Efficient Constructions for the Győri-Lovász Theorem on Almost Chordal Graphs

Katrin Casel[1 [0000−0001−6146−8684]], Tobias Friedrich[1 [0000−0003−0076−6308]], Davis Issac[1 [0000−0001−5559−7471]], Aikaterini Niklanovits[1 [0000−0002−4911−4493]], and Ziena Zeif[1 [0000−0003−0378−1458]]

Hasso Plattner Institute, University of Potsdam, Potsdam 14482, Germany
{First name.Last name}@hpi.de

Abstract. In the 1970s, Győri and Lovász showed that for a k-connected n-vertex graph, a given set of terminal vertices $t_1, \ldots, t_k$ and natural numbers $n_1, \ldots, n_k$ satisfying $\sum_{i=1}^{k} n_i = n$, a connected vertex partition $S_1, \ldots, S_k$ satisfying $t_i \in S_i$ and $|S_i| = n_i$ exists. However, polynomial algorithms to actually compute such partitions are known so far only for $k \leq 4$. This motivates us to take a new approach and constrain this problem to particular graph classes instead of restricting the values of k. More precisely, we consider k-connected chordal graphs and a broader class of graphs related to them. For the first class, we give an algorithm with $\mathcal{O}(n^2)$ running time that solves the problem exactly, and for the second, an algorithm with $\mathcal{O}(n^4)$ running time that deviates on at most one vertex from the required vertex partition sizes.

Keywords: Győri-Lovász theorem · chordal graphs · HHD-free graphs.

1 Introduction

Partitioning a graph into connected subgraphs is a fundamental task in graph algorithms. Such *connected* partitions occur as desirable structures in many application areas such as image processing [8], road network decomposition [9], and robotics [17].

From a theoretical point of view, the existence of a partition into connected components with certain properties also gives insights into the graph structure. In theory as well as in many applications, one is interested in a connected partition that has a given number of subgraphs of chosen respective sizes. With the simple example of a star-graph, it is observed that not every graph admits a connected partition for any such choice of subgraph sizes. More generally speaking, if there exists a small set of t vertices whose removal disconnects a graph (*separator*), then any connected partition into $k > t$ subgraphs has limited choice of subgraph sizes. Graphs that do not contain such a separator of size less than k are called *k-connected*.

On the other hand, Győri and Lovász independently showed that k-connectivity is not just necessary but also sufficient to enable a connected partitioning into k subgraphs of required sizes, formally stated by the following result.

Győri-Lovász Theorem ([4],[7]). Let $k \geq 2$ be an integer, $G = (V, E)$ a k-connected graph, $t_1, \ldots, t_k \in V$ distinct vertices and $n_1, \ldots, n_k \in \mathbb{N}$ such that $\sum_{i=1}^{k} n_i = |V|$. Then G has disjoint connected subgraphs $G_1, \ldots G_k$ such that $|V(G_i)| = n_i$ and $t_i \in V(G_i)$ for all $i \in [k]$.

The caveat of this famous result is that the constructive proof of it yields an exponential time algorithm. Despite this result being known since 1976, to this day we only know polynomial constructions for restricted values of k. Specifically, in 1990 Suzuki et al. [15] provided such an algorithm for $k = 2$ and also for $k = 3$ [14]. Moreover in 1994 Wada et al. [16] also provided an extended result for $k = 3$. For the case of $k = 4$ Nakano et al. [10] gave a linear time algorithm for the case where $k = 4$, G is planar and the given terminals are located on the same face of a plane embedding of G, while in 2016 Hoyer and Thomas [5] provided a polynomial time algorithm for the case of $k = 4$. And so far, this is where the list ends, thus for $k \geq 5$ it remains open whether there even exists a polynomial time construction.

Towards a construction for general k, we consider restricting the class of k-connected graphs instead of the values of k. More precisely, we consider (generalizations of) *chordal* k-connected graphs. A graph is called *chordal*, if it does not contain an induced cycle of length more than three. The restriction to chordal graphs is known to often yield tractability for otherwise NP-hard problems, for example chromatic number, clique number, independence number, clique covering number, stable set and treewidth decomposition [13]. Apart from the interest chordal graphs have from a graph theoretic point of view, their structural properties have also been proven useful in biology when it comes to studying multidomain proteins and network motifs (see e.g. [12,11]).

Our contribution To the best of our knowledge, this paper is the first to pursue the route of restricting the Győri-Lovász Theorem to special graph classes in order to develop a polynomial construction for general values of k on a non-trivial subclass of k-connected graphs. We believe that in general considering the structure of the minimal separators of a graph is promising when it comes to developing efficient algorithms for the Győri-Lovász Theorem.

We give a constructive version of the Győri-Lovász Theorem for *chordal* k-connected graphs with a running time in $\mathcal{O}(|V|^2)$. Observe here that this construction works for all values of k. Then we show how this result can be generalized in two directions.

First, we generalize our result to the vertex weighted version of the Győri-Lovász Theorem (as proven independently by Chandran et al. [2], Chen et al. [3] and Hoyer [5]), specifically deriving the following theorem.

Theorem 1. *Let $k \geq 2$ be an integer, $G = (V, E, w)$ a vertex-weighted k-connected chordal graph with $w\colon V \to \mathbb{N}$, $t_1, \ldots, t_k \in V$ distinct vertices, and $w_1, \ldots, w_k \in \mathbb{N}$ with $w_i \geq w(t_i)$ and $\sum_{i=1}^{k} w_i = w(V)$ for all $i \in [k]$. A partition $S_1, \ldots, S_k$ of V, such that $G[S_i]$ is connected, $t_i \in S_i$ and $w_i - w_{max} < w(S_i) < w_i + w_{max}$, for all $i \in [k]$, can be computed in time $\mathcal{O}(|V|^2)$.*

We further use this weighted version to derive an approximate version of the Győri-Lovász Theorem for a larger graph class. Specifically we define I_j^i to contain all graphs that occur from two distinct chordless C_j's that have at least i vertices in common. We focus on I_4^2-free combined with HH-free graphs. More specifically, we consider the subclass of k-connected graphs that contain no hole or house as subgraph (see preliminaries for the definitions of structures such as hole, house etc.) and that does not contain two distinct induced C_4 that share more than one vertex. We call this class of graphs HHI_4^2-free. Note that HHI_4^2-free, apart from being a strict superclass of chordal graphs, is also a subclass of HHD-free graphs (that is house, hole, domino-free graphs), a graph class studied and being used in a similar manner as chordal graphs as it is also a class where the minimum fill-in set is proven to be polynomially time solvable [1] (see also [6] for NP-hard problems solved in polynomial time on HHD-free graphs). Taking advantage of the fact that given an HHI_4^2-free graph, the subgraph formed by its induced C_4 has a treelike structure, we are able to derive the following result.

Theorem 2. *Let $k \geq 2$ be an integer, $G = (V, E, w)$ a vertex-weighted k-connected HHI_4^2-free graph with $w\colon V \to \mathbb{N}$, $t_1, \ldots, t_k \in V$ distinct vertices, and $w_1, \ldots, w_k \in \mathbb{N}$ with $w_i \geq w(t_i)$ and $\sum_{i=1}^{k} w_i = w(V)$ for all $i \in [k]$. A partition $S_1, \ldots, S_k$ of V, such that $G[S_i]$ is connected, $t_i \in S_i$ and $w_i - 2w_{max} < w(S_i) < w_i + 2w_{max}$, for all $i \in [k]$, can be computed in time $\mathcal{O}(|V|^4)$.*

Notice that the above theorem implies a polynomial time algorithm with an additive error of 1 for the unweighted case.

2 Preliminaries

All graphs mentioned in this paper are undirected, finite and simple. Given a graph G and a vertex $v \in V(G)$ we denote its *open neighborhood* by $N_G(v) := \{u \in V(G) \mid uv \in E(G)\}$ and by $N_G[v]$ its *closed neighborhood*, which is $N(v) \cup \{v\}$. Similarly we denote by $N_G(S) := \bigcup_{v \in S} N_G(v) \setminus S$ the open neighborhood of a vertex set $S \subseteq V(G)$ and by $N_G[S] := N_G(S) \cup S$ its closed neighborhood. We omit the subscript G when the graph we refer to is clear from the context. A vertex $v \in V(G)$ is *universal to a vertex set* $S \subset V(G)$ if $S \subseteq N(v)$. Let G be a graph and $S \subseteq V(G)$. The *induced subgraph* from S, denoted by $G[S]$, is the graph with vertex set S and all edges of $E(G)$ with both endpoints in S.

