds.

Theorem 1. *A spanning tree T of G is minimum if and only if every elementary operation on T has a nonnegative cost.*

Proof. If T is minimum then every elementary operation on T has necessarily a nonnegative cost. Let's suppose that in T every elementary operation has a nonnegative cost and let T_0 be a minimum spanning tree of G. Suppose $a_1, a_2, \ldots, a_s$ are the edges in A_0 (the order is not relevant) and $b_1, b_2, \ldots, b_s$ the ones in A. If the two sets of edges are not equal, let b_k the first edge in A that does not belong to A_0. If we add b_k to A_0 and we take out b_k from A we get a cycle in A_0: in this cycle there is an edge a_h, different from b_k, that does not belong to A and reconnects A after the remotion of b_k. Then
$c(b_k) - c(a_h) \geq 0$ as an elementary operation on A_0, which is minimum;
$c(a_h) - c(b_k) \geq 0$ by the assumption, as an elementary operation on A.

Hence we have a null cost elementary operation on A_0 that gives us a new minimum spanning tree A_1 with one edge more in common with A. If A and A_1 are not equal, we can repeat the step more and more till we get the trees with the same set of edges and the same cost. □

Let us introduce the concept of partition in a graph.

Definition 2. *We call partition forest of the graph G with roots $v_1, \ldots, v_p$ a forest with p connected components which contains all the vertices of G and such that each component contains just one of the nodes $v_1, \ldots, v_p$.*

Also for the partition forests we may define an elementary operation.

Definition 3. *Given a partition forest F with roots $v_1, \ldots, v_p$, adding an edge and removing another edge in the cycle that has been created, in such a way we end up again with a partition forest with the same roots, is called an elementary operation on the partition forest F.*

Remark. For as a partition forest with roots $v_1, \ldots, v_p$ two kinds of elementary operations may exist. The first kind corresponds to the elementary operations on a tree and consists in an operation internal to one of the connected components of the forest. The second kind consists in a connection operation of two different components and the consequent disconnection of the same components. In the same way we did for the trees, it is possible to characterize a minimum spanning forest by means of elementary operations.

Theorem 2. *A partition forest F with roots $v_1, \ldots, v_p$ is minimum if and only if every elementary operation on F has a nonnegative cost.*

Proof. If F is minimum then every elementary operation on T has necessarily a nonnegative cost. Let's suppose in F every elementary operation has a nonnegative cost and let F_0 be a minimum partition forest with roots $v_1, \ldots, v_p$.

Suppose $a_1, a_2, \ldots, a_s$ are the edges in F_0 and $b_1, b_2, \ldots, b_s$ the ones in F. If the two sets of edges are not equal, let b_k the first edge in F that does not belong to F_0. Let's add b_k to F_0 and remove b_k from F. In F_0 either we create a cycle in one connected component or we connect two different components. In F we just create a new connected component C that does not contain any of the roots. Now it is easy to see that either in the created cycle or in the path that connects the two roots we have an edge a_h, different from b_k, that reconnects C to the rest of the forest. Then
$c(b_k) - c(a_h) \geq 0$ as an elementary operation on F_0, which is minimum;
$c(a_h) - c(b_k) \geq 0$ by the assumption, as an elementary operation on F.

Adding in F_0 b_k and removing a_h is then a null cost elementary operation that gives us a new minimum forest with one edge more in common with F. We can repeat the steps till we get the same forests. □

Remark. Clearly Theorem 1 is just a particular case of Theorem 2, as a spanning tree is a partition forest with root v, where v is any node in the graph.

4 Two algorithms to find a minimum partition forest

4.1 A first algorithm for a minimum partition forest

Here is an algorithm for the search of a minimum partition forest with roots $v_1, \ldots, v_p$ in a connected weighted graph $G = (V, E, c)$. We indicate with $F = (V, E')$ the forest; if $|V| = n$ then $|E' = n - p|$.

