Course "Trees – The Ubiquitous Structure in Computer Science and Mathematics", JASS'08

The Number of Spanning Trees in a Graph

Konstantin Pieper

April 28, 2008

1 Introduction

In this paper I am going to describe a way to calculate the number of spanning trees by arbitrary weight by an extension of Kirchhoff's formula, also known as the matrix tree theorem. The ultimate goal is to describe an algorithm that calculates the number of minimal spanning trees of a graph on n vertices in $O(M(n))$, where $M(n)$ is the time required to multiply two $n \times n$ matrices. Most of the results are due to Broder and Mayr [2].

1.1 Preliminaries

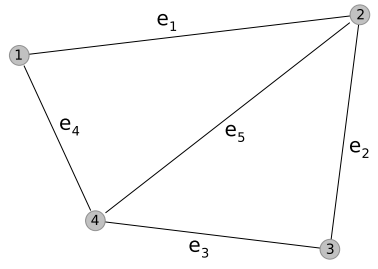
For the following theorems it is useful to introduce the "incidence matrix". It naturally arises in discrete optimization problems, as you can use it to formulate them as linear programs.

Definition 1. Let $G = (V, E)$ with $V = \{1, \dots, n\}$ and $E = \{e_1, \dots, e_m\}$ be a directed graph. Then the incidence matrix $S_G \in M(n, m)$ of G is defined as:

$$(S_G)_{i,j} := \begin{cases} 1 & \text{if } e_j \text{ ends in } i \\ -1 & \text{if } e_j \text{ starts in } i \\ 0 & \text{else} \end{cases}$$

Remark 1. For an undirected graph G every $S_{\bar{G}}$ of some arbitrarily oriented directed variant $\bar{G}$ of G can be taken as the incidence matrix.

Example 2.



$$S_G = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ -1 & -1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 \end{pmatrix}$$

In the following theorems we are going to exploit the following property of the incidence matrix:

Theorem 3. *The rank of the incidence matrix of a graph on n vertices is:*

$$\textit{rank}(S_G) = n - |\text{“weakly” connected components of } G|$$

(“weakly” means not taking into account direction of the edges)

Proof. Reorder the edges and vertices so that:

$$S_G = \begin{pmatrix} S_{G_1} & & \dots & 0 \\ & S_{G_2} & & \vdots \\ \vdots & & \ddots & \\ 0 & \dots & & S_{G_r} \end{pmatrix}$$

Thus we have reduced the problem to connected graphs. The rest is left as an exercise for the reader – The idea is to remove edges that are parts of circles (this does not change the rank of S_G) and to show that for trees S_G has full rank. □

Evidently the incidence matrix can not have maximal rank. Often is more useful to operate on matrices that are at least non-singular in one dimension. Thus we have to modify the incidence matrix:

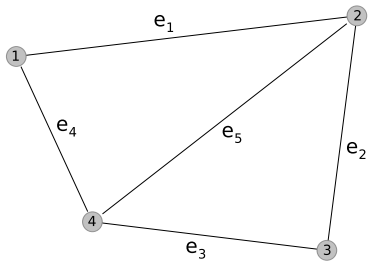
Remark 2. Since $(1, \dots, 1) \cdot S_G = 0$, we can remove an arbitrary row from S_G without losing information.

Definition 4. For every $A \in M(n, m)$ define $\tilde{A} \in M(n-1, m)$ as A without the n-th row.

In the following we are going to count spanning trees – so we can always assume a connect graph G. In this case, $\tilde{S}_G$ has maximal rank.

2 Matrix-Tree Theorems

Casually speaking, the following formulas use the determinant to produce all possible selections of $n-1$ edges and check whether the resulting subgraph is connected. To do this we need to construct a square matrix which contains linear combinations of the columns of S_G.



Example 5.

$\tilde{S}_G \cdot \tilde{S}_G^T =$

$$\begin{pmatrix} -1 & 0 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{pmatrix}$$

2.1 Kirchhoff's theorem

Using the incidence matrix it is easy to state the following famous result, which is mentioned in many books about combinatorics or graph theory (see e.g. [3]).

