Ear decompositions and their applications to network routing problems

Thesis

Author:

Balázs Szepesi
BSc in Mathematics

Supervisors:

Erika Bérczi-Kovács **Gyula Pap**

Institute of Mathematics,

Department of Operations Research

Eötvös Loránd University
Faculty of Science

2025

Contents

Acknowledgements

I express my gratitude to my thesis supervisors Bárczi-Kovács Erika and Pap Gyula for their support and professional insights that helped me to write my thesis. I also would like to thank them for fostering my interest and passion towards operations research. I also express my gratitude to Johanna Becker, with whom I participated in the consultations and explored subject-related topics.

I would like to thank Rózi for her never ending encouragement and motivation during the last three years. Special thanks to my family and friends for their support and patience towards me.

1 Introduction

Consider a communication network that we can represent with a graph G. The nodes are sending messages to each other. However, the connections between the nodes are not reliable entirely; sometimes they fail. The problem of sending messages between two distinct nodes despite having multiple connection failures arises naturally. One possible approach to solve this issue is to find different paths between each pair of nodes that do not intersect each other. That way, if one path fails, then we can use another path, and so on. However, in a larger network, storing these paths would consume a lot of storage space. Instead of maintaining separate paths for each pair of nodes, a more efficient approach is to have multiple spanning trees, where we require edge-disjointness of certain paths.

Definition 1.1. *We are given a graph $G = (V, E)$, a vertex $r \in V$ called a root, and two spanning trees T_1 and T_2. We call T_1 and T_2 edge-independent spanning trees if for any vertex u the paths to r from u in T_1 and T_2 are edge-disjoint. Similarly, we can define vertex-independent spanning trees by requiring the paths to be internally disjoint. We say that k trees are edge- or vertex-independent if the trees are pairwise edge- or vertex-independent, respectively.*

Spanning trees that are independent in some sense have been studied for a long time. Cheng, Wang and Fen gathered the results related to this topic in [1]. We may set the goal to characterize the maximum number of independent spanning trees. There are two famous conjectures regarding independent spanning trees [5, 6]:

Conjecture 1.1. *If a graph G is a k-edge-connected graph, then there are k edge-independent spanning trees rooted at an arbitrary vertex in G.*

Conjecture 1.2. *If a graph G is a k-vertex-connected graph, then there are k vertex-independent spanning trees rooted at an arbitrary vertex in G.*

Although the conjectures have not been proven for general k yet there are partial results for both conjectures. Throughout this thesis, we focus on the edge-independent spanning

trees and show that being k-edge-connected is a sufficient condition for $k = 2, 3$ and 4. For k greater than 4 the question is still open.

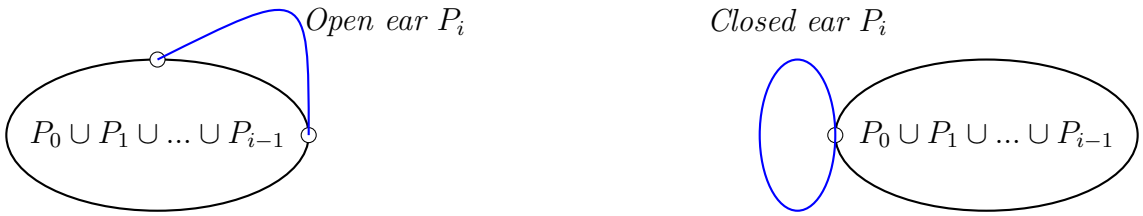
In Section 2, we present an overview of the most simple case $k = 2$. We continue in Section 3 by giving a polynomial-time algorithm for finding $k = 3$ trees. In Section 3, we characterize four-edge-connected graphs with the help of an ear decomposition-like decomposition of the graph, which we use in Section 4 to prove the conjecture in Section 5. Concluding the thesis, we show some applications of edge-independent spanning trees related to communication networks. Section 6.

2 Constructing Two Edge-independent Trees

In this section, we show a way to obtain two edge-independent spanning trees from a two-edge-connected graph G using a partial order on the vertices.

To construct the trees, we use ear decompositions, which are defined as follows.

Definition 2.1. *Given a graph $G = (V, E)$, an ear decomposition is a sequence of subgraphs $P_0, P_1, ..., P_k$ such that P_0 is a cycle in G and for $i \geq 1$ P_i is a cycle or a path such that its internal vertices have not been added in previous P_j, $j < i$, and its endpoints have been added previously. Each subgraph P_i is called an ear. We call an ear closed if it only has one endpoint. In this case, the ear is a cycle. If an ear decomposition contains a closed ear, then we call it a closed ear decomposition. We call an ear with two distinct endpoints an open ear. If an ear decomposition only has open ears we call it an open ear decomposition.*



We use the following well-known characterizations of graphs related to ear decompositions:

Theorem 2.2 (Whitney)**.** *[2] A graph G is 2-connected if and only if it has a open ear decomposition.*

Proof. Using the DFS algorithm, we may easily retrieve an ear decomposition of a 2-edge-connected graph. Once we have obtained a DFS tree, we can build an ear decomposition the following way. First, consider a back edge. Let the starting cycle P_0 be the subgraph containing the root of the DFS tree, the back edge and the path in the DFS tree between the two endpoints. Then choose a back edge connecting to P_0. Add this edge and the path

connecting its other endpoint to P_0 as P_1. Then we continue adding the ears connecting to $\cup_{i=0}^{j-1} P_i$ until the whole graph is covered.

A graph with an ear decomposition is 2-connected by the definition of the ears. □

A consequence of Whitney's theorem is that a graph $\mathcal{G}$ is 2-edge-connected if and only if it has an ear decomposition. If the graph has at least one cut-vertex, then we split the graph at its cut-vertices. Then we have more than one 2-connected graphs, with the cut-vertices appearing in more than one graph. We can find an ear-decomposition in each of them rooted at the component's cut-vertex. We can retrieve an ear-decomposition of G by putting these decompositions next to each other in appropriate order.

From now on we may assume that in addition to the original graph G we have an ear decomposition $G = P_0 \cup P_1 \cup ... \cup P_k$. We may also assume that the root r of the red tree R and the blue tree B is part of the base cycle P_0.

There is a simple algorithm to obtain the two trees from an open ear decomposition by establishing a partial order on the vertices of the graph. However, the two-edge-connected graph may not have an open ear decomposition, so we modify it. If P_i is an closed ear, then it is connected to $P_0 \cup P_1 \cup ... \cup P_{i-1}$ by one vertex v that was added in an ear P_j. We replace v and the edges e_1 and e_2 connected to it in P_j and the edges e_1' and e_2' connected to it in P_i the following way: remove v and add v_1 and v_2 connected by and edge. Also connect v_1 with the other ends of e_1 and e_1' and similarly connect v_2 with the other ends of e_2 and e_2'. For the first ear, we arbitrarily choose a partial ordering of the vertices. For $P_0 = r, v_1, v_2, ..., v_m, r$ we may freely choose the partial order to be $r \prec v_1 \prec v_2 \prec ... \prec v_m$ or the reverse $r \succ v_1 \succ v_2 \succ ... \succ v_m$. For each $P_i = u_1, u_2, ..., u_j$ where i is at least one u_1 and u_j are already ordered if $u_1 \prec u_j$, then $u_1 \prec u_2 \prec ... \prec u_j$, if $u_1 \succ u_j$, then $u_1 \succ u_2 \succ ... \succ u_j$. After creating the partial order, we orient each edge in the following way: if uv is an edge and $u \prec v$, then $u \to v$. In the directed graph, we can find a tree that only uses edges according to the orientation (R) of the graph and one only uses edges opposing the orientation of the graph (B).

The edge-independence of the two trees can easily be observed by noting that for each

$v \in V$, $R(v)$ only uses directed edges following the orientation and $B(v)$ only uses edges opposing the orientation. For them to share an edge, the edge would need to be oriented in both directions. However, since the endpoints follow a partial order, this is impossible. Hence, the trees are edge-independent.