Algorithm 1

```
begin
    E' := ∅;
    while |E'| ≠ n − p do
        begin
            sia e uno spigolo minimo, e ∉ E',
            che aggiunto ad E' non forma cicli e
            non connette due radici;
            E' := E' ∪ {e}
        end
end
```

Theorem 3. *Algorithm 1 finds a minimum forest with roots $v_1, \ldots, v_p$.*

Proof. The output of Algorithm 1 is certainly a forest with roots $v_1, \ldots, v_p$. Let F be this output and let F_0 be a minimum forest with roots $v_1, \ldots, v_p$. If F and F_0 have the same edges, the result is proved; otherwise let's give these edges the same order in which they have been selected by the algorithm. If $a_1, \ldots, a_s$ are the edges, then for every i a_i is a minimum edge that, added to $a_1, \ldots, a_{i-1}$, does not create cycles and does not connect two roots. Moreover let us give the edges in F_0 any ordering we want: be $b_1, \ldots, b_s$ these edges. Let a_k be the first edge in F that does not belong to F_0 and let us add a_k to F_0. We may have two cases:

1. We get a cycle in one of the connected components in F_0. In this cycle there is an edge b that does not belong to F; consider that b, if added to $a_1, \ldots, a_{k-1}$, does not create cycles and does not connect two roots, as $a_1, \ldots, a_{k-1}$ belong to F_0. Then $c(b) \geq c(a_k)$. Hence if in F_0 we add a_k and we remove b we get a forest F_0' with cost less than or equal to the cost of F_0. As the strict inequality can not hold, being F_0 minimum, then $c(F_0') = c(F_0)$. Hence F_0' is a minimum forest with roots $v_1, \ldots, v_p$ that has one more edge than F_0 in common with F. Now, if F and F_0' have the same edges, the result is proved; otherwise we can repeat the procedure.

2. Two roots v_i and v_j are connected in F_0. In the path $v_i \ldots v_j$ just created there is an edge b that does not belong to F; if added to $a_1, \ldots, a_{k-1}$, b does not make a cycle and does not connect two roots, as $a_1, \ldots, a_{k-1}$ belong to F_0. Then $c(b) \geq c(a_k)$. The same as before, in F_0 we may add a_k and remove b, getting a forest F_0' with roots $v_1, \ldots, v_p$ and cost less than or equal to the cost of F_0. As the equality must hold, F_0' is a minimum forest with roots $v_1, \ldots, v_p$ that has one more edge than F_0 in common with F. The conclusions are the same as in the first case.

□

Remarks. Algorithm 1 operates on a partition forest in the same way Kruskal's algorithm operates on a spanning tree. It is a so called greedy algorithm: this is referred to the technique

of choosing at each step a minimal element that preserves the structure of the object we want to construct. It is interesting that the greedy technique works for certain types of problems and does not for others. The concept of matroid investigates on this aspect.

4.2 Another algorithm for a minimum partition forest

Here is a different algorithm for the search of a minimum partition forest with roots $v_1, \ldots, v_p$ in a connected weighted graph $G = (V, E, c)$. We still indicate with $F = (V, E')$ the forest.

Algorithm 2

```
begin
    set up a minimum spanning tree T = (V, S) for G;
    roots := {v_1};
    E' := T;
    for i = 2 to p do
        begin
            let v_s be the root to which v_i is connected in F = (V, E');
            let e be a maximum edge in the path v_i ... v_s;
            E' := E' \ {e};
            roots := roots ∪ {v_i}
        end
end
```

Theorem 4. *Algorithm 2 finds a minimum forest with roots $v_1, \ldots, v_p$.*

Proof. Let F be the output of Algorithm 2: clearly F is a partition forest with roots $v_1, \ldots, v_p$. We prove it is minimum by showing that every elementary operation on F has a nonnegative cost, using Theorem 2.

If we consider an elementary operation internal to one of the components, consisting in adding an edge a and removing an edge b in the cycle that has been created, certainly $c(a) \geq c(b)$ as otherwise the spanning tree T obtained in the first part of the algorithm would have not been minimum. Let's consider now an elementary operation involving two different components that consists in connecting them by adding the edge a and disconnecting them again by removing the edge b in the path that joins the two roots. If a belongs to the tree T then a is maximum in this path and $c(a) \geq c(b)$. If a does not belong to T, then in T the two components are connected by an edge c; obviously $c(a) \geq c(c)$ because T is minimum and $c(c) \geq c(b)$ because c is maximum in the path that joins the two roots: also in this case then $c(a) \geq c(b)$. □

5 Minimal spanning trees with one degree constraint

Let $G = (V, E, c)$ be again a connected weighted graph, let v_0 one of its nodes and k a positive integer.[1] We want to solve the following problem: find a minimum spanning tree of G in which the degree of v_0 is k. As we said in the footnote, we assume that the degree of v_0 in G is at least k. We assume also that we do not disconnect G by removing any of the incident edges in v_0. A spanning tree of G with $\deg(v_0) = k$ can be represented as in Figure 1.

