$k$-Connectivity and decomposition of graphs into forests

Takao Nishizaki\textsuperscript{a}, Svatopluk Poljak\textsuperscript{b,}* \\
\textsuperscript{a} Department of Electrical Communications, Faculty of Engineering, Tohoku University, Sendai, Japan 980 \\
\textsuperscript{b} Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, \\
Malostranske n\\u0161e, 118 00 Praha 1, Czechoslovakia

Received 7 March 1991; revised 13 February 1992

Abstract

We show that, for every $k$-(edge) connected graph $G$, there exists a sequence $T_1, T_2, \ldots, T_k$ of spanning trees with the property that $T_1 \cup T_2 \cup \cdots \cup T_j$ is $j$-(edge) connected for every $j = 1, \ldots, k$. Nagamochi and Ibaraki have recently presented a linear time decomposition procedure by which such a sequence of trees can be constructed. We discuss some properties of this procedure and its relation to the arboricity of a graph.

This paper is motivated by the following question. Given a $k$-(edge) connected graph $G$, find efficiently a spanning subgraph $H$ which is also $k$-(edge) connected, and has a small number of edges. Since the problem of finding $H$ with minimum number of edges is NP-complete by a result of Chung and Graham (see [4, problem GT31]), we are interested in finding a subgraph $H$ with a small (but not necessarily minimal) number of the edges. In fact, there is always such a subgraph with at most $kn$ edges.

In Section 1, we show that, for every $k$-(edge) connected graph $G$, there exists a sequence $T_1, T_2, \ldots, T_k$ of spanning trees with the property that $T_1 \cup T_2 \cup \cdots \cup T_j$ is $j$-(edge) connected for every $j = 1, \ldots, k$. Nagamochi and Ibaraki have recently presented a decomposition procedure by which such a sequence of trees can be constructed in linear time. We discuss some properties of this procedure in Section 2. In particular, we show that the number of resulting partition classes never exceeds $(2e)^{1/2}$ for a connected graph with $e$ edges; however, it can be arbitrarily large with respect to the arboricity of $G$.

We assume a reader to be familiar with the basic notions of the graph theory. For the reference, see the book [1].

*Corresponding author.
1. The existence of tree sequences

We present the vertex connectivity and edge connectivity versions in Theorems 1.1 and 1.2, respectively.

**Theorem 1.1.** Let \( G = (V, E) \) be a \( k \)-connected graph. Then there exists a sequence \( T_1, T_2, \ldots, T_k \) of spanning trees (not necessarily edge disjoint) such that the subgraph formed by \( T_1 \cup T_2 \cup \cdots \cup T_j \) is \( j \)-connected for every \( j = 1, \ldots, k \).

**Proof.** The statement follows from a theorem by Mader [7] (see also [1, Theorem I.4.5]), by which every cycle of a minimally \( k \)-connected graph contains a vertex of degree \( k \). (A \( k \)-connected graph \( G \) is said to be minimally \( k \)-connected if \( G \setminus e \) is not \( k \)-connected for any \( e \in E(G) \).)

Let \( E_1 \subset E_2 \subset \cdots \subset E_k \subset E \) be edge sets chosen so that \( G_j = (V, E_j) \) is minimally \( j \)-connected for every \( j = 1, \ldots, k \). We claim that every set \( E_{j+1} \setminus E_j, j = 1, \ldots, k - 1 \), is a forest. Otherwise let \( C \subset E_{j+1} \setminus E_j \) be a cycle. Since \( C \) contains a vertex of degree \( j + 1 \) in \( G_{j+1} \), then \( G_j \) contains a vertex of degree \( j - 1 \), which is not possible since \( G_j \) is \( j \)-connected. \( \square \)

**Theorem 1.2.** Let \( G = (V, E) \) be a \( k \)-edge connected graph. Then there exists a sequence \( T_1, T_2, \ldots, T_k \) of spanning trees such that the subgraph formed by \( T_1 \cup T_2 \cup \cdots \cup T_j \) is \( j \)-edge connected for every \( j = 1, \ldots, k \).

**Proof.** The statement follows from the following fact. Let \( G_j = (V, E_j) \) be a \( j \)-edge connected spanning subgraph of \( G \), \( j < k \), and let \( F \) be a maximum forest in \( G \setminus E_j \). Then \( G_{j+1} := (V, E_j \cup F) \) is \( (j + 1) \)-edge connected. Assume that \( G_{j+1} \) is not \( (j + 1) \)-edge connected, and let \( S \subset V \) be such that \( |\delta_{G_j}(S)| = |\delta_{G_{j+1}}(S)| = j \leq k - 1 \).