Note that if we contract the edge v_1v_2, whose ends were extensions of a cut vertex v, we maintain edge-independence. However, if there exists a $v_1 \to v_2$ path that is not the edge v_1v_2, after contracting the vertices, a cycle might be created. We can easily eliminate the cycle by not including one edge of the $v_1 \to v_2$ path. Thus, we obtain two edge-independent trees from an ear-decomposition.

3 Constructing Three Edge-independent Trees

Given a 3-edge-connected graph $G = (V, E)$, our goal is to find 3 edge-independent spanning trees rooted at an arbitrary vertex r. We follow the method of Gopalan and Ramasubramanian [3]. For the sake of simplicity, we refer to these trees as red, blue and green trees denoted by R, B and G respectively. We prove the existence of the trees constructively. First, by pruning edges, we reduce the original graph to a minimally 3-edge-connected graph. Then we convert G to a 3-vertex connected cubic graph. Inside this graph, we construct a sequence of paths with special properties. Then from this partition of the modified graph we obtain a sequence of partition of the original graph. Using this partition we may construct the 3 trees.

3.1 Converting a 3-edge-connected graph to 3-edge- 2-vertex connected graphs

Our approach first decomposes the graph to 3-edge 2-vertex connected graphs. If the given graph G is not minimally 3-edge-connected, we remove certain edges to make it so. We decompose the graph into 2-vertex-connected components. If we find a vertex the removal of which disconnects the graph then we may divide our graph into two separate 2-vertex connected components (if required, we iterate this step). If there are at least two components, then some vertices will occur in multiple components. In each component there exists a vertex such that any path from a vertex of the component towards the root in the original graph traverses the vertex. This is the "virtual root" vertex of the component. We perform all the following steps in each component with the virtual root taking the place of the root in some components.

3.2 Converting a 3-edge- and 2-vertex-connected graph to a cubic 3-vertex connected graph

Now we construct a 3-vertex connected cubic graph from a 3-edge- and 2-vertex-connected graph. Each vertex with degree greater than three will be substituted by gadget made of 3-degree nodes. For all vertices n with degree greater than 3 we apply the following procedure:

Procedure: Cubic expansion

1: Remove n from the graph and all edges connected to it

2: Create $d(n) - 2$ subvertices, denoted by $v_1, ..., v_{d-2}$. For $i = 1, 2, ..., d - 3$ add (v_i, v_{i+1}) as an edge.

3: Divide $G - n$ into 2-edge-connected components

4: If there is only one component, then arbitrarily assign edges to the new vertices. Vertices v_1 and v_{d-2} both have two incident edges connected, the other vertices have one edge connected to the rest of the graph.

5: If there are more than one 2-edge-connected components, select two arbitrary 2-edge-connected components which only have one edge connection to another 2-edge-connected component and select two of these edges. Connect these edges to v_1 and v_{d-2}. The rest of the edges can be placed arbitrarily between $v_2, ..., v_{d-3}$.

Theorem 3.1 (Gopalan-Ramasubramanian)**.** *[3] The algorithm maintains the 3-edge- and 2-vertex-connectivity of the graph.*

Proof. In the beginning of each step the graph is 2-vertex connected. Removing one vertex keeps the graph connected. After the expansion procedure is completed it is easy to see, that deleting any vertex does not split the graph either.

To prove that the algorithm maintains the 3-edge-connected property we show that after removing any edge we have a 2-edge-connected graph. Throughout this proof the 2-edge-connected components which only have one edge connection to another 2-edge-connected component (the same as in step 5 of the algorithm) will be called leafs. We introduce a

property called **overlap** for the edges connected to the newly added subvertices. Consider the leaf components of $G-n$, where G is the graph in the beginning of the step and n is the chosen point which is expanded to subvertices. Divide these components to two arbitrary separate parts L_1 and L_2. Then n is replaced by $v_1, ..., v_{d-2}$ vertices. Let min_i be the minimal index for which L_i has an edge connected to v_{min_i}. The same way we define max_i as the maximal index for which L_i has an edge connected to v_{max_i}. We say that L_1 and L_2 overlaps if $(min_1, max_1) \cap (min_2, max_2) \neq \emptyset$. Let $I_n = \{e : e = v_i v_{i+1}, i \in \{1, 2, ..., d-3\}\}$. These are the edges connecting two subvertices of n. It is sufficient to show that by removing $e \in I_n$ or $e \notin I_n$ we still have a 3-edge-connected graph.

a) $e \in I_n$. Removing e splits into the subvertices to two distinct path segments: one of them containing v_1 the other one containing v_{d-2}. Let S_1 and S_2 be the two leaves we connected v_1 and v_{d-2} with. Consider a path between S_1 and S_2. This path and the edges connecting to v_1 and v_{d-2} each form a cycle. Thus the two path segments, S_1, S_2 and the path connecting S_1 to S_2 is 2-edge-connected. It remains to show that the other leaf components are 2-edge-connected as well. These leaf components originally had at least two edges connecting to n, which means the leaves connect to the segments which form a 2-edge-connected component, hence they are part of that component.

b) $e \notin I_n$. We consider three sub-cases. If e is an edge inside a 2-edge-connected component C. Any u vertex in C connects to u with 3 edge-disjoint paths. Removing e can break at most one path, hence u has two edge-disjoint paths to each v and using the two special leaf components connected to v_1 and v_{d-2} we can find two edge-disjoint paths to each v_i. The same argument can be made for any other vertices of the graph, since the existence of two edge-disjoint paths is transitive. Consequently the graph remains 2-edge-connected.

c) If e connects a leaf component to a v_i, then there is still an edge connecting the leaf to some v_j, furthermore the leaf is connected to another 2-edge-connected component which connects to the expansion vertices of n. If e connects a non-leaf component to a v_i, then the component is still connected with two other components. There is a path to a leaf using each edge. Through the leaves we connect to v in the original graph and to two sub-vertex

after replacing v. Hence, 2-edge connectivity is retained.

d) If e connects two components, meaning $G - n - e$ is disconnected, then denote these two components by D_1 and D_2. Observe that the two leaf components connected to v_1 and v_{d-2} are both in a D_i or each is a part of a distinct D_i. Either way the edges connecting from D_1 and D_2 to the subvertices overlap, hence the resulting graph is 3-edge-connected. □

Lemma 3.2. *Any three-edge-connected cubic graph is* 3*-vertex connected.*

Proof. Proof by contradiction. Suppose there are three edge-independent paths between p and p', but there is a $q \in V$ appearing in two paths. Since the paths are edge-independent, $deg(q)$ is at least four, contradicting the assumption that the graph is cubic. □

Definition 3.3. *Given a 3-edge- and 2-vertex-connected graph $G = (V, E)$, we denote the cubic graph obtained by the algorithm $\mathcal{Q} = (\mathcal{V}^*, \mathcal{E}^*)$. Let V^* denote the number of vertices and E^* the number of edges in $\mathcal{Q}$.*

Definition 3.4. *For a vertex $v \in \mathcal{V}$ who is an expansion of a vertex from V, we denote its original vertex n_v. Let $\mathcal{V}_n^*$ denote the set of vertices corresponding to $n \in V$ in $\mathcal{Q}$.*

Since the expansion does not remove edges, only some $E \subset \mathcal{E}^*$ is added.

3.3 Constructing augmenting cycles and paths

Before beginning the description of the construction, let us describe intuitively why we will require certain properties for this decomposition. We want to build the red and blue tree through a degree-like partition of the graph, while we mark a neighbor of the root u through which we can build the green tree not using the decomposition itself, but the fact is that our decomposition preserves a connection between u and the other vertices.

Given a 3-vertex connected cubic graph and a root r, we decompose the graph into a sequence of a cycle and paths. The cycle contains r. Each path starts and ends in different vertex. The cycles and the paths satisfy the following properties:

(A) The removal of vertices in the path keeps every other vertex not added in this path or earlier paths connected with each other.