We may think of $v_1, \ldots, v_k$ as the roots of the tree. Everything is underneath the roots in the picture must be intended as a subtree and will be called a component of the tree. The nodes

[1] For the meaning k has in the following, we assume that k is not greater than the degree of v_0 in the graph G.

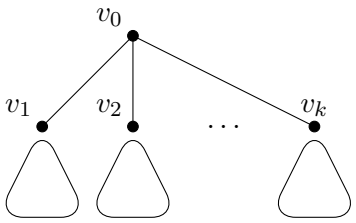


Figure 1: A model for a tree with $\deg(v_0) = k$. The triangular structures below the roots have to be intended as subtrees.

of the components will be the "descendants" of their roots. These names are the same we have with the partition forests: in fact the components of the tree are nothing but a partition forest with roots $v_1, \ldots, v_p$ for the subgraph of G generated by the nodes other than v_0.

The following theorem can be easily proved.

Theorem 5. *If (v_0, v_1) is a minimum cost edge among the incident edges in v_0 then a minimum spanning tree for G exists with $\deg(v_0) = k$ that contains (v_0, v_1).*

Proof. Let $T(k)$ be a minimum spanning tree of G with $\deg(v_0) = k$. If $T(k)$ does not contain (v_0, v_1) suppose in $T(k)$ v_1 is a descendant of the root v_2. If (v_0, v_1) was not a minimum cost edge incident in v_0, the tree we obtain from $T(k)$ by adding (v_0, v_1) and removing (v_0, v_2) would be a tree with $\deg(v_0) = k$ with cost less than $T(k)$. This contradicts the assumption $T(k)$ is minimum. hence (v_0, v_2) is a minimum cost edge incident in v_0, the same as (v_0, v_1): the tree we obtain by adding (v_0, v_1) and removing (v_0, v_2) is then the minimum spanning tree of G with $\deg(v_0) = k$ we were looking for. □

Also for the trees with $\deg(v_0) = k$ it is worthwhile to introduce the concept of elementary operation, as it will be used in the following proofs. Since we want an elementary operation not to modify the essential characteristics of the structure, for trees with $\deg(v_0) = k$ we consider elementary operations preserving the degree of v_0.

Be $T(k)$ then a minimum spanning tree of G with $\deg(v_0) = k$. An elementary operation on $T(k)$ preserving the degree of v_0 may be of three types:

1. (Root change in a component): if in $T(k)$ v_2 descends from v_1, the elementary operation consists in adding the edge (v_0, v_2) and removing the edge (v_0, v_1).

2. (On the components): it is an operation that does not involve the edges incident in v_0; in other words it is an elementary operation on the forest of the components of $T(k)$.

3. It consists in the following sequence of operations: addition of an edge (v_0, v_2), removal of an edge a non incident in v_0 in the generated path, removal of an edge (v_0, v_1) and final addition of an edge b that reconnects the tree.

Remarks. It is evident that the result of an elementary operation preserving the degree of v_0 is a tree with $\deg(v_0) = k$. In the following, talking about elementary operations on a tree with $\deg(v_0) = k$ we will intend elementary operations preserving the degree of v_0.

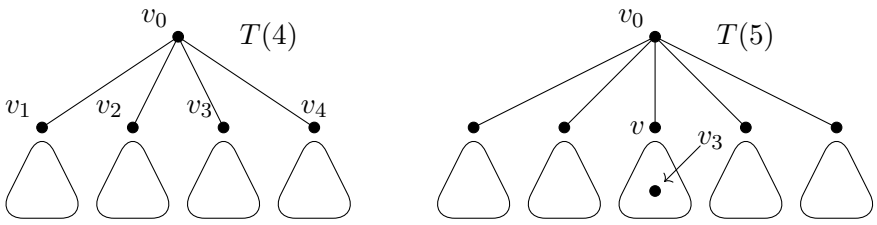
We present an algorithm for the search of a minimum spanning tree of G with $\deg(v_0) = k$.