A graph G is *chordal* if any cycle of G of size at least 4 has a chord (i.e., an edge linking two non-consecutive vertices of the cycle). A vertex $v \in V(G)$ is called *simplicial* if $N[v]$ induces a clique. Based on the existence of simplicial vertices in chordal graphs, the following notion of vertex ordering was given. Given a graph G, an ordering of its vertices $(v_1, \ldots, v_n)$ is called *perfect elimination ordering* (p.e.o.) if v_i is simplicial in $G[\{v_i, v_{i+1}, \ldots, v_n\}]$ for all $i \in [n]$. Given such an ordering $\sigma : V(G) \to \{1, \ldots, n\}$ and a vertex $v \in V(G)$ we call $\sigma(v)$ the *p.e.o. value of* v. Rose et al. [13] proved that a p.e.o. of any chordal graph can be computed in linear time.

Let $e = \{u, v\}$ be an edge of G. We denote by G/e the graph G', that occurs from G by the contraction of e, that is, by removing u and v from G and replacing it by a new vertex z whose neighborhood is $(N(u) \cup N(v)) \setminus \{u, v\}$.

A graph G is *connected* if there exists a path between any pair of distinct vertices. Moreover, a graph is *k-connected* for some $k \in \mathbb{N}$ if after the removal of any set of at most $k-1$ distinct vertices G remains connected. Given a graph G and a vertex set $S \subseteq V(G)$, we say that S is a *separator* of G if its removal disconnects G. We call S a *minimal separator* of G if the removal of any subset $S' \subseteq V(G)$ with $|S'| < |S|$ results in a connected graph.

We now define some useful subgraphs, see also Figure 1 for illustrations. An induced chordless cycle of length at least 5 is called a *hole*. The graph that occurs from an induced chordless C_4 where exactly two of its adjacent vertices have a common neighbor is called a *house*. When referring to just the induced C_3 that is part of a house we call it *roof* while the induced C_4 is called *body*. Two induced C_4 sharing exactly one edge form a *domino*. A graph that contains no hole, house or domino as an induced subgraph is called HHD-free. We call a graph that consists of two C_4 sharing a vertex, and an edge that connects the two neighbors of the common vertex in a way that no other C_4 exists a *double house*.

Lastly, let $G = (V, E)$ be a k-connected graph, let $t_1, \ldots, t_k \in V$ be k distinct vertices, and let $n_1, \ldots, n_k$ be natural numbers satisfying $\sum_{i=1}^{k} n_i = |V|$. We call $S_1, \ldots S_k \subseteq V(G)$ a *GL-Partition of G* if $S_1, \ldots S_k$ forms a partition of $V(G)$, such that for all $i \in [k]$ we have that $G[S_i]$ is connected, $t_i \in S_i$ and $|S_i| = n_i$. When there exists an $l \in \mathbb{N}$, such that for such a partition only $n_i - l \leq |S_i| \leq n_i + l$ holds instead of $|S_i| = n_i$, we say that $S_1, \ldots, S_k$ is a *GL-Partition of G with deviation l*.

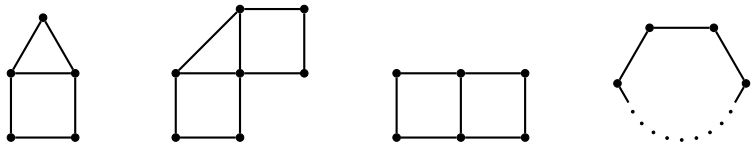

Fig. 1. Specific subgraphs used throughout the paper, from left to right: house, double house, domino and hole example

3 GL-Partition for Chordal Graphs

We present a simple implementable algorithm with quadratic running time that computes GL-Partitions in chordal graphs. We then show that a slight modification of our algorithm is sufficient to compute a GL-Partition on a vertex weighted graph, thus proving Theorem 1.

Due to space restrictions the proof of Lemma 1 has been moved to Appedix B, and the proof of correctness of Algorithm 2 to Appendix A.

3.1 GL-Partition for Unweighted Chordal Graphs

For simplicity, we first prove the restricted version of Theorem 1 to unweighted graphs. We use a p.e.o. to compute a vertex partition, as described formally in Algorithm 1. This algorithm receives as input a k-connected chordal graph $G = (V, E)$, terminal vertices $t_1, \ldots, t_k \in V$, and natural numbers $n_1, \ldots, n_k$ satisfying $\sum_{i=1}^{k} n_i = n$, and outputs connected vertex sets $S_1, \ldots, S_k \subseteq V$ such that $|S_i| = n_i$ and $t_i \in S_i$. In the beginning of the algorithm we initialize each set S_i to contain only the corresponding terminal vertex t_i, and add vertices iteratively to the *non-full sets* (S_i's that have not reached their demanded size). We say a vertex v *is assigned* if it is already part of some S_i and *unassigned* otherwise. At each iteration, the unassigned neighborhood of the union of the previously non-full sets is considered, and the vertex with the minimum p.e.o. value is selected to be added to a non-full set. In case there is more than one non-full set in the neighborhood of this vertex, it is added to the one with lowest priority, where the priority of each set is defined to be the largest p.e.o. value of its vertices so far. The algorithm terminates once all vertices are assigned, in $\mathcal{O}(|V|^2)$ time.

Algorithm 1: `ChordalGL`

Input: k-connected chordal graph $G = (V, E)$, terminal vertices $t_1, \ldots, t_k \in V$, and natural numbers $n_1, \ldots, n_k$ satisfying $\sum_{i=1}^{k} n_i = n$
Output: Connected vertex sets $S_1, \ldots, S_k \subseteq V$ such that $|S_i| = n_i$ and $t_i \in S_i$

$\sigma \leftarrow$ Compute p.e.o. of G as function $\sigma \colon V \to |V|$
$S_i \leftarrow \{t_i\}$, for all $i \in [k]$
while $\bigcup_{i \in [k]} S_i \neq V(G)$ **do**
  $I \leftarrow \{i \in [k] \mid |S_i| < n_i\}$
  $V' \leftarrow N(\bigcup_{i \in I} S_i) \setminus \bigcup_{i \in [k] \setminus I} S_i$
  $v' \leftarrow \arg\min_{v \in V'} \sigma(v)$
  $J \leftarrow \{i \in I \mid v \in N(S_i)\}$
  $j' \leftarrow \arg\min_{j \in J} \max(\sigma(S_j))$
  $S_{j'} \leftarrow S_{j'} \cup \{v'\}$
end
return $S_1, \ldots, S_k$

For the correctness of Algorithm 1 it is enough to show that the unassigned neighborhood V' of all non-full sets is not empty in each iteration of the while-loop, since this implies that we enlarge a non-full set (in the algorithm denoted as $S_{j'}$) by one vertex (in the algorithm denoted as v') while maintaining the size of all remaining sets. That is, in each iteration we make progress in the sense that $|\bigcup_{i \in [k]} S_i|$ increases while maintaining the invariant $|S_i| \leq n_i$ for all S_i's. Note that $v' \in N(S_{j'})$ which in turn implies that $G[S_i]$ is always connected for all $i \in [k]$. Finally, by $\sum_{i=1}^{k} n_i = n$ and through the way we update I we ensure that the algorithm (or while-loop) terminates as $\bigcup_{i \in [k]} S_i = V$ only if we have $|S_i| = n_i$ for all S_i's.

Towards proving the required lemmata for the correctness of Algorithm 1 we make the following observation for the p.e.o. of a graph.

Lemma 1. *Let σ be a p.e.o of a graph $G = (V, E)$ and $P = \{v_1, v_2, \ldots, v_k\}$ a vertex set of G that induces a simple path with endpoints v_1 and v_k. Then $\sigma(v_i) > \min\{\sigma(v_1), \sigma(v_k)\}$ for all $i = 2, \ldots, k-1$.*

Lemma 2. *In each iteration of the while-loop in Algorithm 1 we have $V' \neq \varnothing$.*

Proof. We first define the *z-connecting* neighborhood of a vertex v to be the neighbors of v that are included in some induced path connecting v to z.

