COMBINATORICA
Akadémiai Kiadó - Springer-Verlag

COMBINATORICA 12 (1) (1992) 45–51

SPANNING TREES OF EXTENDED GRAPHS

A. K. KELMANS

Received May 15, 1989

A generalization of the Prüfer coding of trees is given providing a natural correspondence between the set of codes of spanning trees of a graph and the set of codes of spanning trees of the *extension* of the graph. This correspondence prompts us to introduce and to investigate a notion of *the spanning tree volume* of a graph and provides a simple relation between the volumes of a graph and its extension (and in particular a simple relation between the spanning tree numbers of a graph and its uniform extension). These results can be used to obtain simple purely combinatorial proofs of many previous results obtained by the Matrix-tree theorem on the number of spanning trees of a graph. The results also make it possible to construct graphs with the maximal number of spanning trees in some classes of graphs.

1. Prüfer [8] gave an interesting method of coding labeled trees which provides a one-to-one correspondence between the set of spanning trees of the complete graph K_n with n vertices and the set of words with $n-2$ letters from an n-element set. An immediate consequence is the well known Cayley formula for the number $t(K_n)$ of spanning trees of K_n: $t(K_n) = n^{n-2}$. The Prüfer coding was generalized to enumerate the spanning trees of bipartite graphs [9] and of k-partite graphs [7].

In this paper we generalize the Prüfer coding to enumerate the spanning trees of a graph obtained from another graph by a special operation called *extension*.

The suggested coding of spanning trees of a graph extension (see 3) posesses the following properties: the set of codes of spanning trees of an extended graph can be simply described in terms of the set of codes of spanning trees of the extendible graph (4 and 7). From this description it becomes clear how each spanning tree of a graph is "multiplied" in the set of spanning trees of the extension of the graph. As a result we obtain a natural map of the spanning tree set of the extended graph onto the spanning tree set of the original graph (see 9).

The previous results for bipartite graphs [9] and for k-partite graphs [7] are particular cases of the results below when the extendible graph is the complete graph with two and with k vertices respectively.

A structure of this map prompts us to introduce the notion of the *spanning tree volume* of a graph with weighted vertices and makes it possible to reveal a natural and simple interconnection between volumes of a weighted graph and its weighted extension (see 10 and 11). If the weight of each vertex of a graph is 1 then the spanning tree volume turns into the spanning tree number of the graph. Hence we obtain in particular a simple relation between the number of spanning trees of a graph and of its uniform extension (see 12) and also of a bipartite graph and of its bipartite uniform extension (see 13).

The main definitions and notations are given in 2. In Section 3 Procedure P for coding spanning trees of a graph extension is described. In Sections 4 and 8

AMS subject classification code (1991): 05 C 05, 05 C 30

properties of codes obtained by Procedure P are established. Procedure P^{-1} for decoding is described in Section 5. Mutual properties of Procedures P and P^{-1} are given in Section 6.

Each section contains at most one Proposition. So Proposition N means the Proposition in Section N.

2. Undirected graphs without multiple edges but possibly with loops are considered [1]. As usual $V(G)$ and $E(G)$ are the sets of vertices and edges respectively. Let $d(v, G)$ denote the degree of a vertex v in a graph G (i.e. the number of edges incident to v in G; a loop is counted once). Let (a, b) denote an edge with end-vertices a and b. For a given $v \in V(G)$, let $G - v$ denote the graph obtained from G by deleting the vertex v and the edges incident to v.

A *spanning tree* (or simply a *tree*) of a connected graph G is a connected subgraph of G without cycles having $V(G)$ as the vertex set. Spanning trees T_1 and T_2 are *distinct* if $E(T_1) \neq E(T_2)$. Let $\mathcal{T}(G)$ and $t(G)$ denote the set and the number of spanning trees of a graph G respectively.

Let X be a finite set of elements. For $v \in V(G)$ put $X(v) = \{vx: x \in X_v\}$ where $X_v \subseteq X$. Put $\mathcal{X} = \{X(v): v \in VG\}$.