Algorithm 3

begin
set up a minimum spanning tree $T = (V \setminus \{v_0\}, S)$ for G for
the subgraph of G generated by the nodes $V \setminus \{v_0\}$;
let (v_0, v_1) a minimum edge incident in v_0;
$S := S \cup \{(v_0, v_1)\}$;
roots $:= \{v_1\}$;
while $|\text{roots}| \neq k$ do
begin
for each node v such that (v_0, v) exists with $v \notin$ roots, let e be a
maximum edge non incident in v_0 in the path $v \dots v_0$;
compute $c'(v) = c(v_0, v) - c(e)$;
let u be a node with minimum $c'(u)$;
$S := S \cup \{(v_0, u)\} \setminus \{e\}$;
roots $:=$ roots $\cup \{u\}$;
end
end

Theorem 6. *If $T(k)$ is a minimum spanning tree of G with $\deg(v_0) = k$ and $v_1 \dots v_k$ are its roots, then a minimum tree $T(k+1)$ exists with $\deg(v_0) = k+1$ whose roots are $v_1 \dots v_k v_{k+1}$.*

Proof. In order to facilitate the notations we give the proof in the case $n = 4$, but there is a general validity. Suppose $T(4)$ and $T(5)$ are minimum spanning trees with $\deg(v_0) = 4$ and $\deg(v_0) = 5$ respectively.



Let's assume that for example v_3 is not a root in $T(5)$: we want to show that we can always find a minimum $T'(5)$ in which v_3 is a root.

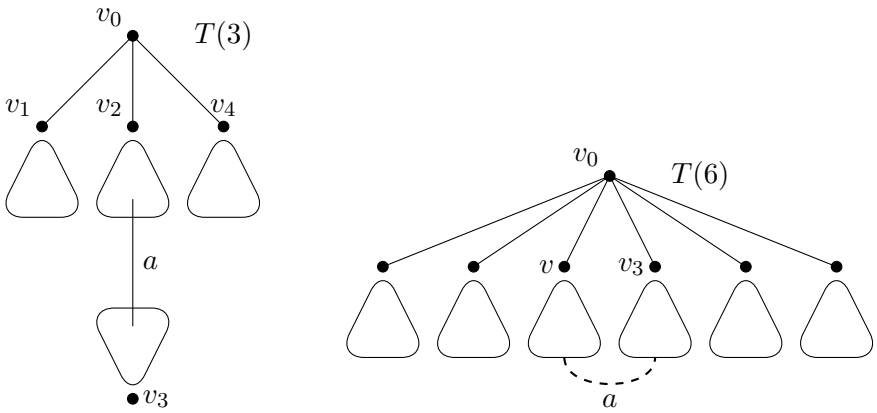
Let's assume v_3 is descendant of the root v. If in $T(4)$ v descends from v_3 we can make in both trees an elementary operation of type 1:
(in $T(5)$): $\cup\{(v_0, v_3)\} \setminus \{(v_0, v)\}$ has nonnegative cost as $T(5)$ is minimum,
(in $T(4)$): $\cup\{(v_0, v)\} \setminus \{(v_0, v_3)\}$ has nonnegative cost as $T(4)$ is minimum.

Clearly the cost is zero and we can obtain a $T'(5)$ simply by interchanging the two roots.