We prove that every non-full set S_i contains a vertex in its neighborhood $N(S_i)$ that is unassigned, which implies that $V' \neq \varnothing$. Assume for a contradiction that at some iteration of our algorithm there is an non-full set S_i whose neighborhood is already assigned to other sets. Let v be the vertex of S_i of maximum σ value among its vertices and z be the vertex of maximum σ value among the unassigned vertices. Note that $vz \notin E(G)$. Let $\mathcal{P}$ be the set of all simple induced paths of G with endpoints z and v. Consider now the following cases:

1. If $\sigma(z) > \sigma(v)$, we get from Lemma 1 that every internal vertex of each path in $\mathcal{P}$ has higher σ value than v. Note that no vertex of S_i is an internal vertex of some path in $\mathcal{P}$, since all of them have smaller σ value than v by the selection of v. Denote the z-connecting neighborhood of v by C.
   Let a, b be two vertices in C and assume that $a, b \in S_j$ for some j. Assume also that during our algorithm, a is added to S_j before b. Since all vertices of S_i have smaller σ value than both a and b, and a is added to S_j before b, the moment b is added to S_j, S_i has already been formed. Consider now the iteration that this happens. Since $b \in N(v)$, $G[S_i \cup \{b\}]$ is connected. Moreover since $\sigma(a) > \sigma(v)$ and S_i is not full, b should be added to S_i instead of S_j. As a result each set apart from S_i contains at most one such neighbor of v, and hence $|C| < k$.
   Observe that $G \setminus C$ has no induced path connecting z and v which in turn implies that $G \setminus C$ has no $z - v$ path in general. However, this contradicts the k-connectivity of G.
2. If $\sigma(z) < \sigma(v)$, since z is the unassigned vertex of the highest σ value among all unassigned vertices, and by Lemma 1 all vertices in $\mathcal{P}$ have greater σ value than z, all of its v-connecting neighbors in $\mathcal{P}$ are already assigned in some set. Denote the set of v-connecting neighbors of z by C.
   Assume now that there are two vertices of C, a and b, that are contained in some S_j and assume also without loss of generality that a was added to S_j before b. Note that since $\sigma(z) < \sigma(b)$ at each iteration of our algorithm z is considered before b to be added to some set if the induced graph remains connected. As a result, after a is added to S_j, the induced subgraph $G[S_j \cup \{z\}]$ is connected and hence z should be added to S_j before b.
   This means that each set contains at most one v-connecting neighbor of z and therefore $|C| < k$. Since $G \setminus C$ has no induced path connecting z and v, there is no z-v-path in $G \setminus C$, which contradicts the k-connectivity.

Corollary 1. *At each iteration of Algorithm 1, unless all vertices are assigned, the neighborhood of each non-full set contains at least one unassigned vertex.*

In the weighted case we use the above corollary of Lemma 2. In particular, it follows from Corollary 1 that as long as we do not declare a set to be full, we ensure that we are able to extend it by a vertex in its neighborhood that is unassigned. Note that in the weighted case we do not know in advance how many vertices are in each part.

3.2 GL-Partition for Weighted Chordal Graphs

With a slight modification of Algorithm 1 we can compute the weighted version of a GL-Partition. In particular, we prove Theorem 1.

The input of our algorithm differs from the unweighted case by having a positive vertex-weighted graph $G = (V, E, w)$ and instead of demanded sizes $n_1, \ldots, n_k$ we have demanded weights $w_1, \ldots, w_k$ for our desired vertex sets $S_1, \ldots, S_k$, where $\sum_{i=1}^{k} w_i = w(V)$. Note also that $w(S_i)$ is not allowed to deviate more than $w_{max} = \max_{v \in V} w(v)$ from w_i, i.e. $w_i - w_{max} < w(S_i) < w_i + w_{max}$.

Algorithm 2: `WeightedChordalGL`

Input: k-connected vertex-weighted chordal graph $G(V, E, w)$, terminal vertices $t_1, \ldots, t_k \in V$, and positive weights $w_1, \ldots, w_k$ satisfying $\sum_{i=1}^{k} w_i = w(V)$

Output: Connected vertex sets $S_1, \ldots, S_k \subseteq V$ such that $w_i - w_{\max} < w(S_i) < w_i + w_{\max}$ and $t_i \in S_i$

$\sigma \leftarrow$ Compute p.e.o. of G as function $\sigma \colon V \to |V|$
$S_i \leftarrow \{t_i\}$, for all $i \in [k]$
$I \leftarrow \{i \in [k] \mid w(S_i) < w_i\}$
while $|I| \neq 1$ ***and*** $\bigcup_{i \in [k]} S_i \neq V(G)$ **do**
  $V' \leftarrow N(\bigcup_{i \in I} S_i) \setminus \bigcup_{i \in [k] \setminus I} S_i$
  $v' \leftarrow \arg\min_{v \in V'} \sigma(v)$
  $J \leftarrow \{i \in I \mid v \in N(S_i)\}$
  $j' \leftarrow \arg\min_{j \in J} \max(\sigma(S_j))$
  if $w(S_{j'}) + w(v') < w_{j'}$ **then**
    $S_{j'} \leftarrow S_{j'} \cup \{v'\}$
  end
  else
    $I \leftarrow I \setminus \{j'\}$
    if $\sum_{i \in [k] \setminus I} (w_i - w(S_i)) \geq 0$ ***or*** $w(S_{j'}) + w(v') = w_{j'}$ **then**
      $S_{j'} \leftarrow S_{j'} \cup \{v'\}$
    end
  end
end
If $|I| = 1$, assign all vertices $V \setminus \bigcup_{i \in [k]} S_i$ (possibly empty) to S_j with $j \in I$.

Again we set each terminal vertex t_i to a corresponding set S_i, and enlarge iteratively the *non-full weighted sets* (S_i's that are not declared as full). One difference to the previous algorithm is that we declare a set S_i as *full weighted set*, if together with the next vertex to be potentially added its weight would exceed w_i. After that, we decide whether to add the vertex with respect to the currently full weighted sets. Similar to Algorithm 1 we interrupt the while-loop if $S_1, \ldots, S_k$ forms a vertex partition of V and the algorithm terminates. However, to ensure that we get a vertex partition in every case, we break the while-loop when only one non-full weighted set is left and assign all remaining unassigned vertices to it.

Observe that we can make use of Corollary 1, since Algorithm 2 follows the same priorities concerning the p.e.o. as Algorithm 1. Basically, it implies that as long we do not declare a set as full weighted set and there are still unassigned vertices then those sets have unassigned vertices in its neighborhood.

We conclude this section by extending the above algorithms to graphs having distance $k/2$ from being chordal. In particular this corollary is based on the observation that an edge added to a graph does not participate in any of the parts those algorithms output if both of its endpoints are terminal vertices.

Corollary 2. *Let G be a k-connected graph which becomes chordal after adding $k/2$ edges. Given this set of edges, a GL-Partition (also its weighted version) can be computed in polynomial time but without fixed terminals.*

4 GL-Partition for $\mathbf{HHI_4^2}$-free

This section is dedicated to the proof of Theorem 2. The underlying idea for this result is to carefully contract edges to turn a k-connected HHI_4^2-free graph into a chordal graph that is still k-connected. Note that we indeed have to be very careful here to find a set of contractions, as we need it to satisfy three seemingly contradicting properties: removing all induced C_4, preserving k-connectivity, and contracting at most one edge adjacent to each vertex. The last property is needed to bound the maximum weight of the vertices in the contracted graph. Further, we have to be careful not to contract terminal vertices.

The computation for the unweighted case of the partition for Theorem 2 is given in Algorithm 3 below, which is later extended to the weighted case as well. Note that we can assume that $n_i \geq 2$ since if $n_i = 1$ for some $i \in [k]$ we simply declare the terminal vertex to be the required set and remove it from G. This gives us a $(k-1)$-connected graph and $k-1$ terminal vertices.

Before starting to prove the Lemmata required for the correctness of Algorithm 3 we give a structural insight which is used in almost all proofs of the following Lemmata. Due to space restrictions the proofs of Lemmata 3 to 6, 8 and 10 have been moved to Appendix B.