(B) Every vertex in the path is connected to at least one vertex that has not been added to this path or to earlier paths.

We start by choosing an arbitrary neighbor u of the root r, and remove the edge between them from the graph. Then we determine a cycle containing r and avoiding u satisfying properties (A) and (B). Then we expand the cycle by adding paths that avoid u starting from two distinct points and with the following properties:

1. every vertex in the path is connected to a vertex not added yet, and
2. every vertex not added yet stays connected to u.

This gives a sequence of paths with the last having u as the only vertex not yet added. We denote the cycle with P_0 and the paths in the order they were added $P_1, P_2, ..., P_l$. J. Cheriyfd S. N. Maheshwari gave an algorithm that finds the starting cycle and the paths using non-separating induced cycles in. The algorithm may also be used to solve the 3 vertex-independent tree problem for 3-connected graphs.

Definition 3.5. *The path index of an edge is the index of the augmenting path/cycle in which the edge was added in.*

3.4 Computing segments

Having computed the augmenting paths in the modified graph $\mathcal{Q}$,we wish to return to the original graph G. Using the augmenting paths we will compute segments of the original graph which will help us to finally compute the red, green and blue trees.

Before presenting the algorithm for computing the segments, we will introduce some notations used in the algorithm.

Definition 3.6. *$next(v)$ is the neighbor of v in a path along the direction of traversal. Let $l(n_v)$ denote the last vertex present in a path containing v that is part of $\mathcal{V}_n^*$.*

At the beginning of the segmentation process only r is marked, the other vertices are unmarked.

Procedure: Compute segment

Input: A path in $\mathcal{Q}$: $x, v_1, v_2, ..., v_l, y$, where x and y are marked. In the case of the first cycle $x = y = r$.

Output: A segment S of the original graph G.

1: $S = \emptyset$

2: Start from x. Traverse the path and find the first v, that v is marked and $next(v)$ is not.

3: Add n_v to S if it exists, if not go to Step 8

4: $v' = next(v)$

5: Add n'_v to S

6: If v' is marked go to step 8

7: $v = l(n_{v'})$

8: Mark all $v \in \mathcal{V}_n^*$ in $\mathcal{Q}$

9: Stop

We follow the procedure for each augmenting cycle and path in the order of $P_0, P_1, ..., P_l$ from the previous procedure. A segment may not cover the whole cycle or path, so it may not be enough to run the procedure only once for each P_i. We repeat the procedure iteratively until all vertices of P_i are marked.

Notice that while u in $\mathcal{Q}$ is avoided by the decomposition until the last augmenting path, n_u in G may not be avoided, since another vertex of u may have been added earlier. However, each vertex not added yet is connected to it.

Let the obtained segments be numbered from 0 to $(S-1)$ and denote the ith segment by S_i. We know that the first segment is a cycle containing r. Except for the starting cycle, all segment's ends are included in earlier segments (the ends of the segment may be the same vertex). Furthermore, each segment contains a vertex that did not appear in earlier segments. Then $\cup_{i=0}^{S-1} S_i$ contains all vertices of G. In addition, we retained the nonseparating

nature of the earlier decomposition, namely that in $V - \cup_{j=0}^{i} S_i$ each vertex is connected to n_u, where u is a fixed neighbor of r in Q.

3.5 Computing The Trees

In order to construct the red and blue trees using the segments, we maintain a partial order between the edges. This partial order helps to maintain the edge-independent property of the trees. We build the two trees segment by segment, adding paths to them.

Procedure: Compute red and blue trees

1: Initialize $\mathcal{T}_r, \mathcal{T}_b$ to be empty.

2: S_0 consists of vertices $r, n_1, ..., n_k, r$. Add the path $n_k \to n_{k-1} \to ... \to r$ to $\mathcal{T}_b$ and $r \to n_1 \to ... \to n_k$ to $\mathcal{T}_r$. The partial order between the edges is as follows: $[r, n_1] \prec [n_1, n_2] \prec ..., \prec [n_k, r]$.

3: From $i = 1$ to $S - 1$:

a) S_i consists of $x, n_1, ..., n_k, y$, where x and y are already added in previous segments. Let v_r and v_b denote the parent of v in $\mathcal{T}_r$ and $\mathcal{T}_b$, respectively. Then x_r, x_b, y_r, y_b are already computed and part of an already processed segment.

b) If $[x, x_r] \prec [y, y_r]$ in the partial order, then we add $x \leftarrow n_1, \leftarrow n_2 \leftarrow ... \leftarrow n_k$ to the red tree, and add $n_1 \to n_2 \to ... \to y$ to the blue tree. The partial order for these edges are updated as follows: $[x, x_r] \prec [x, n_1] \prec [n_1, n_2] \prec ...[n_k, y] \prec [y, y_b]$

c) If $[x, x_r] \succ [y, y_r]$ in the partial order, then we add $x \leftarrow n_1, \leftarrow n_2 \leftarrow ... \leftarrow n_k$ to the blue tree, and add $n_1 \to n_2 \to ... \to y$ to the red tree. The partial order for the edges are updated as follows: $[y_r, y] \prec [y, n_k] \prec ... \prec [n_1, x] \prec [x, x_b]$

When constructing the green tree we use the nonseparating property of the decomposition. We add the edge connecting r to n_u, along with the edge that has the highest path index for every other vertex without creating a cycle.

Theorem 3.7 (Gopalan-Ramasubramanian)**.** *[3] The red, blue and green trees are edge-independent.*

Proof. To prove that the green and red as well as the green and blue trees are edge-independent, it is enough to observe that the path from a vertex to r in the red and blue trees traverses through edges with non-increasing path-indexes, while in the green tree it is strictly increasing. To prove that the red and blue trees are edge-independent, we may use a similar argument made for the two edge-independent case.

□

With this method, we showed Conjecture 1.1 for $k = 3$ and also gave a polynomial time algorithm to find three trees.

4 Chain Decomposition of a four-edge-connected Graph

In this section, we present the results of Hoyer and Thomes [4] regarding the characterization of four-edge-connected graphs. The characterization resembles the ear decomposition of two-edge-connected graphs in that it involves building the graph from 'chains.' However, it differs in that there are more types of 'chains.'

4.1 Basic definitions

Our main goal is finding four edge-independent trees, but similarly to the two edge-independent tree case we would like to characterize the 4-edge-connected graphs with something resembling to degrees. We will use chain decompositions with respect to a root vertex r. Before defining a chain decomposition, we define different kinds of chains that will build up the decomposition:

Definition 4.1 (Up Chain)**.** *An up chain of G with respect to a pair of edge-disjoint subgraphs (H, L) is a subgraph of G, edge-disjoint from H and L, which is either*

1. *(open chain) a path with at least one edge such that each vertex is either r or has at least degree two in L, and ends either at r or at H, or*
2. *(closed chain) a cycle such that every vertex either is r or has degree at least two in L, and some vertex v, either is r or has degree at least two in H. Then v is considered to be both ends of the chain and all other vertices in the chain are to be considered internal vertices.*

Definition 4.2 (Down Chain)**.** *A down chain of G with respect to (H, L) is an up chain with respect to (L, H).*

Definition 4.3 (One-way Chain)**.** *A one-way chain of G with respect to (H, L) is a subgraph of G, induced by an edge $e \notin H \cup L$. The edge $e = uv$, such that u is either r or has a degree at least two in H, and v is either r or has a degree at least two in L. Then u is considered to be the tail of the chain and v the head of the chain*

Definition 4.4 (Chain Decomposition)**.** *Let $G_0, G_1, ..., G_m$ be a sequence of subgraphs of G. Denote $H_i = G_0 \cup G_1 \cup ... \cup G_{i-1}$ and $L_i = G_{i+1} \cup G_{i+2} \cup ... \cup G_m$. H_0 and L_m are the null graph. Then $G_0, G_1, ..., G_m$ is a chain decomposition of G rooted at r, if the following hold:*

1. *The sets $E(G_0), E(G_1), ..., E(G_m)$ partition $E(G)$*
2. *For $i = 0, 1, ..., m$ the subgraph G_i is either an up chain, a down chain or a one-way chain with respect to the subgprahs (H_i, L_i).*

Definition 4.5. *The chain index of $e \in E(G)$ denoted by $CI(e)$, is the index of the chain containing e.*

Chain decompositions have a certain symmetry. Namely, if $G_0, G_1, ..., G_m$ is a chain decomposition of G with root r, then $G_m, G_{m-1}, ..., G_0$ is a chain decomposition with root r and the up/down chains of the first decomposition are down/up chains of the second one, while in the case of one-way chains the heads and tails are switched. From now on, this property of chain decompositions will be known as **symmetry**. The first chain has to be either a closed up chain or a one-way chain with r as the tail. Furthermore the last chain has to be either a closed down chain or a one-way chain with r as the head.