Hence \( F \subset \langle S \rangle \cup \langle V \setminus S \rangle \). Since \( |\delta_{G}(S)| \geq k \), there is an edge \( e \in (E \setminus (E_{j+1} \cup F)) \cap \delta_{G}(S) \), and \( F \cup e \) is a forest, which contradicts the maximality of \( F \). \( \square \)

We recall that the arboricity \( a(G) \) of a graph \( G = (V, E) \) is defined as the minimum number of spanning trees whose union covers the edge set of \( G \). Theorem 1.2 can be slightly strengthened as follows.

**Theorem 1.3.** Let \( G = (V, E) \) be a \( k \)-edge connected graph with the arboricity \( a(G) = a \). Then there exists a sequence \( T_1, T_2, \ldots, T_a \) of spanning trees such that

(i) the subgraph formed by \( T_1 \cup T_2 \cup \cdots \cup T_j \) is \( j \)-edge connected for every \( j = 1, \ldots, k \), and

(ii) \( T_1 \cup T_2 \cup \cdots \cup T_a = E \).

**Proof.** Since \( a(G) = a \), the edge set \( E(G) \) can be decomposed into \( a(G) \) forests \( F_1, F_2, \ldots, F_a \) such that (ii) holds. Move edges from \( F_2 \cup \cdots \cup F_a \) into \( F_1 \) until \( F_1 \) is maximal. Then move edges from \( F_3 \cup \cdots \cup F_a \) into \( F_2 \) until \( F_2 \) is maximal, etc. Since
this is an implementation of the construction of Theorem 1.2, it obviously achieves properties (i) and (ii). □

The above proof was suggested by one of the referees of our paper. Our previous proof was based on the matroid theory (for the reference, see the book [12]). Since it may be interesting to mention this connection, we present our original proof in the following remark.

Remark. Let $M(G)$ be the cycle matroid of $G$, and $M_j(G) = M(G) \cup \cdots \cup M(G)$ ($j$ times) be the matroid union of $j$ copies of $M(G)$, $j = 1, \ldots, \alpha(G)$. We recall that a set is independent in $M_j(G)$ if and only if it can be written as a union of $j$ forests of $G$. The claim follows from the facts that matroid union is a matroid, and that each independent set (of the union) is contained in a base (of the union), which is a maximum independent set. Hence a selection of spanning trees $T_1, \ldots, T_{\alpha(G)}$ such that $T_1 \cup T_2 \cup \cdots \cup T_j$ is a base of $M_j(G)$ satisfies the above Theorem 1.3.

We do not know whether a statement analogous to Theorem 1.3 is valid also for the vertex connectivity. Let us also mention a related result of [6], by which every $k$-edge connected graph contains at least $\lceil (k - 1)/2 \rceil$ disjoint spanning trees.

We will now briefly discuss the question of the complexity of finding the tree sequences whose existence is proved in Theorems 1.1 and 1.2. We start with the edge connectivity case.

The maximum forest $F$, considered in the proof of Theorem 1.2, consists of a spanning tree in each component of $G \setminus E_j$. Since it can be found in $O(m)$ time, there is an $O(km)$ time algorithm to construct a $k$-edge connected spanning subgraph $H$ with at most $kn$ edges. However, it has been proved in [8,9] that this can be also done in $O(m)$ time by their algorithm.

Next we consider the vertex connectivity case. Let $x_G(x, y)$ denote the local connectivity between $x$ and $y$, i.e. the maximum number of openly vertex disjoint paths between two vertices $x$ and $y$ in a graph $G$.

Given a $k$-connected graph $G = (V, E)$, a minimally $k$-connected subgraph $H = (V, F)$ can be constructed by the following procedure.

For $e = xy \in E$ do
   if $x_G(x, y) > k$ then $G := G \setminus xy$;
   $H := G$;

The correctness of the procedure follows from a simple fact that if deletion of an edge $e = xy$ decreases the connectivity of a graph, then it decreases also the local connectivity between $x$ and $y$. Since $x_G(x, y)$ can be computed in $O(m\sqrt{n})$ time by the network flow algorithm, the complexity of the procedure is $O(m^2\sqrt{n})$ for a graph with $n$ vertices and $m$ edges. It has been, for some time, an open question (formulated by the first author), whether the time efficiency can be improved.
Recently, Nagamochi and Ibaraki ([8] and [9]) presented a linear time algorithm which can be used to find the trees of Theorem 1.1, and also Theorem 1.2. We recall this algorithm in the next section. For $k \leq 3$, a linear algorithm has been earlier found in [10]. Some applications of the sparse graph connectivity certificates to parallel algorithms are given in [2].

2. The number of forests in the Nagamochi–Ibaraki decomposition

We recall the original formulation of the Nagamochi–Ibaraki decomposition procedure as it appeared in [8,9].