Theorem 6 (Kirchhoff)**.** *The number of spanning trees of a graph G can be calculated as:*

$$\det(D_G) \text{ where } D_G = \tilde{S}_G \cdot \tilde{S}_G^T$$

Remark 3.

$$(D_G)_{i,j} = \begin{cases} \deg(i) & \text{if } i = j \\ -1 & \text{if } \{i,j\} \in E \\ 0 & \text{else} \end{cases}$$

The shortest way to prove this theorem is probably to use the Cauchy-Binet-Formula [3] for $\det(A \cdot B)$ where A and B are not square. It allows to express the determinant in terms of the determinants of their minors. We are not going to use it here but instead do a more elementary proof. Let us start with some trivial considerations:

Lemma 7. *Let $T = (V, E)$ be a directed tree that is rooted at n. We can order E so that e_i ends in i.*

Proof. Take $e_i := (p(i), i)$, where $p(i)$ is the parent of i. □

Remark 4. Every undirected tree on V has exactly one undirected variant that is rooted at n. So for constructing/counting spanning trees we only have to consider graphs with $i \in e_i$.

Lemma 8. *Let $G = (V, E)$ with $|E| = n - 1$ be a directed graph which is not a tree. Then $\det(\tilde{S}_G) = 0$.*

Proof. Since $|E| = n - 1$, G is not "weakly" connected. So $\operatorname{rank}(\tilde{S}_G) = \operatorname{rank}(S_G) \leq n - 2$. □

Lemma 9. *Let $T = (V, E)$ be a tree with $e_i \in E$ ending in $i \in V$. Then* $\det(\tilde{S}_T) = 1$.

Proof. Order the vertices (and edges simultaneously – e_i has to end in i) so that $p(i) > i$. Then,

$$\tilde{S}_T = \begin{pmatrix} 1 & * & \dots & * \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & * \\ 0 & \dots & 0 & 1 \end{pmatrix}$$

□

Now we are ready to prove the theorem:

Proof of Kirchhoff's theorem. We observe that the i-th column of D_G is the sum of "incidence vectors" that correspond to edges in G that have endpoints in the i-th vertex. If we now use the linearity of the determinant in every column we obtain:

$$\det(D_G) = \sum_{H \in \mathcal{H}} \det(\tilde{S}_H)$$

where $\mathcal{H}$ is the set of all subgraphs of G which correspond to a selection of $n-1$ edges, with e_i ending in i.

Figure 1: First column of D_G

$$D_{\cdot,1} = \begin{pmatrix} 3 \\ -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

Figure 2: Expansion of the determinant

$$\det(D) = \begin{vmatrix} 3 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 4 \end{vmatrix}$$

$$= \begin{vmatrix} 1 & -1 & -1 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{vmatrix} + \begin{vmatrix} 1 & -1 & -1 \\ 0 & 4 & -1 \\ -1 & -1 & 4 \end{vmatrix} + \begin{vmatrix} 1 & -1 & -1 \\ 0 & 4 & -1 \\ 0 & -1 & 4 \end{vmatrix} = \dots$$

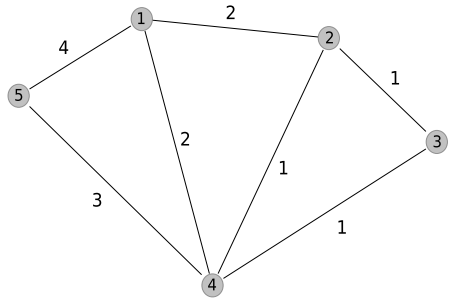
To prove the theorem it is now sufficient to use the preceding lemmata. □

2.2 Extension towards MST

Now we know how to obtain the number of spanning trees via calculation of an n-dimensional determinant. This theorem can be generalized towards weighted graphs by extending the base ring of the incidence matrix.