Lemma 3. *Given an HHI_4^2-free graph G and an induced C_4, $C \subseteq V(G)$, then any vertex in $V(G) \setminus C$ that is adjacent to two vertices of C is universal to C. Moreover, the set of vertices that are universal to C induces a clique.*

Algorithm 3: HHI_4^2-free GL

Input: k-connected HHI_4^2-free graph $G(V, E)$, terminal vertices $t_1, \ldots, t_k \in V$, and positive integers $n_1, \ldots, n_k \geq 2$ satisfying $\sum_{i=1}^{k} n_i = n$

Output: Connected vertex sets $S_1, \ldots, S_k \subseteq V$ such that $n_i - 1 \leq |S_i| \leq n_i + 1$ and $t_i \in S_i$

Add an edge between each pair of non-adjacent terminals that are part of an induced C_4

$\mathcal{C} \leftarrow$ Set of all induced C_4 in G.

$G' \leftarrow (\bigcup_{C \in \mathcal{C}} V(C), \bigcup_{C \in \mathcal{C}} E(C))$

$E' \leftarrow \varnothing$

while $\mathcal{C} \neq \varnothing$ **do**

  Select three vertices v_1, v_2, v_3 in G' and the corresponding cycle $C \in \mathcal{C}$ that satisfies that for all $C' \in \mathcal{C} \setminus \{C\}$ we have $V(C') \cap \{v_1, v_2, v_3\} = \varnothing$.

  Pick a vertex v from v_1, v_2, v_3 that is not a terminal vertex and add an incident edge of v in $G'[\{v_1, v_2, v_3\}]$ to E'.

  Remove the cycle C from $\mathcal{C}$ and the vertices v_1, v_2, v_3 from G'.

end

Transform G to a weighted graph G'' by contracting each edge of E' in G, assigning to each resulting vertex as weight the number of original vertices it corresponds to.

$S_1, \ldots S_k \leftarrow$ Run Algorithm 2 with G'', the given set of terminals $t_1, \ldots, t_k$, and the size (or weight) demands $n_1, \ldots, n_k$ as input.

Reverse the edge contraction of E' in the sets $S_1, \ldots, S_k$ accordingly.

Lemma 4. *Let G be an HHI_4^2-free graph. If G contains a double house as a subgraph then at least one of the two C_4 in it has a chord.*

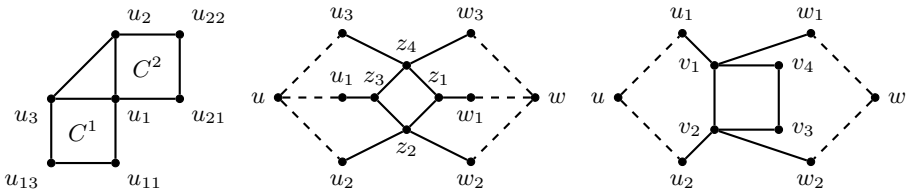

Fig. 2. Illustrations for the vertex namings used in proofs, from left to right: Lemma 4, Lemma 7 and Lemma 8

The following lemma captures the essence of why the algorithm provided in this section cannot be applied also on HHD-free graphs, since it holds for HHI_4^2-free graphs but not for HHD-free graphs. Think for example of a simple

path P of length 5 and a vertex disjoint induced chordless C_4, C. Consider also each vertex of P being universal to C. Observe that this graph is HHD- but not HHI_4^2-free. Every two non adjacent vertices of C together with the endpoints of P create an induced chordless C_4. Adding a chord connecting the two endpoints of P creates a hole and hence the resulting graph is not HHD-free.

Lemma 5. *Let G be a HHI_4^2-free graph and $C = \{v_1, v_2, v_3, v_4\}$ an induced C_4 in G. Then the graph G' created by adding the chord v_1v_3 to G is HHI_4^2-free and has one less induced C_4 than G.*

An essential property of the graph class we work on is being closed under contraction, since our algorithm is based on contracting edges iteratively until the resulting graph becomes chordal. Before proving this property though, although "after an edge contraction a new cycle is created" is intuitively clear, we formally define what it means for a C_4 to be "new".

Definition 1. *Let G be a graph, $uv \in E(G)$ and $G' = G/uv$. Let also w be the vertex of G' that is created by the contraction of uv. We say that an induced cycle C containing w in G' is new if $N_C(w) \not\subseteq N_G(v)$ and $N_C(w) \not\subseteq N_G(u)$.*

Lemma 6. *HHI_4^2-free graphs are closed under contraction of an edge of an induced C_4.*

In order to prove that the contractions of our algorithm do not affect the connectivity, we first study the possible role of vertices on an induced C_4 in minimal separators in HHI_4^2-free graphs.

Lemma 7. *Let G be a k-connected HHI_4^2-free graph for $k \geq 5$. Then no three vertices of an induced C_4 belong in the same minimal separator.*

Proof. Let G be a k-connected HHI_4^2-free graph for $k \geq 5$ and v_1, v_2, v_3, v_4 vertices that induce a C_4, C. Assume that v_1, v_2, v_3 belong in the a same minimal separator S and hence, $(G \setminus \{v_1, v_2, v_3\})$ is only $k-3$ connected. Let also u and w be two distinct vertices belonging in different connected components of $G \setminus S$.

Consider now the chordal graph G' created, by adding v_2v_4 to C and one chord to each other induced C_4 of G. By Lemma 5 this is possible by adding exactly one chord to each induced C_4 of G - in particular each addition does not create new induced C_4. Since G' is chordal each minimal separator induces a clique, and hence v_1, v_2, v_3 cannot be part of the same minimal separator in G' because they do not induce a triangle in G'. Thus $G' \setminus S$ remains connected.

Let P_1 be a $u - w$ path in $G' \setminus S$ that contains a minimal number of added edges. Let $z_1z_3 \in E(P_1)$ be one of the added edges, such that z_3 is closer to u on P_1 than z_1. Note that z_1 and z_3 are part of some induced C_4, $C' = \{z_1, z_2, z_3, z_4\}$ in G. Since z_1z_3 cannot be replaced by neither z_1z_2, z_2z_3, nor z_1z_4, z_4z_3 (otherwise we get a path with strictly less added edges than P_1) it follows that $z_2, z_4 \in S$.

We will use the $u - w$ paths through S in G to reach a contradiction. Since S is a minimal $u - w$ separator in G, there are two internally vertex disjoint

$u - w$ paths P_2 and P_3, with $P_2 \cap S = \{z_2\}$ and $P_3 \cap S = \{z_4\}$. Let w_2 be the neighbor of z_2 on P_2 closer to w, w_1 the respective neighbor of z_1 on P_1 and w_3 the respective neighbor of z_4 on P_3. Let also u_1, u_2, u_3 be the corresponding neighbors of these paths closer to u. See the illustration in Figure 2 for these namings, keeping in mind that it could be $w_1 \in \{w_2, w_3\}$ or $u_1 \in \{u_2, u_3\}$ or also $w_1 = w_2 = w_3 = w$ or $u_1 = u_2 = u_3 = u$.

We claim that, in G, z_3 is adjacent to a vertex on $P_2^{[w_2,w]}$ or $P_3^{[w_3,w]}$. Assume otherwise, and assume that $P_2^{[w_2,w]}$, $P_3^{[w_3,w]}$ are induced paths in G (shortcut them otherwise). If $w_2 = w_3$ then notice that $w_2 = w = w_3$. In order for z_3, z_2, z_4, w not to induce a C_4 with three common vertices to C, z_3 has to be adjacent to w which is on $P_2^{[w_2,w]}$. If $w_2 \neq w_3$ then assume without loss of generality that $w_2 \neq w$. In order to not be a hole, there has to be a chord in the cycle build by $P_2^{[w_2,w]}$, $P_3^{[w_3,w]}$ with z_4, z_3, z_2. By assumption, this chord cannot be from z_3, so it has to involve z_4 or z_2. Since $P_2^{[w_2,w]}$ and $P_3^{[w_3,w]}$ are induced and $w_2 \neq w$, either w_2 is adjacent to z_4, or $w_3 \neq w$ is adjacent to z_2. Both cases create a C_4 that has three vertices in common with C, (w_2, z_2, z_3, z_4, and w_3, z_2, z_3, z_4, resp.) and since $z_4z_2 \notin E(G)$, the added chord for these C_4 has to be w_2z_3, resp. w_3z_3, leading again to z_3 being adjacent to some vertex on $P_2^{[w_2,w]}$ or $P_3^{[w_3,w]}$.