If instead in $T(4)$ v does not descend from v_3 then in $T(5)$ in the path $v_3 \dots v$ there is an edge a that can reconnect $T(4)$ after (v_0, v_3) has been removed. Let's set

$$T(3) = T(4) \setminus \{(v_0, v_3)\} \cup \{a\} \qquad \text{e} \qquad T(6) = T(5) \cup \{(v_0, v_3)\} \setminus \{a\}$$

Let u be a root in $T(6)$, different from v_3, that is not a root in $T(3)$. If either u is descendent in $T(3)$ of a root v_i not belonging to the component of u in $T(6)$ or u is in the component of v_3, the in the path $u \dots v_i$ there is an edge b that reconnects $T(6)$ after (v_0, u) has been removed. In this case:



(in $T(5)$): $\cup\{(v_0, v_3)\}\setminus\{a\}\setminus\{(v_0, u)\}\cup\{b\}$ in an elementary operation of type 3 with nonnegative cost as $T(5)$ is minimum,
(in $T(4)$): $\setminus\{(v_0, v_3)\}\cup\{a\}\cup\{(v_0, u)\}\setminus\{b\}$ in an elementary operation of type 3 with nonnegative cost as $T(4)$ is minimum.

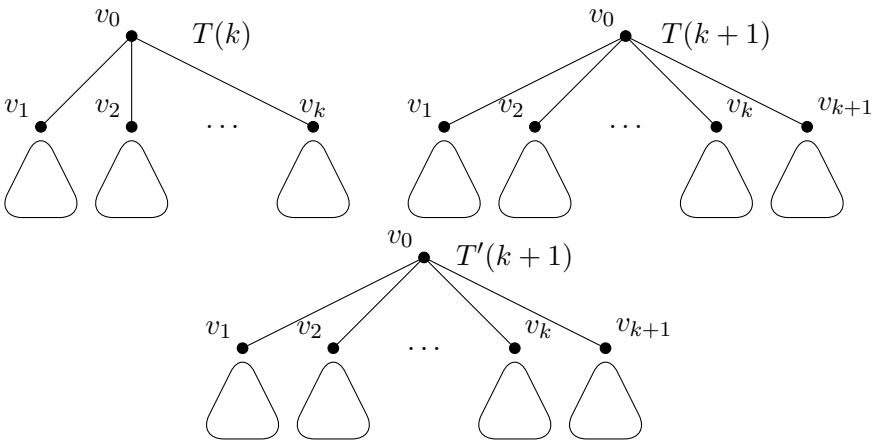
The cost of the operation being null, it allows me to find the required tree $T'(5)$.

Finally, if u is descendent in $T(3)$ of a root v_i belonging to the component of u in $T(6)$, then there is still the possibility for an elementary operation of type 1 by simply interchanging the roots u and v_i. The procedure can be repeated until all the roots in $T(4)$ are also roots in $A(5)$. □

Theorem 7. *Algorithm 3 finds a minimum spanning tree with* $\deg(v_0) = k$.

Proof. By induction on the degree k.

If $k = 1$ the thesis is clearly true. Suppose at step k a minimum tree with $\deg(v_0) = k$ is obtained and let $T(k)$ such a tree. Let $T(k+1)$ be the tree we obtain after a further iteration of Algorithm 3. Let finally $T'(k+1)$ be a minimum tree with $\deg(v_0) = k+1$. If $v_1 \dots v_k$ are the roots in $T(k)$, from Theorem 6 we may suppose that $v_1 \dots v_k\ u$ be the roots in $T'(k+1)$.



The components of $T(k)$ form a minimal forest with roots $v_1 \dots v_k\ u$ for the same subgraph. hence we can think of this second forest as obtained from the first with Algorithm 2, by removing a maximum edge a in the path connecting u to its root. Remembering that $T(k+1)$ is obtained from $T(k)$ by adding (v_0, v_{k+1}) and removing a maximum edge b in the path connecting in $T(k)$ v_{k+1} to its root and remembering also this operation has minimum cost among all possible

operations, we have that

$$c(T(k+1)) = c(T(k)) + c(v_0, v_{k+1}) - c(b) \leq c(T(k)) + c(v_0, u) - c(a) = c(T'(k+1)).$$

Therefore, being $T'(k+1)$ of minimal cost, $T(k+1)$ is also minimum. □

6 An example

Just to provide an application let us consider a simple numerical example. Here is a weighted graph with six vertices in Figure 2. Let $V = \{v_1, v_2, \ldots, v_6\}$ is the vertex set of the graph.