Definition 10. For a weighted graph G with edge weights $w_{i,k}$ let $S_G(x)$ be the incidence matrix S_G with every column corresponding to $(i,k) \in E$ rescaled by $x^{w_{i,k}}$.

$$(S_G(x))_{i,j} = \begin{cases} x^{w_{k,i}} & \text{if } e_j = (k,i) \\ -x^{w_{i,k}} & \text{if } e_j = (i,k) \\ 0 & \text{else} \end{cases}$$



Example 11 (Toy graph H).

$$S_G(x) = \begin{pmatrix} -x^2 & -x^2 & -x^4 & 0 & 0 & 0 & 0 \\ x^2 & 0 & 0 & -x & -x & 0 & 0 \\ 0 & 0 & 0 & x & 0 & -x & 0 \\ 0 & x^2 & 0 & 0 & x & x & -x^3 \\ 0 & 0 & x^4 & 0 & 0 & 0 & x^3 \end{pmatrix}$$

Now we only have to construct a square matrix in the same way as above and apply the determinant. For every spanning tree we will then get x to the power of his weight.

Theorem 12 (Matrix-Tree Theorem for weighted graphs)**.** *The generating function of the number of spanning trees by weight w is the determinant of*

$$D_G(x) = \tilde{S}_G(x) \cdot \tilde{S}_G^T$$

where $\tilde{S}_G(x)$ and $\tilde{S}_G$ are defined as above. In other words:

$$\det\left(D_G(x)\right) = \sum_{w=0}^{\infty} |ST \text{ by weight } w| \cdot x^w$$

Example 13. For our toy graph H (see above) we get:

$$D_G(x) = \begin{pmatrix} x^4 + 2x^2 & -x^2 & 0 & -x^2 \\ -x^2 & x^2 + 2x & -x & -x \\ 0 & -x & 2x & -x \\ -x^2 & -x & -x & x^3 + x^2 + 2x \end{pmatrix}$$

$$\det\left(D_G(x)\right) = 2x^{10} + 5x^9 + 8x^8 + 6x^7$$

Remark 5. It is easy to check that we can also write down $D_G(x)$ for any given graph G directly:

$$(D_G(x))_{i,j} = \begin{cases} \sum_{\{i,k\}\in E} x^{w_{i,k}} & \text{if } i = j \\ -x^{w_{i,j}} & \text{if } \{i,j\} \in E \\ 0 & \text{else} \end{cases}$$

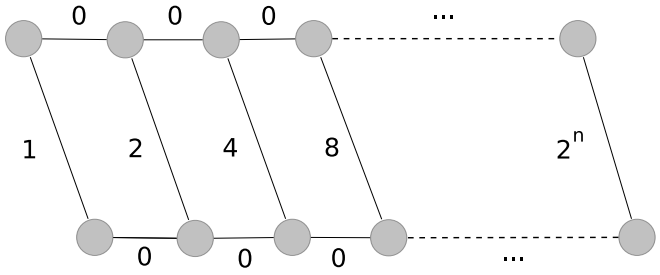
Proof. As above. Here each spanning tree by weight w contributes x^w to the determinant of $D_G(x)$. □

3 Algorithms

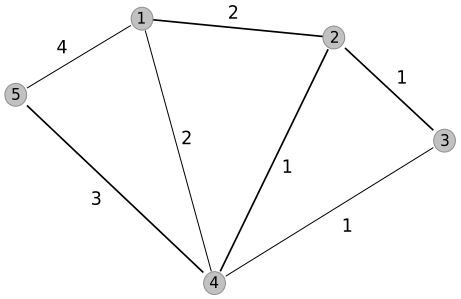
Now we are going to see some implementation ideas of the above results. In the unweighted case not much is to do. Efficient algorithms for calculating the determinant over the real numbers are well known: Simply compute an LU-factorization $D_G = L \cdot U$ and calculate the product of the diagonal elements of U. This is possible in $O(M(n))$, where $M(n)$ is the time required to multiply two $n \times n$ matrices [1]. Note however that we have to account addition and multiplication on the entries of D_G as elementary operations to obtain this bound. As the complete graph on n vertices has $n^{(n-2)}$ spanning trees, our algorithm has to operate on numbers of this magnitude.

3.1 Computing the number of MSTs

Clearly we should not try to calculate the polynomial $\det(D_G(x))$ in the weighted case. If we consider the following example graph on $2n$ vertices, we see that $\det(D_G(x))$ can have $\Omega(2^n)$ coefficients.