There is a certain family of chain decompositions which are easier to deal with and have desirable properties. These are minimal chain decompositions:

Definition 4.6. *An up chain / down chain G_i is minimal if no internal vertex of G_i is in $\{r\} \cup V(H_i)$ / $\{r\} \cup V(L_i)$. A chain decomposition is minimal if all up chains and down chains are minimal.*

Observe that a minimal chain is an ear (Theorem 2.1). If we are given a chain decomposition then we may subdivide up and down chains into minimal chains by cutting them at the critical vertices. Then we may replace the original chain with a series of minimal chains in the decomposition. Using the previous method, we may convert a chain decomposition to a minimal chain decomposition. From now on, we may assume that the chain decomposition is minimal if convenient.

The main theorem of this section is as follows.

Theorem 4.7 (Hoyer-Thomas)**.** *[4] Given a four-edge-connected graph $G = (V, E)$ and a vertex $r \in V$ there is a chain decomposition of G with root r.*

To prove this theorem, we use the Mader construction.

Definition 4.8. *A Mader operation is one of the following operations:*

- *Add an edge between two (not necessarily distinct) vertices.*
- *Consider two distinct edges, $e_1 = xy$ and $e_2 = zw$. "Pinch" them as follows: delete e_1 and e_2. Add a new vertex v and edges e_x, e_y, e_z, e_w with endpoints v and x, y, z, w, respectively. Although e_1 and e_2 must be distinct, the new edges may not.*

We will use the following theorem, which characterizes four-edge-connected graphs with Mader operations.

Theorem 4.9 (Mader)**.** *[4] A graph is four-edge-connected if and only if, for any $r \in V(G)$, we can construct G in the following way. Begin with a graph G^0 consisting of r and one other vertex of G, with four parallel edges. Then repeatedly perform Mader operations to obtain G.*

4.2 Preparatory claims

The following claims will help us to prove that we can maintain a chain decomposition through Mader-operations. First, we show that loop edges can be omitted in some sense from the graph. Then, we prove the existence of two important chain indices, which will help us to examine different subcases in later steps of the proof. Finally, we show that the existence of a chain decomposition implies a degree property.

Lemma 4.10. *Given a graph $G = (V, E)$ and a chain decomposition $G_0, G_1, ..., G_m$ with root $r \in V$. If $v \neq r$ is in H_i (respectively, L_i) then v is incident to a nonloop edge in H_i (respectively, L_i). If $v \neq r$ has degree at least two in H_i (respectively, L_i), then v is incident to two distinct nonloop edges in H_i (respectively, L_i).*

Proof. A vertex with a loop has at least degree two, hence it is sufficient to prove only the second part of the lemma since it implies the first part.

Suppose $v \neq r$ is incident to a loop. We may assume that it is in H_i by symmetry. Choose the loop with minimal chain index $j < i$. Notice that by the minimality of j the vertex v is incident to no loops in H_j. By recalling the definitions of the different types of chains we can conclude that a loop can only be a closed chain. A closed chain requires v to have at least degree two in H_j. We showed that in $H_j \subset H_i$ v is distinct to at least two nonloop edges. □

An immediate consequence of the previous lemma:

Corollary 4.11. *Given a graph $G = (V, E)$ and a chain decomposition $G_0, G_1, ..., G_m$ with root $r \in V$. Suppose $e \in E(G_i)$ is a loop. Then $G_0, G_1, ..., G_{i-1}, G_{i+1}, ..., G_m$ is a chain decomposition of $G - e$ rooted in r.*

Notice that if G has no isolated vertices, then $G-e$ has neither, since it would not satisfy any chain definition. Now, we show a useful property of minimal chain decompositions:

Lemma 4.12. *Given a graph $G = (V, E)$ with no isolated vertices and a minimal chain decomposition $G_0, G_1, ..., G_m$ with root $r \in V$. For a vertex $v \in V$ there are indices i and j such that the degree of v in H_i and L_j is two.*

Proof. By symmetry, we only need to find i. Since G has no isolated vertices v is contained in a chain. Let G_{i_0} be the chain containing v with a minimal chain index i_0.

G_{i_0} cannot be a down chain, since $v \notin V(H_{i_0})$.

If G_{i_0} is an up chain, then v has to be an internal vertex, since $v \notin V(H_{i_0})$ so v has to have degree two in G_{i_0}. Furthermore, v has to have degree at least two in L_{i_0} which means that $L_{i_0} \neq \emptyset$, so H_{i_0+1} is a sufficient i.

If G_{i_0} is a one-way chain, then v has to be the head, since $v \notin V(H_{i_0})$. Therefore, v has to have degree at least two in L_{i_0} so we may consider the next chain G_{i_1} such that $v \in V(G_{i_1})$. Note that the degree of v in H_{i_1} is exactly one.

G_{i_1} cannot be a down chain, since the degree of v in H_{i_1} is not at least two. If G_{i_1} is an up chain, then it has to be an open chain. Since the chain decomposition is minimal, v has to be an end of the chain, therefore, the degree of v in G_{i_1} is exactly one. If G_{i_1} is a one-way chain, then it has to be the head since v does not have degree two in H_{i_1}. In either way v has exactly degree one in G_{i_1}, at least degree two in L_{i_1}, so H_{i_1+1} is nonempty, hence a sufficient choice. □

As a closure of this section we show that a chain decomposition implies a minimum degree result:

Lemma 4.13. *Given a graph $G = (V, E)$ with no isolated vertices and a minimal chain decomposition $G_0, G_1, ..., G_m$ with root $r \in V$. For a vertex $r \neq v \in V$ the degree of v is at least four.*

Proof. By 4.11, we may assume that there are no loops in the graph. Suppose v is in an up chain G_i. In an open chain v has degree at least two in L_i. If it is an internal vertex of the chain, then it has degree at least two in G_i. If it is the end of the chain, then it has degree at least one in G_i and degree at least one in H_i. We showed that if v is in a down chain, then it has degree at least four. By symmetry, the claim regarding the degree is also true for vertices in down chains.

Since there are no isolated vertices for the rest of the vertices, we may assume that they are in only one-way chains. If v is in a one-way chain G_i, then it has degree one in G_i and degree two in either H_i or L_i. Assume for the sake of contradiction that v is in only three one-way chains. Let these chains be $G_{l_1}, G_{l_2}, G_{l_3}$ with increasing chain indexes. Consider the definition of one-way chains and the chain G_{l_2}. Notice that v should have degree at least two in H_{l_2} or L_{l_2}, but it has not, contradicting that v is in only three one-way chains. We can conclude that $v \in V - r$ has a degree at least four. □

4.3 Proof of characterization

We will prove Theorem 4.7 using Mader operations. Due to the characterization of four-edge-connected graphs in Theorem 4.9, it suffices to show that we can maintain a chain decomposition through a Mader operation. First, we need a chain decomposition of G^0. The chain decomposition of G^0 will be a closed up chain consisting of two edges and a closed down chain also consisting of two edges.