Thus we conclude that z_3 is adjacent to a vertex x on $P_2^{[w_2,w]}$ or $P_3^{[w_3,w]}$ in G. This however allows to create a path from u to w with (at least) one added edge less than P_1 in G (since P_2, P_3 do not contain any added edges). Specifically, if x is on P_2 we get $P_1' = P_1^{[u,z_3]}xP_2^{[x,w]}$ and if $x \in P_3$, $P_1' = P_1^{[u,z_3]}xP_3^{[x,w]}$.

Since C' was an arbitrary cycle we conclude that v_1, v_2, v_3 cannot be part of the same minimal separator in G.

Lemma 8. *Let G be an HHI$_4^2$-free k-connected graph and $C = \{v_1, v_2, v_3, v_4\}$ be an induced C_4. The graph $G' = G/v_1v_2$ is still k-connected.*

Now, we finally look specifically at Algorithm 3, and first show that its subroutine creating G'' works correctly.

Lemma 9. *Given an HHI$_4^2$-free graph G, the vertices selected in line 6 of Algorithm 3 indeed exist as long as an induced C_4 exists.*

Proof. Let G be an HHI$_4^2$-free graph and $\mathcal{C}$ the set of all induced C_4 in G, consider the bipartite graph T constructed through the following procedure: Its vertices are partitioned into two sets B, and S referred to as *big* and *small* vertices of T, respectively. Each big vertex represents an induced C_4 of $\mathcal{C}$ while each small vertex represents a vertex of G participating in at least two induced C_4. Each small vertex is adjacent to the big vertices which represent a C_4 this vertex participates in. We claim that with this definition T is indeed a tree (actually a forest). Assume now for a contradiction that T contains a cycle and let C be one of the shortest such cycles in T.

First, consider the case that C has length $l \geq 6$. Since T is bipartite, due to its construction, l is even and the vertices of $C = \{s_1, b_1, s_2, b_2, \ldots, s_{l/2}, b_{l/2}\}$

alternate between big and small. We denote by $P_{b_i}^{s_w,s_z}$ a shortest path containing edges from the C_4 represented by the big vertex b_i with endpoints the vertices represented by s_w and s_z. Due to C, the cycle $P_{b_1}^{s_1,s_{l/2}} \dots P_{b_{l/2}}^{s_{l/2-1},s_{l/2}}$ exists in G as a subgraph. Note that since we have assumed that C is a minimal length cycle of T it is also chordless. Hence, in order for a hole not to be an induced subgraph of G at least one chord must exist connecting two vertices corresponding to two small ones of T. This however would create a double house as a subgraph with the two C_4 forming it being the two that correspond to big vertices of T. By Lemma 4 this means that one of the C_4 is not induced, a contradiction to the construction of T. Notice also that in the case where $l = 6$ we directly find a double house and reach a contradiction using the same arguments.

Moreover the assumption that $l = 4$, leads us to a contradiction to the fact that two C_4 have at most one vertex in common. Hence, T is a forest and the vertices mentioned in line 6 are the ones belonging only to a cycle represented by one leaf belonging in B.

Lemma 10. *Given an HHI$_4^2$-free graph G, lines 1-10 of Algorithm 3 transforms G into a weighted chordal graph G'', with the same connectivity as G and such that each vertex from G is involved in at most one edge contraction to create G''.*

At last, notice that we can easily alter Algorithm 3 to also work for weighted graphs, with the simple change of setting the weights of a vertex in G'' in line 10 to the sum of the weights of the original vertices it was contracted from. With this alteration, we can conclude now the proof of Theorem 2 with the following.

Lemma 11. *Algorithm 3 works correctly and runs in time $\mathcal{O}(|V|^4)$.*

Proof. By Lemma 10, G'' is a chordal graph with maximum vertex weight $2w_{max}$. Further, observe that we did not merge terminal vertices with each other, thus we can properly run Algorithm 2 on it. By the correctness of this algorithm (Theorem 1), we know that $S_1, \dots, S_k$ in line 11 is a GL-partition for G'' with deviation $2w_{max}$. Since reversing edge-contraction does not disconnect these sets, the unfolded sets $S_1, \dots, S_k$ are thus also a GL-partition for G with deviation $2w_{max}$; note here that the only edges we added to create G'' are between terminal vertices, which are in separate sets S_i by definition.

The most time consuming part of Algorithm 3 is the preprocessing to transform the input graph into a weighted chordal graph which requires $\mathcal{O}(|V|^4)$ time in order to find all the induced C_4 (note that the induced C_4 are at most $(n-4)/3$ since they induce a tree).

Moreover, as is the case for chordal graphs, we can sacrifice terminals to enlarge the considered graph class.

Corollary 3. *Let G be a k-connected graph which becomes HHI$_4^2$-free after adding $k/2$ edges. Then, given those edges, a GL-Partition of G with deviation 1 (also its weighted version with deviation $2w_{max} - 1$) can be computed in polynomial time but without fixed terminals.*

References

1. Broersma, H., Dahlhaus, E., Kloks, T.: Algorithms for the treewidth and minimum fill-in of HHD-free graphs. In: International Workshop on Graph-Theoretic Concepts in Computer Science (WG). pp. 109–117 (1997). https://doi.org/10.1007/BFb0024492, https://doi.org/10.1007/BFb0024492
2. Chandran, L.S., Cheung, Y.K., Issac, D.: Spanning tree congestion and computation of generalized Győri-Lovász partition. In: International Colloquium on Automata, Languages, and Programming, (ICALP). LIPIcs, vol. 107, pp. 32:1–32:14 (2018). https://doi.org/10.4230/LIPIcs.ICALP.2018.32, https://doi.org/10.4230/LIPIcs.ICALP.2018.32
3. Chen, J., Kleinberg, R.D., Lovász, L., Rajaraman, R., Sundaram, R., Vetta, A.: (almost) tight bounds and existence theorems for single-commodity confluent flows. Journal of the ACM **54**(4), 16 (2007). https://doi.org/10.1145/1255443.1255444, https://doi.org/10.1145/1255443.1255444
4. Győri, E.: On division of graphs to connected subgraphs, combinatorics. In: Colloq. Math. Soc. Janos Bolyai, 1976 (1976)
5. Hoyer, A.: On the independent spanning tree conjectures and related problems. Ph.D. thesis, Georgia Institute of Technology (2019)
6. Jamison, B., Olariu, S.: On the semi-perfect elimination. Advances in Applied Mathematics **9**(3), 364–376 (1988)
7. Lovász, L.: A homology theory for spanning tress of a graph. Acta Mathematica Hungarica **30**(3-4), 241–251 (1977)
8. Lucertini, M., Perl, Y., Simeone, B.: Most uniform path partitioning and its use in image processing. Discrete Applied Mathematics **42**(2), 227–256 (1993). https://doi.org/10.1016/0166-218X(93)90048-S, https://doi.org/10.1016/0166-218X(93)90048-S
9. Möhring, R.H., Schilling, H., Schütz, B., Wagner, D., Willhalm, T.: Partitioning graphs to speedup Dijkstra's algorithm. ACM Journal of Experimental Algorithmics **11** (2006). https://doi.org/10.1145/1187436.1216585, https://doi.org/10.1145/1187436.1216585
10. Nakano, S., Rahman, M.S., Nishizeki, T.: A linear-time algorithm for four-partitioning four-connected planar graphs. Information Processing Letters **62**(6), 315–322 (1997). https://doi.org/10.1016/S0020-0190(97)00083-5, https://doi.org/10.1016/S0020-0190(97)00083-5
11. Przytycka, T.M.: An important connection between network motifs and parsimony models. In: International Conference on Research in Computational Molecular Biology (RECOMB). vol. 3909, pp. 321–335. Springer (2006). https://doi.org/10.1007/11732990_27, https://doi.org/10.1007/11732990_27
12. Przytycka, T.M., Davis, G.B., Song, N., Durand, D.: Graph theoretical insights into evolution of multidomain proteins. Journal of Computational Biology **13**(2), 351–363 (2006). https://doi.org/10.1089/cmb.2006.13.351, https://doi.org/10.1089/cmb.2006.13.351
13. Rose, D.J., Tarjan, R.E., Lueker, G.S.: Algorithmic aspects of vertex elimination on graphs. SIAM Journal on Computing **5**(2), 266–283 (1976). https://doi.org/10.1137/0205021, https://doi.org/10.1137/0205021
14. Suzuki, H., Takahashi, N., Nishizeki, T., Miyano, H., Ueno, S.: An algorithm for tripartitioning 3-connected graphs. Journal of Information Processing Society of Japan **31**(5), 584–592 (1990)