Thus we restrict ourselves to only compute the number of minimal spanning trees (i.e. the coefficient of the minimum degree monomial). If $w_{\min}$ is the minimal weight for a spanning tree, clearly $x^{w_{\min}}$ must divide $\det(D_G(x))$. The idea is to factor out the minimum degree monomial of each column and use the linearity of the determinant.



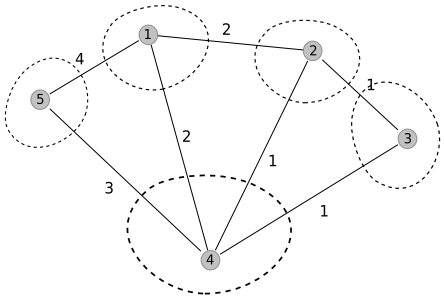
Example 14 (Factor $\det(D_G(x))$).

$\det(D_G(x)) =$

$$\begin{vmatrix} x^4+2x^2 & -x^2 & 0 & -x^2 \\ -x^2 & x^2+2x & -x & -x \\ 0 & -x & 2x & -x \\ -x^2 & -x & -x & x^3+x^2+2x \end{vmatrix} = x^5 \cdot \begin{vmatrix} x^2+2 & -x & 0 & -x \\ -1 & x+2 & -1 & -1 \\ 0 & -1 & 2 & -1 \\ -1 & -1 & -1 & x^2+x+2 \end{vmatrix}$$

We observe that the product of the minimum degree monomials can be a strict divisor of $x^{w_{\min}}$. In the above example we can only factor out x^5 but we see that any spanning tree of H has to have at least weight 7. We will see that there is a way to modify $D_G(x)$ so we can use this trick.

The entries of the i'th column of $D_G(x)$ correspond to edges in the cut $(i, \mathcal{N}(i))$:



So we only have to change the cuts to achieve our gaol.

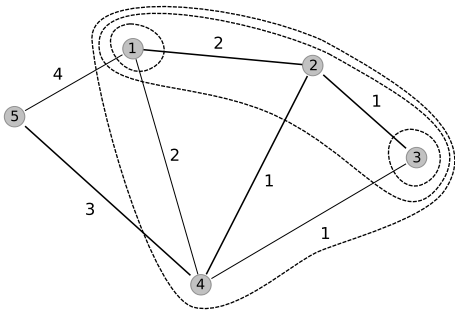
Theorem 15. *If we have a minimal spanning tree T of G, we can modify $D_G(x)$ (without changing the determinant) so that the product of the minimum degree monomials of each column is w_{min}.*

Algorithm A: Modify $D_G(x)$

T := mst(G)
$D'(x)$:= $D_G(x)$
while $T \neq \{\}$ do
    i := arbitrary leaf of T with $i \neq n$
    p := parent of i
    add the i-th column in D' to the p-th column
    $T := T \setminus \{i\}$
od

If we execute Algorithm A we obtain a matrix $D'(x)$ that contains linear combinations of the columns of $D_G(x)$. Let $\sigma(i) \subset V$ denote the vertices in the subtree of T that is rooted at $i \in V$ (consider T as a tree that is rooted at n). Then the i'th column of $D'(x)$ contains the sum of the columns of $D_G(x)$ with index in $\sigma(i)$.

Example 16 (MST of H and corresponding $\sigma(i)$).



$$\det(D'(x)) = \begin{vmatrix} x^4+2x^2 & x^4+x^2 & 0 & x^4 \\ -x^2 & x & -x & 0 \\ 0 & x & 2x & 0 \\ -x^2 & -x^2-2x & -x & x^3 \end{vmatrix} = x^7 \cdot \begin{vmatrix} x^2+2 & x^3+x & 0 & x \\ -1 & 1 & -1 & 0 \\ 0 & 1 & 2 & 0 \\ -1 & -x-2 & -1 & 1 \end{vmatrix}$$

Lemma 17. *The i-th column of $D'(x)$ contains the sum of the columns in $D_G(x)$ corresponding to $\sigma(i)$. Writing out the entries of $D'(x)$ we get: For $i \notin \sigma(j)$*

$$D'_{i,j}(x) = -\sum_{\{i,k\} \in E : k \in \sigma(j)} x^{w_{k,i}}$$

For $i \in \sigma(j)$ the above cancels with $D_{i,i}$

$$D'_{i,j}(x) = \sum_{\{i,k\} \in E : k \notin \sigma(j)} x^{w_{k,i}}$$

Using the identities above it is easy to see why Theorem 15 holds.