Given a graph G, with chain decomposition $G_0, G_1, ..., G_m$. We may assume that the decomposition is minimal. We obtain G' from G by a Mader-operation. Our goal is to give a chain decomposition of G'.

a) Adding an edge

Suppose that we obtained G' by adding an edge uv in G. If either vertex is r, then we can classify the edge uv as a one-way chain with tail r and make it first in the chain decomposition. The head has to have degree at least two in later chains, but all chains are later, so any other vertex is an eligible head.

If neither vertex is r, then let i be the minimal index for which u or v has degree exactly two in H_i, guaranteed to exist by 4.12. Without loss of generality, we may assume that it is u. Since H_0 is null, i has to be greater than one. By the choice of i, the degree of v in H_i is at most two and at least two in L_{i-1}. Classify the edge uv a one-way chain with tail u and head v and in the decomposition, we place it between G_{i-1} and G_i.

Consider the impact of the added chains on other chains. Increasing the degree of any vertex does not interfere with any chain definitions. The relative position of the other chains did not change. We obtained a chain decomposition of G', but notice that some chains may not be minimal.

b) Pinching edges

Suppose that we obtained G' by pinching the edges $e_1 = xy$ and $e_2 = zw$ in G replacing them with e_x, e_y, e_z and e_w. We denote the chain containing e_1 by $J_1 = G_{CI(e_1)} = P_x e_1 P_y$,

where P_x is the path of the chain from an end to x not using e_1 and P_y is the path from the other end to y not using e_1. Note that P_x and P_y may be empty. Similarly, we define $J_2 = G_{CI(e_2)} = P_z e_2 P_w$.

Now, we show several claims proving that for any type of chain pairs and index combinations of J_1 and J_2 we can obtain a chain decomposition for G'. First, we show that if the chain indices are the same, then we can obtain a chain decomposition. We then distinguish between cases based on the degree properties of the one-way chains. Naturally, the claims also cover those cases where the pinched edges are not one-way chains.

Claim 4.14. *If $CI(e_1) = CI(e_2)$, then G' has a chain decomposition rooted at r.*

Proof. If $CI(e_1) = CI(e_2)$, then $J_1 = J_2$. Without loss of generality $e_1 \in E(P_z)$ and $e_2 \in E(P_y)$, so the chain can be written as $J_1 = J_2 = P_x e_1 (P_y \cap P_z) e_2 P_w$. Note that $P_y \cap P_z$ may be empty if $y = z$. By the definition of pinching e_1 and e_2 are distinct, consequently $J_1 = J_2$ is not a one-way chain. We may assume by symmetry that $J_1 = J_2$ is an up chain. In G', replace $J_1 = J_2$ with the following chains in this order.

1. $P_x e_x e_w P_w$ as an up chain. Since e_y and e_z have not yet been used, v has degree at least two in later chains.
2. e_y as a one-way chain with v as tail and y as head. Then v is incident to two edges in the chain above. Then y has degree at least two in later chains, since it was an internal vertex of $J_1 = J_2$.
3. e_z as a one-way chain with head z and tail v. Then v is incident to at least two edges in earlier chains (the ones above), and the head z has degree at least two in later chains since it was an internal vertex of an up chain.
4. $P_y \cap P_z$ as an up chain if it is not empty. The ends y and z are incident to two later chains since they were interior points of an up chain.

Consider the impact of the replacement for the other chains. For earlier chains, there is no impact. Most of the edges of $G_{CI(e_1)} = G_{CI(e_2)}$ are added with the same chain index except for e_1 and e_2, since these were replaced with other edges. Since replacing these two

edges does not change the degree of the edges outside of $CI(e_1) = CI(e_2)$, we maintained a chain decomposition.

□

Without loss of generality, for the remainder of the proof we may assume the following:

- $CI(e_1) < CI(e_2)$
- If J_1 is a one-way chain, then y is the head and x is the tail
- If J_2 is a one-way chain, then w is the head and z is the tail.

Claim 4.15. *If J_1 is either a one-way chain whose head y has degree exactly one in $H_{CI(e_2)}$, or J_2 is a one-way chain whose tail z has degree exactly one in $L_{CI(e_1)}$ then G' has a chain decomposition rooted at r.*

Proof. The two cases are the "complement" of each other, so by symmetry we may assume the first case, where y is the head of the one-way chain J_1 with degree at least two in $H_{CI(e_2)}$.

Replace J_1 with e_x the edge between x and the new vertex v with chain index $CI(e_1)$. The tail x is tail of the original one-way chain, so it has degree at least two in earlier chains. The head v has edges incident to it in later chains e_y, e_z, e_w. We now give chain decompositions for different chain types of J_2.

a) J_2 is an up chain. By the condition in the claim y has degree one in $H_{CI(e_2)}$, if J_2 is closed, y cannot be the end of it. If it is an open chain, then by swapping w and z we may assume that y is not the other end of P_z. The other end of P_z must be a vertex with degree at least one in earlier chains despite not adding e_y to the graph yet. Replace J_2 with the following chains at the index $CI(e_2)$.

1. $P_z e_z$ as an up chain. The original end of P_z is an eligible end, since it was an end for J_2 and v is also a valid end, since it is incident to e_x from an earlier chain and has degree at least two in later chains by e_y and e_w.
2. e_y as a one-way chain with head y and tail v. Then v is incident to two edges in earlier chains e_x and e_z, and y has degree one in $H_{CI(e_2)}$ by assumption, so y has degree at least two in later chains.

3. e_w as a one-way chain with head w and tail v. We already seen in the last chain that v is an eligible tail. The head w is either r or has degree at least two in later chains, because it was a part of an up chain J_2.
4. P_w as an up chain if it is not empty. Then w is incident to e_w from an earlier chain and has degree at least two in later chains, since it was an internal vertex of the up chain J_2. The other end of P_w is an end of J_2. It is either r or incident to an edge in earlier chain. Note that it may be y but by adding e_y earlier it is an eligible end.

b) J_2 is a down chain. By assumption y has degree exactly one in $H_{CI(e_2)}$, therefore y cannot be part of J_2. This implies that each vertex in J_2 is either the root r or has degree at least two in earlier chains, even though e_y has not been placed yet. Replace J_2 with the following chains at the index $CI(e_2)$.

1. P_w as a down chain if it is not empty. Its end w has an incident edge in later chains (namely e_w) and degree at least two in later chains, since it was an internal vertex of a down chain J_2.
2. e_w as a one-way chain with head v and tail w. Since w was a part of J_2 it has a degree at least two. Then v is incident to e_y and e_z so it is an eligible head.
3. $P_z e_z$ as a down chain. Its end v is incident to one edge in later chains e_y and two edges in earlier chains e_x and e_w.
4. e_y as a one-way chain with head y and tail v. The tail has a degree at least two in earlier chains, and the head y is either r or has degree at least two later chains, since by assumption it has degree one in $H_{CI(e_2)}$.

c) J_2 is a one-way chain. We assumed, that y has degree exactly one in $H_{CI(e_2)}$, so it is not an eligible candidate for the tail (z). Consequently z is either r or has degree at least two in earlier chains. Replace J_2 with the following chains at the index $CI(e_2)$.

1. e_z as a one-way chain with head v and tail z. Then z is an eligible tail as described above. The head v has degree exactly two in later chains e_y and e_w.

2. e_w as a one-way chain with head w and tail v. Then v has degree two in earlier chains e_z and e_x, so it is an eligible tail. The head w is either r or has degree two in later chains, since it was the head of J_2.
3. e_y as a one-way chain with head y and tail v. Then v has three edges incident to it from earlier chains, namely e_x, e_z, e_w, hence it is an eligible tail. The head y by assumption is either r or has degree at exactly one in $H_{CI(e_2)}$, so it has to have at least two edges incident to it in later chains.