15. Suzuki, H., Takahashi, N., Nishizeki, T.: A linear algorithm for bipartition of biconnected graphs. Information Processing Letters **33**(5), 227–231 (1990). https://doi.org/10.1016/0020-0190(90)90189-5, https://doi.org/10.1016/0020-0190(90)90189-5
16. Wada, K., Kawaguchi, K.: Efficient algorithms for tripartitioning triconnected graphs and 3-edge-connected graphs. In: International Workshop on Graph-Theoretic Concepts in Computer Science (WG). vol. 790, pp. 132–143. Springer (1993). https://doi.org/10.1007/3-540-57899-4_47, https://doi.org/10.1007/3-540-57899-4_47
17. Zhou, X., Wang, H., Ding, B., Hu, T., Shang, S.: Balanced connected task allocations for multi-robot systems: An exact flow-based integer program and an approximate tree-based genetic algorithm. Expert Systems with Applications **116**, 10–20 (2019). https://doi.org/10.1016/j.eswa.2018.09.001, https://doi.org/10.1016/j.eswa.2018.09.001

A Correctness of Algorithm 2 (`WeightedChordalGL`)

For the correctness Algorithm 2, we need to prove that all vertices are assigned, i.e. the algorithm terminates, and if this is the case, then $S_1, \ldots, S_k$ corresponds to a connected vertex partition satisfying the required weight conditions for each S_i. We start by proving that the Algorithm 2 eventually ends up assigning all vertices into connected vertex sets.

Lemma 12. *Algorithm 2 terminates, such that all vertices are assigned, where $G[S_i]$ is connected for all $i \in [k]$. Further, the while-loop iterates at most $|V|$ times.*

Proof. Similarly to Algorithm 1, because we add only vertices to non-full weighted sets from its unassigned neighborhood, the S_is correspond always to connected vertex sets. Note that by Corollary 1 as long as we have non-full weighted sets and unassigned vertices, the while-loop makes progress in the sense that either an unassigned vertex becomes assigned or a set is declared to be a full weighted set. Thus, if there are unassigned vertices after the while-loop, and we have $|I| = 1$ and we assign the remaining ones to the last non-full weighted set S_j with $j \in I$ (cf. line 19). Observe that Corollary 1 implies that $G[S_j]$ is still connected after adding the remaining vertices.

For the second part of the lemma, when we reach the while loop, there are exactly $|V| - k$ unassigned vertices. Except of at most k times an unassigned vertex becomes assigned in an iteration of the while-loop. This in turn implies that we have not more than $|V|$ iterations.

The running time in Theorem 1 is $\mathcal{O}(|V|^2)$ since the while-loop iterates at most $|V|$ times and each operation in this loop runs in $\mathcal{O}(|V|)$ time. Hence, to prove Theorem 1 it remains to show that the required weight condition for each part of the connected vertex partition $S_1, \ldots, S_k$ is satisfied.

The indices in I denote the non-full weighted sets and therefore, $\overline{I} := [k] \setminus I$ the indices of the full weighted sets. Declaring a set $S_{j'}$ as complete weighted set, i.e. we remove j' from I, implies that $w(S_{j'}) + w(v') \geq w_{j'}$. If $w(S_{j'}) + w(v') \neq w_{j'}$, whether we add v' to $S_{j'}$ depends on whether the value of $\sum_{i \in \overline{I}}(w_i - w(S_i))$ is less than zero or not. Basically, this sum serves to balance the variations in the required weights of the sets $S_1, \ldots, S_k$, and determines the moment we declare $S_{j'}$ to be a full weighted set whether we want $w(S_{j'}) < w_{j'}$ or $w(S_{j'}) > w_{j'}$. During the algorithm the sum $\sum_{i \in \overline{I}}(w_i - w(S_i))$ satisfies the following invariant.

Lemma 13. *In the Algorithm 2, before reaching line 19 we have each time $|\sum_{i \in \overline{I}}(w_i - w(S_i))| < w_{max}$.*

Proof. We prove this lemma by induction on $|\overline{I}|$. After assigning each terminal to a corresponding vertex set, we initialize I by $I = \{i \in [k] \mid w(S_i) < w_i\}$. That is, $\sum_{i \in \overline{I}}(w_i - w(S_i)) = 0$, since either $\overline{I} = \varnothing$ or each $i \in \overline{I}$ satisfy $w(S_i) = w_i$ by $w(t_i) \leq w_i$.

Assume $|\sum_{i \in \overline{I}}(w_i - w(S_i))| < w_{max}$ for $|\overline{I}| \leq \ell < k$ and we now add j' to $\overline{I}$ according to the algorithm, i.e. we remove j' from I. First, we show that

$|w_j - w(S_{j'})| < w_{max}$ in both possible future cases $v' \in S_{j'}$ or $v' \notin S_{j'}$. By $w(S_j \cup \{v'\}) = w_j$ we have added v' to $S_{j'}$ and $|w_j - w(S_{j'})| = 0 < w_{max}$ holds. Thus, we can assume that $w(S_{j'} \setminus \{v'\}) < w_j$ and $w(S_{j'} \cup \{v'\}) > w_{j'}$ which in turn results to $|w_{j'} - w(S_{j'} \cup \{v'\})| < w_{max}$ and $|w_{j'} - w(S_{j'} \setminus \{v'\})| < w_{max}$ by $w(v') \leq w_{max}$.

If $w(S_{j'} \cup \{v'\}) = 0$, we have $\sum_{i \in \overline{I} \setminus \{j'\}} (w_i - w(S_i)) = \sum_{i \in \overline{I}} (w_i - w(S_i))$ and we are done by the induction hypotheses

We can assume that $w(S_{j'} \cup \{v'\}) > w_{j'}$. If $0 \leq \sum_{i \in \overline{I} \setminus \{j'\}} (w_i - w(S_i)) < w_{max}$ we add v' to $S_{j'}$. It follows that $\sum_{i \in \overline{I}} (w_i - w(S_i)) < \sum_{i \in \overline{I} \setminus \{j'\}} (w_i - w(S_i))$ by $w_{j'} - w(S_{j'}) < 0$. By $-w_{max} < w_j - w(S_{j'}) < 0$ the sums might deviate by at most $w_{max} - 1$ from each other. Thus, by $\sum_{i \in \overline{I} \setminus \{j'\}} (w_i - w(S_i)) \geq 0$ we obtain $|\sum_{i \in \overline{I}} (w_i - w(S_i))| < w_{max}$.

In case $-w_{max} < \sum_{i \in \overline{I} \setminus \{j'\}} (w_i - w(S_i)) < 0$ we do not add v' to $S_{j'}$ and obtain $\sum_{i \in \overline{I}} (w_i - w(S_i)) > \sum_{i \in \overline{I} \setminus \{j'\}} (w_i - w(S_i))$ by $w_{j'} - w(S_{j'}) > 0$. Furthermore, by $0 < w_j - w(S_{j'}) < w_{max}$ the sums deviate by at most $w_{max} - 1$ from each other and finally, by $\sum_{i \in \overline{I} \setminus \{j'\}} (w_i - w(S_i)) < 0$ we obtain $|\sum_{i \in \overline{I}} (w_i - w(S_i))| < w_{max}$.

With Lemma 13 we prove now the last part of the proof Theorem 1.