Proof. So the j'th column of D' contains only terms corresponding to edges in the cut $(\sigma(j), V \setminus \sigma(j))$. The only edge of T in the j'th cut is $(p(j), j)$. All the

other edges from the cut must have higher weight because T is minimal. Thus we can factor exactly $w_{p(j),j}$ from column j:

$$\prod_{i=1}^{n-1} w_{p(i),i} = w_{\min}$$

□

So if we calculate $D'(x)$ for a minimal spanning tree and factorize $D'(x) = x^{w_{\min}} \cdot \tilde{D}(x)$, we can obtain the number of spanning trees by evaluating $\det(\tilde{D}(0))$.

3.2 Optimizations

The runtime of our implementation so far is $O(mn + M(n)) \cdot O(M(n))$ can be thought of as "$O(n^{2+\epsilon})$" [1]. We can further optimize the $O(mn)$ part by

- only calculating the minimum degree monomials for each entry.
- calculating the negative and positive entries separately.

For convenience we sort E so that $p(j) > j$. We also precompute $\sigma(j)$ for all $j \in \{1 \dots n-1\}$ and initialize D' with zeros.

3.2.1 D' for $i \notin \sigma(j)$

For $i \notin \sigma(j)$ no entries cancel – the naive approach works.

Algorithm B1: Negative Entries of D'

```
for j = 1 to n − 1 do
    for i ∉ σ(j) do
        D'_{i,j} := min_d(D_G(x)_{i,j} + Σ_{k child of j in T} D'_{i,k})
        /* min_d computes the minimum degree monomial */
    od
od
```

This is $O(n^2)$.

3.2.2 D' for $i \in \sigma(j)$

For $i \in \sigma(j)$ we use the explicit formula

$$D'_{i,j} = \sum_{\{i,k\} \in E : k \notin \sigma(j)} x^{w_{k,i}}$$

and run over the rows first.

A different way to state the relation $i \in \sigma(j)$ is to call j an ancestor of i. Let $j_1 < j_2 < \cdots < j_s = n$ denote the ancestors of i. We are going to use the fact that the sets $\sigma(j_p)$ form a tower: $\sigma(j_1) \subset \sigma(j_2) \subset \cdots \subset \sigma(j_s)$

So for every column i we can walk along the ancestors of i in increasing order and have to find the minimum degree monomial of $\sum_{k \in \mathcal{N}(i) \setminus \sigma(j_p)} x^{w_{i,k}}$ ($\mathcal{N}(i)$ are the neighbors of i in G). But since $\mathcal{N}(i) \setminus \sigma(j_p)$ is decreasing, we can use the following trick to do this efficiently: We sort $\mathcal{N}(i)$ in increasing weight order and within each weight class so that $k \in \sigma(j_p)$ with small p come first. For every entry we also precompute the number of elements by same weight following it.

Algorithm B2: Positive Entries of D'

```
for i = 1 to n − 1 do
    L := sort(N(i))
    Z := same_weight_after(L)
    p := 1
    for j = 1 to n − 1, i ∈ σ(j) do
        while p ≤ |L| and L[p] ∈ σ(j) do
            p := p+1;
        od
        if p ≤ |L| and L[p] ∉ σ(j)
            D'_{i,j} := Z[p]·x^{w_{i,L[p]}}
        else
            D'_{i,j} := 0
        fi
    od
od
```

This is $O(n^2 \log n)$. Thus the whole algorithm is $O(M(n))$.

References

[1] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 2 edition, 1975.

[2] Andrei Z. Broder and Ernst W. Mayr. Counting minimum weight spanning trees, 1997.

[3] Shimon Even. *Algorithmic Combinatorics*. Macmillan, 1973.