Given G and $\mathcal{X}$, let us construct the new graph $\Gamma = G(\mathcal{X})$ as follows:

(a1) $V(\Gamma) = \cup\{X(v) : v \in V(G)\}$ and

(a2) for a pair of vertices v_1x_1 and v_2x_2 of Γ $(v_1x_1, v_2x_2) \in E(\Gamma)$ if and only if $(v_1, v_2) \in E(G)$.

We call $G(\mathcal{X})$ the *$\mathcal{X}$-extension* of G. Put $|V(G)| = n$, $|X(v)| = k(v)$ and $|V(G(\mathcal{X}))| = s$ so that $s = \sum\{k(v) : v \in V(G)\}$.

3. Assume $|V(G)| = n \geq 2$. Let T be a spanning tree of $G(\mathcal{X})$. Let us construct a code $P(T)$ of a tree T similar to Prüfer code [1]. The code $P(T)$ consists of $n + 1$ words with letters from $V(G(\mathcal{X}))$. Except for one of these words, the words are in one-to-one correspondence with the vertices of G. Thus,

$$P(T) = \{w\} \cup \{w(v) : v \in VG\},$$

The word w consists of $n - 2$ letters while $w(v)$ consists of $k(v) - 1$ letters.

In particular if $n = 2$ then w is an empty word : $w = \emptyset$; if $k(v) = 1$ then $w(v)$ is an empty word : $w(v) = \emptyset$.

Hence if $|V(T)| = |V(G(\mathcal{X}))| = 2$ then $P(T)$ is an empty code, i.e. each word of $P(T)$ is empty.

From here on we suppose that linear orders $\succ$ are defined on the sets $V(G)$ and X (for example $V(G)$ and X are the sets of distinct integers). Let $v_1, v_2 \in VG$ and $x_1, x_2 \in X$. We say that v_1x_1 *is greater then* v_2x_2: $v_1x_1 \succ v_2x_2$ if $v_1 \succ v_2$ or $v_1 = v_2$ and $x_1 \succ x_2$.

Procedure P for constructing the code $P(T)$ of a tree T from $\mathcal{T}(G)$ makes $s - 2$ steps. At each step the procedure adds a new letter to some of the words. If $s = 2$ then by definition $P(T) = \emptyset$. Roughly speeking the procedure works as follows. Find the minimal (in the sense of $\succ$) end-vertex vx of the tree T. Let $v'x'$ be the (unique) vertex of T adjacent to the vertex vx. If the word $w(v)$ has not been "filled" yet (i.e. in this case $k(v) > 1$) then add the letter $v'x'$ to $w(v)$ from right hand. If $w(v)$ has already been filled (i.e. in this case $k(v) = 1$) then add $v'x'$ to the word w. After that delete from T the vertex vx and the edge of T incident to vx. Do such steps until T has two verticies (i.e. do $s - 2$ steps).

Below we give a complete description of Procedure P. The content of the square brackets illustrates the recursive character of the procedure. The recursiveness of the procedure is used in the inductive proofs of the results in Sections 4 and 6.

Procedure P.

Input: *A spanning tree* T *of a graph* $G(\mathcal{X})$.

Output: *A collection of words* $W' = \{w'\} \cup \{w(a) : a \in V(G)\}$.

(p0) *Put* $N := 0$, $T' := T$, $w' := \emptyset$, $w(a) := \emptyset$ *for* $a \in V(G)$
[Put $G' := G$, $X'(a) := X(a)$ *for* $a \in V(G)$, $\mathcal{X}' = \{X'(a) : a \in V(G)\}$ *so that* T' *is a spanning tree of the graph* $G'(\mathcal{X}')$*].*

(p1) *Find in the tree* T' *the minimal (in the sense of* $\succ$*) end-vertex* vx*. Find the (unique) vertex* $v'x'$ *of* T' *adjacent to* vx *so that* $(vx, v'x')$ *is the only edge of* T' *incident to* vx*). Put* $N := N + 1$ *and* $T' := T' - vx$.
If $|w(v)| < k(v) - 1$ *then put* $w(v) := w(v), v'x'$ *(i.e. add the letter* $v'x'$ *to the end of the word* $w(v)$ *)*
[and put $X'(v) := X(v) - vx$*].*
If $|w(v)| = k(v) - 1$ *then put* $w := w, v'x'$
[and put $X'(v) := X'(v) - vx$ *(so that new* $X'(v) = \emptyset$*),* $\mathcal{X}' := \mathcal{X}' - \{X'(v)\}$ *and* $G' := G' - vx$ *. As a result we have as before :* T' *is a spanning tree of the graph* $G'(\mathcal{X}')$*].*