Lemma 14. *If Algorithm 2 terminates, then we have $w_i - w_{max} < w(S_i) < w_i + w_{max}$ for each $i \in [k]$.*

Proof. The sets in $S_1, \ldots, S_k$ with indices $[k] \setminus I$ in the initialization of I, i.e. $I = \{i \in [k] \mid w(S_i) < w_i\}$, satisfy clearly its weight conditions as $w(t_i) \leq w_i$ for all $i \in [k]$. Next, we show that each set that is declared as full weighted set in the while-loop satisfies its weight condition, i.e. $|w_i - w(S_i)| < w_{max}$ for $i \in \overline{I}$. According to Algorithm 2 let j' be the index that we remove from I and consider $S_{j'}$ before possibly adding v' to it. $w(S_{j'}) < w_{j'}$ and $w(S_{j'}) + w(v') \geq w_{j'}$ implies that $w(S_{j'}) < w_{j'} + w_{max}$ independent of v' being added to $S_{j'}$ or not by $w(v') \leq w_{max}$. Similarly, $w(S_{j'}) + w(v') > w_{j'}$ implies that $w(S_{j'}) > w_{j'} - w_{max}$. In case $w(S_{j'}) = w_{j'} - w_{max}$ and $w(v') = w_{max}$ the algorithm adds v' to $S_{j'}$ (cf. line 14) and we have $w_{j'} - w(S_{j'} \cup \{v'\}) = 0 < w_{max}$.

Hence, it remains to prove that the weight conditions are satisfied from the non-full sets if either the while-loop terminates with all vertices assigned, or with $|I| = 1$. We start with the former case. If all vertices are assigned we have $\sum_{i=1}^{k} (w_i - w(S_i)) = \sum_{i=1}^{k} w_i - \sum_{i=1}^{k} w(S_i)) = w(V) - w(V) = 0$. Let I be the indices of the non-full weighted sets after the while-loop is terminated with all vertices assigned and recall $\overline{I} = [k] \setminus I$. Each non-full weighted set S_i for $i \in I$ satisfies $w(S_i) < w_i$ and therefore $\sum_{i \in I} w_i - w(S_i) > 0$ as each value $w_i - w(S_i)$ is greater than zero. Thus, by $\sum_{i=1}^{k} (w_i - w(S_i)) = \sum_{i \in \overline{I}} (w_i - w(S_i)) + \sum_{i \in I} (w_i - w(S_i)) = 0$ we have $\sum_{i \in \overline{I}} (w_i - w(S_i)) < 0$ and by Lemma 13 $-w_{max} < \sum_{i \in \overline{I}} (w_i - w(S_i)) < 0$. Suppose there is an $i \in I$ with $w_i - w(S_i) \geq w_{max}$. This would imply that $0 = \sum_{i \in \overline{I}} (w_i - w(S_i)) + \sum_{i \in I} (w_i - w(S_i)) > -w_{max} + \sum_{i \in I} (w_i - w(S_i)) \geq -w_{max} + w_{max} = 0$, which is a contradiction.

It remains to consider the case that the while-loop terminates when $|I| = 1$. Let say $I = \{\ell\}$ and the remaining unassigned vertices are already added to S_ℓ

according to line 19. Same as above, the S_i's with $i \in [k] \setminus \{\ell\} = \overline{I}$ satisfy its required weight condition and hence, we need to show that $w_\ell - w_{max} < w(S_\ell) < w_\ell + w_{max}$ holds. By $\sum_{i\in\overline{I}}(w_i - w(S_i)) + \sum_{i\in I}(w_i - w(S_i)) = \sum_{i\in\overline{I}}(w_i - w(S_i)) + (w_\ell - w(S_\ell)) = 0$ we obtain $\sum_{i\in\overline{I}}(w_i - w(S_i) = w(S_\ell) - w_\ell$. As a result, since $|\sum_{i\in\overline{I}}(w_i - w(S_i)| < w_{max}$ by Lemma 13, it follows that $|w(S_\ell) - w_\ell| < w_{max}$.

B Omitted Proofs

Proof of Lemma 1

Proof. Assume for a contradiction that there is an index $j \in \{2, \ldots, k-1\}$ such that $\sigma(v_j) = \min_{i\in[k]}\{\sigma(v_i)\}$. As a result $\sigma(v_{j-1}) > \sigma(v_j)$ and $\sigma(v_j) < \sigma(v_{j+1})$. This means however when v_j is assigned a σ value, none of v_{j-1} and v_{j+1} have been assigned such a value. Since v_j is a simplicial vertex at that point, its neighbors that still have not been assigned a σ value induce a clique. Hence, $v_{j-1}v_{j+1} \in E(G)$, which contradicts the fact that P induces a simple path.

Proof of Lemma 3

Proof. Let $v \in V \setminus C$ be adjacent to two vertices u and w in C. If $uw \in E(G)$ then, since G is house-free, v is also adjacent to at least one vertex of C other than u and w. Then however, if v is only adjacent to three vertices of C, it induces a C_4 with two of the adjacent and the non adjacent vertex, that shares 3 vertices with C. In the case where $uw \notin E(G)$, in order to not have two induced C_4 sharing three vertices, v has to also be adjacent to another vertex of C, which leads us to the previous case where v is adjacent to two vertices of C inducing an edge and hence universal to C.

Moreover notice that any set of universal vertices to C, induces a clique since otherwise two induced C_4 exist that share two vertices (consider C and the C_4 induced by two non adjacent vertices that are universal to C and two non adjacent vertices of C).

Proof of Lemma 4

Proof. Let G be an HHI_4^2-free graph and H be a subgraph of G that is a double house. As illustrated in Figure 2, we denote by C^1 and C^2 the two induced C_4 of H, by u_1 their common vertex, by u_2 and u_3 the ones adjacent to u_1 and to each other, belonging in C^2 and C^1, respectively and finally by u_{ij} the vertex of C_i that is adjacent to u_j.

Notice that by Lemma 3, since u_2 is adjacent to two vertices of C^1 it is universal to C^1, while the same holds for u_3 and C^2. After adding those edges to H however u_{21} is adjacent to two of the vertices of C^1, and hence is universal to C^1 in G, while the same holds for u_{11} and C^2. After this addition however u_{13} is adjacent to both u_2 and u_{21} of C^2 and hence universal to C^2, which creates the chord $u_{13}u_1$ in C^1 that concludes this proof.

Proof of Lemma 5

Proof. Assume for a contradiction that the addition of v_1v_3 creates a new induced C_4, $C' = \{v'_1, v_1, v_3, v'_3\}$, where v'_1, and v'_3 are the neighbors on C' of v_1 and v_3 respectively. Notice that v'_1 and v'_3 do not belong in C. In order for $v_1, v_2, v_3, v'_3, v'_1$ not to induce a hole in G, either v'_1 or v'_3, say v'_1, has to also be adjacent to either v_2 or v_3. This however, due to Lemma 3, means that v'_1 is universal to C in G, which creates a chord on C'.

It remains now to show that G' is still HHI_4^2-free. Using similar arguments as before we see that no hole is formed from the addition of v_1v_3 and since no new C_4 is formed also, the remaining C_4 keep having pairwise at most one vertex in common. Assume now for a contradiction that adding v_1v_3 creates an induced house H in G'. Since, as we showed above, v_1v_3 does not participate in any induced C_4 it must be one of the two roof's edges. Notice also that from the vertices of C, only v_1 and v_3 participate in this house because otherwise G would contain two induced C_4 sharing two vertices. Let u be the third vertex of the roof and notice that by Lemma 3, u is also adjacent to v_2 and v_4, and let w and z be the remaining two vertices of the house (assume that $wu, zv_3 \in E(G)$). In order for wzv_2v_3u not to induce a house in G either $v_2z \in E(G)$ or $v_2w \in E(G)$. Notice that through these cases we conclude, again by Lemma 3, that either w or z is universal to C. We have assumed however that H is an induced house, hence w is not universal to C because that would create a chord in the body of the house. As a result z is universal to C, which again leads to a contradiction to the fact that H is an induced house because of the edge v_1z.

Proof of Lemma 6

Proof. As we have stated before HHI_4^2-free graphs are in particular HHD-free. Since HHD-free graphs are closed under edge contraction, no hole or house occurs after contracting any edge of an HHI_4^2-free graph.

Let $C = \{v_1, v_2, v_3, v_4\}$ be an induced C_4 in G and consider contracting the edge v_1v_2. Let G' be the graph resulting from this contraction, and let v_{12} be the newly added vertex.

We first show that contracting v_1v_2 does not create any new C_4. Assume for a contradiction that G' contains a new induced C_4, $C' = \{v_{12}, u_1, u_2, u_3\}$.