Now we consider how the replacement of the chains affected the other chains. Note that for the vertices x, w, z their degree after the step $CI(e_1)$ and $CI(e_2)$ did not change, since we inserted in each case e_x with chain index $CI(e_1)$ and e_w and e_z with chain index $CI(e_2)$. The only vertex not mentioned yet is y. We inserted e_y with chain index $CI(e_2)$ in each case, but by the condition of the claim, that is the degree of y in $H_{CI(e_2)}$ is exactly one shows that y is not in any chains between the chain indices $CI(e_1)$ and $CI(e_2)$, so it could not affect any chains.

□

For the remaining cases, we may assume the following, which are symmetrical counterparts of each other.

- If J_1 is a one-way chain, then y has degree at least two in $H_{CI(e_2)}$.
- If J_2 is a one-way chain, then z has degree at least two in $L_{CI(e_1)}$.

The following definitions of indices, which are also symmetrical counterparts of each other help separate the different cases from each other.

- If J_1 is a one-way chain and y is not in $H_{CI(e_1)}$ define the minimal index i for which $y \in V(G_i)$ and $CI(e_1) < i < CI(e_2)$.
- If J_2 is a one-way chain and z is not in $L_{CI(e_2)}$ define the maximal index j for which $y \in V(G_i)$ and $CI(e_1) < i < CI(e_2)$.

Note that by the minimality of i and maximality of j, if we use minimal chain decompositions G_i can only be an open up chain with y as its end or a one-way chain with y as its head. Similarly G_j can only be a down chain with z as its end or a one-way chain with z as its tail.

Claim 4.16. *If $i < j$ or either one of them is not defined, then G' has a chain decomposition.*

Proof. We replace J_1 with other chains with chain index $CI(e_1)$ or i (if i is defined). Similarly we replace J_2 with chains with chain index $CI(e_2)$ or j (if j is defined). By assumption, the chains replacing J_1 have smaller chain index, than the ones replacing J_2, foreshadowing that we gain a chain decomposition after the procedure. We replace J_1 accordingly depending on its chain type.

a) If J_1 is an up chain, then replace it with $P_x e_x e_y P_y$ as an up chain. Then v is valid as internal vertex of an up chain, since it is incident to e_w and e_z in later chains.

b) If J_1 is a down chain, then replace it with the following chains:

1. P_x as a down chain, if it is not empty. One end of it is the end of J_1, the other one is x, which is incident to e_x in a later chain, so it is valid.
2. P_y as a down chain, if it is not empty. One end of it is the end of J_1, the other one is y, which is incident to e_y in a later chain, so it is valid.
3. e_x as a one-way chain with head v and tail x. Then x is either r or has degree at least two in earlier chain, since it was an internal vertex of a down chain. The head v is incident to two edges in later chains, namely e_w and e_z.
4. e_y as a one-way chain with head v and tail y. Then y is either r or has degree at least two in earlier chain, since it was an internal vertex of a down chain. The head v is incident to two edges in later chains, namely e_w and e_z.

c) J_1 is a one-way chain with head y and y has been added in earlier chains. Replace J_1 with the following chains.

1. e_x as a one-way chain with head v and tail x. Then x is an eligible tail since it was the tail of J_1. Then v at least two edges incident to it in later chains (e_y, e_z, e_w).

2. e_y as an up chain. Since y was the head of J_1 it has degree at least two in later chains and by assumption has already an edge incident to it. The other end v has two edges incident to it in later chains e_w and e_z, and an edge from earlier chains e_x.

d) If J_1 is a one-way chain with head y that has not been added in earlier chains.

Add e_x with chain index $CI(e_1)$ as a one-way chain with head v and tail x. Then$x$ is an eligible tail since it was the tail of J_1, and v is incident to at least two edges in later chains (e_y, e_z, e_w).

- If y is one of two distinct ends of the up chain G_i, then replace G_i with $G_i e_y$ as an up chain. Then v is an eligible end since it is incident to an edge from earlier chains, namely e_x and to e_z and e_w in later chains.
- If y is a head of the one-way chain G_i, which is not a loop, then y is an eligible head at the chain index i without having e_y placed yet. Thus do not need to replace G_i, it is enough to add e_y after G_i as an up chain. Then y is incident to an edge from G_i and either is r or has a degree at least two in later chains, since it was the head of G_i in the original decomposition. Then v is incident to one edge from earlier chains e_x and to two from later chains e_w and e_z.

Now we consider how the replacement of the chains affected the other chains. In a), b) and c) the new chains are inserted with chain index $CI(e_1)$ and each vertex's degree remains the same so we can conclude that these cases are valid. In d) the edge e_y is inserted with chain index i. By the definition of i and the assumptions the change at chain index $CI(e_1)$ does not impact any other chains than G_i but has been proved to be valid already.

To replace J_2 consider the reversed chain decomposition and follow the same procedure as for J_1. The proof for the validity of the chain decomposition regarding J_2 is the same as for J_2 also. We may conclude that we maintained a chain decomposition. □

Claim 4.17. *If both i and j are defined and $i = j$ then G' has a chain decomposition rooted at r.*

Proof. When we introduced i and j we showed that G_i can only be an open up chain with end y or a one-way chain with head y, and G_j can only be an open down chain with end z or a one-way chain with tail z. Since $G_i = G_j$ it has to be a one-way chain with head y and tail z. Both i and j are defined, which means that the chain cannot be a loop, thus y and z are distinct. We replace J_1 and J_2 with the following chains in this order. Before $i = j$ we insert the two following chains.

1. e_x as a one-way chain with head v and tail x. Then x is an eligible tail, since it was the tail of J_1 and i is grater than $CI(e_1)$. The head v three edges incident to it in later chains e_x, e_w, e_z.
2. e_z as a one-way chain with head v and tail z. By the definition of j that tail z is either r or has two edges incident to it in earlier chains than G_j. Since we insert this chain immediately before the chain G_j z is a valid tail. Then v has two edges incident to it in later chains, namely e_w, e_y.

We insert the following two chains immediately after $i = j$.

1. e_y as a one-way chain with head y and tail v. Then v is incident to two edges from earlier chains, namely e_x and e_z. By the definition of i, y is either r or has two edges incident to it in chains later, than $i = j$. Since we insert this chain immediately after $i = j$ this chain is also valid.
2. e_w as a one-way chain with head w and tail v. The tail v was already valid in the previous case, and it is still valid. The head w was the head of the one-way chain J_2 and $i = j$ is smaller than $CI(e_2)$, so it is valid.

Now we consider how the replacement of the chains affected the other chains. We inserted e_x with greater chain index than $CI(e_1)$, so we must examine how this affects the chains between J_1 and $G_i = G_j$. We know, that x was the tail of J_2, so it already has a degree at least two at chain index $i = j$. By the definitions of chains losing one degree at vertex x between $CI(e_1)$ and $i = j$ does not affect the validity of other chains. The edge e_y is inserted later, than $i = j$, but by the definition of i the only chain affected is $G_i = G_j$. This chain

remains valid, because G_i is a one-way chain, with head y, so if y loses a degree in H_i it does not affect the validity of G_i. We can make a similar argument in the reversed graph for e_z and e_w.

□

Claim 4.18. *If both i and j are defined, and $i > j$ then G' has a chain decomposition.*

Proof. We replace J_1 and J_2 with the following chains. Their chain index is specified in their respective description.