Procedure FOREST; \{input: $G = (V, E)$, output: $E_1, E_2, \ldots, E_{|E|}\}

\begin{algorithm}
begin
1 $E_1 := E_2 := \cdots := E_{|E|} := \emptyset$;
2 Label all nodes $v \in V$ and all edges $e \in E$ “unscanned”;
3 $r(v) := 0$ for all $v \in V$;
4 while there exists “unscanned” nodes do
\begin{algorithm}
begin
5 Choose an “unscanned” node $x \in V$ with the largest $r$;
6 for each “unscanned” edge $\rho$ incident to $x$ do
\begin{algorithm}
begin
7 $E_{r(x)+1} := E_{r(y)+1} \cup \{e\}$; \{y is the other end node (\neq x) of e\}
8 if $r(x) = r(y)$ then $r(x) := r(x) + 1$;
9 $r(y) := r(y) + 1$;
10 Mark $e$ “scanned”
\end{algorithm}
end;
11 Mark $x$ “scanned”
end;
end.

The main properties of the above procedure can be summarized as follows.

**Theorem 2.1** (Nagamochi and Ibaraki [8,9]). The procedure FOREST decomposes the edge set $E$ of a graph $G = (V, E)$ into forests $E_1, E_2, \ldots, E_{|E|}$ in $O(|E|)$ time. The decomposition has the following properties.

(i) If $G$ is $k$-connected, then $(V, E_1 \cup E_2 \cup \cdots \cup E_j)$ is $j$-connected for every $j = 1, \ldots, k$.

(ii) If $G$ is $k$-edge connected, then $(V, E_1 \cup E_2 \cup \cdots \cup E_j)$ is $j$-edge connected for every $j = 1, \ldots, k$.

We will study the number of nonempty classes which may appear in the decomposition of a graph by the procedure FOREST. It is not difficult to see that if some
decomposition class $E_i$ is empty, then also $E_j = \emptyset$ for all $j = i, i + 1, \ldots, |E|$. The number of nonempty classes may also depend on the initial order of vertices. Let $k(G)$ denote the maximum number of nonempty classes among $E_1, E_2, \ldots, E_{|E|}$ into which $G$ can be partitioned by the procedure FOREST.

Theorem 2.2. We have

$$k(G) \leq (2e - n + 1)^{1/2}$$

for any graph $G$ with $n$ vertices, $e$ edges, and without isolated vertices. (The bound is exact for complete graphs.)

Theorem 2.2 can be proved by induction from the following property of the algorithm.

Lemma 2.3. Let $(E_1, E_2, E_3, \ldots)$ be a partition of $G = (V, E)$ obtained by the algorithm. Then $(E_2, E_3, \ldots)$ is a possible output of the algorithm when applied to the input $G\setminus E_1 = (V, E\setminus E_1)$.

Proof. Let $(x_1, x_2, \ldots, x_n)$ be the order in which the vertices of $G$ were scanned. Then $x_1$ is incident only to edges from $E_1$, and hence it is isolated in $G\setminus E_1$. We claim that $(x_2, x_3, \ldots, x_n)$ is an admissible order of vertices of $G\setminus E_1$ for the algorithm. Let $r(x)$ and $r'(x)$ denote the labels of vertices used when processing $G$ and $G\setminus E_1$, respectively. Let us imagine that both $G$ and $G\setminus E_1$ are processed simultaneously, with a break in $G\setminus E_1$ while an edge belonging to $E_1$ is scanned in $G$. At arbitrary time, we have $r'(x) \leq r(x)$ for all vertices, and equality holds for the vertex $y$ at step 7, because $E_1$ is a spanning forest, and some edge of $E_1$ terminating at $y$ must have been scanned before scanning any edge of $G\setminus E_1$. \xqed

Proof of Theorem 2.2. Without loss of generality, we may assume that the graph $G$ is connected. We prove the statement by the induction on $k$, the number of forests in the decomposition. The statement is trivially valid for $k = 1$, because $G$ is a tree in this case. Assume that $k > 1$, and that the statement is valid for $k - 1$. Let $(E_1, \ldots, E_k)$ be a partition obtained by the procedure. Let us denote by $G' = (V', E')$ the graph obtained from $G$ after deleting the edge set $E_1$, and also deleting the isolated vertices of $G_1\setminus E_1$. Let $p$ be the number of vertices of $G'$; whereas the number of the edges of $G'$ is $e - (n - 1)$. Since $(E_1, \ldots, E_k)$ is a possible output of the procedure, we have