(p2) *If* $N = s - 2$ *then stop. If* $N < s - 2$ *then go to (p1).*

Evidently if $k(v) = 1$ then $w(v) = \emptyset$. If in particular $s = n$ (i.e. $k(v) = 1$ for any $v \in V(G)$), then $G(\mathcal{X}) = G$, T is a spanning tree of G and $P(T)$ consists of a single word w. It follows from Procedure P that in this case $P(T)$ is nothing but the Prüfer code of T with respect to the mentioned above linear order of the set $V(G)$.

4. Let $\mathcal{W}(G,\mathcal{X})$ denote the set of collections $W = \{w\} \cup \{w(v) : v \in VG\}$ with the following properties :

(c0) w and $w(v)$ for $v \in VG$ are words with letters from $V(G(\mathcal{X}))$,
(c1) the word w consists of $n - 2$ letters,
(c2) the word $w(v)$ for $v \in VG$ consists of $k(v) - 1$ letters,
(c3) if vx is a letter of $w(u)$ then $(u, v) \in EG$ so that the letters of $w(u)$ always belong to $\cup\{X(v) : (u, v) \in E(G)\}$ and
(c4) if $w = (v_1x_1, \ldots, v_{n-2}x_{n-2})$ then $(v_1, \ldots, v_{n-2})$ is the Prüfer code of some spanning tree of the graph G; let us denote this tree of G by $\tau'(W)$ so that $P(\tau'(W)) = (v_1, \ldots, v_{n-2})$.

Lemma. *Let* T *be a spanning tree of the graph* $G(\mathcal{X})$ *and* $P(T)$ *be the code of* T *obtained by Procedure* P*. Then* $P(T) \in \mathcal{W}(G,\mathcal{X})$.

Proof. Properties (c0)–(c3) of $P(T)$ follow immediately from Procedure P. Property (c4) can be proved easily by induction on $s = |V(G(\mathcal{X}))|$ using the recursive character of Procedure P. ∎

5. It turns out that converse of the Lemma is also true. Namely we give Procedure P^{-1} which for any W from $\mathcal{W}(G,\mathcal{X})$ constructs a spanning tree T of $G(\mathcal{X})$ such that $P(T) = W$.

Roughly speeking Procedure P^{-1} works as follows: Find in $V = VG(\mathcal{X})$ the $\succ$-minimal vertex-letter vx which does not belong to W. Delete vx from V. If the

word $w(v)$ from W is not empty then take the first letter $v'x'$ of $w(v)$. Connect the vertices vx and $v'x'$ by an edge and delete the letter $v'x'$ from $w(v)$. If $w(v)$ is empty but w is not empty then take the first letter $v'x'$ of w . Connect the vertices vx and $v'x'$ by an edge and delete $v'x'$ from w. If both $w(v)$ and w are empty (it can be proved that in this case V consists of exactly two vertices, say a and b), then connect a and b by an edge and stop. Do the same with new W and V. Repeat this step until V has two vertices.

Below we give a complete description of Procedure P^{-1}. The content of the square brackets is given to illustrate the recursive character of the procedure. The recursiveness of the procedure is used in the proof of the results in Section 6.

Let $\Gamma = G(\mathcal{X})$ where $\mathcal{X} = \{X(a) : a \in VG\}$. We suppose that $|VG| \geq 2$.

Procedure P^{-1}.

Input: *A collection of words $W = w \cup \{w(a) : a \in V(G)\}$ from $\mathcal{W}(G,\mathcal{X})$ and $V = V(\Gamma)$.*

Output: *a graph T with $V(T) = V(\Gamma)$ and $E(T) = E$.*

(p0) *Put $E := \emptyset$, $V' := V$, $w' := w$, $w'(a) := w(a)$ for $a \in V(G)$, $W' := \{w'\} \cup \{w'(a) : a \in V(G)\}$.*
[Put $G' := G$, $\Gamma' := \Gamma$, $X'(a) := X(a)$ for $a \in V(G)$, $\mathcal{X}' = \{X'(a) : a \in VG\}$ so that $\Gamma' = G'(\mathcal{X}')$, $V' = V(\Gamma')$ and $W' \in \mathcal{W}(G',\mathcal{X}')$].