- $v_3 \notin C'$
  Let u_1, u_2 be the neighbors of v_{12} on C'. Since C' is new we have that $u_1v_1, u_2v_2 \in E(G)$ and $u_1v_2, u_2v_1 \notin E(G)$. In order for u_3, u_2, v_2, v_1, u_1 to not induce a hole in G, at least one of u_3v_1, u_3v_2, u_1u_2 has to also exists as edges in G. This however would create a chord in C' which contradicts our assumption that C' is an induced C_4 in G'.
- $v_3 \in C'$
  In order for $v_{12}v_3$ not to be a chord in C', this edge participates in the induced C_4, and also $u_1, u_3 \neq v_2, v_4$ (assuming $v_3 = u_2$ and that $u_3v_3, u_1v_{12} \in E(G')$). Since C' is new and $v_2v_3 \in E(G)$ we conclude that $v_1u_1 \in E(G)$

and $v_2u_1 \notin E(G)$. Notice now that in order for v_1, u_1, v_2, v_3, u_3 not to induce a hole in G, u_3 should be adjacent in G to either v_1 or v_2. This however would create a chord on C' in G' which leads to a contradiction.

Now it remains to show that contracting v_1v_2 does not create two induced C_4 that share more than one vertex. Assume now for a contradiction that G' contains two C_4 $Z = \{z_1, z_2, z_3, z_4\}$, $W = \{w_1, w_2, w_3, w_4\}$ that share more than one vertex.

First, assume that Z and W share an edge, thus let $z_1 = w_1$ and $z_2 = w_2$ be two of the common vertices of Z and W. We can directly assume that $z_1 = v_{12}$, and $z_3, z_4 \notin W$ and $w_3, w_4 \notin Z$, since previously any pair of C_4 shared at most one vertex. By Lemma 5, contracting v_1v_2 did not create any new induced C_4, thus Z and W with the vertex v_{12} replaced by either v_1 or v_2 were already induced C_4 in G, so assume that $\{v_1, w_2, w_3, w_4\}$ induces a C_4 in G.

Since G was HHI_4^2-free it follows that v_1 is not adjacent to z_4 (otherwise $\{v_1, w_2, w_3, w_4\}$ and $\{v_1, z_2, z_3, z_4\}$ are two induced C_4 sharing more than one vertex), thus $\{v_2, z_2, z_3, z_4\}$ also induces a C_4 in G.

Then, however in order for v_1, v_2, z_2, w_3, w_4 not to induce a house in G either w_3 or w_4 is adjacent to v_2, which would create a chord in W after the contraction of v_1v_2 that leads to a contradiction.

It remains to consider the case that Z and W share two non-adjacent vertices, i.e. $w_1 = z_1 = v_{12}$ and $w_3 = z_3$ are the two common vertices. Similarly to the previous case, we can use Lemma 5 to assume that $\{v_1, w_2, w_3, w_4\}$ and $\{v_2, z_2, z_3, z_4\}$ are induced C_4 in G. Since v_1, v_2, z_2, z_3 does not create an induced house or hole together with w_2 or w_4, it follows that either v_1 is adjacent to z_3, which would create a chord for Z in G', or w_2 and w_4 are adjacent to v_1 or z_2. In the latter case, w_2 and w_4 are adjacent to more than one vertex of the induced C_4 Z, which means they are both universal to Z and have to form a clique by Lemma 3. Then however w_2w_4 is a chord for W in G'.

Proof of Lemma 8

Proof. Assume for a contradiction that G' is only $k-1$-connected, thus there is a separator of size $k-1$ that disconnects two distinct vertices u and w in G'.

Since the connectivity between u and w dropped after contracting v_1v_2, in G there are two internally vertex disjoint paths P_1 and P_2 that connect u and w such that $v_1 \in P_1$ and $v_2 \in P_2$. We can further assume that P_1 and P_2 are two such paths of minimal length, respectively. Denote also by u_1 and by w_1 the neighbors of v_1 on P_1 which are closer to u and w, respectively. (See the right illustration in Figure 2 for an example of these namings, keeping in mind that it could be that $u_1 = u_2 = u$ and/or $w_1 = w_2 = w$.) Similarly, denote by u_2 and w_2 these neighbors of v_2 on P_2.

In order for v_1, v_2 and u not to be part of an induced hole (given that u_1 and u_2 are not u) either $u_1u_2 \in E(G)$ or $u_1v_2 \in E(G)$ or $u_2v_1 \in E(G)$. The first case, however, if no other edges existed, would create a domino while the other cases form a house. Hence at least one of the edges u_1v_3, u_1v_4, u_2v_3, u_2v_4

exists. Similarly, assuming that neither w_1 nor w_2 is equal to w, since our graph is HHI_4^2-free, also one of w_1v_3, w_1v_4, w_2v_3, w_2v_4 exists. We show now that by using the remaining vertices of C we can recreate the two vertex disjoint paths that previously existed in G.

1. $u_1v_3 \in E(G)$
   (a) $w_1v_3 \in E(G)$: Notice that the two $u-w$ vertex disjoint paths are preserved in G', specifically $P_1' = P_1^{[u,u_1]}v_3P_1^{[w_1,w]}$ and $P_2' = P_2$ (where in P_2 we have instead of v_2 the newly created vertex v_{12}).
   (b) $w_1v_4 \in E(G)$: $P_1' = P_1^{[u,u_1]}v_3v_4P_1^{[w_1,w]}$, $P_2' = P_2$
   (c) $w_2v_3 \in E(G)$: $P_1' = P_1^{[u,u_1]}v_3w_2P_2^{[w_2,w]}$, $P_2' = P_2^{[u,v_{12}]}P_1^{[v_{12},w]}$
   (d) $w_2v_4 \in E(G)$: $P_1' = P_1^{[u,u_1]}v_3v_4w_2P_2^{[w_2,w]}$, $P_2' = P_2^{[u,v_{12}]}P_1^{[v_{12},w]}$
2. $u_1v_4 \in E(G)$
   (a) $w_1v_3 \in E(G)$: $P_1' = P_1^{[u,u_1]}v_4v_3P_1^{[w_1,w]}$ and $P_2' = P_2$
   (b) $w_1v_4 \in E(G)$: $P_1' = P_1^{[u,u_1]}v_4P_1^{[w_1,w]}$, $P_2' = P_2$
   (c) $w_2v_3 \in E(G)$: $P_1' = P_1^{[u,u_1]}v_4v_3w_2P_2^{[w_2,w]}$, $P_2' = P_2^{[u,v_{12}]}P_1^{[v_{12},w]}$
   (d) $w_2v_4 \in E(G)$: $P_1' = P_1^{[u,u_1]}v_4w_2P_2^{[w_2,w]}$, $P_2' = P_2^{[u,v_{12}]}P_1^{[v_{12},w]}$
3. The remaining cases are symmetrical to the ones written above.

Similar arguments can be used to find such paths if $u_1 = u_2 = u$ or $w_1 = w_2 = w$.

Due to Lemma 7 we know that v_3 and v_4 are free to be used for the creation of the above paths since they can not be part of the same minimal separator as v_1 and v_2.

Proof of Lemma 10

Proof. Observe that by Lemma 5 we can safely add the edges in line 1, in the sense that we still have an HHI_4^2-free graph, and that we do not create new C_4 that our terminal vertices might participate in. Thus, the graph we consider moving forward in the algorithm is HHI_4^2-free and the induced C_4 considered in line 2 do not contain two non-adjacent terminals.

Consider now the tree T constructed by $\mathcal{C}$ in Lemma 9. We proceed in arguing that the contraction order described in line 6 of Algorithm 3 is indeed the desired one.

Let v be a leaf of T. If $v \in S$ (representing a single vertex in G) then delete v and update T accordingly. If $v \in B$ then consider the induced C_4 corresponding to v. The fact that v is a leaf in T means that there are at least three vertices in the corresponding C_4 that are not included in any other induced C_4 of G. These three vertices v_1, v_2, v_3 are candidates for line 6 of Algorithm 3, and after removal of them and the C_4 corresponding to v in G' in line 8, removing v from T yields a tree for which this procedure can be repeated.

Since we remove all vertices involved in a contracted edge from G' as soon as their first adjacent edge is chosen, we can ensure that no vertex is contracted twice. Thus when we create G'' all vertices indeed have maximum weight at most 2.