1. e_x as an up chain, with head v and tail x and chain index $CI(e_1)$. In J_1 x was an eligible tail, so it is also an eligible tail for e_x. The head v has three edges incident to it in later chains, namely e_y, e_z, e_w.
2. e_z as a one-way chain, with head v and tail z. We insert this chain immediately after G_j. By the definition of j, z is an eligible tail since z is either r or incident to two edges in earlier chains than G_j. The head v is incident to two edges in later chains, namely e_y and e_w.
3. e_y as a one-way chain with head y and tail v. We insert this chain immediately before G_i. The tail v has two edges incident to it in earlier chains, namely e_x and e_z. By the definition of i the head y is either r or incident to two edges in later chains than G_i. Since we insert this chain before G_i, this chain is valid.
4. e_w as a one-way chain with head w and tail v and chain index $CI(e_2)$. The tail v has three incident edges in earlier chains, namely e_x, e_y, e_z. The head w was a valid head for J_2, so it is also valid for e_w.

Now we consider how the replacement of the chains affected the other chains. The degree of vertex x did not change after replacing e_1 with e_x at $CI(e_1)$. The edge e_y is inserted immediately before G_i. By the definition of i, no other chains are affected. By symmetry and a similar argument for e_2 we may conclude that we maintained a chain decomposition.

□

We covered all possibilities of chain types when pinching edges. In summary, we proved that we can maintain a chain decomposition through Mader-operations, which also implies Theorem 4.7.

5 Constructing Four Edge-independent Trees

In this section, we show that Conjecture 1.1 holds for $k = 4$ following the method of Hoyer and Thomas [4]. Using the result of the previous section, namely that a four-edge-connected graph has a chain decomposition. It is enough to prove the following theorem.

Theorem 5.1 (Hoyer-Thomas)**.** *[4] Suppose G is a graph without isolated vertices. If G has a chain decomposition rooted at a vertex r, then it has four edge-independent spanning trees rooted at r.*

Given a graph G and its chain decomposition $G_0, G_1, ..., G_m$. We may assume that the chain decomposition is minimal. We obtain four edge-independent spanning trees by constructing two partial numberings of the edges of G using the chain decomposition. From each numbering, we gain two spanning trees, whose independence is provided by monotonic paths with respect to the numberings from the vertices to the root r. The spanning trees that are obtained from different numberings gain their independence from monotonic chain indices in the paths.

By Theorem 4.11 we may assume that there are no loops in G. Using Theorem 4.10, we can select for each vertex special edges incident to it. For each vertex $v \neq r$, there are two distinct nonloop edges incident to v whose chain indices are strictly smaller than any other edge's chain index incident to v. Likewise, there are two edges, whose chain indices are strictly greater than any other edge's chain index incident to v. We use these edges when constructing the trees, so we name these edges.

Definition 5.2. *For each vertex $v \neq r$, the two f-edges of v are the two edges incident to v with the lowest chain index. The two g-edges of v are the two edges incident to v with the highest chain index.*

Note that by definition f-edges cannot be edges from down chains. Similarly, g-edges cannot be edges from up chains.

We define the numbering f iteratively on the edges of all up chains and one-way chains. The numbering assigns distinct values from $\mathbb{R}$ to each edge. We consider two edges **consec-**

utive if they belong to the same up chain and are incident to the same internal vertex of the chain. Note that in case of a closed up chain, the two neighboring edges of the end are not consecutive.

We are numbering the edges of the graph in the order of the chain decomposition. We begin by numbering $E(G_0)$ and then we number each edge of the up chains and the one-way chains. Note that when we number G_i each edge in H_i belonging to up chains or one-way chains has been numbered. Since f-edges cannot be edges from down chains, we numbered every f-edge from H_i. We denote the f-, and g-values of an edge e by $f(e)$ and $g(e)$, respectively. When we number G_i, we number the edges based on the following rules:

- If G_i is a closed up chain containing r, then number the edges of $E(G_i)$ so that the values monotonically change between consecutive edges. The values can be chosen arbitrarily.
- If G_i is a closed up chain not containing r, then the end's f-edges have already been numbered. We call these edges the **numbering edges** of G_i. Number the edges of $E(G_i)$ so that the values monotonically change between consecutive edges and their values are between the values of the two numbering edges.
- If G_i is an open up chain containing r, then r is an end of the chain. Let the other end be $u \neq r$. At least one f-edge of u has already been numbered. We call this edge the numbering edge of G_i. Number the edges of $E(G_i)$ so that the values grow monotonically between consecutive edges and their value is greater than the numbering edge's value.
- If G_i is an open up chain not containing r, then at least one f-edge of each end has been numbered. Let the two ends be u and v. Choose an f-edge of each end. We can choose these edges to be distinct. If we could not, then v and u would have only one f-edge in $E(H_i)$, the one between them, but this would imply that the graph is disconnected. We denote them e_v and e_u and call them the numbering edges of G_i. We may assume that $f(e_v) < f(e_u)$. Starting from the edge incident to v in the chain, we umber the edges of $E(G_i)$ so that the values grow monotonically between consecutive

edges and their value stays between the values of the numbering edges of G_i.

- If G_i is a one-way chain with tail r, then arbitrarily number the edge G_i.
- If G_i is a one-way chain whose tail is not r, then both f-edges of the tail have been numbered with distinct values. Number the edge of G_i between the two values.

Symmetrically, we define the numbering g. The numbering assigns a number from $\mathbb{R}$ to each edge of one-way chains and down chains. To obtain the g-numbering, we follow the procedure in the reverse chain decomposition.

We construct the four trees R, B, G, Y as follows. For each vertex $v \neq r$, consider its two f-edges. Assign the one with the lower f-value to R and the one with the higher f-value to B. Similarly assign the g-edge of $v \neq r$ with lower g-value to G and the one with higher g-value to Y.

The following claim prepares us to prove that the trees are edge-independent.

Claim 5.3. *For any $v \neq r$, consider the edge e_1 assigned to the vertex v in R. Let v' be the other end of v_1. If $v' \neq r$, then let e'_1 be the edge assigned to v' in R. Then $CI(e'_1) \leq CI(e_1)$ and $f(e'_1) < f(e_1)$.*

Proof. Let e_2 be the edge assigned to B. We know that since e_1 is an f-edge, it cannot be in a down chain. We consider the two chain-types e_1 can be the part of:

- Suppose e_1 is in an up chain G_i. Since the chain decomposition is minimal and $v' \in V(G_i)$, the f-edges of v' are in $E(H_{i+1})$, thus $CI(e'_1) \leq i = CI(e_1)$.

  The edge e_2 is part of G_i or is the covering edge of v, which is an end of G_i. By definition $f(e_1) < f(e_2)$. Furthermore, by the numbering procedure there exists an edge e^* of v' for which $f(e^*) < f(e_1) < f(e_2)$. Using the definition of R: $f(e'_1) \leq f(e^*)$. Both desired conditions are fulfilled.

- Suppose e_1 is the one-way chain G_i. By assumption e_1 is an f-edge of v, consequently the degree of v can be at most one. By the definition of a one-way chain v is not eligible to be the tail of G_i, so it must be the head and v' must be the tail. The f-edges of v' have chain indices smaller than i, consequently $e_1 \neq e'_1$ and $CI(e'_1) < CI(e_1)$.

By the numbering procedure, $f(e_1)$ is between the value of the two f-edges of v'. By the definition of e_1' it is the smaller one, thus $f(e_1') < f(e_1)$. Both desired conditions are fulfilled.

□

We may conclude that to each vertex, we assign a different edge in R. Also, R does not contain a cycle, but has $|V| - 1$ edges, thus it is a spanning tree of G. From each vertex $v \neq r$ there is a unique path from v to r, through which the chain indices are decreasing, and the f-values are also decreasing. In the case of B, the chain indices are also decreasing, but the f-values are increasing. The independence of the two trees is demonstrated in the different behavior of f-values.

By symmetry, the independence of G and Y is also guaranteed by the different behavior of the f-values of the two trees. Their independence from R and B is ensured by their distinct behavior with respect to the chain indices. Although the chain indices show different monotonicity, their starting edges may be the same, but considering Theorem 4.12 and Theorem 4.13 the starting edges are different in each tree.