(p1) *If $|V'| = 2$, say $V' = \{a,b\}$, then put $E := E \cup \{(a,b)\}$ and stop (here (a,b) is the edge with the end-vertices a and b). If $|V'| > 2$ then go to (p2). (Since $|V(G)| \geq 2$, we have $|V(\Gamma)| \geq 2$).*

(p2) *Find in V' the $\succ$-minimal vertex-letter vx which does not belong to W'. Put $V' := V' - vx$.*
[Put $\Gamma' := \Gamma - vx$ and $X'(v) := X'(v) - vx$]. If $w'(v) \neq \emptyset$, then put $E := E \cup (vx, v'x')$ and $w'(v) := w'(v) - v'x'$ where $v'x'$ is the first letter of the word $w'(v)$, $(vx, v'x')$ is the edge with the end-vertces vx , $v'x'$ and $w'(v) - v'x'$ is the word obtained from $w'(v)$ by deleting the letter $v'x'$. If $w'(v) = \emptyset$ but $w \neq \emptyset$ then put $E := E \cup (vx, v'x')$ and $w' := w' - v'x'$ where $v'x'$ is the first letter of the word w'.
[If $w'(v)$ or w' becomes empty then delete it from W'. If $X'(v)$ becomes empty then delete it from $\mathcal{X}'$. As a result we have as above: $V' := V(\Gamma')$, $\Gamma' = G'(\mathcal{X}')$ and $W' \in \mathcal{W}(G',\mathcal{X}')$].
Go to (p1).

6. The recursive character of Procedures P and P^{-1} make it possible to prove easily by induction the following

Theorem.

(a1) *$P^{-1}(W)$ is a spanning tree of $G(\mathcal{X})$ for any $W \in \mathcal{W}(G,\mathcal{X})$,*

(a2) *$P(P^{-1}(W)) = W$ for any $W \in \mathcal{W}(G,\mathcal{X})$, and*

(a3) *$P^{-1}(P(T)) = T$ for any $T \in \mathcal{T}(G(\mathcal{X}))$.* ∎

7. Corollary. *$P: \mathcal{T}(G(\mathcal{X})) \to \mathcal{W}(G,\mathcal{X})$ is a one-to-one map of the set $\mathcal{T}(G(\mathcal{X}))$ of all spanning trees of the graph $G(\mathcal{X})$ onto the set $\mathcal{W}(G,\mathcal{X})$ of collections of the words with properties (c0)–(c4).* ∎

8. It is easy to prove the following

Lemma. *Let $P(T)$ be the Prüfer code of a tree T. Then a letter-vertex v from $V(T)$ occurs in $P(T)$ exactly $d(v,T)-1$ times (here $d(v,T)$ is the degree of v in T).* ∎

9. In Section 4 the map $\tau' : \mathcal{W}(G,\mathcal{X}) \to \mathcal{T}(G)$ has been defined. Due to Lemma 4 the map $\tau = \tau' P : \mathcal{T}(G(\mathcal{X})) \to \mathcal{T}(G)$ is correctly defined.

From Theorem 6 we have

Corollary. *$\tau : \mathcal{T}(G(\mathcal{X})) \to \mathcal{T}(G)$ is a map of the set $\mathcal{T}(G(\mathcal{X}))$ of spanning trees of $G(\mathcal{X})$ onto the set $\mathcal{T}(G)$ of spanning trees of G (so that for any $D \in \mathcal{T}(G)$ there exists a tree $T \in \mathcal{T}(G(\mathcal{X}))$ such that $D = \tau(T)$).* ∎

10. Let H be a graph and $h : V(H) \to \mathsf{K}$ be an arbitrary function where K is a commutative ring. Suppose first that H is a tree T.