$$k - 1 \leq (2(e - n + 1) - p + 1)^{1/2}$$

by the induction hypothesis. It is not difficult to check that

$$1 + (2(e - n + 1) - p + 1)^{1/2} \leq (2e - n + 1)^{1/2},$$

because the number $e$ of edges is at most $(\xi)$ if $p = n$, and $(\xi) + n - 1$ if $p < n$. \xqed
The statement of our Theorem 2.2 was motivated by a recent result [3], where an upper bound
\[ a(G) \leq (e/2)^{1/2} \]
on the arboricity \( a(G) \) of a graph has been given. Observe that the ratio between this bound and the bound of Theorem 2.2 is two. Hence one may expect a close relation between the numbers \( a(G) \) and \( k(G) \). As we prove in Corollary 2.5 below, this is true for regular graphs, where the ratio between \( k(G) \) and \( a(G) \) is at most 2.

Given a graph \( G \), let \( d(x) \) denote the degree of a vertex \( x \). Further, let \( \delta = \delta(G) \) and \( \Delta = \Delta(G) \) denote the minimum and maximum degree of a vertex in \( G \), respectively.

**Theorem 2.4.** We have \( \delta \leq k(G) \leq \Delta \) for every graph \( G \).

**Proof.** The number \( k \) of nonempty decomposition classes after executing the procedure FOREST is equal to the maximum label \( r(x) \) of a vertex \( x \). During the run of the procedure, the label \( r(x) \) of an unscanned vertex \( x \) is increased by one whenever a neighbor \( y \) of \( x \) is scanned. Hence \( r(x) \leq d(x) \), and \( k \leq \Delta \) follows. On the other hand, we have \( r(x) = d(x) \) for the last scanned vertex. Hence \( k \geq \delta \) follows. \( \square \)

**Corollary 2.5.** Let \( G \) be a \( d \)-regular graph. Then \( k(G) = d \), and \( a(G) > d/2 \).

**Proof.** We have \( k(G) = d \) by Theorem 2.4. The arboricity \( a(G) \) is at least \( dn/(2(n - 1)) > d/2 \) since \( G \) has \( dn/2 \) edges. \( \square \)

It is well known that, given arbitrary \( \varepsilon > 0 \), almost all graphs satisfy \( \Delta/\delta \leq 1 + \varepsilon \). Hence the ratio \( 1 \leq k(G)/a(G) \leq 2 + \varepsilon \) remains valid for almost all graphs. However, in the worst case, there are graphs of arboricity two, and with \( k(G) \) arbitrarily large.

**Theorem 2.6.** For every \( k \geq 2 \), there exists a graph \( G \) with the arboricity \( a(G) = 2 \), and for which a possible output of the procedure FOREST is a decomposition into \( k \) nonempty forests.

**Proof.** We will construct a sequence \( G_k \), \( k = 1, 2, \ldots \), of graphs as follows. Set \( G_1 := K_2 \) (the complete graph on two vertices), and assume that \( G_k = (V_k, E_k) \) has already been constructed. Let \( v_1, v_2, \ldots, v_n \) denote the vertices of \( G_k \). We define \( G_{k+1} = (V_{k+1}, E_{k+1}) \) as follows. Set \( V_{k+1} = V_k \cup \{w_1, w_2, \ldots, w_n, z\} \), where \( z \) and \( w_i \)'s are new vertices, and \( E_{k+1} = E_k \cup \{v_iw_i | i = 1, \ldots, n\} \cup \{w_iz | i = 1, \ldots, n\} \). We claim that \( k(G_k) \geq k \), while \( a(G_k) = 2 \) for \( k \geq 2 \).

(i) We show, by the induction on \( k \), that \( a(G_k) \leq 2 \) for all \( G_k \). It is trivially valid for \( G_1 \). Assume that the statement is valid for \( G_k \), and let \( T_1 \) and \( T_2 \) be a pair of trees with
$T_1 \cup T_2 = E_k$. Then $T_1 \cup \{w_i| i = 1, \ldots, n\}$, and $T_2 \cup \{w_i| i = 1, \ldots, n\}$ is a pair of forests covering the graph $G_{k+1}$.

(ii) We check that $k(G_k) \geq k$. Assume that the procedure FOREST begins to scan the vertices of $G_{k+1}$ in the order $v, w_1, w_2, \ldots, w_n$, which is an admissible order. After scanning these $n+1$ vertices, we have $E_1 = \{v_iw_i| i = 1, \ldots, n\}$ \cup $\{w_i| i = 1, \ldots, n\}$, and $r(v_i) = 1$ for all $v_i \in V_k$. In this situation, the procedure FOREST will process $G_k$ in the same manner as if $r(v_i) = 0$ for all $v_i \in V_k$. By the induction hypothesis, $G_k$ could be decomposed into $k$ forests.

References