We showed that the four trees are indeed edge-disjoint, proving Conjecture 1.1 for $k = 4$. The steps of the proof also imply a polynomial-time algorithm to construct the trees.

We summarize the algorithm without proving the correctness of each step:

Procedure: Four trees

Input: A graph G and its chain decomposition $G_0, G_1, ..., G_m$ rooted at $r \in V(G)$.

Output: Four edge-independent spanning trees R, B, G, Y.

1: $R = \emptyset, B = \emptyset, G = \emptyset, Y = \emptyset$

2: for $i = 1, 2, ..., m$:

Let the edges of G_i be $e_1, e_2, ..., e_k$ in this order

- If G_i is a closed up chain and $r \in G_i$, then $f(e_j) = \dfrac{j}{k+1}$
- If G_i is a closed up chain and $r \notin G_i$. Let the value of the numbering edges of the end be l and u such that $l < u$. Then $f(e_j) = l + \dfrac{j(u-l)}{(k+1)}$
- If G_i is an open up chain and $r \in G_i$. The two endpoints are r and u. The vertex u is incident to e_1. Then $f(e_j) = \dfrac{j}{k+1}$
- If G_i is an open up chain and $r \notin G_i$. The two ends are u and v, such that u is incident to e_1. The value of one of the f-edges of u and v are l and u respectively, such that $l < u$. Then $f(e_j) = l + \dfrac{u-l}{k+1}$
- If G_i is a one-way chain with tail r, then $e_1 = i$
- If G_i is a one-way chain with tail $u \neq r$. The value of the two f-edges of u are l and u such that $l < u$. Then $f(e_1) = \dfrac{l+u}{2}$

3: for $i = m, m-1, ..., 1$: do the same as above but replace f by g

4: for all $v \in V - r$:

Let the two f-edges of v be e_1 and e_2 and the two g-edges of v be e_3 and e_4.

$R = R + v_1$ $B = B + v_2$ $G = G + v_3$ $Y = Y + v_4$

6 Applications

In this section, we define and study a routing problem that has been investigated in different settings. Itai and Rodeh proposed a solution in the context of general networks [5]. Gopalan and Ramasubramanian proposed a protocol for a similar problem [3]. We begin by presenting the approach of Itai and Rodeh, and subsequently discuss the algorithm by Gopalan and Ramasubramanian.

6.1 General routing problem

Consider a network $G = (V, E)$, where the vertices intend to send packets to each other. The vertices u and v can only communicate with each other if $uv \in E$. Each vertex has an information storage capacity. A vertex v (the sender) intends to send a packet to the vertex u (the receiver). If the graph G is connected, then this problem can be solved easily with a spanning tree. Our goal is to solve the problem by using limited storage space, which can result in faster message transmissions. It is sufficient to store one neighbor to a given receiver for each vertex. However, assume that some edges may "fail" in the sense that we cannot send packets through them. When an edge fails, its endpoints are not aware that it is faulty. Our goal is to be able to send the packets despite as many failures as possible. We will refer to a packet sending method as a routing mechanism. We say that a routing mechanism is k-reliable in case of at most $k-1$ edge failures we can send a packet from any vertex to any other vertex.

Edge-independent spanning trees are appilicable structures for solving these problems. Consider a graph $G = (V, E)$ with $T_1, T_2, ..., T_k$ edge-independent spanning trees with root r. Each vertex stores its parents and children, one for each of the k spanning trees rooted at r. If the vertex v intends to send a packet to the vertex u, then u sends a packet to the root r, then from the root r it is transmitted to u. Note that the packet contains its receiver vertex, so when a packet passes through a vertex it knows if it is the receiver of the packet. In addition, the packet also contains information that indicates which tree it was sent on. We can store this information with $\log k$ bits. We propagate the packet from the root r to

its children in all trees, who in turn forward it to their respective children, and so on. This way, we cover the whole graph. Summarizing how a vertex acts if it receives a packet:

- If the root receives a packet, it forwards it to all of its children in all trees.
- If vertex is not the root and the packet was sent by one of its children, then it forwards it to its parent in the respective tree.
- If the packet was sent by its parent, then it sends the packet to all of its children in the respective tree.

In case of $k-1$ edge failures, the vertex u is guaranteed to receive the packet due to the nature of edge-independent spanning trees. Note that in case of no edge failure, the vertex u receives the packet at most k^2+k times, since it may be part of each $v \to r$ paths, and when the packet arrives at r, it forwards it in all k trees. Thus, when we send the packet through one tree, the vertex u can receive it at most $k+1$ times. This is a theoretical result, as the method has a large number of messages, making it impractical to use.

6.2 A more practical routing approach

The previous routing protocol guarantees that the packet is received, but the same packet has been sent, transmitted, and received many times, leading to an overload in the network. Our goal now is not to send the packet to some vertex, but specifically to the root vertex. We present a routing protocol for the case $k = 3$. This routing protocol operates under stronger network assumptions, which allow for reduced communication overhead.

In this network, the vertex who sends a message on an edge is aware whether the vertex on the other end of the edge has successfully received the packet. This condition gives us the opportunity to redirect the packet when we encounter an edge failure. We refer to the three trees as red, blue and green (R, B, G, respectively). An additional characteristic of the network is that each vertex stores the color of the edges to which it is connected. We can think of the edges of the trees as arcs, directed towards the root. Consequently, an edge uv may have a $u-v$ color and a $v-u$ color.

The following method is called "Red tree first". We add an extra bit to the end of the packet indicating if it has already encountered two faulty edges. The bit is set to 0 at the beginning and switches to 1 if we have encountered two edge failures. The starting vertex sends the packet along the red edges towards the root.

a) One edge failure. Assume that the packet cannot be sent from u to v through the edge uv . The packet is sent towards the root through either the blue or the green tree. If the color of edge $v-u$ is blue, then the packet continues its path on the blue tree, if the color of $v-u$ is green, it continues on the green tree. If the edge does not have a $v-u$ color, then the next edge is chosen arbitrarily. The extra bit remains 0.

b) Two edge failures. Assume that the packet cannot be sent from u to v through the uv edge. The packet used either the blue or green tree until now. We send the packet towards the root on the green tree, if we have used the blue tree until now, and on the blue one, if we have used the green tree. The overhead bit changes to 1. If we encounter any following edge failures, the packet is dropped.

Alulírott Szepesi Balázs nyilatkozom, hogy szakdolgozatom elkészítése során az alább felsorolt feladatok elvégzésére a megadott MI alapú eszközöket alkalmaztam:

Feladat	Felhasznált eszköz	Felhasználás helye	Megjegyzés
Nyelvhelyesség ellenőrzése	Writefull	Teljes dolgozat	

References

[1] Baolei Cheng, Dajin Wang, and Jianxi Fan, *Independent spanning trees in networks: A survey*, ACM Comput. Surv. **55** (July 2023), no. 14s.

[2] J Cheriyan and S.N Maheshwari, *Finding nonseparating induced cycles and independent spanning trees in 3-connected graphs*, Journal of Algorithms **9** (1988), no. 4, 507–537.

[3] Abishek Gopalan and Srinivasan Ramasubramanian, *Ip fast rerouting and disjoint multipath routing with three edge-independent spanning trees*, IEEE/ACM Transactions on Networking **24** (2016), no. 3, 1336–1349.

[4] Alexander Hoyer and Robin Thomas, *Four edge-independent spanning trees*, SIAM Journal on Discrete Mathematics **32** (2018), no. 1, 233–248, available at `https://doi.org/10.1137/17M1134056`.

[5] Alon Itai and Michael Rodeh, *The multi-tree approach to reliability in distributed networks*, Information and Computation **79** (1988), no. 1, 43–59.

[6] Avram Zehavi and Alon Itai, *Three tree-paths*, Journal of Graph Theory **13** (1989), no. 2, 175–188, available at `https://onlinelibrary.wiley.com/doi/pdf/10.1002/jgt.3190130205`.