Put

$$R(T,h) = \prod \{h(v)^{d(v,T)-1} : v \in V(T)\}.$$

For a graph H put

$$R(H,h) = \sum \{R(T,h) : T \in \mathcal{T}(H)\}.$$

$R(H,h)$ is called the *spanning tree volume of a weighted graph* (H,h).

Note that $\{h(v) : v \in V(H)\}$ may be considered as *indeterminates* and so $R(H,h)$ may also be considered as the *spanning tree generating function of a graph* H.

Let $f : V(G(\mathcal{X})) \to \mathsf{K}$ and $g : V(G) \to \mathsf{K}$.

Theorem 6 makes it possible to reveale the interconnection between $R(G,g)$ and $R(G(\mathcal{X}),f)$ under some relation between g and f.

Given $D \in \mathcal{T}(G)$, put

$$r(D,\mathcal{X},f) = \sum \{R(T,f) : T \in \mathcal{T}(G(\mathcal{X})) \quad \text{and} \quad \tau(T) = D\}.$$

Now put

$$q(v,G,\mathcal{X},g) = \Big(\sum \{g(a) : a \in V(G) \quad \text{and} \quad (a,v) \in E(G)\}\Big)^{k(v)-1}$$

and

$$Q(G,\mathcal{X},g) = \prod \{q(v,G,\mathcal{X}g) : v \in V(G)\}.$$

Finally put

$$f^{\mathcal{X}}(v) = g(v) = \sum \{f(vx) : vx \in X(v)\}$$

for $v \in V(G)$.

From Lemma 8 and Corollary 9 we have

Corollary. *Let $f : V(G(\mathcal{X})) \to \mathsf{K}$ and $D \in \mathcal{T}(G)$. Then*

$$r(D,\mathcal{X},f) = Q(G,\mathcal{X},f^{\mathcal{X}}) \cdot R(D,f^{\mathcal{X}}).$$ ∎

11. Theorem. *Let $f : V(G(\mathcal{X})) \to \mathsf{K}$. Then*

$$R(G(\mathcal{X}),f) = Q(G,\mathcal{X},f^{\mathcal{X}}) \cdot R(G,f^{\mathcal{X}}).$$

Proof.

$$R(G(\mathcal{X}), f) = \sum\{R(T, f): T \in \mathcal{T}(G(\mathcal{X}))\}$$
$$= \sum\Big\{\sum\{R(T, f): T \in \mathcal{T}(G(\mathcal{X})) \quad \text{and} \quad \tau(T) = D\}: D \in \mathcal{T}(G)\Big\}$$
$$= \sum\{r(D, \mathcal{X}, f): D \in \mathcal{T}(G)\}.$$

Since $R(G, g) = \sum\{R(D, g): D \in \mathcal{T}(G)\}$, we obtain from Corollary 10 the required assertion. ∎

Note that Theorem 11 revealed and proved by the above described coding can also be proved by the Matrix-tree theorem [1].

12. Put $f \equiv 1$. Then $R(G(\mathcal{X}), f) = t(G(\mathcal{X}))$. In particular suppose that the sets $X(v)$ in $\mathcal{X} = \{X(v): v \in V(G)\}$ have the same cardinality : $|X(v)| = k$ for any $v \in V(G)$. Let $G[k]$ denote the graph $G(\mathcal{X})$ in the case. Then $f^{\mathcal{X}} \equiv k$, $R(T, f^{\mathcal{X}}) = k^{n-2}$ for any $T \in \mathcal{T}(G)$ so that $R(G, f^{\mathcal{X}}) = k^{n-2} \cdot t(G)$. Also $q(v, G, \mathcal{X}, f^{\mathcal{X}}) = (k \cdot d(v, G))^{k-1}$ (here $d(v, G)$) is the degree of v in G and a loop is counted in $d(v, G)$ once). Hence $Q(G, \mathcal{X}, f^{\mathcal{X}}) = (k^n \prod\{d(v, G): v \in V(G)\})^{k-1}$.

Thus Theorem 11 implies

Corollary.

$$t(G[k]) = k^{nk-2} \prod\{d(v, G): v \in V(G)\} \cdot t(G). \quad ∎$$

A particular case of this assertion when each vertex of the original graph has a loop was obtained earlier by means of the Matrix-tree theorem in [2].