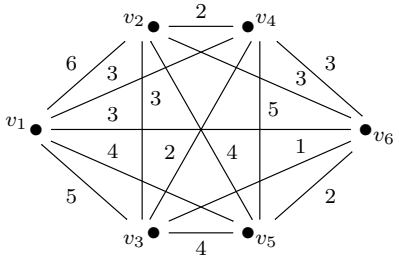


Figure 2: A weighted graph with six vertices

We want to find a minimum spanning tree with $\deg(v_1) = 3$. First of all it is necessary to set up a minimum spanning tree for the vertices $V \setminus \{v_1\} = \{v_2, \ldots, v_6\}$. With the classical algorithms for the (unconstrained) minimum spanning tree we get the pair vertices/edges

$$T = (V, S), \text{ where } V = \{v_2, \ldots, v_6\} \text{ and } S = \{(v_2, v_4), (v_3, v_4), (v_3, v_6), (v_5, v_6)\}.$$

The tree is represented here below on the left.

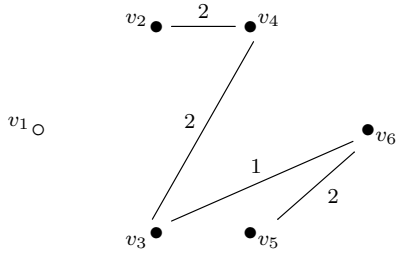


Figure 3: A minimum spanning tree for $V \setminus \{v_1\}$

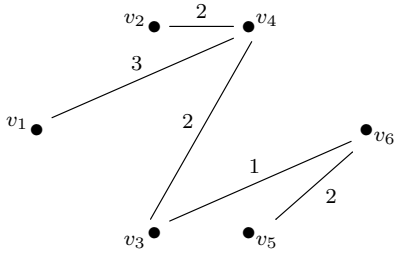


Figure 4: A minimum spanning tree with $\deg(v_1) = 1$

Now, by following the first step of Algorithm 3, (v_1, v_4) is a minimum edge incident in v_1 and then we firstly update $S = S \cup \{(v_1, v_4)\}$, roots $:= \{v_4\}$. The new tree is represented above on the right.

As $|\text{roots}| \neq 3$, we have to consider the nodes v such that (v_1, v) exists with $v \notin$ roots. These nodes are v_2, v_3, v_5, v_6. We have to compute for these nodes the "weights" $c'(v_i) = c(v_1, v_i) - c(e)$,

where e is a maximum edge non incident in v_1 in the path $v_i \dots v_0$. We get

$$\begin{aligned} c'(v_2) &= c(v_1, v_2) - c(v_2, v_4) = 6 - 2 = 4 \\ c'(v_3) &= c(v_1, v_3) - c(v_3, v_4) = 5 - 2 = 3 \\ c'(v_5) &= c(v_1, v_5) - c(v_5, v_6) = 4 - 2 = 2 \\ c'(v_6) &= c(v_1, v_6) - c(v_3, v_4) = 3 - 2 = 1 \end{aligned}$$

The vertex v_6 is chosen and consequently the arc (v_3, v_4) is removed from the tree.

$$V = \{v_1, v_2, \dots, v_6\} \text{ and } S = S \cup \{(v_1, v_6)\} \setminus \{(v_3, v_4)\}.$$

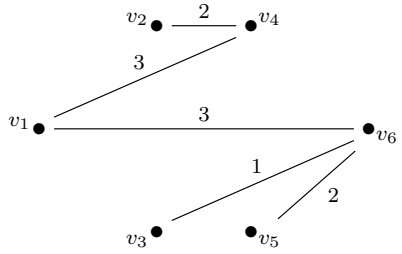


Figure 5: A minimum spanning tree with $\deg(v_1) = 2$

As $|\text{roots}| \neq 3$, we have to consider again the nodes v such that (v_1, v) exists with $v \notin$ roots. These nodes are now v_2, v_3, v_5. The weights are

$$\begin{aligned} c'(v_2) &= c(v_1, v_2) - c(v_2, v_4) = 6 - 2 = 4 \\ c'(v_3) &= c(v_1, v_3) - c(v_3, v_6) = 5 - 1 = 4 \\ c'(v_5) &= c(v_1, v_5) - c(v_5, v_6) = 4 - 2 = 2 \end{aligned}$$

The vertex v_5 is chosen and consequently the arc (v_5, v_6) is removed from the tree.

$$V = \{v_1, v_2, \dots, v_6\} \text{ and } S = S \cup \{(v_1, v_5)\} \setminus \{(v_5, v_6)\}.$$

A minimum spanning tree with $\deg(v_1) = 3$ is here below.

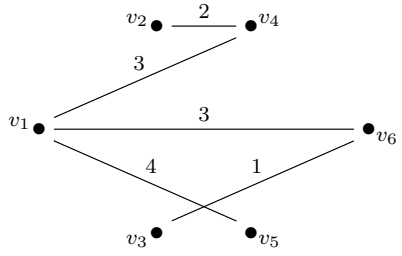


Figure 6: A minimum spanning tree with $\deg(v_1) = 3$

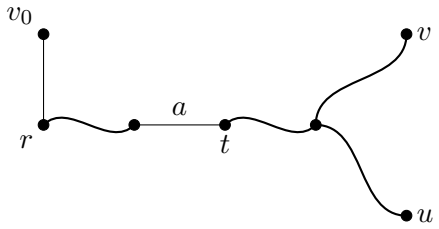
7 Some considerations on the computational complexity

Suppose G has n nodes. At each iteration of Algorithm 3 we must find, for each of the roots, the path that connects them to v_0 and find in this path a maximum cost edge. This operation can be done easily if an orientation is given to the edges. We may think that in the tree every node has a depth, given by the number of edges in the only path that connects the node to v_0: a node with depth i is a descendant of a node with depth $i-1$. For the purposes of Algorithm 3 it is important to know, for every node its (unique) predecessor, in order to get the path that connects it to a root.

Here is an algorithm that, with a fixed node v_0, constructs an order of the nodes based on the depths in relation to the root.

```
begin
    predecessors := {v_0};
    S := set of the nodes of the tree;
    while predecessors ≠ ∅ do
        begin
            descendants := ∅;
            for each x ∈ predecessors, if the edge (x, y)
            exists in the tree and y ∈ S, then
                descendants := descendants ∪ {y};
                pred(y) := x;
            S := S \ predecessors;
            predecessors := descendants;
        end
end
```

Since the algorithm considers the nodes just once and for each of them the search for descendants can be performed in n steps, the algorithm terminates in at most n^2 steps. Once the nodes are sorted this way, the search for a maximum edge in the path connecting a root to v_0 can be performed in at most n steps. Clearly after the removal of this edge the descendants have to be updated: the picture shows what can happen.



Here r is a root and u and v are two possible roots. Suppose the algorithm decides on the basis of costs to add edge (v_0, u) and remove edge a. At the next step v no longer descend from r but from u. We just need to reverse the order of descendants in the path $u \dots t$, that can be done in n steps. Remembering that a minimum spanning tree can be found in $O(n^2)$ steps, Algorithm 3 for the search of a minimum spanning tree with $\deg(v_0) = k$ requires $O(kn^2)$ steps to terminate.

8 Further developments and conclusions

It is known that the greedy algorithm, namely the way of "choosing the best" one step after the other, not always takes to the correct best final result. It depends on the kind of problem we have. Interesting connections with the so called matroid theory exist. We have seen that with the minimum spanning tree with one degree constraint, that is a degree constraint on one of the vertices of the graph, the technique works. It is interesting that if we consider a slightly generalised version of this problem, for example just the degree constraint on two of the vertices, the greedy technique does not work any more. We intend to consider this problem in the general context of degree-constrained minimum spanning tree in a future work.

References

[1] C. Papadimitriou, M. Yannakakis, The complexity of restricted spanning tree problems, Journal of the Associationn for Computing Machinery, Vol. 29, No. 2, 1982, pp. 285–309.

[2] C. Papadimitriou, K. Steiglitz, Combinatorial optimization algorithms, Prentice–Hall, Englewood Cliffs, N.J., 1982.

[3] A.V. Aho, J.E. Hopcroft, J.D. Ullman, The design and analysis of computer algorithms, Addison–Wesley, Reading, Mass., 1974.

[4] C. Berge, Graphs and hypergraphs, North–Holland Publishing Company, 1973.