13. Put as above $f \equiv 1$ so that $R(G(\mathcal{X}), f) = t(G(\mathcal{X}))$. Now suppose that G is a bipartite graph with parts V_1 and V_2 so that $V(G) = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$ (and therefore G has no loop). Let $|V_i| = n_i$. Suppose also that $|X(v)| = k_i$ for any $v \in V_i$ and $i = 1, 2$. Let $G[k_1, k_2]$ denote the graph $G(\mathcal{X})$ in the case. Then $f^{\mathcal{X}}(v) = k_i$ for $v \in V_i$, $i = 1, 2$, and $R(T, f^{\mathcal{X}}) = k_1^{n_2-1} \cdot k_2^{n_1-1}$ for any $D \in \mathcal{T}(G)$ so that

$$R(G, f^{\mathcal{X}}) = k_1^{n_2-1} k_2^{n_1-1} \cdot t(G).$$

Now

$$q(v, G, \mathcal{X}, f^{\mathcal{X}}) = (k_i \cdot d(v, G))^{k_j-1}$$

for $v \in V_j$ and $\{i, j\} = \{1, 2\}$.

Hence

$$Q(G, \mathcal{X}, f^{\mathcal{X}}) = k_1^{n_2(k_2-1)} k_2^{n_1(k_1-1)}$$
$$\times \prod\{d(v, G)^{k_2-1}: v \in V_2\} \cdot \prod\{d(v, G)^{k_1-1}: v \in V_1\}.$$

Thus from Theorem 11 we have

Corollary.

$$t(G[k_1, k_2]) = k_1^{n_2 k_2 - 1} k_2^{n_1 k_1 - 1}$$
$$\times \prod\{d(v, G)^{k_2-1}: v \in V_2\} \cdot \prod\{d(v, G)^{k_1-1}: v \in V_1\} \cdot t(G). \quad ∎$$

This assertion has earlier been obtained in another way in [2].

In [3] using the Matrix-tree theorem an algorithm was given for obtaining formulas of the spanning tree numbers of so called decomposable graphs. The above coding of trees enable us to obtain the same result in a purely combinatorial way.

Corollaries 12 and 13 together with some previous results in [4, 5, 6] make it possible to construct graphs with the maximal number of spanning trees in some classes of graphs.

References

[1] J. A. BONDY, and U. S. R. MURTY: *Graph Theory with Applications*, New York: Elsevier, London: MacMillan, 1976.

[2] A. K. KELMANS: The number of spanning trees of a graph containing a given forest *Acta Math. Acad. Sci. Hungar.*, **27** (1976), 89–95.

[3] A. K. KELMANS: The number of trees of a graph I, II *Avtomat. i Telemekh*, **3**, (1965), 2194–2204; **2**, (1966), 56–65. (English transl. in Automat. Remote Control **26** (1965); **27** (1966)).

[4] A. K. KELMANS: Operations on graphs that increase the number of their spanning trees, in: Issledovaniya po Diskretnoy optimizacii, Nauka, Moskva 1976, 406–424. (in Russian, MR 56 #3342).

[5] A. K. KELMANS: Comparison of graphs by their number of spanning trees, *Discrete Mathematics* **16** (1976), 241–261.

[6] A. K. KELMANS, and V.N. CHELNOKOV: A certain polynomial of a graph and graphs with extremal number of trees, *J. Combinatorial Theory*, (B) **16** (1974), 197–214.

[7] GY. OLÁH: A problem on the number of trees, *Studia Sci. Math. Hungar.* **3** (1968), 71–80.

[8] A. PRÜFER: Neuer Beweis eines Satzes über Permutationen, *Archiv für Math. u. Phys.* **27** (1918), 142–144.

[9] A. RÉNYI: Új módszerek és eredmények a kombinatorikus analízisben I, *Magyar Tud. Akad. Mat. Fiz. Oszt. Közl.* **16** (1966), 77–105.

Alexander K. Kelmans

RUTCOR, Rutgers University
New Brunswick, NJ 08903
U. S. A.
kelmans@rutcor.rutgers.edu